**Example 1:**

Code:

clc

clear

pkg load symbolic

syms n z

h = 2^n;

H = ztrans(h,z)

H = simplify(H)

Result:

H = (sym)

/ 1 1

| ----- for --- < 1/2

| 2 |z|

| 1 - -

| z

|

< oo

| \_\_\_

| \ `

| \ n -n

| / 2 \*z otherwise

| /\_\_,

\n = 0

H = (sym)

/ z

| ----- for Or(z > 2, z < -2)

| z - 2

|

| oo

< \_\_\_

| \ `

| \ n -n

| / 2 \*z otherwise

| /\_\_,

\n = 0

>>

**Example 2:**

Code:

clc

clear

pkg load symbolic

syms n z X Y

Y1 = (z^-1)\*Y;

X1 = (z^-1)\*X;

G = Y-Y1-X-X1;

Y=solve(G,Y);

H=Y/X

Result:

H = (sym)

X\*z + X

---------

X\*(z - 1)

>>

**Example 3:**

**First way**

Code:

n=[2 1];

d=[1 3 2];

zer = roots(n);

pol = roots(d);

plot(real(pol),imag(pol),'\*',

real(zer),imag(zer),'o')

xlim([-3 1]);

legend('poles', 'zeros');

Result:

Chart, scatter chart

Description automatically generated

**Second way**

Code:

clear

clc

pkg load control

n=[2 1];

d=[1 3 2];

H=tf(n,d,0.1);

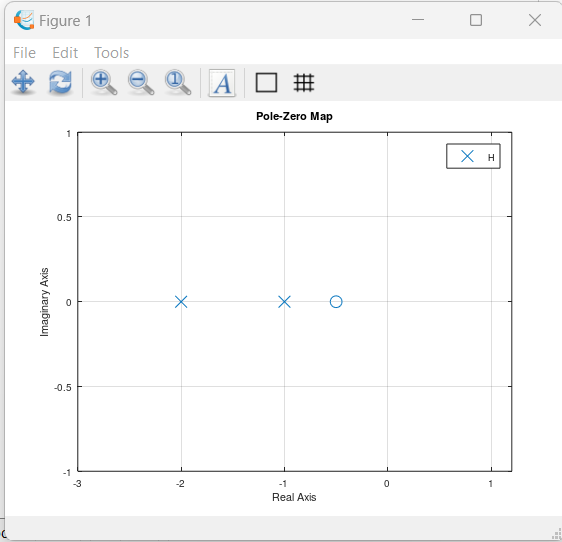
poles=pole(H)

zeros=zero(H)

pzmap(H)

xlim([-3 1.2])

Result:



**Example 4:**

**First way**

Code:

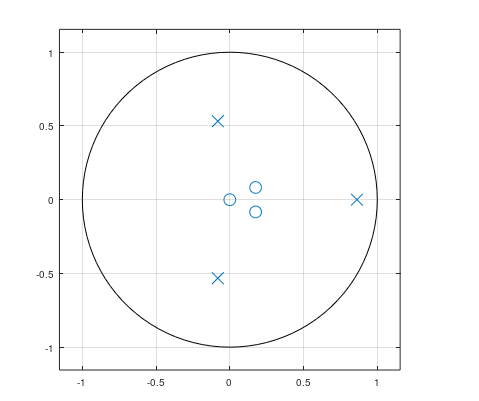
pkg load signal

n=[4 -1.4 .15];

d=[1 -.7 .15 -.25];

zplane(n,d)

Result:



**Second way**

Code:

pkg load signal

n=[4 -1.4 .15];

d=[1 -.7 .15 -.25];

H=tf(n,d,0.1);

pzmap(H)

Result:

Chart, scatter chart

Description automatically generated

**Example 5:**

Code:

pkg load control

n = [2 0 -1];

d = [1 1 -12];

H = tf(n,d,0.5);

zpk(H)

Result:

>> Lab5\_Ex5

Transfer function 'ans' from input 'u1' to output ...

2 z^2 - 1

y1: ------------

z^2 + z - 12

Sampling time: 0.5 s

Discrete-time model.

**Example 6:**

**a)**

Code:

clear

clc

pkg load control

num=5;

den=[1 2 5];

sys=tf(num,den);

step(sys);

Result:

Chart, line chart

Description automatically generated

**b)**

Code:

clear

clc

pkg load control

num=5;

den=[1 2 5];

sys=tf(num,den);

s=step(sys);

stairs(0:length(s)-1,s);

legend('Step response')

Result:

Graphical user interface, chart, line chart

Description automatically generated

**c)**

Code:

clear

clc

pkg load control

num=5;

den=[1 2 5];

n=0:80;

sys=tf(num,den);

s=step(sys,n);

stairs(n,s);

legend('Step response')

Result:

Graphical user interface

Description automatically generated

**Example 7:**

Code:

