MENG 366
System Dynamics and Control Laboratory
Experiment 4: Transfer Function of LTI Systems

Objectives: This experiment has following two objectives:

1. We will learn commands in MATLAB that would be used to represent the linear systems in terms of transfer function or pole-zero gain representations.
2. We will also learn how to make preliminary analysis of such systems using plots of poles and zeros locations as well as time response due to impulse, step and arbitrary inputs.

List of Equipment/Software

Following equipment/software is required:

 MATLAB

Category

Soft-Experiment

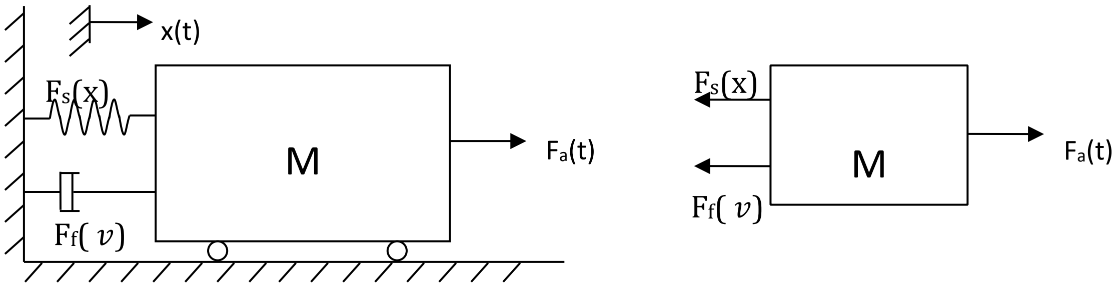
Deliverables

A complete lab report including the following:

- Summarized learning outcomes.
- MATLAB scripts and their results should be reported properly.

Mass-Spring System Model

The spring force is assumed to be either linear or can be approximated by a linear function $F_s(x)$= Kx, B is the friction coefficient, x(t) is the displacement and $F_a(t)$ is the applied force:



The differential equation for the above Mass-Spring system can be derived as follows

$$M\frac{d\ \ x(t)}{dt} + B\frac{dx(t)}{dt} + Kx(t) = F_{a_{()}}$$

Transfer Function:

Applying the Laplace transformation while assuming the initial conditions are zeros, we get

$$(Ms\ \ + Bs + K) * X(s) = F_a(s)$$

Then the transfer function representation of the system is given by

$$TF = \frac{Output}{Input} = \frac{F_a(s)}{X(s)} = \frac{1}{(Ms \quad + Bs + K)}$$

Linear Time-Invariant Systems in MATLAB:
Control System Toolbox in MATLAB offers extensive tools to manipulate and analyze linear time-invariant (LTI) models. It supports both continuous- and discrete-time systems. Systems can be single-input/single-output (SISO) or multiple-input/multiple-output (MIMO). You can specify LTI models as:

Transfer functions (TF), for example,

$$P(s) = \frac{s + 2}{s \quad + s + 10}$$

<u>**Note:**</u> All LTI models are represented as a ratio of polynomial functions

Examples of Creating LTI Models
Building LTI models with Control System Toolbox is straightforward. The following sections show simple examples. Note that all LTI models, i.e. TF, ZPK and SS are also MATLAB objects.

Example of Creating Transfer Function Models
You can create transfer function (TF) models by specifying numerator and denominator coefficients. For example,

```
>>num = [1 0];
>>den = [1 2 1];
>>sys = tf(num,den)
```

```
Transfer    function:
s
------------- s^2
+ 2 s + 1
```

A useful trick is to create the Laplace variable, s. That way, you can specify polynomials using s as the polynomial variable.

```
>>s=tf('s');
>>sys= s/(s^2 + 2*s + 1)
```

```
Transfer function:
    s
------------- s^2
+ 2 s + 1
```

This is identical to the previous transfer function.

Example of Creating Zero-Pole-Gain Models

To create zero-pole-gain (ZPK) models, you must specify each of the three components in vector format. For example,

```
>>sys = zpk([0],[-1 -1],[1])
```

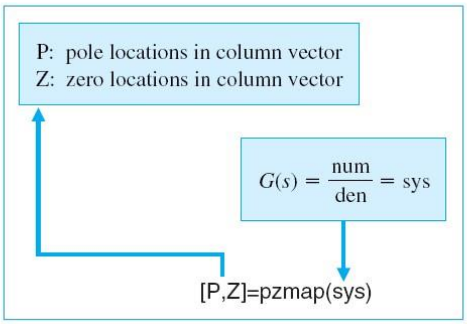
```
Zero/pole/gain:

s -------
(s+1)^2
```

produces the same transfer function built in the TF example, but the representation is now ZPK. This example shows a more complicated ZPK model.

```
>>sys=zpk([1 0], [-1 -3 -.28],[.776])
```

```
Zero/pole/gain:
0.776 s (s-1)
--------------------
(s+1) (s+3) (s+0.28)
```

Plotting poles and zeros of a system:


P: pole locations in column vector
Z: zero locations in column vector
$G(s) = \frac{\text{num}}{\text{den}} = \text{sys}$
[P,Z]=pzmap(sys)

pzmap

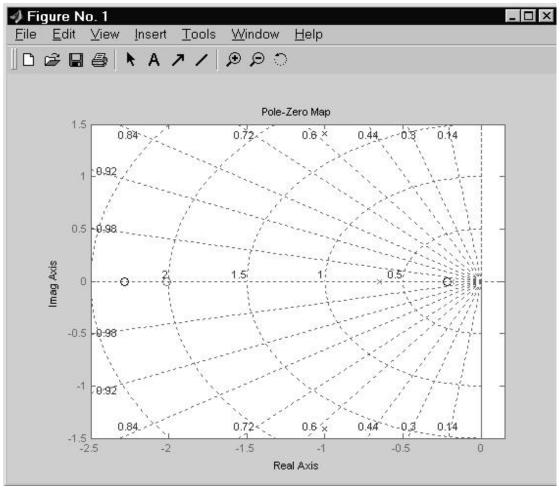
Compute pole-zero map of LTI models

```
pzmap(sys)  pzmap(sys1,sys2,...,sysN)
[p,z] = pzmap(sys)
```

Description:

pzmap(sys) plots the pole-zero map of the continuous- or discrete-time LTI model sys. For SISO systems, pzmap plots the transfer function poles and zeros. The poles are plotted as x's and the zeros are plotted as o's.

pzmap(sys1,sys2,...,sysN) plots the pole-zero map of several LTI models on a single figure. The LTI models can have different numbers of inputs and outputs. When invoked with left-hand arguments, [p,z] = pzmap(sys) returns the system poles and zeros in the column vectors p and z. No plot is drawn on the screen. You can use the functions sgrid or zgrid to plot lines of constant damping ratio and natural frequency in the s- or z- plane.



Example

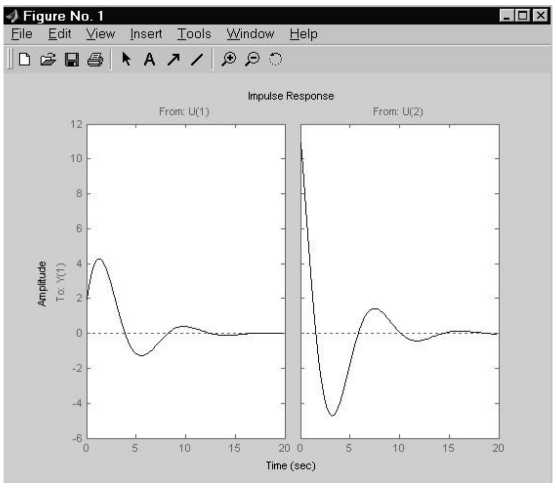
Plot the poles and zeros of the continuous-time system.

$$H(s) = \frac{2s \quad + 5s + 1}{s \quad + 2s + 3}$$

```
>>H = tf([2 5 1],[1 2 3]); sgrid
```

```
>>pzmap(H)
```

