**Introduction to Different functions in Matlab**

**Lab report #01**

****

Fall 2022

CSE-310L Control Systems

Submitted by: **Ashfaq Ahmad**

Registration No: **19PWCSE1795**

Class Section: **B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Dr: Muniba Ashfaq**

**Oct** 12, 2022

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

1) Use the “help” command of MATLAB to get familiar with the following functions.

1. **Roots:**

**Syntax:** r=roots (p)

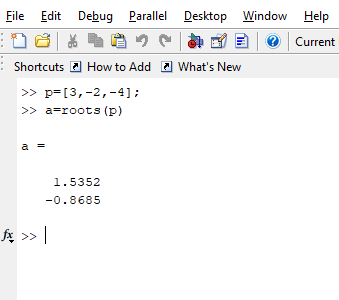
**Description:**

r = roots (p) returns the roots of the polynomial represented by p as a column vector. Input p is a vector containing n+ 1 polynomial coefficient, starting with the coefficient of xn. A coefficient of 0 indicates an intermediate power that is not present in the equation. For example, p = [3 2 -2] represents the polynomial 3*x*2+2*x*−2.

The roots function solves polynomial equations of the form *p*1*xn*+...+*pnx*+*pn*+1=0. Polynomial equations contain a single variable with nonnegative exponents.

**Example:**

Solve the equation 3*x*2−2*x*−4=0.

****

1. **Ploy:**

**Syntax:** p=poly(r)

**Description:**

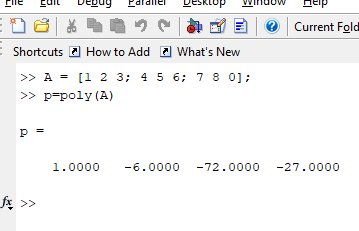
It’s opposite to roots function I,e return roots to polynomial.

p=poly(r) returns the coefficients of polynomial p, where r is the root vector of polynomial p

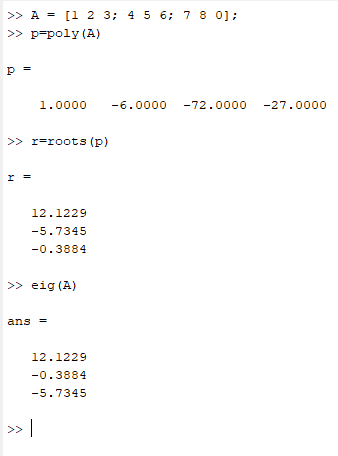
**Example:**

Use poly to calculate the characteristic polynomial of a matrix, A.

A = [1 2 3; 4 5 6; 7 8 0]



Note: if we wanna find roots of p it will give Eigen values of vector A. I,e



1. **Polyval:**

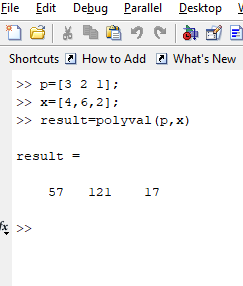
**Syntax:** y=poly val(p,x)

**Description:**

Y=polyval(p) evaluates the polynomial p at each point in x. The argument p is a vector of length n+1 whose elements are the coefficients (in descending powers) of an nth-degree polynomial:

**Example:**

Evaluate the polynomial *p*(*x*)=3*x*2+2*x*+1 at the points *x*=5,7,9

****

1. **Conv:**

**Syntax:** c=conv(a,b)

**Description:**

c = conv(a,b) returns the convolution of input vectors a and b, at least one of which must be a fi object.

1. **Residue:**

**Syntax:**

[r,p,k]=residue[b,a]

[b,a]=residue[r,p,k]

**Description:**

[r,p,k]=residue[b,a] finds the residues, poles, and direct term of a partial fraction expansion of ratio of two polynomials.

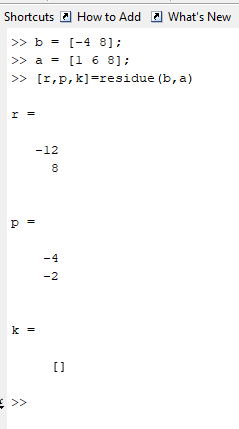
[b,a]=residue[r,p,k] converts the partial fraction expansion back to the ratio of two polynomials and returns the coefficients in b and a.

Simply residue is the inter-conversion of partial fraction expansion and ratio of two polynomials.

**Example:**

Find the partial fraction expansion of the following ratio of polynomials F(s) using residue

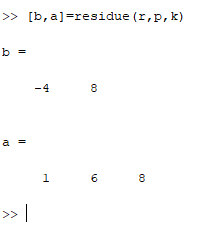
F(s)=b(s)/a(s)=-4s+8/s2+6s+8

****

This represents the partial fraction expansion.

