Lab report no 1

Fall 2022

Control System Lab

Submitted By

Names	Registration No
Muhammad Ali	**19pwcse1801**

Section: **A**

Date: 12,10,22

Submitted to: Dr Muniba Ashfaq

Department of Computer Systems Engineering
University of Engineering and Technology, Peshawar

Objectives: -

- **To learn and get familiar with the basics function using help command.**
- **To practice each command in MATLAB.**

With help: Description and example of following functions: -

```
>> help roots

ROOTS  Find polynomial roots.

ROOTS(C) computes the roots of the polynomial whose coefficients

are the elements of the vector C. If C has N+1 components,

the polynomial is C(1)*X^N + ... + C(N)*X + C(N+1).


>> help poly

 POLY Convert roots to polynomial.

   POLY(A), when A is an N by N matrix, is a row vector with

   N+1 elements which are the coefficients of the

   characteristic polynomial, DET(lambda*EYE(SIZE(A)) - A) .

   POLY(V), when V is a vector, is a vector whose elements are

   the coefficients of the polynomial whose roots are the

   elements of V . For vectors, ROOTS and POLY are inverse

   functions of each other, up to ordering, scaling, and

   roundoff error.


>> help polyval

 POLYVAL Evaluate polynomial.

   Y = POLYVAL(P,X) returns the value of a polynomial P evaluated at X. P

   is a vector of length N+1 whose elements are the coefficients of the

   polynomial in descending powers.


       Y = P(1)*X^N + P(2)*X^(N-1) + ... + P(N)*X + P(N+1)
```

Example:

Evaluate the polynomial p(x) = 3x^2+2x+1 at x = 5,7, and 9:

```
p = [3 2 1];
polyval(p,[5 7 9])%
```

```
>> help conv
 CONV Convolution and polynomial multiplication.
    C = CONV(A, B) convolves vectors A and B.  The resulting vector is
    length MAX([LENGTH(A)+LENGTH(B)-1,LENGTH(A),LENGTH(B)]). If A and B are
    vectors of polynomial coefficients, convolving them is equivalent to
    multiplying the two polynomials.
```

```
>> help tf
 TF  Construct transfer function or convert to transfer function.

   Construction:
     SYS = TF(NUM,DEN) creates a continuous-time transfer function SYS with
     numerator NUM and denominator DEN. SYS is an object of type TF when
     NUM,DEN are numeric arrays, of type GENSS when NUM,DEN depend on tunable
     parameters (see REALP and GENMAT), and of type USS when NUM,DEN are
     uncertain (requires Robust Control Toolbox).

     SYS = TF(NUM,DEN,TS) creates a discrete-time transfer function with
     sampling time TS (set TS=-1 if the sampling time is undetermined).

     S = TF('s') specifies the transfer function H(s) = s (Laplace variable).
     Z = TF('z',TS) specifies H(z) = z with sample time TS.
     You can then specify transfer functions directly as expressions in S
     or Z,
```

for example,

```
s = tf('s');  H = exp(-s)*(s+1)/(s^2+3*s+1)
```

help pzmap

PZMAP Pole-zero map of dynamic systems.

PZMAP(SYS) computes the poles and (transmission) zeros of the dynamic system SYS and plots them in the complex plane. The poles are plotted as x's and the zeros are plotted as o's.

PZMAP(SYS1,SYS2,...) shows the poles and zeros of several systems SYS1,SYS2,... on a single plot. You can specify distinctive colors for each model, for example:

```
pzmap(sys1,'r',sys2,'y',sys3,'g')
```

[P,Z] = PZMAP(SYS) returns the poles and zeros of the system in two column vectors P and Z. No plot is drawn on the screen.

>> help impulse

IMPULSE Impulse response of dynamic systems.

IMPULSE(SYS) plots the impulse response of the dynamic system SYS. For systems with more than one input, independent impulse commands are applied to each input channel. The time range and number of points are chosen automatically. For continuous-time systems with direct feedthrough, the infinite pulse at t=0 is ignored.

IMPULSE(SYS,TFINAL) simulates the impulse response from t=0 to the final time

t=TFINAL (expressed in the time units specified in SYS.TimeUnit). For
discrete-time models with unspecified sampling time, TFINAL is interpreted
as the number of samples.

```
>> help step
```

STEP Step response of dynamic systems.

STEP(SYS) plots the step response of the dynamic system SYS. For systems with more than one input, independent step commands are applied to each input channel. The time range and number of points are chosen automatically.

STEP(SYS,TFINAL) simulates the step response from t=0 to the final time t=TFINAL (expressed in the time units specified in SYS.TimeUnit). For discrete-time models with unspecified sampling time, TFINAL is interpreted as the number of samples.

```
>>help series
```

SERIES Series connection of two input/output models.

