SYSTEM ANALYSIS – I

Control Engineering Laboratory
Experiment Number 2

Name: Pragyaditya Das.
Roll No.: 110113062

Date:

Attestation:

Problem Statement:

The problem statement is to be familiar with various system representations and their analysis using various analytical tools available in MATLAB. A brief outline of the experiments done are as follows:

- Transfer function and transfer function matrix
- From state space form to Transfer function and transfer function matrix
- From Transfer function and transfer function matrix form to state space form

Pole zero form

- Investigation of Poles& zeros of transfer function matrix
- Stability and stabilization
- Study of bode plot and polar plot
- Root locus plot
- Study of sisotool in matlab
- Plot system output for various inputs

Theory:

Representation:

A Transfer Function is defined as the ratio of the output of a system to the input of a system, in the Laplace domain considering its initial conditions and equilibrium point to be zero. This assumption is relaxed for systems observing transience. If we have an input function of *X(s)*, and an output function *Y(s)*, we define the transfer function *H(s)* to be:

$$H(s) = \frac{Laplace\ of\ the\ Output}{Laplace\ of\ the\ input}$$

The transfer function is a convenient representation of a linear time invariant dynamical system. Mathematically the transfer function is a function of complex variables. For finite dimensional systems the transfer function is simply a rational function of a complex variable. The transfer function can be obtained by inspection or by simple algebraic manipulations of the differential equations that describe the systems. Transfer functions can describe systems of very high order, even infinite dimensional systems governed by partial differential equations. The transfer function of a system can be determined from experiments on a system. However, transfer function models are hugely used in case of Single-Input-Single-Output (SISO) systems.

When we move on to the Multiple-Input-Multiple-Output (MIMO) systems, we start using State Space Models. State Space models form the very foundation of Modern Control Theory, which is the field of Control System which developed after 1950.

In control engineering, a **state-space representation** is a mathematical model of a physical system as a set of input, output and state variables related by first-order differential equations. "State space" refers to the space whose axes are the state variables. The state of the system can be represented as a vector within that space.

To abstract from the number of inputs, outputs and states, these variables are expressed as vectors. Additionally, if the dynamical system is linear, time-invariant, and finite-dimensional, then the differential and algebraic equations may be written in matrix form. The state-space method is characterized by significant algebraization of general system theory, which makes possible to use Kronecker vector-matrix structures.

The standard equation for state space model is given as,

$$\frac{dx}{dt} = Ax(t) + Bu(t) \qquad (1)$$
$$y(t) = Cx(t) + Du(t) \qquad (2)$$

The internal state variables are the smallest possible subset of system variables that can represent the entire state of the system at any given time. The minimum number of state variables required to represent a given system is usually equal to the order of the system's defining differential equation.

There are two very important concepts that we must keep in mind to define any MIMO system effectively. They are:

1. Controllability.
2. Observability.

State controllability condition implies that it is possible – by admissible inputs – to steer the states from any initial value to any final value within some finite time window. A continuous time-invariant linear state-space model is **controllable** if and only if:

$$rank[A\ AB\ A^2B \dots A^{N-1}] = N$$

Observability is a measure for how well internal states of a system can be inferred by knowledge of its external outputs. The observability and controllability of a system are mathematical duals (i.e., as controllability provides that an input is available that brings any initial state to any desired final state, observability provides that knowing an output trajectory provides enough information to predict the initial state of the system). A continuous time-invariant linear state-space model is **observable** if and only if:

$$rank\begin{bmatrix} C \\ \vdots \\ CA^{N-1} \end{bmatrix} = N$$

Stability of a system:

Bode Plot:

Bode plot is a way of getting the frequency response of a system. It is usually a combination of a Bode Magnitude Plot and a Bode Phase Plot. Both plots are plotted against horizontal axis proportional to the logarithm of frequency. Given that amplitude itself is a logarithmic scale, the Bode Magnitude plot is a log-log plot, whereas the Bode Phase plot is a lin-log plot.

Nyquist Plot:

A Nyquist plot is a parametric plot of a frequency response used in automatic control and signal processing. The most common use of Nyquist plots is for assessing the stability of a system with feedback. In Cartesian coordinates, the real part of the transfer function is plotted on the X axis. The imaginary part is plotted on the Y axis. The frequency is swept as a parameter, resulting in a plot per frequency. Alternatively, in polar coordinates, the gain of the transfer function is plotted as the radial coordinate, while the phase of the transfer function is plotted as the angular coordinate.

Routh-Hurwitz Criteria:

The stability of a process control system is extremely important to the overall control process. System stability serves as a key safety issue in most engineering processes. If a control system becomes unstable, it can lead to unsafe conditions. For example, instability in reaction processes or reactors can lead to runaway reactions, resulting in negative economic and environmental consequences.

