```
close all; clear all; clc;
```

Given and setup

```
M = [60.1429,18.5714;18.5714,14.2857]
```

```
M = 2×2
   60.1429   18.5714
   18.5714   14.2857
```

```
D = [0, 0 ;0, 2]
```

```
D = 2×2
     0     0
     0     2
```

```
K = [0, 0 ; 0, 1000]
```

```
K = 2×2
           0           0
           0        1000
```

```
B = [1 ; 0 ]
```

```
B = 2×1
     1
     0
```

```
F = [-0.042857;0.071429]
```

```
F = 2×1
   -0.0429
    0.0714
```

```
syms f(theta) f(phi) f(Tc) f(force)
q_state = [f(theta) f(phi)]'
```

q_state =

$$\begin{pmatrix} \overline{f(\theta)} \\ \overline{f(\phi)} \end{pmatrix}$$

```
q_state_dot = [diff(q_state(1)) diff(q_state(2))]'
```

q_state_dot =

$$\begin{pmatrix} \frac{\partial}{\partial \theta}\, f(\theta) \\ \frac{\partial}{\partial \phi}\, f(\phi) \end{pmatrix}$$

```
q_state_dot_dot = [diff(q_state_dot(1)) diff(q_state_dot(2))]'
```

q_state_dot_dot =

$$\begin{pmatrix} \overline{\frac{\partial^2}{\partial \theta^2}\ f(\theta)} \\ \overline{\frac{\partial^2}{\partial \phi^2}\ f(\phi)} \end{pmatrix}$$

```
big_eqn = M * q_state_dot_dot + D*q_state_dot + K*q_state == B *  f(Tc) + F *  f(force)
```

```
big_eqn =
```

$$\begin{pmatrix} \frac{92857\,\sigma_2}{5000} + \frac{1058045086050707\,\sigma_1}{17592186044416} = f(\mathrm{Tc}) - \frac{6176344615366955\ f(\ \mathrm{force})}{144115188075855872} \\ 1000\,\overline{f(\phi)} + \frac{8042134149590837\,\sigma_2}{562949953421312} + \frac{92857\,\sigma_1}{5000} + 2\,\frac{\partial}{\partial \phi}\ f(\phi) = \frac{5147001884535155\ f(\ \mathrm{force})}{72057594037927936} \end{pmatrix}$$

where

$$\sigma_1 = \overline{\frac{\partial^2}{\partial \theta^2}\ f(\theta)}$$

$$\sigma_2 = \overline{\frac{\partial^2}{\partial \phi^2}\ f(\phi)}$$

```
%solved by hand
C = (-M(2,1)*M(1,2)/(M(1,1)))+M(2,2)
```

```
C = 8.5511
```

1-state space

```
A = [0 1;-K(2,2)/C -D(2,2)/C]
```

```
A = 2×2
         0    1.0000
 -116.9443   -0.2339
```

```
B_f = [0; (-F(1) +F(2))/C]
```

```
B_f = 2×1
         0
    0.0134
```

```
B_Tc = [ 0 ; (-B(1)/C) ]
```

```
B_Tc = 2×1
         0
   -0.1169
```

```
B = [B_f,B_Tc]
```

```
B = 2×2
         0         0
    0.0134   -0.1169
```

```
C = [1 ,0]
```

```
C = 1×2
     1     0
```

```
D = [0 ,0]
```

```
D = 1×2
     0     0
```

2

```
[eig_vals,eig_vects] = eig(A)
```

```
eig_vals = 2×2 complex
  -0.0010 - 0.0921i  -0.0010 + 0.0921i
   0.9958 + 0.0000i   0.9958 + 0.0000i
eig_vects = 2×2 complex
  -0.1169 +10.8134i   0.0000 + 0.0000i
   0.0000 + 0.0000i  -0.1169 -10.8134i
```

3

```
tf_final = ss(A,B,C,D)
```

```
tf_final =

  A =
             x1       x2
   x1         0        1
   x2    -116.9  -0.2339

  B =
             u1       u2
   x1         0        0
   x2   0.01337  -0.1169

  C =
        x1  x2
   y1    1   0

  D =
        u1  u2
   y1    0   0

Continuous-time state-space model.
```

```
[wn,zeta,poles] = damp(tf_final)
```

```
wn = 2×1
   10.8141
   10.8141
zeta = 2×1
    0.0108
    0.0108
poles = 2×1 complex
  -0.1169 +10.8134i
```

```
  -0.1169 -10.8134i
```

```
control_mat_both = ctrb(A,B);
trust_torque_bool = length(control_mat_both)-rank(control_mat_both) == 0
```

```
trust_torque_bool = logical
   0
```

```
control_mat_thrust = ctrb(A,B_f);
thrust_bool = length(control_mat_thrust)-rank(control_mat_thrust) == 0
```

```
thrust_bool = logical
   1
```

```
control_mat_torque = ctrb(A,B_Tc);
torque_bool = length(control_mat_torque)-rank(control_mat_torque) == 0
```

```
torque_bool = logical
   1
```

```
observe_sensor = obsv(A,C)
```

```
observe_sensor = 2×2
     1     0
     0     1
```

```
sensor_bool = length(observe_sensor)-rank(observe_sensor) == 0
```

```
sensor_bool = logical
   1
```

4 settime - 30; %OS - 16;

```
B_nof = [0, 0; B_Tc'] % only the torquer
```

```
B_nof = 2×2
         0         0
         0   -0.1169
```

```
rank(ctrb(A,B_nof)) % system controllability
```

```
ans = 2
```

```
% choose new eigenvalues based on performance for an appropriate gain controller
ts = 30; % settling time in seconds
OS = 16/100; % 16 percent overshoot

% zeta = -ln(OS)/sqrt(pi^2 + ln^2(OS))
% Ts = 4 / zeta*wn

syms zeta w_n
```

```
damping_ratio_condition = OS == exp(-pi*zeta/sqrt(1-power(zeta,2)));
zeta = vpa(solve(damping_ratio_condition,zeta));
```

```
Warning: Possibly spurious solutions.
```

```
spec_zeta = double(zeta(1)) % system damping ratio
```

```
spec_zeta = 0.5039
```

```
settling_time_condition = ts == 4 / (zeta(1) * w_n);
spec_wn = double(vpa(solve(settling_time_condition,w_n))) % natural frequency (radians/second)
```

```
spec_wn = 0.2646
```

```

% find poles
poles_spec = [-spec_zeta*spec_wn + spec_wn*i*sqrt(1-power(spec_zeta,2)); -spec_zeta(1)*spec_wn
```

```
poles_spec = 2×1 complex
  -0.1333 + 0.2286i
  -0.1333 - 0.2286i
```