M = SERIES(M1,M2,OUTPUTS1,INPUTS2) connects the input/ouput models M1 and M2 in series. The vectors of indices OUTPUTS1 and INPUTS2 specify which outputs of M1 and which inputs of M2 are connected together. The resulting model M maps u1 to y2.

```
>> help parallel
```

parallel is both a directory and a function.

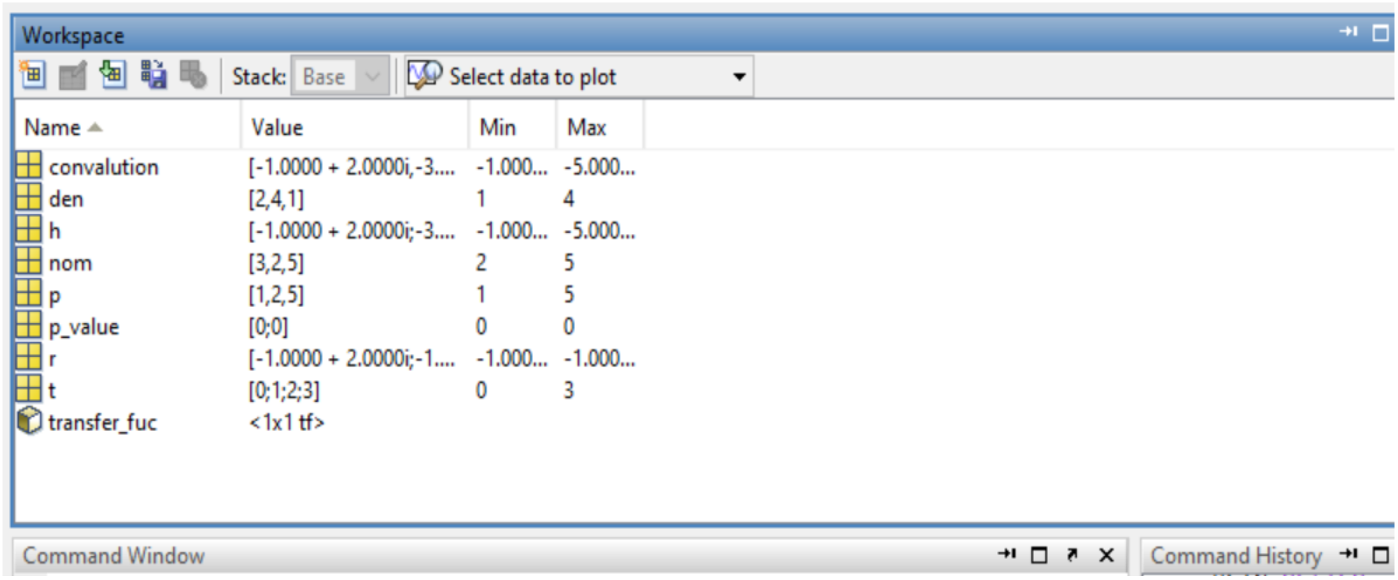
PARALLEL Parallel connection of two input/output models.

M = PARALLEL(M1,M2,IN1,IN2,OUT1,OUT2) connects the input/output models M1 and M2 in parallel. The inputs specified by IN1 and IN2 are connected and the outputs specified by OUT1 and OUT2 are summed. The resulting model M maps [v1;u;v2] to [z1;y;z2]. The vectors IN1 and IN2 contain indices into the input vectors of M1 and M2, respectively, and define the input channels u1 and u2 in the diagram. Similarly, the vectors OUT1 and OUT2 contain indexes into the outputs of M1 and M2.

Practical examples and Results: -

```
clc
clear all
close all

%initial values for d/f function
p=[1 2 5];
den = [2 4 1];
nom = [3 2 5];

%finding roots
r=roots(p);

%finding poly
p=poly(r);

%finding ployval
p_value = polyval(p,r);

%finding conv
convalution = conv(r,p);

%finding transfer function
transfer_fuc = tf(nom,den);

%finding pzmap
pzmap(transfer_fuc);

%finding impulse
[h,t]= impz(convalution);

%finding step
step(transfer_fuc);
```

Resulted values: -



Name	Value	Min	Max
convalution	[-1.0000 + 2.0000i,-3....	-1.000...	-5.000...
den	[2,4,1]	1	4
h	[-1.0000 + 2.0000i;-3....	-1.000...	-5.000...
nom	[3,2,5]	2	5
p	[1,2,5]	1	5
p_value	[0;0]	0	0
r	[-1.0000 + 2.0000i;-1....	-1.000...	-1.000...
t	[0;1;2;3]	0	3
transfer_fuc	<1x1 tf>		

Step: