MECH 6323 - HW 07

Author: Jonas Wagner

Date: 2022-05-08

Problem 1

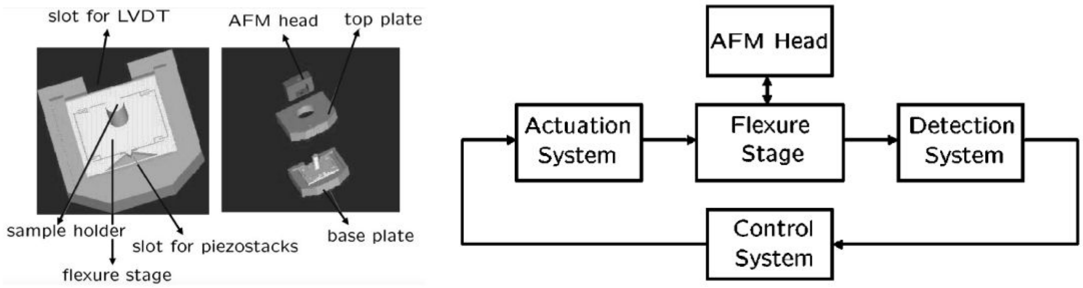


Fig. 1: Nanopositioning flexure stage (left) and feedback diagram (right); figures adapted from Salapaka et. al, *Rev. Sci. Instrum.* 2002.

```
clear
close all
```

Part a

Load data: G, w, nano_sp.Gfr

```
% load('C:\Users\Jonas\OneDrive - The University of Texas at Dallas\2022_Spring\MECH6323\Homewo
nano_rsp = load('npresp.mat')
```

```
nano_rsp = struct with fields:
    Gfr: [1×1 frd]
      w: [1748×1 double]
      G: [1×1×1748 double]
```

```
omega_min = min(nano_rsp.w);
omega_max = max(nano_rsp.w);
```

Estimate system transfer function

```
tf_order = 6;
G_sys = fitfrd(nano_rsp.Gfr, tf_order)
```

```
G_sys =

  A = 
          x1     x2     x3     x4     x5     x6
   x1  -1468   8540  -8540   8540  -8540   4270
   x2  -4681   6559  -3757   3757  -3757   1878
   x3  -2174   4348  -7150   9952  -9952   4976
   x4  -4695   9390  -9390   6588  -3786   1893
   x5  -1687   3374  -3374   3374  -6176   4489
   x6  -3428   6856  -6856   6856  -6856  625.7
```

```
  B = 
         u1
   x1  5.156
   x2  11.36
   x3   16.7
   x4  18.15
   x5  11.59
   x6  11.18

  C = 
           x1      x2      x3      x4      x5      x6
   y1   114.1  -228.1   228.1  -228.1   228.1  -114.1

  D = 
           u1
   y1  0.08876

Continuous-time state-space model.
```

```
G_sys_tf = tf(G_sys)
```

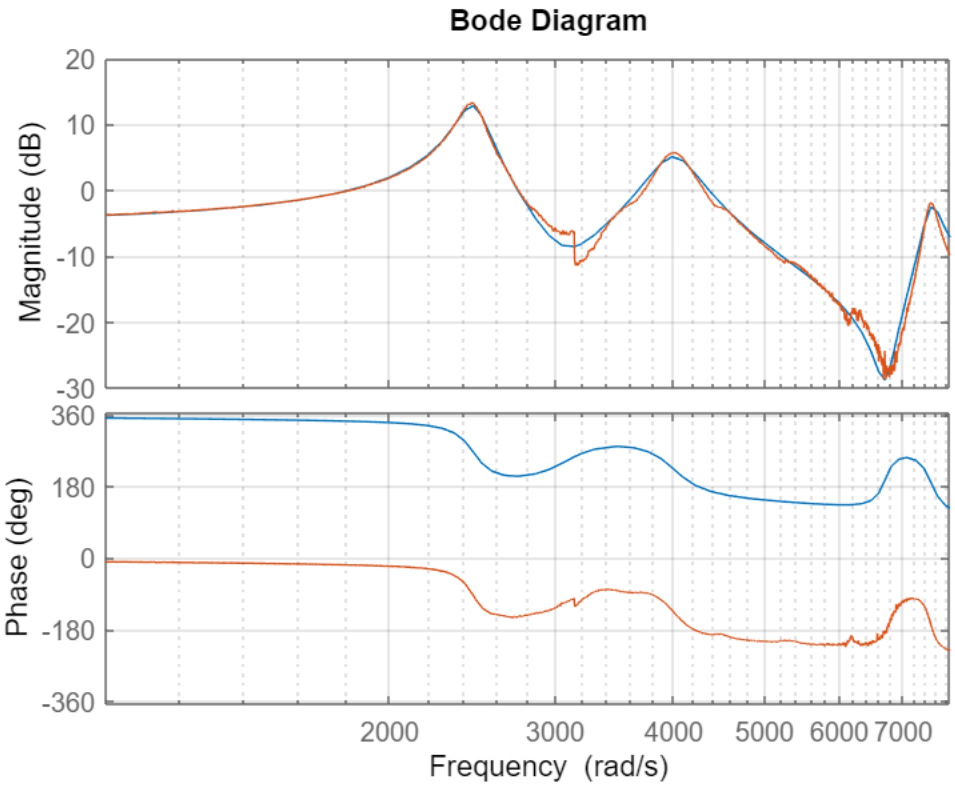
```
G_sys_tf =
 
  0.08876 s^6 - 876.1 s^5 + 1.136e07 s^4 - 4.345e10 s^3 + 4.097e14 s^2 - 2.095e17 s + 3.082e21
  ---------------------------------------------------------------------------------------------
       s^6 + 1021 s^5 + 7.856e07 s^4 + 5.129e10 s^3 + 1.342e15 s^2 + 3.65e17 s + 5.421e21
 
Continuous-time transfer function.
```

```
G_sys_zpk = zpk(G_sys)
```

```
G_sys_zpk =
 
  0.088762 (s^2 + 526.7s + 9.417e06) (s^2 + 276.6s + 4.494e07) (s^2 - 1.067e04s + 8.207e07)
  ------------------------------------------------------------------------------------------
         (s^2 + 186.2s + 6.029e06) (s^2 + 482.7s + 1.6e07) (s^2 + 352.5s + 5.621e07)
 
Continuous-time zero/pole/gain model.
```