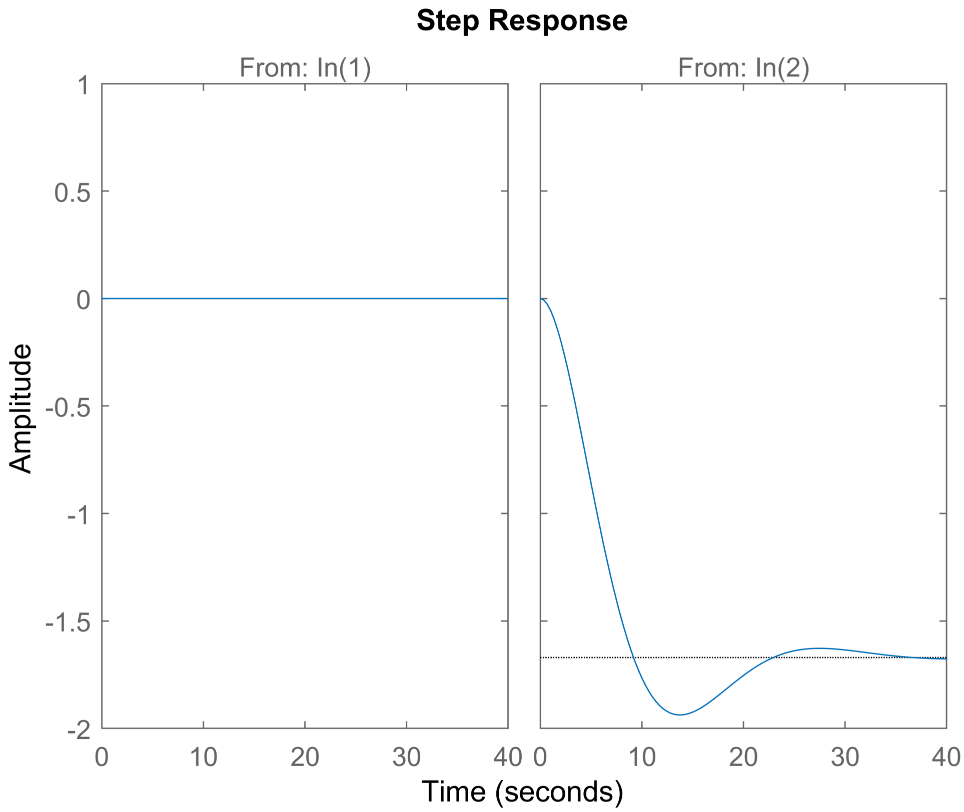
```
% These poles are the new eigenvalues for our system specifications, placed
% as G (our controllable/regulator gain.)
G = place(A,B_nof,poles_spec)
```

```
G = 2×2
         0         0
  999.4012   -0.2803
```

```

A_cl = A-(B_nof*G);
cPlant = ss(A_cl,B_nof,C,0) % new Ax + Bu
```

```
cPlant =

  A =
             x1        x2
   x1         0         1
   x2  -0.07002   -0.2667

  B =
            u1       u2
   x1        0        0
   x2        0  -0.1169

  C =
       x1  x2
   y1   1   0

  D =
       u1  u2
   y1   0   0

Continuous-time state-space model.
```

```
step(cPlant)
```




```
stepinfo(cPlant)
```

ans = 1×2 struct

Fields	RiseTime	SettlingTime	SettlingMin	SettlingMax	Overshoot	Undershoot	Peak	PeakTime
1	0	0	0	0	Inf	0	0	0
2	6.2244	30.3730	-1.9372	-1.5403	15.9972	0	1.9372	13.8155

4b

```
poles_est = poles_spec*2;
bPlant = place(A,B, poles_est)
```

```
bPlant = 2×2
 -112.5423    0.2889
  984.7429   -2.5276
```

```
A_obs = A_cl'
```

```
A_obs = 2×2
         0   -0.0700
    1.0000   -0.2667
```

```
C_obs = B_nof'
```

```
C_obs = 2×2
```

```
         0         0
         0   -0.1169
```

```
B_obs = C'
```

```
B_obs = 2×1
     1
     0
```

```
bPlant = place(A_obs,B_obs, poles_est)
```

```
bPlant = 1×2
    0.2667    0.1390
```

```
syms s L_1 L_2 L_3
poles_est_eqn = vpa(prod(expand(s*poles_est)),5)
```

poles_est_eqn = $0.28009\,s^2$

```
L_mat = [0 L_1; 0 L_2]
```

L_mat =

$$\begin{pmatrix} 0 & L_1 \\ 0 & L_2 \end{pmatrix}$$

```
eqn_1_obs = vpa(A_obs - L_mat * C_obs,5)
```

eqn_1_obs =

$$\begin{pmatrix} 0 & 0.11694\,L_1 - 0.070023 \\ 1.0 & 0.11694\,L_2 - 0.26667 \end{pmatrix}$$

```
eqn_2_obs = eye(size(A_obs))*s - eqn_1_obs
```

eqn_2_obs =

$$\begin{pmatrix} s & 0.070023488292250135600625071674585 - 0.11694434261622177473327610641718\,L_1 \\ -1.0 & s - 0.11694434261622177473327610641718\,L_2 + 0.26666666666666515084216371178627 \end{pmatrix}$$

```
L_1_answer = vpa(solve(vpa(det(eqn_2_obs) == poles_est_eqn,5),L_1),5) == L_1
```