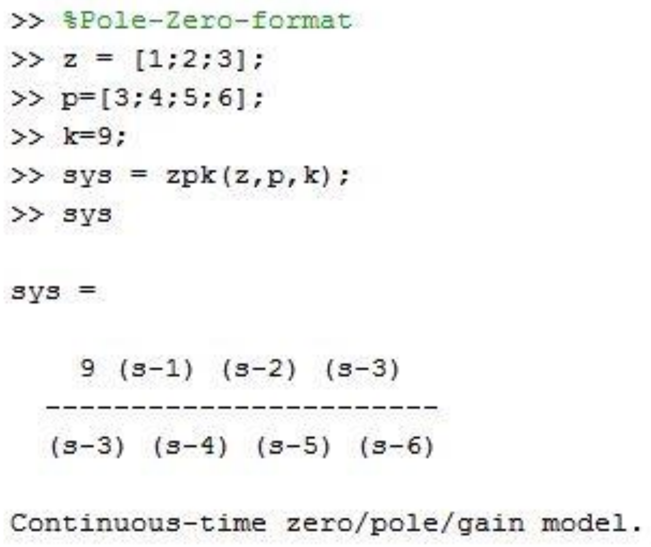
The absolute stability of a control process can be defined by its response to an external disturbance to the system. The system may be considered stable if it exists at a consistent state or setpoint and returns to this state immediately after a system disturbance. In order to determine the stability of a system, one often must determine the eigenvalues of the matrix representing the system’s governing set of differential equations. Unfortunately, sometimes the characteristic equation of the matrix (the polynomial representing its eigenvalues) can be difficult to solve; it may be too large or contain unknown variables. In this situation, a method developed by British mathematician Edward Routh can yield the desired stability information without explicitly solving the equation.

Recall that in order to determine system stability one need only know the signs of the real components of the eigenvalues. Because of this, a method that can reveal the signs without actual computation of the eigenvalues will often be adequate to determine system stability.

Procedure:

1. Before we start, we must read about the various MATLAB commands we use in this experiment.
   Some of the common tools used are,
   ss2tf, tf2ss, c2d, d2c, zpk, sisotool, rltool, margin and isstable.
2. In the beginning of any system analysis, we must correctly enter details given.
3. For checking the system stability, we use some graphical and computational tools, the common commands used are,
   bode (transfer function), nyquist (transfer function), margin (transfer function) and isstable (transfer function).
4. We cross verify each result with other methods available in MATLAB.
5. We analyse each result and draw inference out of it.

Simulation and Analysis:
a.



```
>> %Pole-Zero-format
>> z = [1;2;3];
>> p=[3;4;5;6];
>> k=9;
>> sys = zpk(z,p,k);
>> sys

sys =

     9 (s-1) (s-2) (s-3)
  -----------------------
  (s-3) (s-4) (s-5) (s-6)

Continuous-time zero/pole/gain model.
```

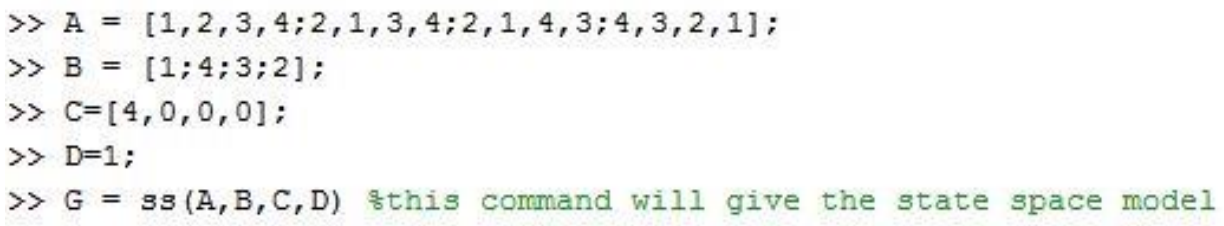
In this experiment, we provide the information about the zeroes, poles and gain of the system to MATLAB and it generates the transfer function for us. Here is the part of the official MATLAB documentation on the zpk command used,

The official MATLAB documentation is as follows:

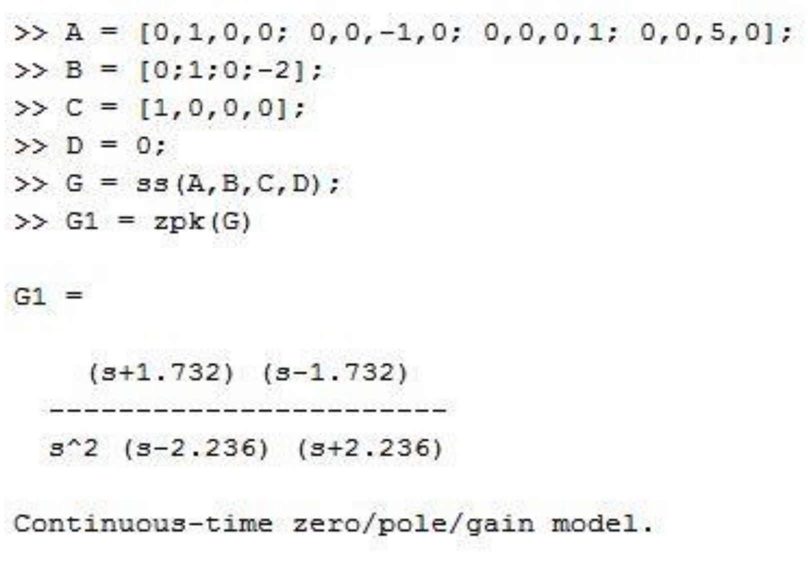
```
     Use zpk to create zero-pole-gain models (zpk model objects),
or to convert dynamic systems to zero-pole-gain form. Creation of
Zero-Pole-Gain Models. System =zpk (z,p,k) creates a continuous-time
zero-pole-gain model with zeros z, poles p, and gain(s) k. The
output sys is a zpk model object storing the model data.
In the SISO case, z and p are the vectors of real- or complex-valued
zeros and poles, and k is the real- or complex-valued scalar gain:
```

$$Transfer\ fuction = \frac{k(s-z_1)(s-z_2)\ldots(s-z_n)}{(s-p_1)(s-p_2)\ldots(s-p_n)}$$

b. An ss command is used to form the state space from the given A, B, C and D matrices. We then use the command to analyse the zeroes, poles and the gain of the system.