Bode Diagram Data

```
figure()
bode(G_sys)
hold on
bode(nano_rsp.Gfr)
grid on
xlim([omega_min, omega_max])
```



Clearly this plot is a good estimation for the frequency response of the system, noting that the phase of the system is offset by a 360 degree phase shift (which implies a need for another set of integrators)

Part b

PI - Implimentation

```
PM_min = 75;
opt = pidtuneOptions( ...
    'PhaseMargin', PM_min, ...
    'DesignFocus', 'disturbance-rejection' ...
    )
```

```
opt = 
  pidtune with properties:

        PhaseMargin: 75
    NumUnstablePoles: 0
        DesignFocus: 'disturbance-rejection'
```

```
[C_pi, info] = pidtune(G_sys, 'pi', opt)
```

```
C_pi =
 
        1
  Ki * ---
        s

  with Ki = 240
```

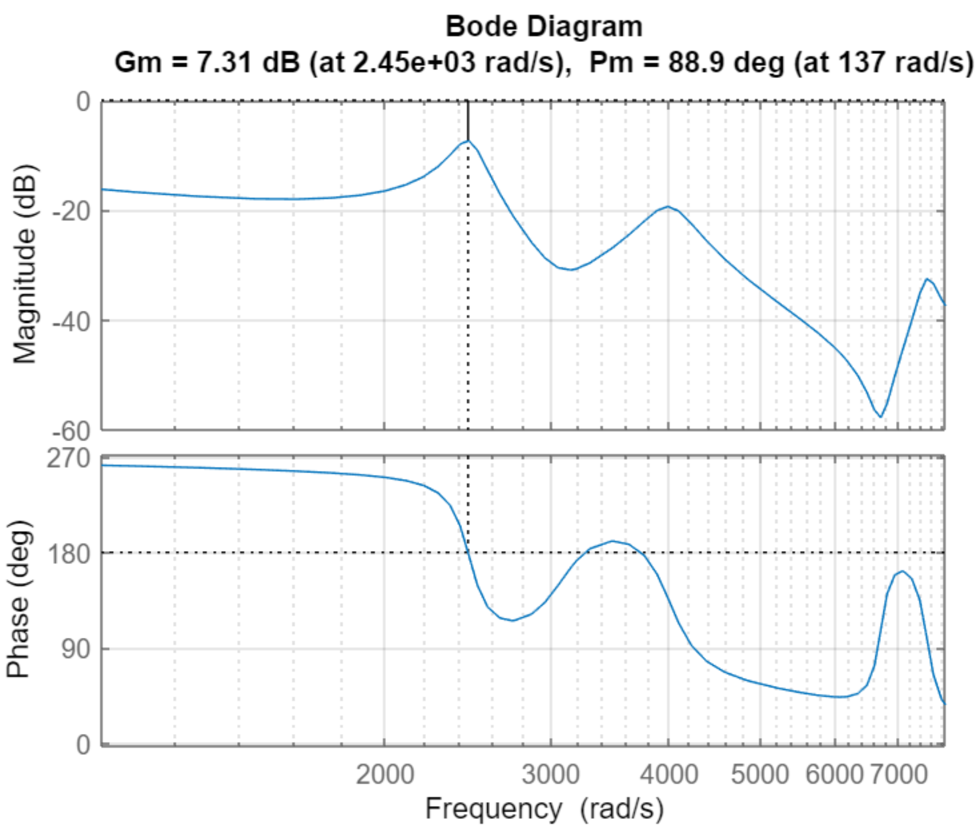
```
Continuous-time I-only controller.
info = struct with fields:
               Stable: 1
    CrossoverFrequency: 136.5946
           PhaseMargin: 88.9408
```

```
allmargin(C_pi * G_sys)
```

```
ans = struct with fields:
     GainMargin: [2.3213 31.8215 13.3190]
    GMFrequency: [2.4472e+03 3.2556e+03 3.7346e+03]
    PhaseMargin: 88.9408
    PMFrequency: 136.5946
    DelayMargin: 0.0114
    DMFrequency: 136.5946
         Stable: 1
```

Bode Diagram of Margin

```
figure
margin(C_pi * G_sys_tf)
xlim([omega_min, omega_max])
grid on
```



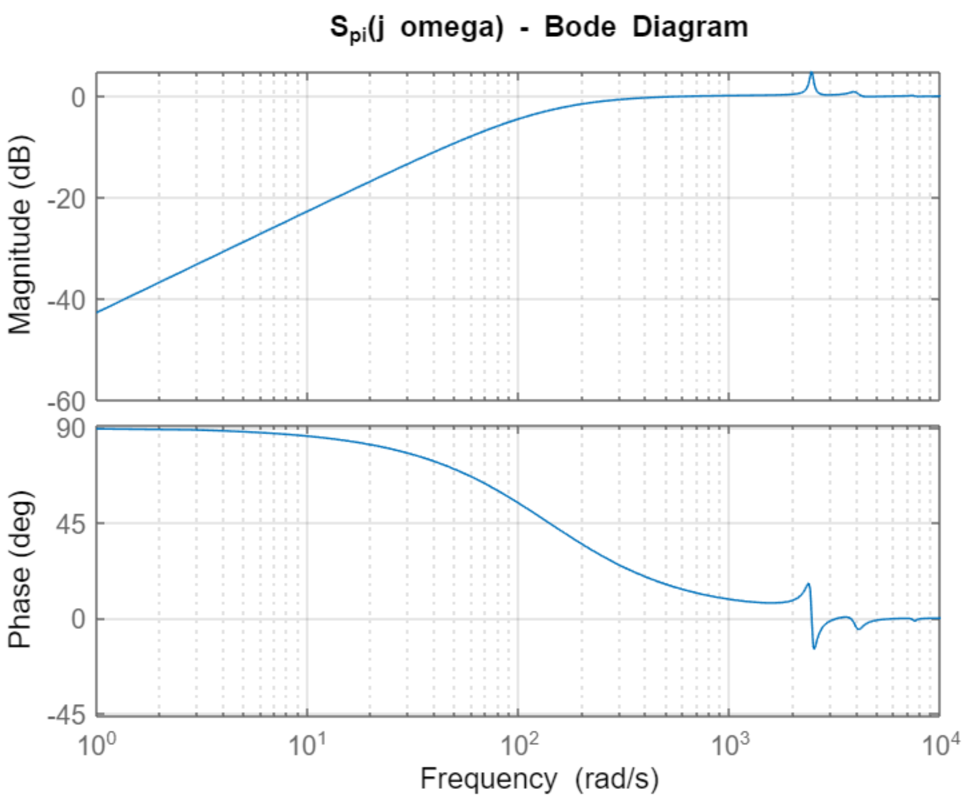
As a result, the single only integral controller is a pretty weird result. However, looking at the results it does make sense.

Sensivity Transfer Function

```
S_pi = zpk(1/(1+C_pi*G_sys))
```

```
S_pi =

       s (s^2 + 186.2s + 6.029e06) (s^2 + 482.7s + 1.6e07) (s^2 + 352.5s + 5.621e07)
  ---------------------------------------------------------------------------------------
  (s+138.6) (s^2 + 108.1s + 5.995e06) (s^2 + 446.2s + 1.584e07) (s^2 + 349.8s + 5.615e07)

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

```
figure
bode(S_pi)
title('S_{pi}(j omega) - Bode Diagram')
grid on
```

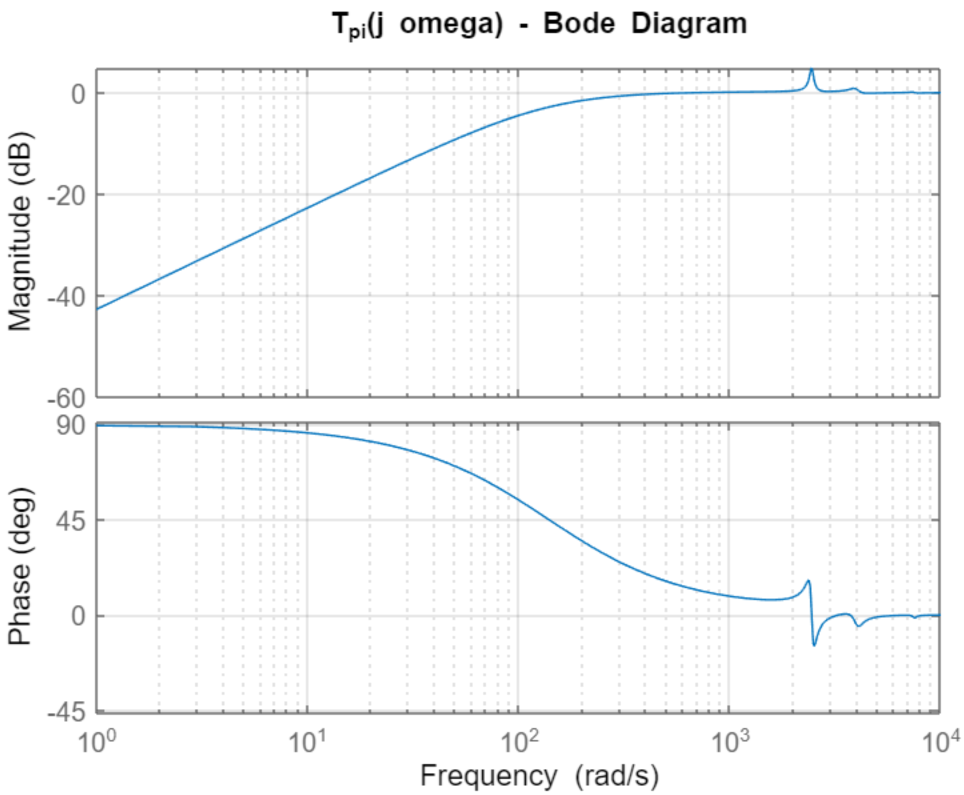


Complimentary Transfer Function

```
T_pi = zpk(1/(1+C_pi*G_sys))
```

```
T_pi =

       s (s^2 + 186.2s + 6.029e06) (s^2 + 482.7s + 1.6e07) (s^2 + 352.5s + 5.621e07)
  ---------------------------------------------------------------------------------------
  (s+138.6) (s^2 + 108.1s + 5.995e06) (s^2 + 446.2s + 1.584e07) (s^2 + 349.8s + 5.615e07)