L_1_answer = $2.2802870211667785803351587785971\,s - 1.0\,L_2\,s + 6.1559715564311417467801356575303\,s^2 + 0.598$

```
L_2_answer = vpa(solve(vpa(det(eqn_2_obs) == poles_est_eqn,5),L_2),5) == L_2
```

L_2_answer =

$$\frac{8.5510763293759057486703468750689\,\left(0.71990604683099945759749971330166\,s^2 + 0.266666666666\right.}{}$$

```
mat = equationsToMatrix([L_1_answer,L_2_answer],[L_1, L_2])
```

mat =

$$\begin{pmatrix} -1.0 & -1.0\,s \\ -\frac{1.0}{s} & -1.0 \end{pmatrix}$$

```
linsolve(mat,[L_1;L_2])
```

Warning: Solution does not exist because the system is inconsistent.

ans =

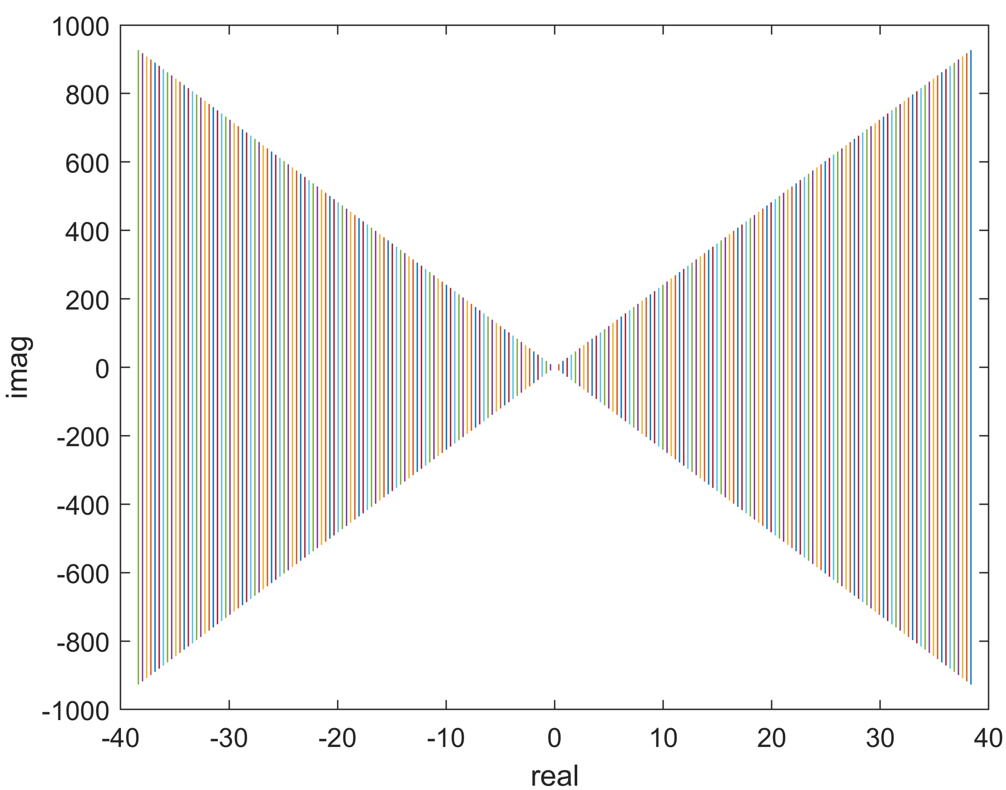
$$\begin{pmatrix} \infty \\ \infty \end{pmatrix}$$

```
error_vals = [-100:100]
```

error_vals = 1×201

```
  -100   -99   -98   -97   -96   -95   -94   -93   -92   -91   -90   -89   -88 ···
```

```
plot(real(eig((A-bPlant*C')) *error_vals),imag(eig((A-bPlant*C')) *error_vals))
xlabel('real')
ylabel('imag')
```



4C