Simulation of Linear systems to different inputs



impulse, step and lsim

You can simulate the LTI systems to inputs like impulse, step and other standard inputs and see the plot of the response in the figure window. MATLAB command 'impulse' calculates the unit impulse response of the system, 'step' calculates the unit step response of the system and 'lsim' simulates the (time) response of continuous or discrete linear systems to arbitrary inputs. When invoked without left-hand arguments, all three commands plots the response on the screen. For example:

To obtain an impulse response

```
>> H = tf([2 5 1],[1 2 3]);  >>impulse(H)
```

To obtain a step response type

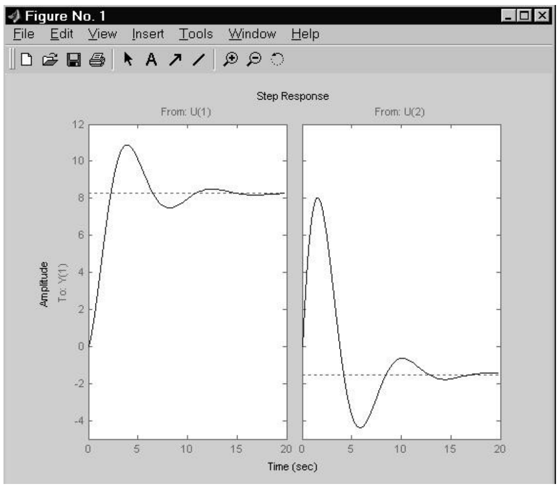
```
>>step(H)
```

Time-interval specification:

To contain the response of the system you can also specify the time interval to simulate the system to.

```
>> t = 0:0.01:10;
>> impulse(H,t)
```

For example,



```
>> step(H,t)
```

Or

Simulation to Arbitrary Inputs:

To simulates the (time) response of continuous or discrete linear systems to arbitrary inputs use 'lsim'. When invoked without left-hand arguments, 'lsim' plots the response on the screen. lsim(sys,u,t) produces a plot of the time response of the LTI model sys to the input time history 't','u'. The vector 't' specifies the time samples for the simulation and consists of regularly spaced time samples.

T = 0:dt:Tfinal

The matrix u must have as many rows as time samples (length(t)) and as many columns as system inputs. Each row u(I,☺ specifies the input value(s) at the time sample t(i).

Simulate and plot the response of the system

to a square period of four $H(s) = \frac{2s\quad + 5s + 1}{s\quad + 2s + 3}$ wave with seconds.

First generate the square wave with gensig. Sample every 0.1 second during 10 seconds:

```
>>[u,t] = gensig('square',4,10,0.1);
```

Then simulate with lsim.

```
>> H = tf([2 5 1],[1 2 3])
```

Transfer function: 2
s^2 + 5 s + 1

---

s^2 + 2 s + 3

```
>> lsim(H,u,t)
```

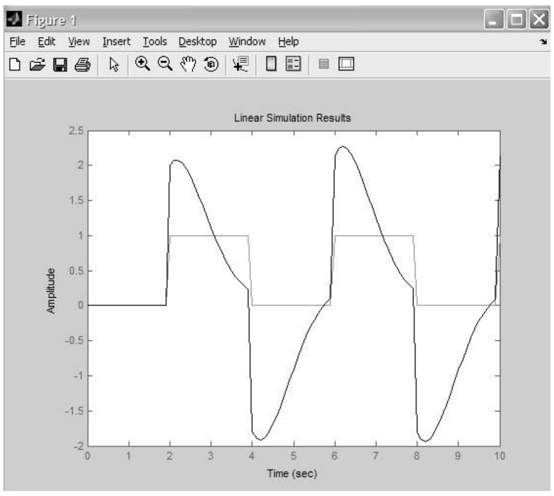


Figure 1
File Edit View Insert Tools Desktop Window Help
Linear Simulation Results
Amplitude
Time (sec)
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0 1 2 3 4 5 6 7 8 9 10