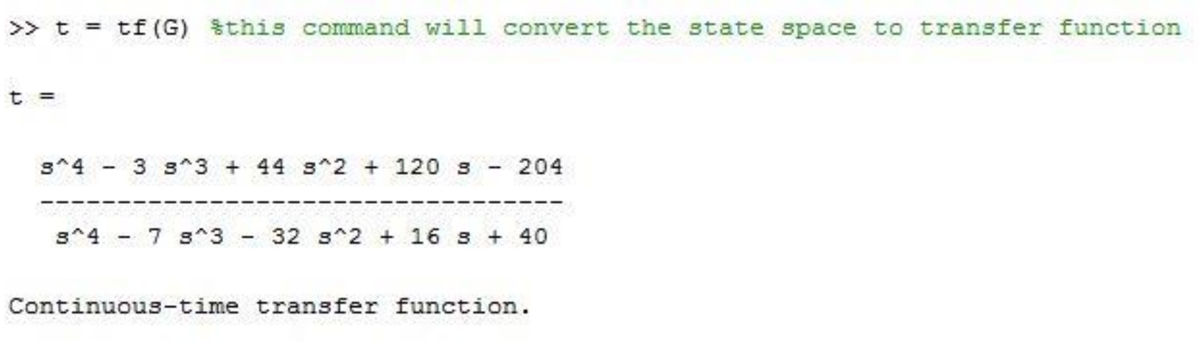


```
>> A = [1,2,3,4;2,1,3,4;2,1,4,3;4,3,2,1];
>> B = [1;4;3;2];
>> C=[4,0,0,0];
>> D=1;
>> G = ss(A,B,C,D) %this command will give the state space model
```



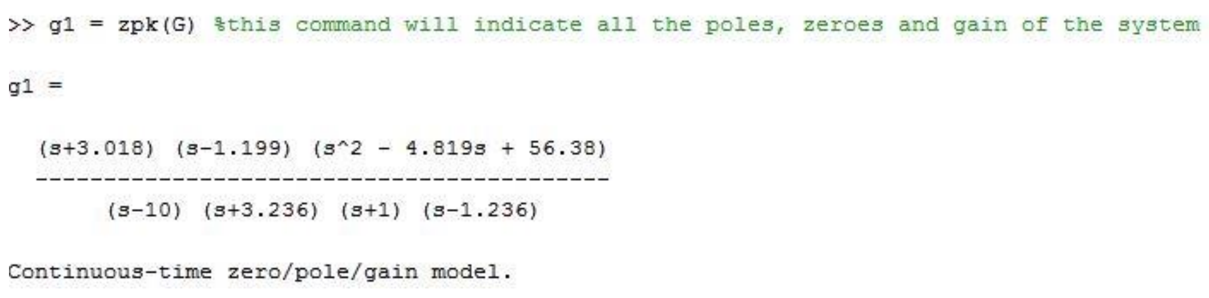
```
>> A = [0,1,0,0; 0,0,-1,0; 0,0,0,1; 0,0,5,0];
>> B = [0;1;0;-2];
>> C = [1,0,0,0];
>> D = 0;
>> G = ss(A,B,C,D);
>> G1 = zpk(G)

G1 =

     (s+1.732) (s-1.732)
  ----------------------
  s^2 (s-2.236) (s+2.236)

Continuous-time zero/pole/gain model.
```

c. We now just run the tf command to obtain the transfer function model from the state space model.



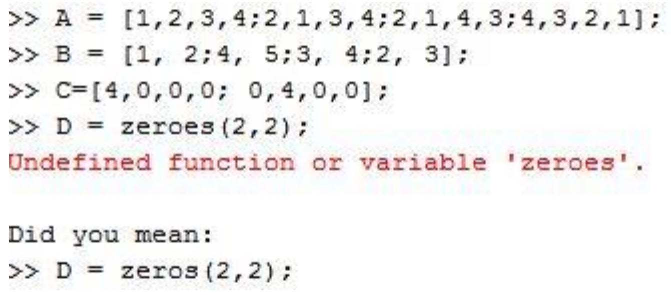
```
>> t = tf(G) %this command will convert the state space to transfer function

t =

  s^4 - 3 s^3 + 44 s^2 + 120 s - 204
  ----------------------------------
   s^4 - 7 s^3 - 32 s^2 + 16 s + 40

Continuous-time transfer function.
```

d. To indicate the zeroes, poles and the gain of the transfer function, we run the zpk command.



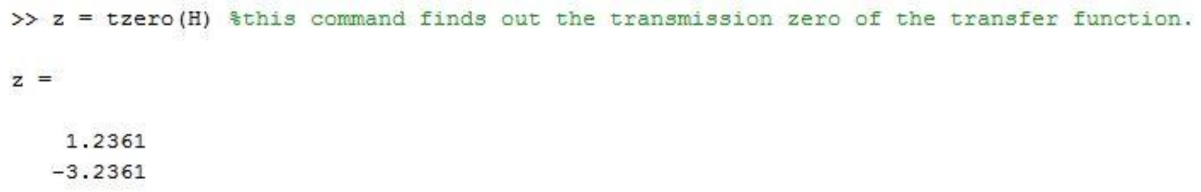
```
>> g1 = zpk(G) %this command will indicate all the poles, zeroes and gain of the system

g1 =

  (s+3.018) (s-1.199) (s^2 - 4.819s + 56.38)
  ------------------------------------------
       (s-10) (s+3.236) (s+1) (s-1.236)

Continuous-time zero/pole/gain model.
```

e. Now, we define another State Space model, and this stage, we try to find the Transmission Zeroes of the given model.



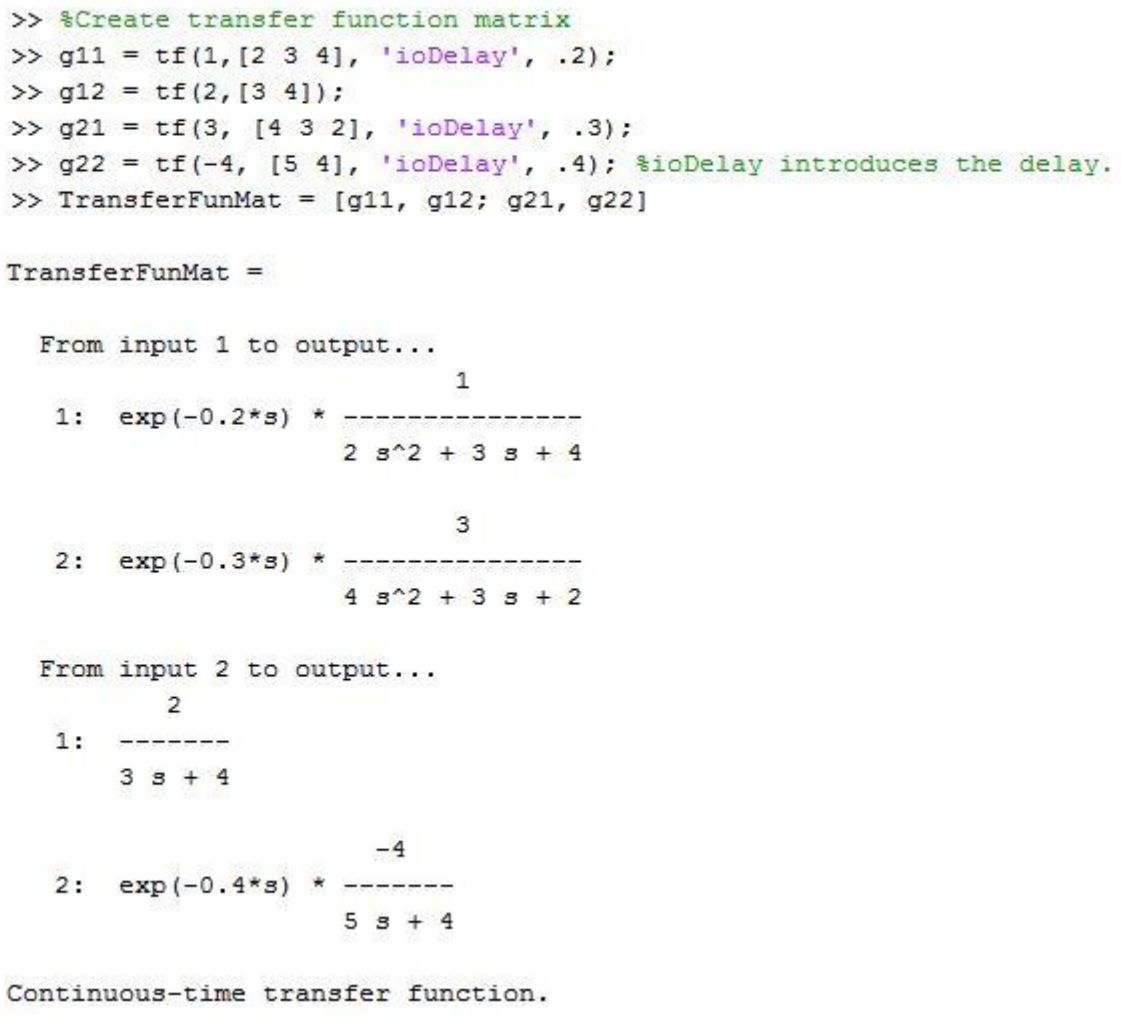
```
>> A = [1,2,3,4;2,1,3,4;2,1,4,3;4,3,2,1];
>> B = [1, 2;4, 5;3, 4;2, 3];
>> C=[4,0,0,0; 0,4,0,0];
>> D = zeroes(2,2);
Undefined function or variable 'zeroes'.

Did you mean:
>> D = zeros(2,2);
```

now, we use H = ss(A,B,C,D). Which will give us the state space.
Now, we use the tzero command to find the transmission zeroes.



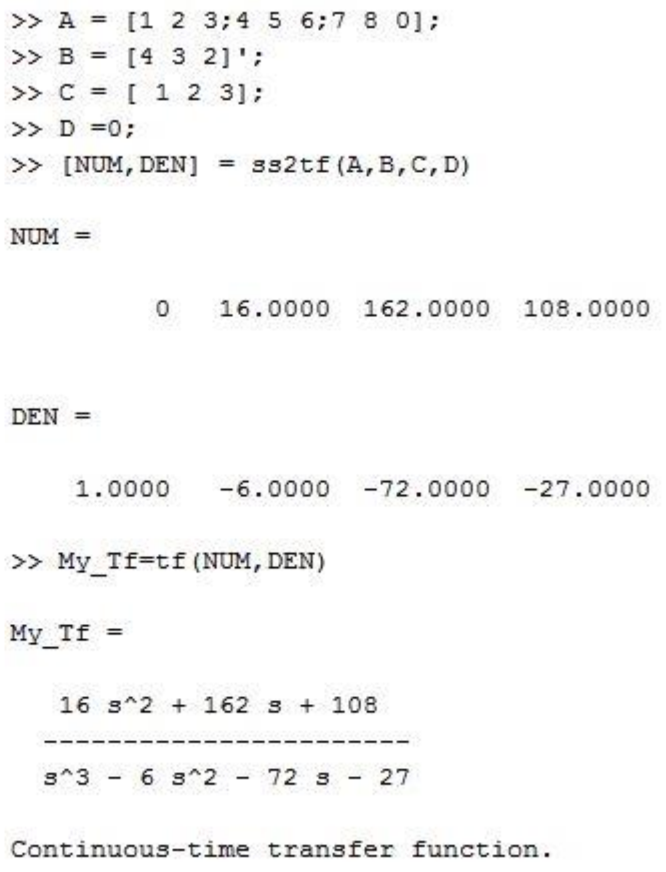
```
>> z = tzero(H) %this command finds out the transmission zero of the transfer function.

z =

    1.2361
   -3.2361
```

f. We will now create a transfer function model with some input delay.

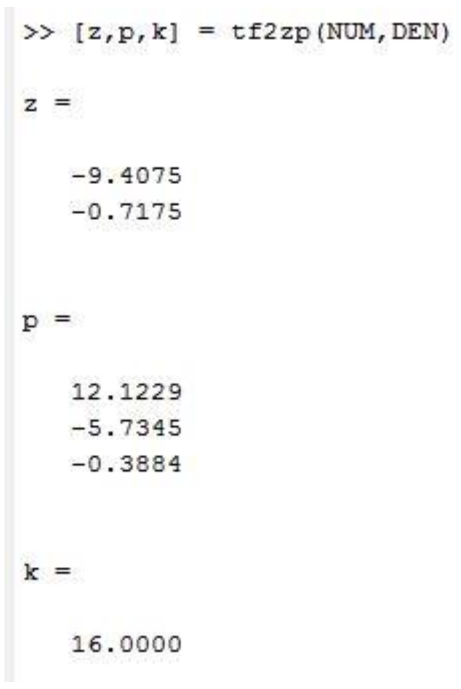


```
>> %Create transfer function matrix
>> g11 = tf(1,[2 3 4], 'ioDelay', .2);
>> g12 = tf(2,[3 4]);
>> g21 = tf(3, [4 3 2], 'ioDelay', .3);
>> g22 = tf(-4, [5 4], 'ioDelay', .4); %ioDelay introduces the delay.
>> TransferFunMat = [g11, g12; g21, g22]

TransferFunMat =

  From input 1 to output...
                              1
   1:  exp(-0.2*s) * ---------------
                     2 s^2 + 3 s + 4

                              3
   2:  exp(-0.3*s) * ---------------
                     4 s^2 + 3 s + 2

  From input 2 to output...
          2
   1:  -------
       3 s + 4

                          -4
   2:  exp(-0.4*s) * -------
                     5 s + 4

Continuous-time transfer function.
```

g. In this experiment, we do similar, but here we separately extract the numerator and denominator of the Transfer function and do the usual exercises.