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

```
figure
bode(T_pi)
title('T_{pi}(j omega) - Bode Diagram')
grid on
```



Bandwith Calculation

```
bw_threshold = -3; %db
bw = getGainCrossover(S_pi, db2mag(bw_threshold))
```

```
bw = 134.4361
```

Double Integrator implimenation

Instead we can also include an additional integrator into the controller, i.e.

```
G_int = tf(1,[1, 0]);
[C_pi_int, info] = pidtune(G_int * G_sys, 'pi', opt)
```

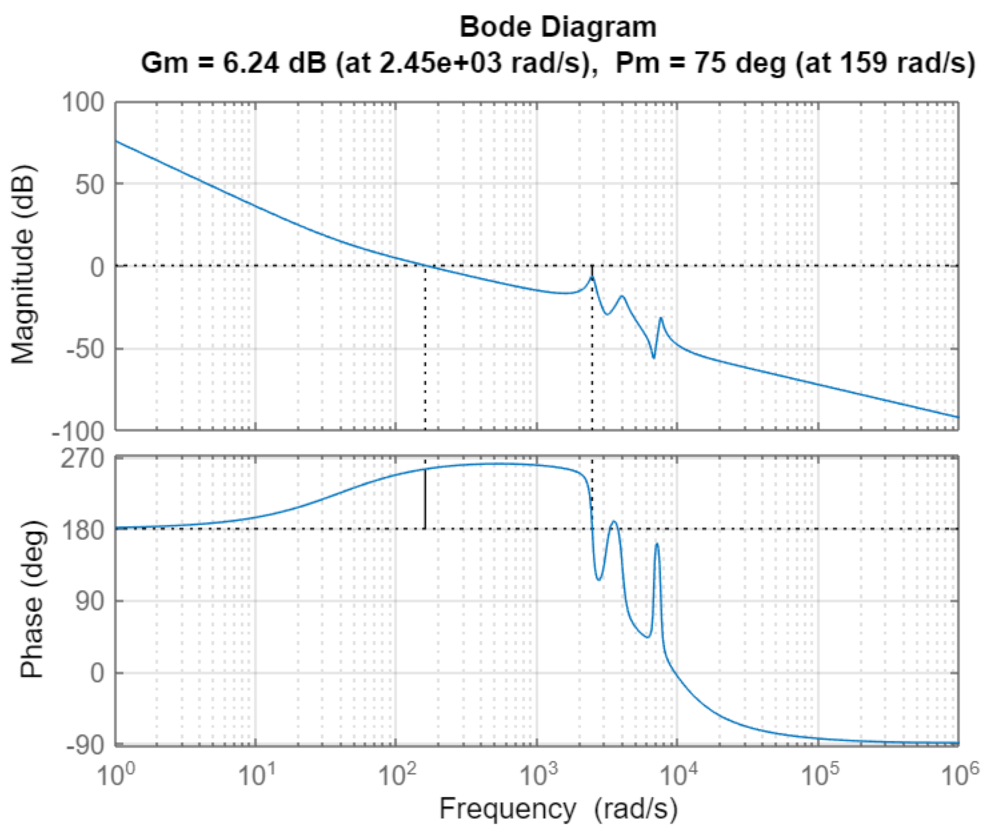
```
C_pi_int =

             1
  Kp + Ki * ---
             s

  with Kp = 271, Ki = 1.06e+04

Continuous-time PI controller in parallel form.
info = struct with fields:
                Stable: 1
    CrossoverFrequency: 159.0729
           PhaseMargin: 75.0000
```

```
allmargin(C_pi_int * G_int * G_sys)
```

```
ans = struct with fields:
```

```
  GainMargin: [0 2.0518 27.8933 11.9425]
 GMFrequency: [0 2.4457e+03 3.2626e+03 3.7282e+03]
 PhaseMargin: 75.0002
 PMFrequency: 159.0759
 DelayMargin: 0.0082
 DMFrequency: 159.0759
      Stable: 1
```

```
figure
margin(C_pi_int * G_int * G_sys)
grid on
```

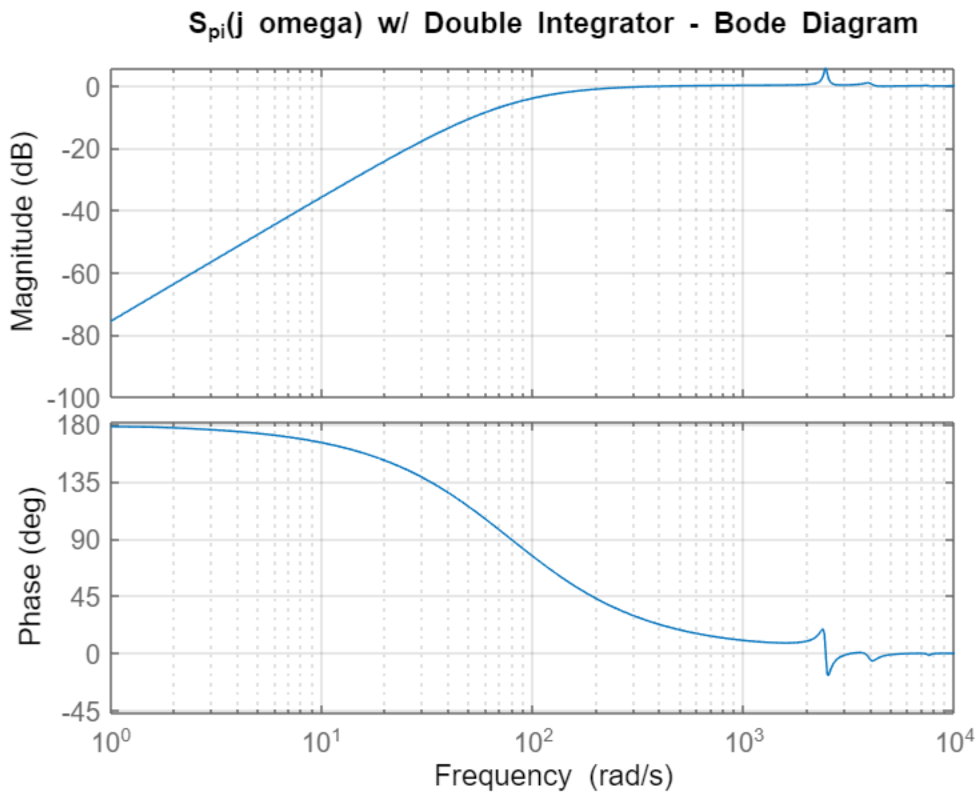


Sensivity Transfer Function

```
S_pi_int = zpk(1/(1+C_pi_int*G_int*G_sys))
```

```
S_pi_int =

           s^2 (s^2 + 186.2s + 6.029e06) (s^2 + 482.7s + 1.6e07) (s^2 + 352.5s + 5.621e07)
  ----------------------------------------------------------------------------------------------------
  (s^2 + 156.3s + 6120) (s^2 + 97.81s + 5.989e06) (s^2 + 441.9s + 1.582e07) (s^2 + 349.5s + 5.614e07)

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

```
figure
bode(S_pi_int)
title('S_{pi}(j omega) w/ Double Integrator - Bode Diagram')
grid on
```

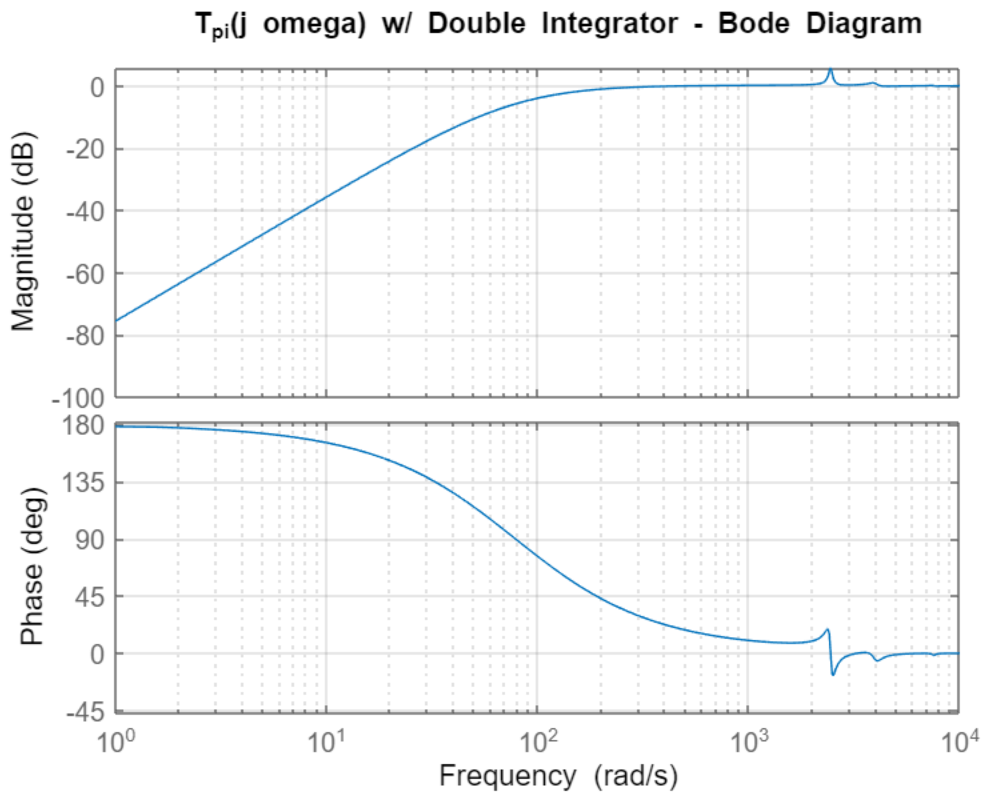


Complimentary Transfer Function

```
T_pi_int = zpk(1/(1+C_pi_int*G_int*G_sys))
```

```
T_pi_int =

            s^2 (s^2 + 186.2s + 6.029e06) (s^2 + 482.7s + 1.6e07) (s^2 + 352.5s + 5.621e07)
  ----------------------------------------------------------------------------------------------------
  (s^2 + 156.3s + 6120) (s^2 + 97.81s + 5.989e06) (s^2 + 441.9s + 1.582e07) (s^2 + 349.5s + 5.614e07)

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

```
figure
bode(T_pi_int)
title('T_{pi}(j omega) w/ Double Integrator - Bode Diagram')
grid on
```



Bandwith Calculation

```
bw_threshold = -3; %db
bw = getGainCrossover(S_pi_int, db2mag(bw_threshold))
```

```
bw = 117.7945
```

The bandwidth of this method is not as great as the original PI implimenation. I think the performance is better in certain situation though and may be worth implimenting (even if it isn't really a PI controller)

Part c

Specs:

1. Bandwidth ($|S(j\omega)| = -3$ dB) is around 250 Hz
2. $|S(j\omega)| \leq 1.5 \ \ \forall_\omega$
3. Slope below bandwidth = 20 dB/decade
4. DC gain of $S \leq -80$dB
5. $|T(j\omega)| < -3$dB @ 500 Hz
6. $|T(j\omega)| \leq 1.5 \ \ \forall_\omega$
7. $|T(j\omega)| < -40$dB as $\omega \rightarrow \infty$
8. $|C_\infty S(j\omega)| \leq 10 \ \ \forall_\omega$

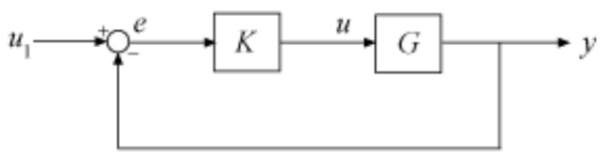
Design Weights

In order design using the mixed sensitivity design approach, we shape $S_{pi}(s)$ and $T_{pi}(s)$ to achieve the desired performance and robustness specs using weighting functions that are inverse of thoose desired shapes.

`[K,CL,gamma,info] = mixsyn(G,W1,W2,W3)` computes a controller that minimizes the H_∞ norm of the weighted closed-loop transfer function

$$M(s) = \begin{bmatrix} W_1 S \\ W_2 KS \\ W_3 T \end{bmatrix},$$

where $S = (I + GK)^{-1}$ and $T = (I - S)$ is the complementary sensitivity of the following control system.



`mixsyn` computes the controller `K` that yields the minimum $||M(s)||_\infty$, which is returned as `gamma`. For the returned controller K,

$$||S||_\infty \le \gamma|W_1^{-1}|$$
$$||KS||_\infty \le \gamma|W_2^{-1}|$$
$$||T||_\infty \le \gamma|W_3^{-1}|.$$

Description

`makeweight` is a convenient way to specify loop shapes, target gain profiles, or weighting functions for applications such as controller synthesis and control system tuning.

`W = makeweight(dcgain,[freq,mag],hfgain)` creates a first-order, continuous-time weight $W(s)$ satisfying these constraints: example

$$W(0) = \text{dcgain}$$
$$W(\text{Inf}) = \text{hfgain}$$
$$|W(j \cdot \text{freq})| = \text{mag}.$$

In other words, the gain of `W` passes through `mag` at the finite frequency `freq`.

W_1 - Shaping S: $\quad ||S||_\infty \le \gamma|W_1^{-1}|$

```
dcgain_1 = db2mag(-80);% Spec 4
hfgain_1 = 1.5; % Spec 2
bw_1 = 250; % Spec 1
W_1 = makeweight(dcgain_1, [2*pi*bw_1, db2mag(-3)], hfgain_1)
```

```
W_1 =

  A = 
           x1
   x1   -2934
```

```
  B =
        u1
   x1   64

  C =
            x1
   y1   -68.77

  D =
         u1
   y1   1.5

Continuous-time state-space model.
```

W_2 - Shaping KS: $\quad ||KS||_\infty \le \gamma |W_2^{-1}|$

```
u_max = 10;
W_2 = tf(1/u_max)
```

```
W_2 =

  0.1

Static gain.
```

W_3 - Shaping T: $\quad ||T||_\infty \le \gamma |W_3^{-1}|$

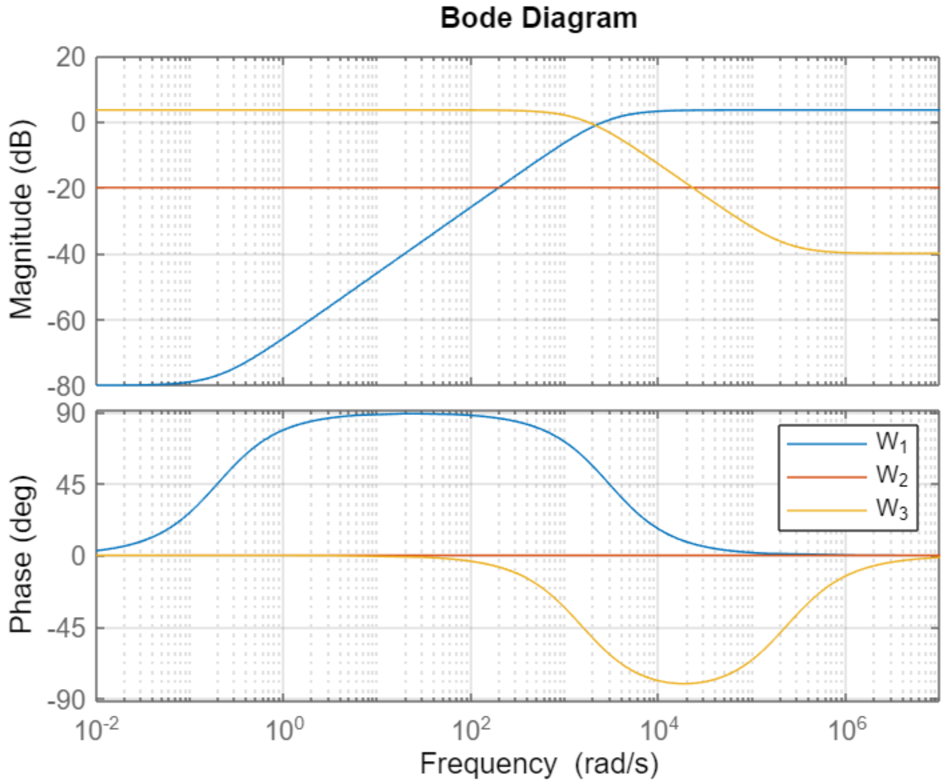
```
dcgain_3 = 1.5; % Spec 6 (max KS should be less then 1.5)
hfgain_3 = db2mag(-40); % Spec 7
bw_3 = 450; %should be less then 500 to ensure below -3dB @ 500 Hz
W_3 = makeweight(dcgain_3, [2*pi*bw_3, db2mag(-3)], hfgain_3)
```

```
W_3 =

  A =
           x1
   x1   -1513

  B =
        u1
   x1   64

  C =
           x1
   y1   35.24

  D =
          u1
   y1   0.01

Continuous-time state-space model.
```

Plotting Weighting functions

```
figure
hold on
bode(W_1)
bode(W_2)
```

```
bode(W_3)
legend('W_1','W_2','W_3')
grid on
```



H_∞ controller Calculation

```
[C_Hinf,CL,gamma,info] = mixsyn(G_sys,W_1,W_2,W_3)
```

```
C_Hinf =

  A = 
                x1          x2          x3          x4          x5          x6          x7          x8
   x1        -2934   -1.51e-10   2.547e-11   4.657e-10  -9.313e-10   6.985e-10  -1.164e-10   8.731e-11
   x2    9.353e+04  -1.527e+04  -5.738e+04  -3.866e+06   4.894e+06  -4.182e+06   2.764e+06  -6.636e+05
   x3    8.488e+04  -1.249e+04  -6.017e+04  -3.487e+06    4.42e+06  -3.774e+06   2.486e+06  -5.914e+05
   x4    1.871e+05  -2.752e+04   -1.34e+05  -7.697e+06   9.756e+06  -8.331e+06   5.494e+06  -1.311e+06
   x5     2.75e+05  -4.046e+04  -1.924e+05  -1.132e+07   1.434e+07  -1.224e+07   8.074e+06  -1.925e+06
   x6    2.989e+05  -4.397e+04  -2.114e+05   -1.23e+07   1.558e+07  -1.331e+07   8.781e+06  -2.095e+06
   x7    1.908e+05  -2.807e+04  -1.336e+05  -7.853e+06    9.95e+06  -8.497e+06   5.601e+06  -1.334e+06
   x8    1.841e+05  -2.709e+04  -1.308e+05  -7.576e+06     9.6e+06  -8.198e+06   5.405e+06  -1.291e+06

  B = 
            u1
   x1       64
   x2    113.4
   x3      103
   x4    226.9
   x5    333.6
   x6    362.5
   x7    231.4
   x8    223.3
```

```
  C = 
             x1          x2          x3          x4          x5          x6          x7          x8
   y1   1.646e+04       -2422  -1.139e+04   -6.78e+05    8.59e+05  -7.336e+05   4.839e+05  -1.155e+05
 
  D = 
          u1
   y1  19.97
 
Continuous-time state-space model.


CL =
 
  A = 
              x1          x2          x3          x4          x5          x6          x7          x8          x9
   x1      -2934           0       -2633        5266       -5266        5266       -5266        2633  -3.373e+04
   x2          0       -1513        2633       -5266        5266       -5266        5266       -2633   3.373e+04
   x3          0           0       -5704   1.701e+04  -1.701e+04   1.701e+04  -1.701e+04        8506   3.062e+04
   x4          0           0  -1.402e+04   2.523e+04  -2.243e+04   2.243e+04  -2.243e+04   1.121e+04   6.747e+04
   x5          0           0   -1.59e+04    3.18e+04   -3.46e+04    3.74e+04   -3.74e+04    1.87e+04    9.92e+04
   x6          0           0  -1.961e+04   3.922e+04  -3.922e+04   3.642e+04  -3.362e+04   1.681e+04   1.078e+05
   x7          0           0  -1.121e+04   2.241e+04  -2.241e+04   2.241e+04  -2.522e+04   1.401e+04    6.88e+04
   x8          0           0  -1.262e+04   2.523e+04  -2.523e+04   2.523e+04  -2.523e+04        9815   6.641e+04
   x9          0           0       -2633        5266       -5266        5266       -5266        2633  -3.667e+04
   x10         0           0       -4667        9335       -9335        9335       -9335        4667   3.373e+04
   x11         0           0       -4236        8472       -8472        8472       -8472        4236   3.062e+04
   x12         0           0       -9335   1.867e+04  -1.867e+04   1.867e+04  -1.867e+04        9335   6.747e+04
   x13         0           0  -1.372e+04   2.745e+04  -2.745e+04   2.745e+04  -2.745e+04   1.372e+04    9.92e+04
   x14         0           0  -1.492e+04   2.983e+04  -2.983e+04   2.983e+04  -2.983e+04   1.492e+04   1.078e+05
   x15         0           0       -9520   1.904e+04  -1.904e+04   1.904e+04  -1.904e+04        9520    6.88e+04
   x16         0           0       -9189   1.838e+04  -1.838e+04   1.838e+04  -1.838e+04        9189   6.641e+04
 
             x16
   x1    2.367e+05
   x2   -2.367e+05
   x3   -2.148e+05
   x4   -4.734e+05
   x5    -6.96e+05
   x6   -7.564e+05
   x7   -4.828e+05
   x8    -4.66e+05
   x9    2.367e+05
   x10   -2.44e+05
   x11  -2.106e+05
   x12  -4.715e+05
   x13  -6.911e+05
   x14  -7.545e+05
   x15  -4.783e+05
   x16  -4.654e+05
 
  B = 
           u1
   x1   23.08
   x2   40.92
   x3   37.13
   x4   81.83
   x5   120.3
   x6   130.8
   x7   83.45
   x8   80.55
   x9   23.08
   x10  40.92
   x11  37.13
   x12  81.83
```

```
   x13  120.3
   x14  130.8
   x15  83.45
   x16  80.55

  C = 
            x1       x2       x3       x4       x5       x6       x7       x8       x9
   y1   -68.77        0   -61.72    123.4   -123.4    123.4   -123.4    61.72   -790.6
   y2        0        0   -82.16    164.3   -164.3    164.3   -164.3    82.16    593.8
   y3        0    35.24   0.4114  -0.8229   0.8229  -0.8229   0.8229  -0.4114    5.271

           x16
   y1     5548
   y2    -4167
   y3   -36.99

  D = 
           u1
   y1     0.541
   y2    0.7203
   y3  0.006393

Input groups:
    Name    Channels
     U1        1

Output groups:
    Name    Channels
     Y1      1,2,3

Continuous-time state-space model.
gamma = 1.4549
info = 
  hinfINFO with properties:

    gamma: 1.4549
        X: [8×8 double]
        Y: [8×8 double]
       Ku: [-325.4596 42.8192 1.8687e+03 9.4406e+03 -1.4372e+04 1.3922e+04 -1.0845e+04 3.5761e+03]
       Kw: [-869.6446 127.2423 943.6919 3.5037e+04 -4.4783e+04 3.8441e+04 -2.5510e+04 6.1675e+03]
       Lx: [8×1 double]
       Lu: 7.2026
     Preg: [5×2 ss]
       AS: [2×2 ss]
```