-4s+8/s2+6s+8=-12/(s+4) + 8/(s+2)

Now find coefficients of ratio of polynomials from above partial fraction.



1. **tf:**

**Syntax:**

Sys=tf(num,denum)

Sys=tf(num,denum,ts)

**Description:**

SYS = TF(NUM,DEN) creates a continuous-time transfer function SYS with

numerator(s) NUM and denominator(s) DEN. The output SYS is a TF object.

SYS = TF(NUM,DEN,TS) creates a discrete-time transfer function with

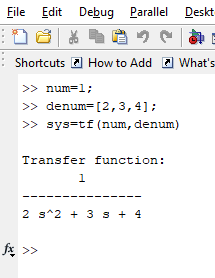
sample time TS (set TS=-1 if the sample time is undetermined).

Example:

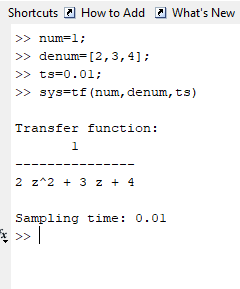
consider the following SISO transfer function model:

sys(s)=1/2s2+3s+4

**Continues time tf:**



**Discrete time tf:**

****

1. **pzmap:**

**Syntax:**

pzmap(sys)

pzmap(sys1,sys2,………)

**Description:**

PZMAP(SYS) computes the poles and (transmission) zeros of the LTI model 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 multiple LTI models SYS1,SYS2,... on a single plot. You can specify distinctive colors for each model,

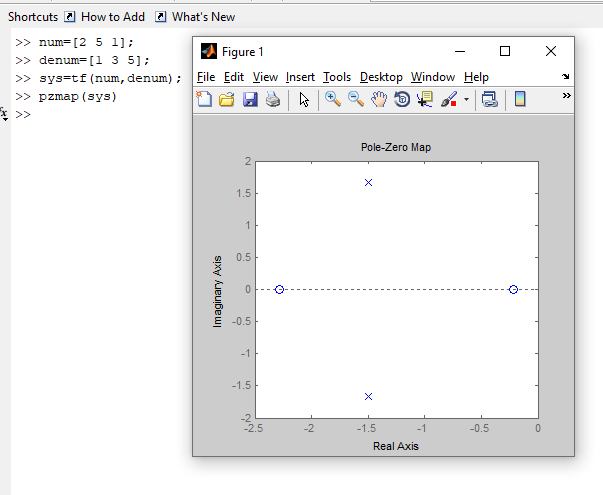
**as in pzmap(sys1,'r',sys2,'y',sys3,'g')**

[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.

**Example:**

Plot the poles and zeros of the continuous-time system represented by the following transfer function.

H(s)=2s2+5s+1/s2+3s+5

****

1. **impulse:**

**Syntax:** impulse (sys)

**Description**:

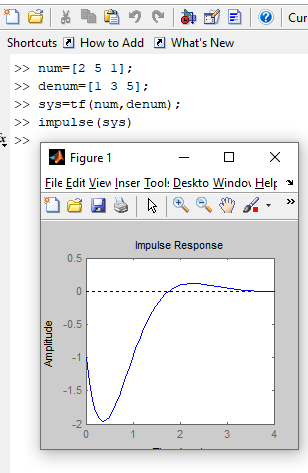
IMPULSE (SYS) plots the impulse response of the LTI model SYS (created with TF, ZPK, or SS). For multi-input models, independent impulse commands are applied to each input channel. The time range and number of points are chosen automatically**.**

IMPULSE (SYS, TFINAL) simulates the impulse response from t=0 to the final time t=TFINAL. For discrete-time systems with unspecified sampling time, TFINAL is interpreted as the number of samples.

**Example:**

Find impulse response of given function.

H(s)=2s2+5s+1/s2+3s+5



1. **step:**

**Syntax:** step (sys)

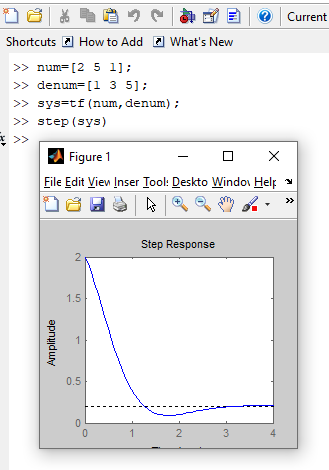
**Description:**

STEP (SYS) plots the step response of the LTI model SYS (created with either TF, ZPK, or SS). For multi-input models, 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. For discrete-time models with unspecified sampling time, TFINAL is interpreted as the number of samples.

**Example:**

Find step response of given function.