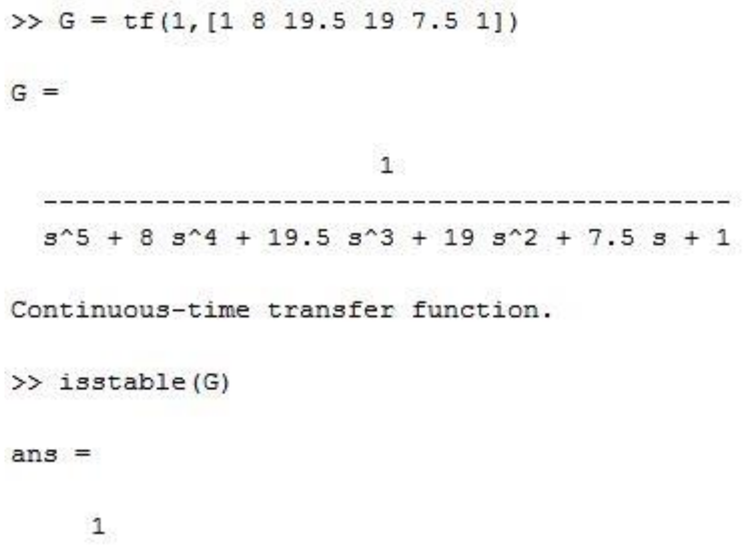
```
>> A = [1 2 3;4 5 6;7 8 0];
>> B = [4 3 2]';
>> C = [ 1 2 3];
>> D =0;
>> [NUM,DEN] = ss2tf(A,B,C,D)

NUM =

         0   16.0000  162.0000  108.0000


DEN =

    1.0000   -6.0000  -72.0000  -27.0000

>> My_Tf=tf(NUM,DEN)

My_Tf =

   16 s^2 + 162 s + 108
  -----------------------
  s^3 - 6 s^2 - 72 s - 27

Continuous-time transfer function.
```



```
>> [z,p,k] = tf2zp(NUM,DEN)

z =

   -9.4075
   -0.7175


p =

   12.1229
   -5.7345
   -0.3884


k =

   16.0000
```

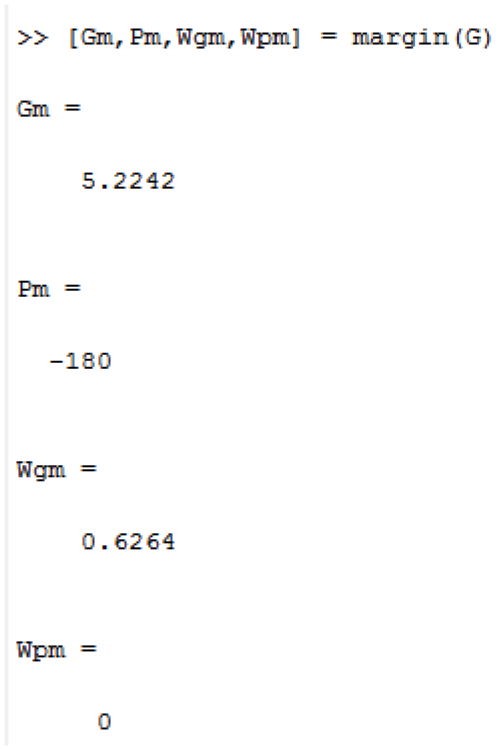
h. We are given with a complex structure. We will analyse the transfer function using two methods, one is graphical and the other is computational.

We will first use the isstable function to determine if the system is stable. Isstable gives 1 as a output if the system is stable, and 0 if not.



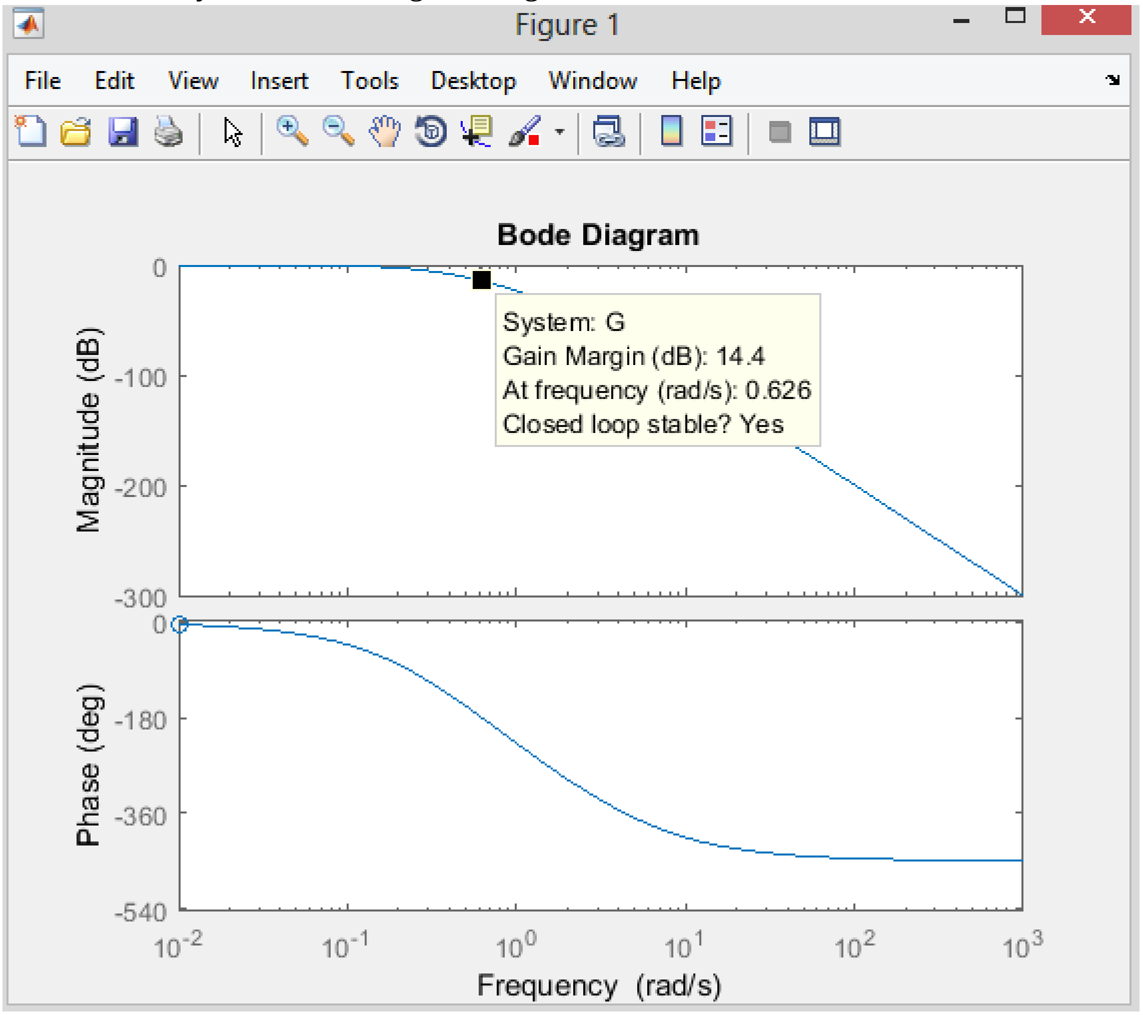
```
>> G = tf(1,[1 8 19.5 19 7.5 1])

G =

                          1
  -------------------------------------------
  s^5 + 8 s^4 + 19.5 s^3 + 19 s^2 + 7.5 s + 1

Continuous-time transfer function.

>> isstable(G)

ans =

     1
```

