NAME	Urwa Maryam, Umar Hayyat. Mateen Abbas
REG NO.	2019-EE-352, 2019-EE-360, 2019-EE-384
MARKS	5 15-11-21

Lab # 6

Block Diagram Reduction

Objectives:

- To study about block diagram
- Usage of block diagram in control system
- To study Block diagram reduction
- Use MATLAB for block diagram reduction

Introduction:

- **Block Diagram:**

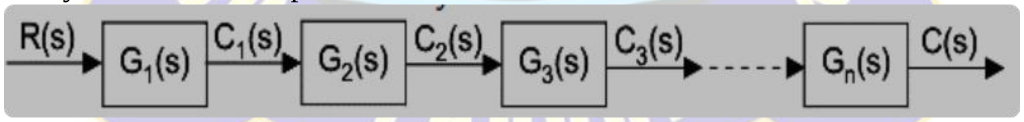
A *block diagram* is a visual representation of a system that uses simple, labeled *blocks* that represent single or multiple items, entities or concepts, connected by lines to show relationships between them. It is also a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. They are heavily used in engineering in hardware design, electronic design, software design, and process flow diagrams.

- **Block Diagram Reduction:**

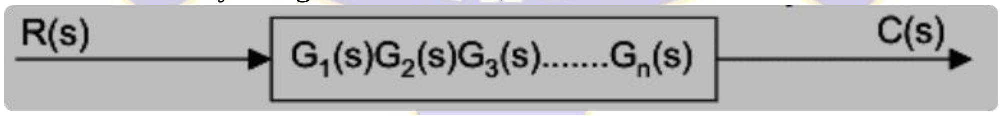
The technique of combining of these blocks is referred to as **block diagram reduction technique**. For successful implementation of this technique, some rules for block diagram reduction to be followed.

  - **Rule-1-Check for the blocks connected in series and simplify:**

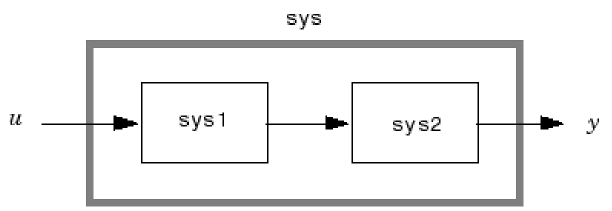
When several systems or control blocks are connected in cascaded manner, the transfer function of the entire system will be the product of transfer function of all individual blocks.



Now, we can reduce this by using the above rule,



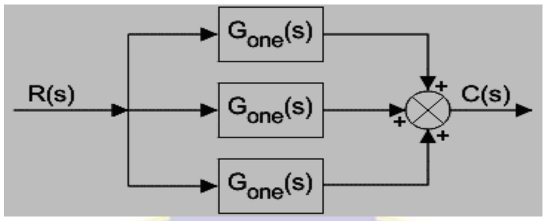
MATLAB Command:`sys=series(sys1,sys2)` forms the basic series connection for:



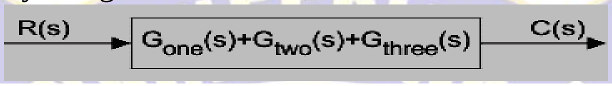
This MATLAB command is equivalent to the direct multiplication: sys = sys2 * sys1

- **Rule-2-Check for the blocks connected in parallel and simplify:**

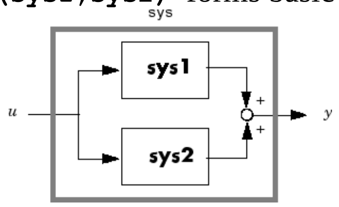
When same input signal is applied different blocks and the output from each of them are added in a summing point for taking final output of the system then over all transfer function of the system will be the algebraic sum of transfer function of all individual blocks.



Now, we can reduce this by using the above rule,



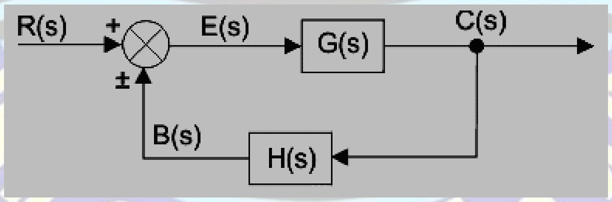
`MATLAB Command:` **`sys=parallel(sys1,sys2)`** forms basic parallel connection for:



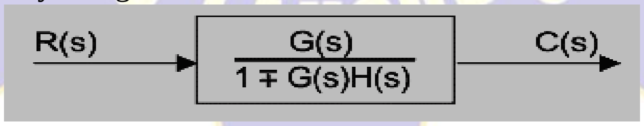
This MATLAB command equals the direct addition: sys = sys1 + sys2

- **Rule-3-Check for the blocks connected in feedback loop and simplify:**

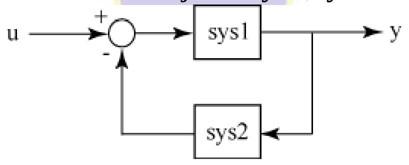
In a **closed loop control system**, a fraction of output is fed-back and added to input of the system. If ***H (s)*** is the transfer function of feedback path, then the transfer function of feedback signal will be ***B(s) = C(s)H(s).*** At summing point, the input signal ***R(s)*** will be added to ***B(s)*** and produces actual input signal or error signal of the system.



Now, we can reduce this by using the above rule,



Matlab Commands: `sys=feedback(sys1,sys2)` returns a model object sys for the negative feedback interconnection of model objects sys1,sys2.



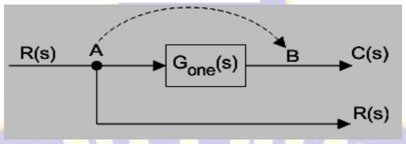
From the figure, the closed-loop model `sys` has *u* as input vector and *y* as output vector. Both models, `sys1` and `sys2`, must either be continuous or discrete with identical sample times.

`sys=feedback(sys1,sys2,sign)`

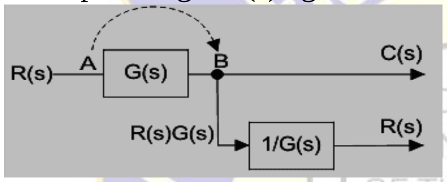
By default, `feedback` assumes negative feedback and is equivalent to `feedback(sys1,sys2,-1)`. To compute the closed-loop system with positive feedback, use `sign = +1`.

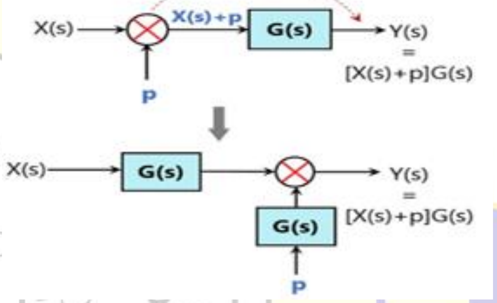
- **Rule-4-If there is difficulty with node while simplifying, shift it towards right:**

If same signal is applied to more than one system, then the signal is represented in the system by a point called take off point. Principle of **shifting of node** is that, it may be shifted either side of a block but final output of the branches connected to the take off point must be un-changed. The take off point can be shifted either sides of the block.



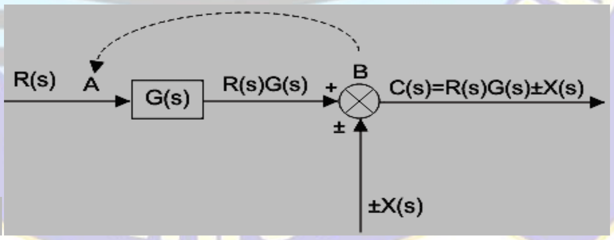
In the figure above the take off point is shifted from position A to B. The signal R(s*)* at node A will become G(s)R(s) at point B. Hence another block of inverse of transfer function G(s) is to be put on that path to get R(s) again.





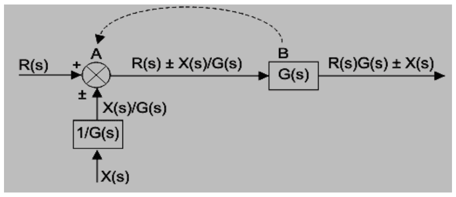
- **Rule-5-If there is difficulty with summing point while simplifying, shift it towards left:**

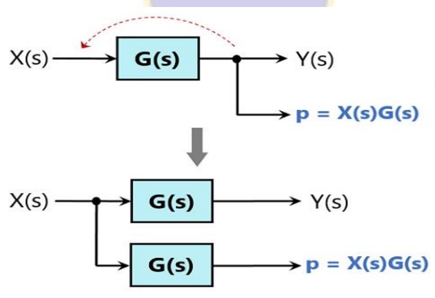
Let us examine the shifting of summing point from a position before a block to a position after a block. There are two input signals R(s) and ± X(s) entering in a summing point at position A. The output of the summing point is R(s) ± X(s). The resultant signal is the input of a control system block of transfer function G(s) and the final output of the system is:

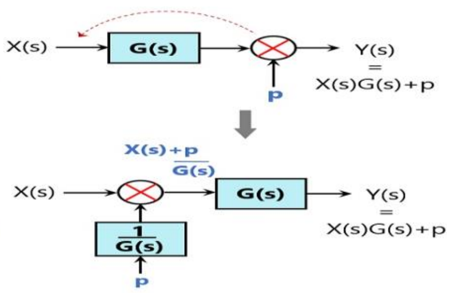


$C(s)=R(s)G(s)\pm X(s)$

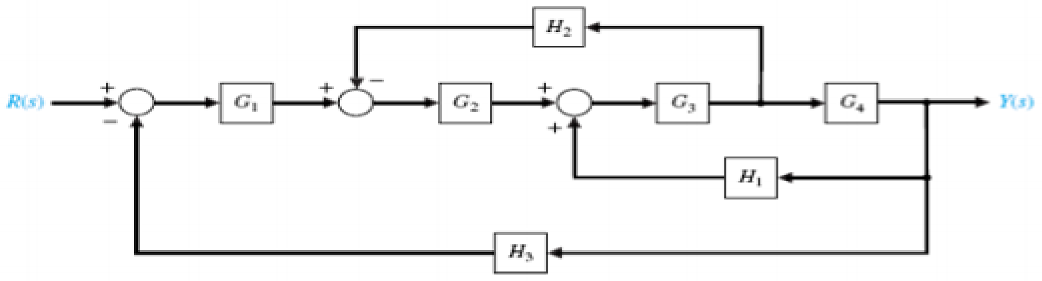
Finally, it can be drawn as below.







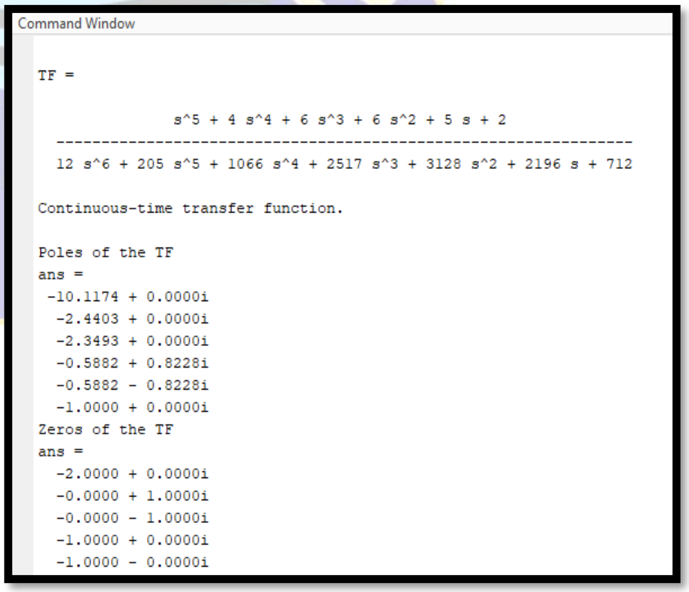
Exercise 1: For the following multi-loop feedback system, get closed loop transfer function and the corresponding pole-zero map of the system.



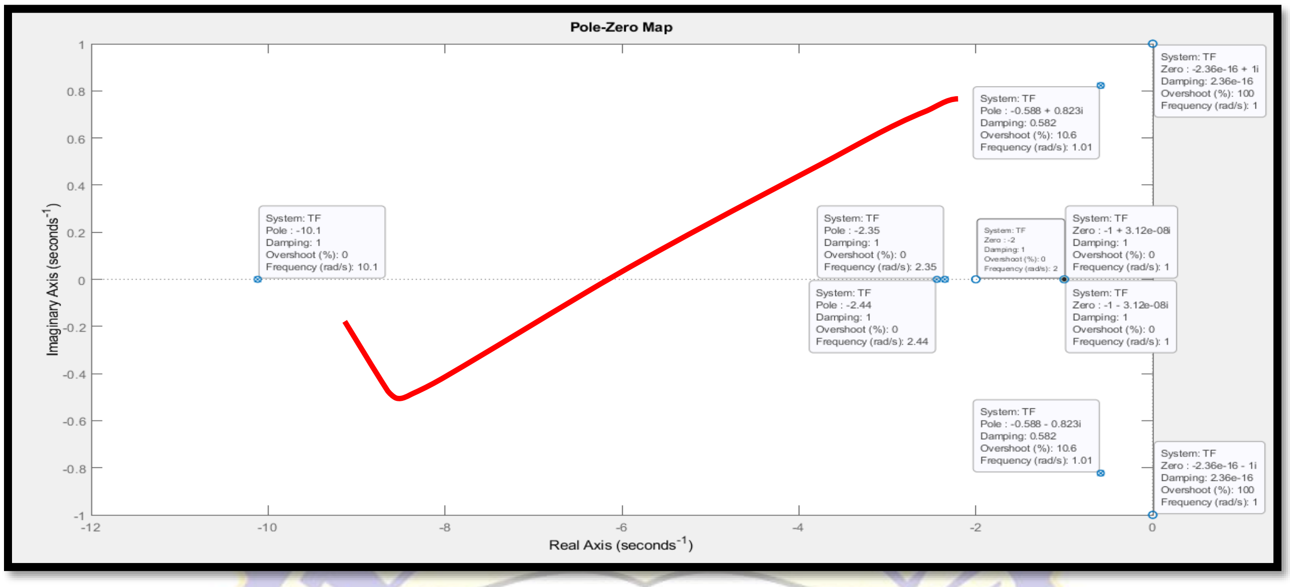
Given $G_1 = \frac{1}{(s+10)}$; $G_2 = \frac{1}{(s+1)}$; $G_3 = \frac{s^2+1}{(s^2+4s+4)}$; $G_4 = \frac{s+1}{(s+6)}$; $H_1 = \frac{s+1}{(s+2)}$; $H_2 = 2$; $H_3 = 1$

Code:

```
clc;clear all;close all;
%given transfer funtion
G1=tf([1],[1 10]);
G2=tf([1],[1 1]);
G3=tf([1 0 1],[1 4 4]);
G4=tf([1 1],[1 6]);
H1=tf([1 1],[1 2]);
H2=2;
H3=1;
%Reduction of block diagram
TF0=series(H2,1/G4);         %shifting pick-off point to the right of G4
TF1=series(G3,G4);           %cascading G3 G4
TF2=feedback(TF1,H1,+1);     %+1 for +ve feedback.Solving feedback H1 to the input
of G3*G4
TF3=series(G2,TF2);          %cascading TF2 function with G2
TF4=feedback(TF3,TF0);       %no need to write -1 for -ve feedback as builtin is -ve
TF5=series(G1,TF4);          %cascading
TF4 with G1
TF=feedback(TF5,H3)        %-ve
feedback %Finally feedback H3 with
TF5
pzmap(TF);
format compact;
disp('Poles of the TF');
pole(TF)    %pole values
disp('Zeros of the TF');
zero(TF)    %zero values
```

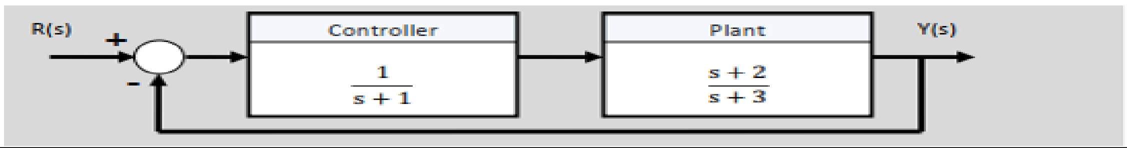


```
TF =

              s^5 + 4 s^4 + 6 s^3 + 6 s^2 + 5 s + 2
  -----------------------------------------------------------------
  12 s^6 + 205 s^5 + 1066 s^4 + 2517 s^3 + 3128 s^2 + 2196 s + 712

Continuous-time transfer function.

Poles of the TF
ans =
 -10.1174 + 0.0000i
  -2.4403 + 0.0000i
  -2.3493 + 0.0000i
  -0.5882 + 0.8228i
  -0.5882 - 0.8228i
  -1.0000 + 0.0000i
Zeros of the TF
ans =
  -2.0000 + 0.0000i
  -0.0000 + 1.0000i
  -0.0000 - 1.0000i
  -1.0000 + 0.0000i
  -1.0000 - 0.0000i
```



Exercise 2: Consider the feedback system depicted in the figure below

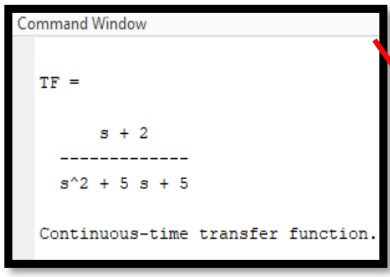
a. Compute the closed-loop transfer function using the 'series' and 'feedback' functions
b. Obtain the closed-loop system unit step response with the 'step' function and verify that final value of the output is 2/5.



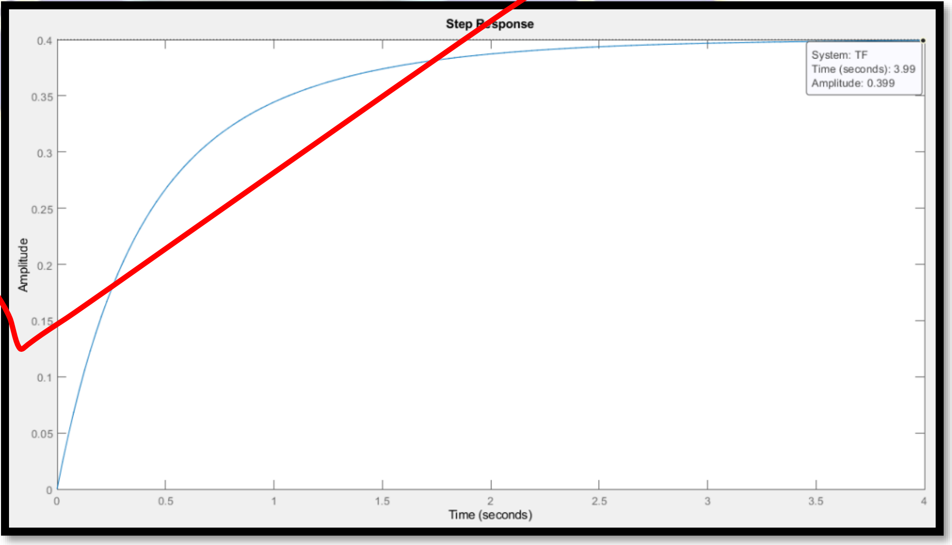
Code:

```
controller = tf([1],[1 1]);  %controller transfer function
H = 1;   %feedback transfer function
plant = tf([1 2],[1 3]);     %plant transfer function
Tf1 = series(controller,plant);  %controller and plant is in cascade (series)
TF = feedback(Tf1,H,-1)          %feedback H = 1 to the input side of controller
step(TF)
```

(b): From the response plot shown below, the final value or steady state value of the system is 2/5 = 0.4


Command Window

```
TF =

      s + 2
  -------------
  s^2 + 5 s + 5

Continuous-time transfer function.
```



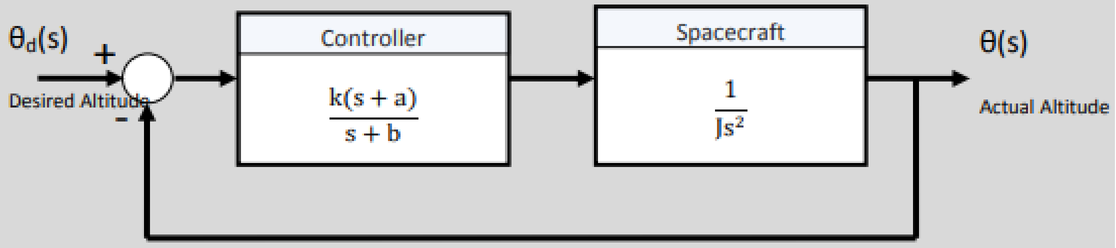
Exercise 3: A satellite single-axis altitude control system can be represented by the block diagram in the figure given. The variables 'k', 'a' and 'b' are controller parameters, and 'J' is the spacecraft moment of inertia. Suppose the nominal moment of inertia is 'J' = 10.8E8, and the controller parameters are k=10.8E8, a=1, and b=8.

a. Develop an m-file script to compute the closed-loop transfer function

$$T(s) = \theta(s)/\theta_d(s).$$

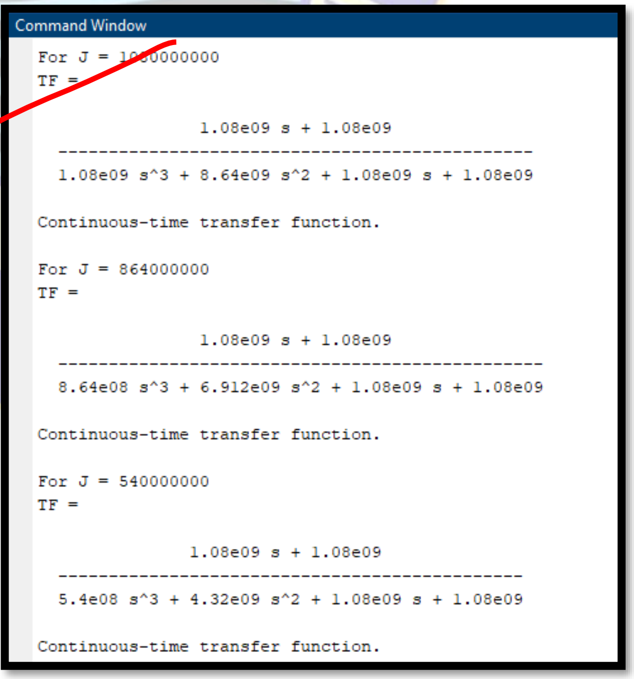
b. Compute and plot the step response to a 10° step input.

c. The exact moment of inertia is generally unknown and may change slowly with time. Compare the step response performance of the spacecraft when J is reduced by 20% and 50%. Discuss your results.

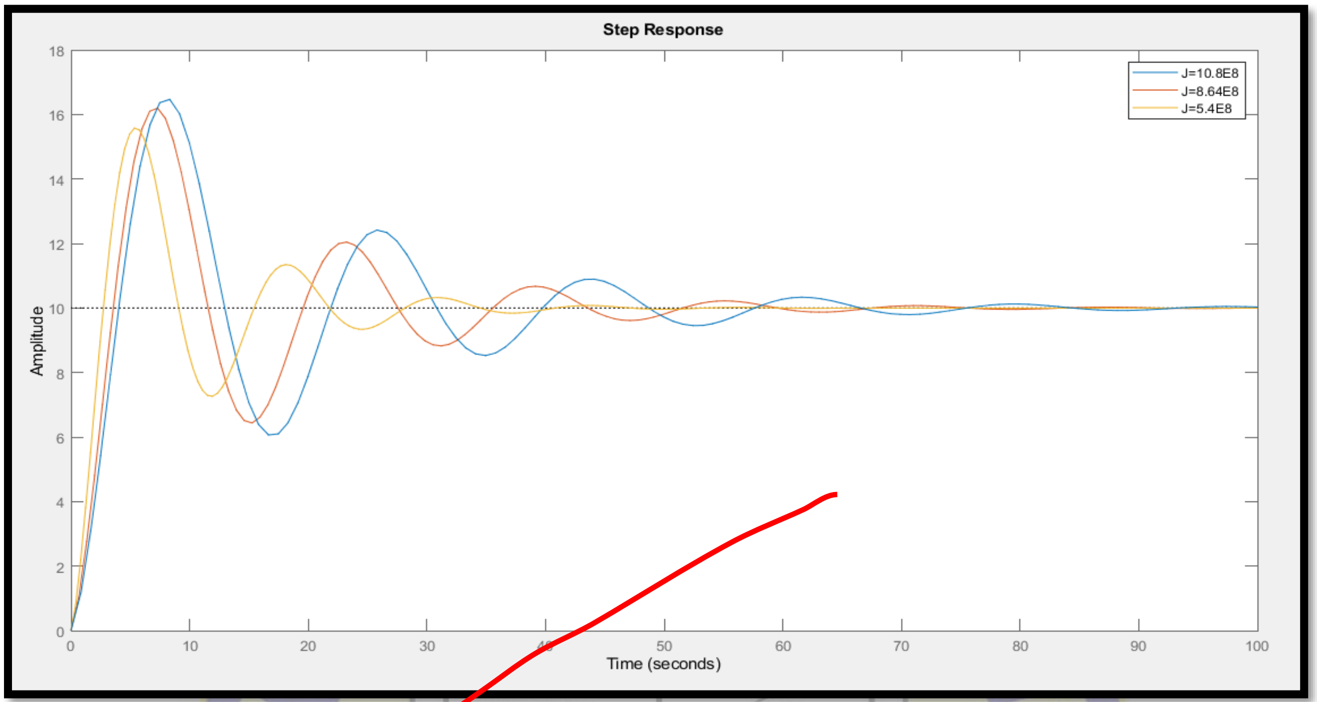


Code:

```
clc; clear all; close all;
k=10.8E8;  a=1;  b=8;
JA = [10.8E8,8.64E8, 5.4E8];  %JA(2) = 20% reduced J   JA(3) = 50% reduced J
controller = tf([k k*a],[1 b]);  %controller transfer function
H = 1;  %feedback
for i=1:3
    spacecraft = tf([1],[JA(i) 0 0]);   %spacecreaft Transfer function
    Tf1 = series(controller,spacecraft);  %controller and space craft is in cascade
    fprintf('For J = %d',JA(i));
    TF = feedback(Tf1,H,-1)
%feedback 1 to the input side of
controller
    step(10*TF)
    hold on
end
legend('J=10.8E8','J=8.64E8','J=5.4E8')
```



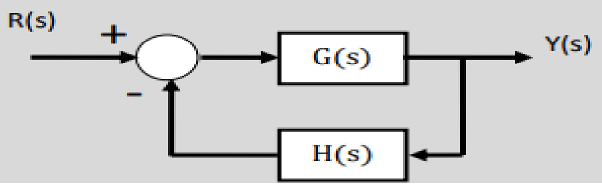
```
For J = 1080000000
TF =

                  1.08e09 s + 1.08e09
  --------------------------------------------------
  1.08e09 s^3 + 8.64e09 s^2 + 1.08e09 s + 1.08e09

Continuous-time transfer function.

For J = 864000000
TF =

                  1.08e09 s + 1.08e09
  ---------------------------------------------------
  8.64e08 s^3 + 6.912e09 s^2 + 1.08e09 s + 1.08e09

Continuous-time transfer function.

For J = 540000000
TF =

                 1.08e09 s + 1.08e09
  -------------------------------------------------
  5.4e08 s^3 + 4.32e09 s^2 + 1.08e09 s + 1.08e09

Continuous-time transfer function.
```



(c): This show that more the value of J (by keeping k,a,b constants) more will be the Tp (Peak time) and Ts (Settling time) and vice versa.

Exercise 4: Consider the feedback control system given in figure, where

$$G(s) = \frac{s+1}{s+2} \text{ and } H(s) = \frac{1}{s+1}.$$



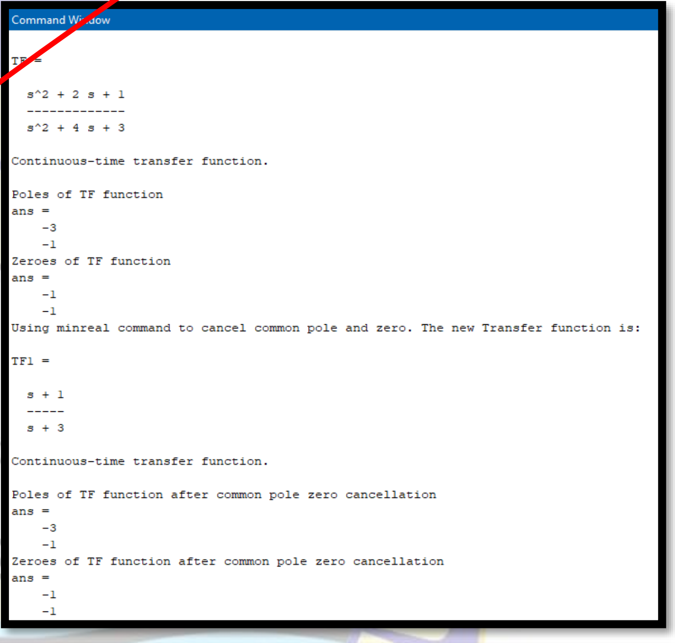
a. Using an m-file script, determine the close-loop transfer function.
b. Obtain the pole-zero map using the 'pzmap' function. Where are the closed-loop system poles and zeros?
c. Are there any pole-zero cancellations? If so, use the 'minreal' function to cancel common poles and zeros in the closed-loop transfer function.
d. Why is it important to cancel common poles and zeros in the transfer function?

Code:

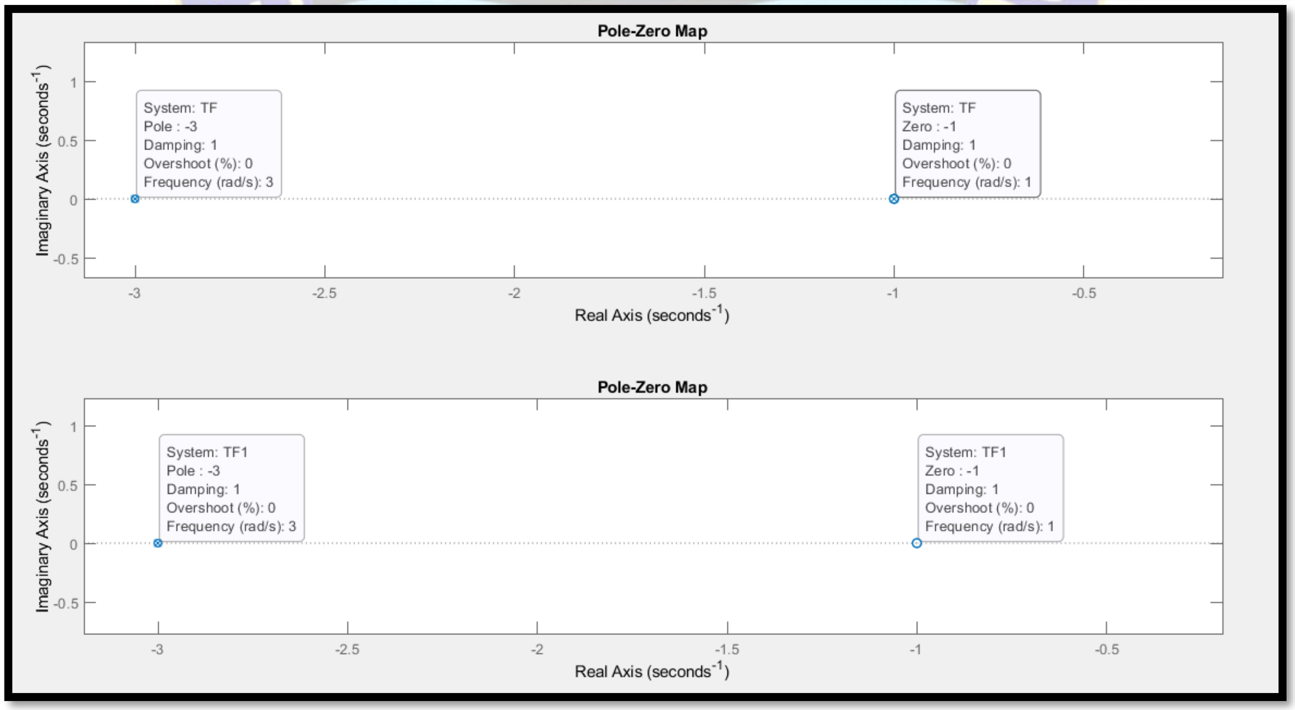
```
clc; clear all; close all; format compact;
G = tf([1 1],[1 2]);
H = tf([1],[1 1]);
TF = feedback(G,H,-1)
%pole and zero plot of transfer function
subplot(2,1,1)
pzmap(TF)
%poles and zeros of transfer function
disp('Poles of TF function');
pole(TF)
```

```
disp('Zeroes of TF function');
zero(TF)
%pole and zero plot after cancellation
disp('Using minreal command to cancel common pole and zero. The new Transfer
function is: ');
subplot(2,1,2)
TF1 = minreal(TF)
pzmap(TF1)
%poles and zeros of transfer function after common pole zero cancellation
disp('Poles of TF function after common pole zero cancellation');
pole(TF)
disp('Zeroes of TF function after
common pole zero cancellation');
zero(TF)
```

(d): In figure shown above, there is a common pole and zero at -1. Common pole and zero indicates that the system under study is unstable. To make it stable, one needs to cancel the common pole and zero.



```
TF =

  s^2 + 2 s + 1
  -------------
  s^2 + 4 s + 3

Continuous-time transfer function.

Poles of TF function
ans =
    -3
    -1
Zeroes of TF function
ans =
    -1
    -1
Using minreal command to cancel common pole and zero. The new Transfer function is:

TF1 =

  s + 1
  -----
  s + 3

Continuous-time transfer function.

Poles of TF function after common pole zero cancellation
ans =
    -3
    -1
Zeroes of TF function after common pole zero cancellation
ans =
    -1
    -1
```



Conclusion:

In this lab, we have learnt the block diagram reduction and rules how to reduce a closed loop block diagram. The advantage of reduction is that we can get a single transfer function defining the overall response of the system but the disadvantage is that we can't describe by just looking at the transfer function how does the system processes the input. We have learned series() and feedback() command to solve a closed loop circuit in MATLAB. We have implemented the rules to find out the transfer function and plot the desired output. We have also plotted the poles and zeros of the transfer function using pzmap() command and then analyzed the system to make it stable by cancelling the common pole and zero.