Part d

H_∞-controller Margin Calculations

```
allmargin(C_Hinf*G_sys)
```

```
ans = struct with fields:
     GainMargin: 6.5542
    GMFrequency: 4.2065e+03
    PhaseMargin: [53.6057 -125.7859]
    PMFrequency: [1.2243e+03 1.6588e+06]
    DelayMargin: [7.6422e-04 2.4644e-06 0]
    DMFrequency: [1.2243e+03 1.6588e+06 Inf]
         Stable: 1
```

```
figure
```

```
margin(C_Hinf * G_sys)
grid on
```

PI - vs H_∞ - controller Bode Comparrision

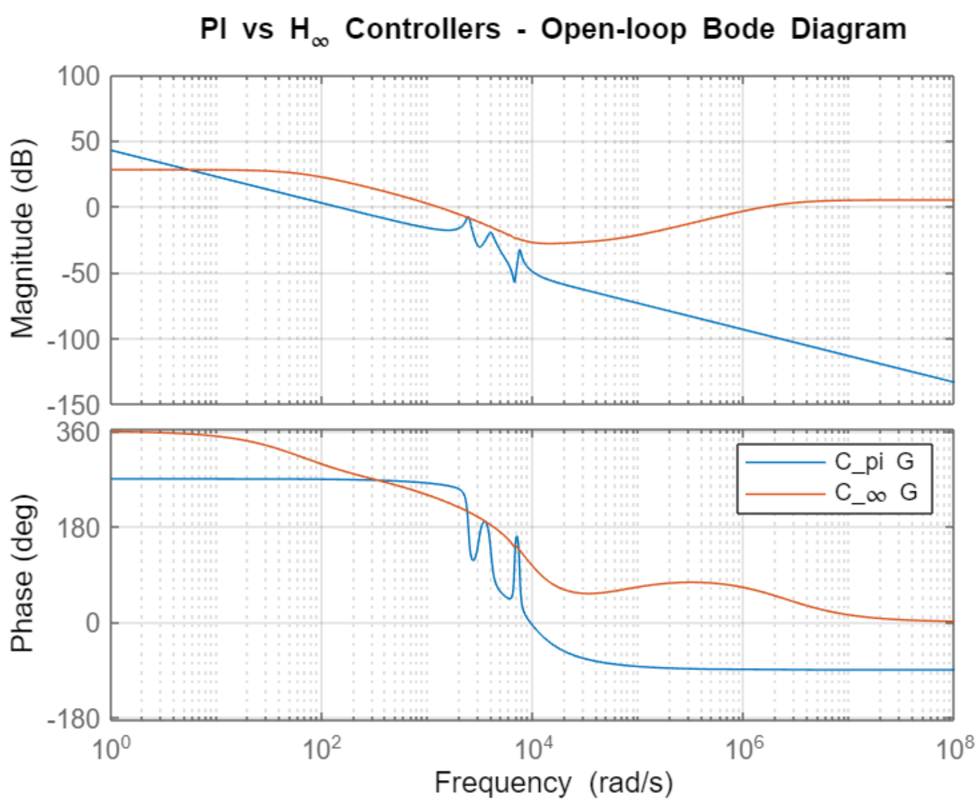
```
figure
hold on
bode(C_pi)
bode(C_Hinf)
legend('C_{pi}', 'C_{H_\infty}')
grid on
```

Loop Transfer Functions

```
sys_pi = feedback(C_pi * G_sys, 1);
sys_Hinf = feedback(C_Hinf * G_sys, 1);
```

Bode Open Loop

```
figure
hold on
bode(C_pi * G_sys, C_Hinf * G_sys)
legend('C_{pi} G', 'C_\infty G')
title('PI vs H_\infty Controllers - Open-loop Bode Diagram')
grid on
```



Bode Closed Loop

```
figure
hold on
bode(sys_pi,sys_Hinf)
legend
title('PI vs H_\infty Controllers - Closed-loop Bode Diagram')
grid on
```

Nyquist Loops

```
figure
nyquist(sys_pi, sys_Hinf)
legend
title('PI and H_\infty Closed-loop Comparrision - Nyquist')
grid on
```

Problem 2

```
```

Final Parts

```
fname = matlab.desktop.editor.getActiveFilename;
export(fname, 'MECH6323_HW07.pdf')
```