H(s)=2s2+5s+1/s2+3s+5

****

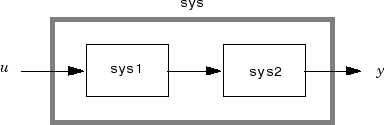
1. **series:**

**Syntax:**   
sys = series(sys1,sys2)  
sys = series(sys1,sys2,outputs1,inputs2)

**Description:**

series connects two model objects in series. This function accepts any type of model. The two systems must be either both continuous or both discrete with identical sample time. Static gains are neutral and can be specified as regular matrices.

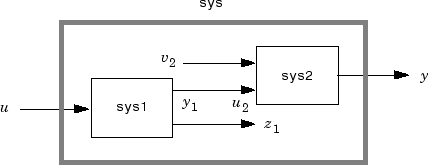
sys = series(sys1,sys2) forms the basic series connection shown below.



This command is equivalent to the direct multiplication

sys = sys2 \* sys1

sys = series(sys1,sys2,outputs1,inputs2) forms the more general series connection.



The index vectors outputs1 and inputs2 indicate which outputs y1 of sys1 and which inputs u2 of sys2 should be connected. The resulting model sys has u as input and y as output.

**Example:**

Consider a state-space system sys1 with five inputs and four outputs and another system sys2 with two inputs and three outputs. Connect the two systems in series by connecting outputs 2 and 4 of sys1 with inputs 1 and 2 of sys2.

outputs1 = [2 4];

inputs2 = [1 2];

sys = series(sys1,sys2,outputs1,inputs2)

1. **parallel:**

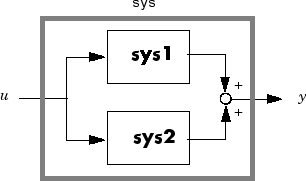
**Syntax:**

sys = parallel(sys1,sys2)  
sys = parallel(sys1,sys2,inp1,inp2,out1,out2)

**Description:**

Parallel connects two model objects in parallel. This function accepts any type of model. The two systems must be either both continuous or both discrete with identical sample time. Static gains are neutral and can be specified as regular matrices.

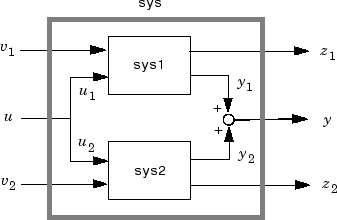
sys = parallel(sys1,sys2) forms the basic parallel connection shown in the following figure.



This command equals the direct addition

sys = sys1 + sys2

sys = parallel(sys1,sys2,inp1,inp2,out1,out2) forms the more general parallel connection shown in the following figure.



The vectors inp1 and inp2 contain indexes into the input channels of sys1 and sys2, respectively, and define the input channels u1 and u2 in the diagram. Similarly, the vectors out1 and out2 contain indexes into the outputs of these two systems and define the output channels y1 and y2 in the diagram. The resulting model sys has [v1 ; u ; v2] as inputs and [z1 ; y ; z2] as outputs.

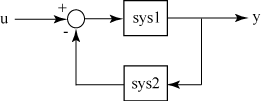
1. **Feedback:**

**Syntax:**

Sys=feedback (sys1, sys2)

Sys=feedback (sys1, sys2, feedin, feedout)

**Description**:

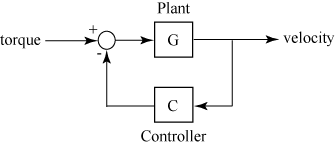
Sys=feedback (sys1,sys2) returns a model object sys for the negative feedback interconnection of model objects sys1,sys2.

From the figure, the closed-loop model sys has u as input vector and y as output vector. Both models, sys1 and sys2, must either be continuous or discrete with identical sample times.

Sys=feedback (sys1, sys2, feedin, feedout) computes a closed-loop model sys using the input and output connections specified using feedin and feedout. Use this syntax when you want to connect only a subset of the available I/Os of MIMO systems.

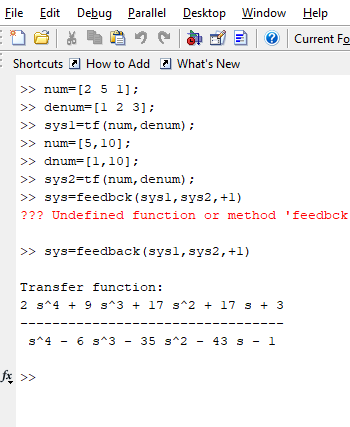
**Example:**

For this example, consider two transfer functions that describe a plant G and controller C respectively.



G(s)=2*s*2+5*s*+1/s2+2s+3 C(s)=5(s+2)/s+10

Create the plant and controller transfer functions.

****