Now, we use the margin command to find the stability margins for analysis. Margin uses Routh Hurwitz Criteria.



```
>> [Gm,Pm,Wgm,Wpm] = margin(G)

Gm =

    5.2242


Pm =

  -180


Wgm =

    0.6264


Wpm =

     0
```

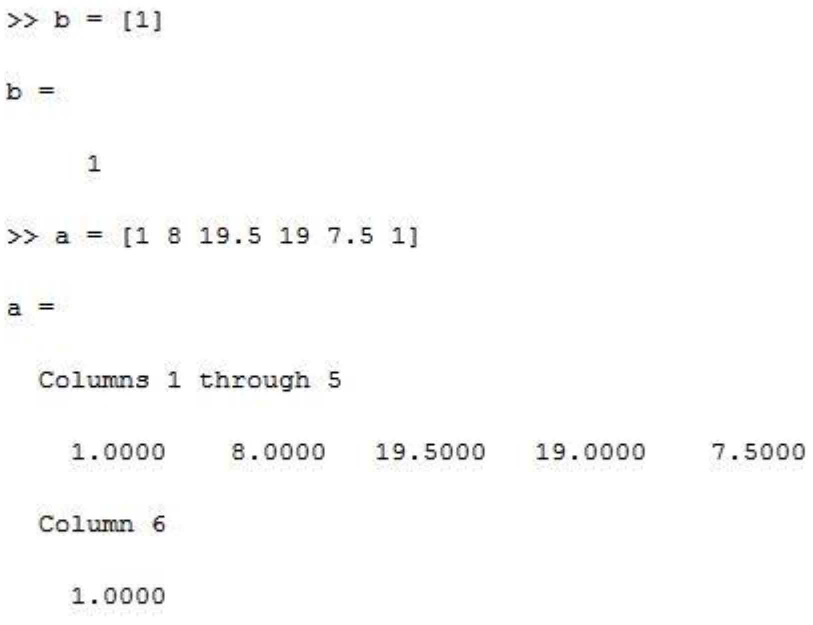
We will now try to find the margins using Bode method.



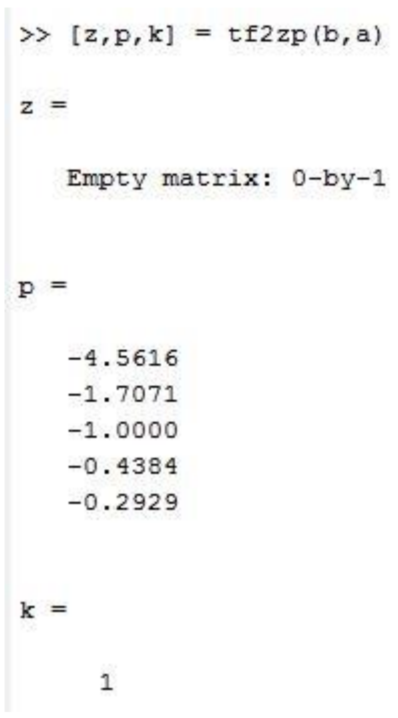
System: G
Gain Margin (dB): 14.4
At frequency (rad/s): 0.626
Closed loop stable? Yes

We see the difference in the gain margin. This is because of the limitation of bode plot to correctly ascertain the stability of complex systems.