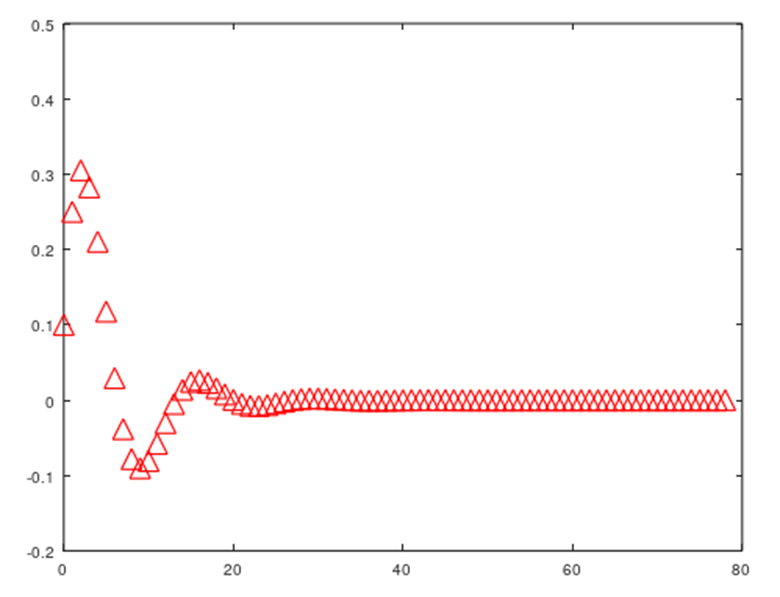
pkg load signal

num = [0.1 0.1];

den = [1 -1.5 0.7];

impz(num,den)

Result:



**Example 8:**

Code:

pkg load symbolic

n=0:3;

h=[3 5 2 1];

syms z w

Htf=sum(h.\*z.^-n);

H=subs(Htf,z,exp(j\*w));

H=simplify(H)

Hw=sum(h.\*exp(-j\*w\*n))

Result:

>> Lab5\_Ex8

H = (sym)

-I\*w -2\*I\*w -3\*I\*w

3 + 5\*e + 2\*e + e

Hw = (sym)

-I\*w -2\*I\*w -3\*I\*w

3 + 5\*e + 2\*e + e

**Example 9:**

Code:

pkg load symbolic

pkg load signal

syms n z w

h = (2/3)^n\*heaviside(n);

Hz = ztrans(h,z);

H = subs(Hz,z,exp(j\*w));

H = simplify(H)

h = (2/3)^n;

Hw = symsum(h\*exp(-j\*w\*n),n,0,inf)

Result:

>> Lab5\_Ex9

warning: passing floating-point values to sym is dangerous, see "help sym"

warning: called from

double\_to\_sym\_heuristic at line 50 column 7

sym at line 379 column 13

mpower at line 70 column 5

Lab5\_Ex9 at line 4 column 3

H = (sym)

oo

\_\_\_

\ `

\ -n

) n / I\*w\

/ 2/3 \*\e / \*Heaviside(n)

/\_\_,

n = 0

warning: passing floating-point values to sym is dangerous, see "help sym"

warning: called from

double\_to\_sym\_heuristic at line 50 column 7

sym at line 379 column 13

mpower at line 70 column 5

Lab5\_Ex9 at line 8 column 3

Hw = (sym)

oo

\_\_\_

\ `

\ n -I\*n\*w

/ 2/3 \*e

/\_\_,

n = 0

**Example 10:**

Code:

pkg load signal

pkg load symbolic

syms z n w Yz X

Y1z=z\*(-1)\*Yz;

G=Yz-0.9\*Y1z-X;

Yz=solve(G,Yz);

Hz=Yz/X

Hw=subs(Hz,z,exp(j\*w))

Result:

>> Lab4\_Ex10

warning: passing floating-point values to sym is dangerous, see "help sym"

warning: called from

double\_to\_sym\_heuristic at line 50 column 7

sym at line 379 column 13

mtimes at line 54 column 5

Lab4\_Ex10 at line 6 column 2

Hz = (sym)

10

--------

9\*z + 10

Hw = (sym)

10

-----------

I\*w

9\*e + 10

**Problem 2:**

Code:

clear

clc

% Define the transfer function H(z)

num = [8 10 -6];

den = [1 2 -1 -2];

H = tf(num, den, 1);

% Check if the system is stable

if isstable(H)

printf("The system is stable.\n");

else

printf("The system is unstable.\n");

end

% Express the transfer function in zero-pole-gain form

[z, p, k] = zpkdata(H);

printf("Zeros: %s\n", mat2str(z{1}));

printf("Poles: %s\n", mat2str(p{1}));

printf("Gain: %f\n", k);

% Express the transfer function in partial fraction form

[r, p, k] = residue(num, den);

for i = 1:length(r)

printf("Residue %d: %f\n", i, r(i));

printf("Pole %d: %f\n", i, p(i));

end

printf("Constant term: %f\n", k);