Now, we find the poles, zeroes and gain of the system.



```
>> b = [1]

b =

     1

>> a = [1 8 19.5 19 7.5 1]

a =

  Columns 1 through 5

    1.0000    8.0000   19.5000   19.0000    7.5000

  Column 6

    1.0000
```

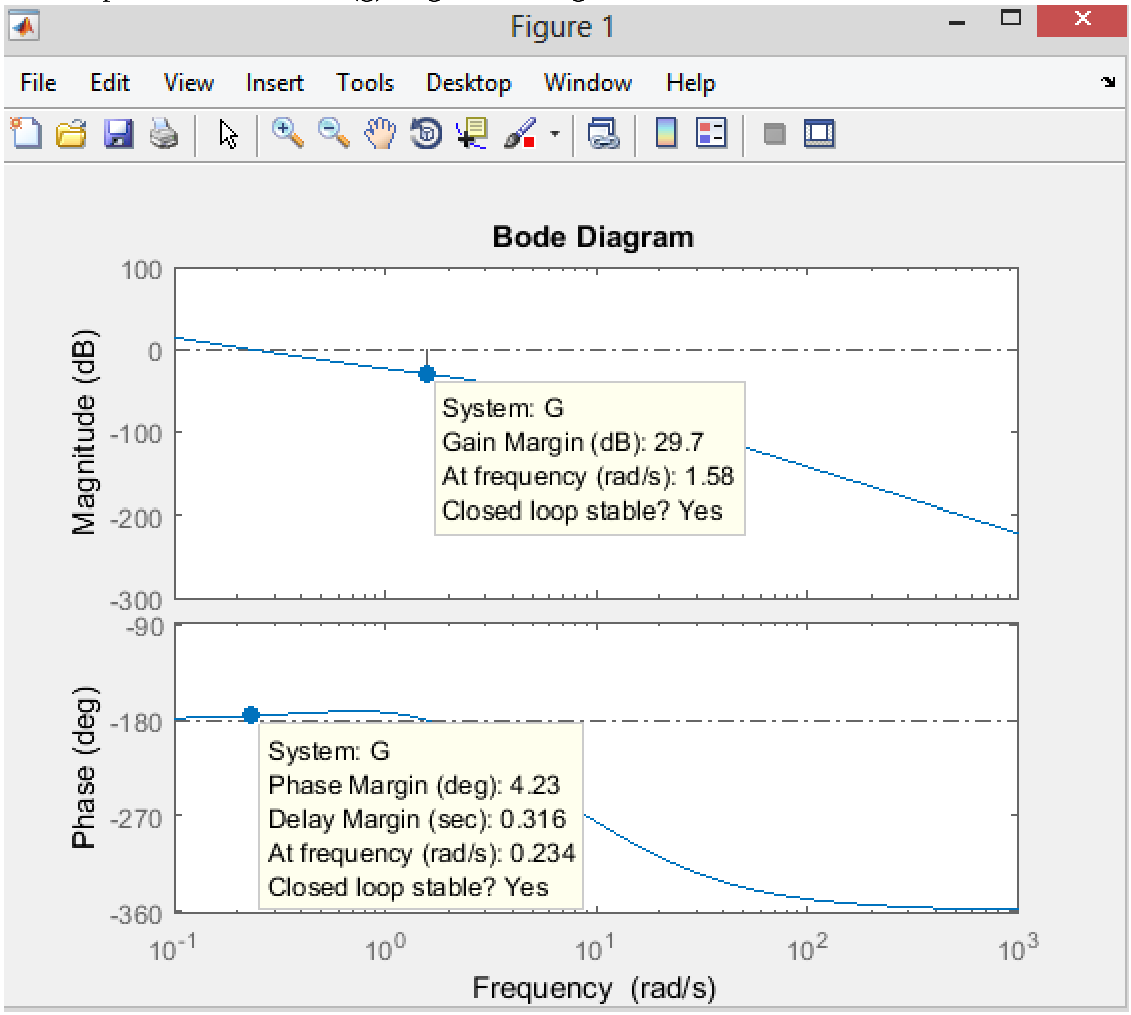


```
>> [z,p,k] = tf2zp(b,a)

z =

   Empty matrix: 0-by-1


p =

   -4.5616
   -1.7071
   -1.0000
   -0.4384
   -0.2929


k =

     1
```

i. Now, we find the gain and phase margins of a complex transfer function using bode plot. The given transfer function is factored, for simplicity of use in MATLAB, we use it in non factored form.

```
G =

              8 s + 8
  --------------------------------
  s^5 + 21 s^4 + 100 s^3 + 150 s^2

Continuous-time transfer function.
```

next step is to run the bode(g) to get the margins.



System: G
Gain Margin (dB): 29.7
At frequency (rad/s): 1.58
Closed loop stable? Yes

System: G
Phase Margin (deg): 4.23
Delay Margin (sec): 0.316
At frequency (rad/s): 0.234
Closed loop stable? Yes

j. The next step is to find the maximum extent upto which the gain of a system can vary using the SISOTOOL.

```
H =

                     1
  -----------------------------------
  s^4 + 12 s^3 + 69 s^2 + 198 s + 200

Continuous-time transfer function.
```

We assume a K in the denominator. After running the SISOTOOL command, we set the damping ratio to 0.707 and we see that the closed loop gain is set to around 64.4.

Conclusion:

1. We have seen that the margin analysis using Bode Plot fails when we use fairly complex systems.

2. Routh Hurwitz criteria is a very reliable method. This method also gives us the idea of the number of poles in the RHP in the system.

Result:

Hence, we studied various ways of representing and analysing a Control System given any type of model representation.