MATLAB Assignment DP10.1

Table of Contents

Calculate p and z for phase lead network

Constants

```
open_loop_pole_1 = 0;
open_loop_pole_2 = -2;

% Requirements
Ts_max = 1;
PO_max = 15;
ess_ramp = 0.02;

% SET DESIGN PARAMETERS
damping = 0.8;
wn = 5.5;

% Check that Ts requirement met
Ts_estimate = 4 / (damping * wn);
if Ts_max < Ts_estimate
   error(sprintf('Ts requirement not met. Ts_max = %f. Ts_estimate =
 %f', Ts_max, Ts_estimate));
   return
end


% Calculate desired poles
desired_pole_1 = -damping*wn + wn*sqrt(1 - damping^2) * i;
desired_pole_2 = conj(desired_pole_1);

% Calculate z and p for phase lead compensator
z = real(desired_pole_1);
theta0 = rad2deg(atan2(imag(desired_pole_1) - imag(open_loop_pole_1),
 real(desired_pole_1) - real(open_loop_pole_1)));
theta1 = rad2deg(atan2(imag(desired_pole_1) - imag(open_loop_pole_2),
 real(desired_pole_1) - real(open_loop_pole_2)));
theta2 = rad2deg(atan2(imag(desired_pole_1) - imag(z),
 real(desired_pole_1) - real(z)));
```

```
phi = -theta0 - theta1 + theta2;
theta_p = 180 + phi;
l = imag(desired_pole_1) / tan(deg2rad(theta_p));
p = z - l;
```

Setup transfer function

```
K = abs(open_loop_pole_1 - desired_pole_1) * abs(open_loop_pole_2 -
 desired_pole_1) * abs(p - desired_pole_1) / abs(z - desired_pole_1);
G = tf([20], [1, 2, 0]);
G_lead = tf(K .* [1, -z], [1, -p]);
loop_tf = G * G_lead;
closed_loop_tf = feedback(loop_tf, [1]);
```

Calculate step response performance

```
stepInfo = stepinfo(closed_loop_tf);

[y,t] = step(closed_loop_tf);

Ess = abs(1-y(end));

disp(sprintf('Selected design parameters: damping: %f. wn: %f. K: %f',
 damping, wn, K));
disp(sprintf('Resulting z and p: z: %f. p: %f', z, p));
disp(sprintf('Step Response performance: T_s: %f. PO: %f. Ess: %f',
 stepInfo.SettlingTime, stepInfo.Overshoot, Ess));
```

```
Selected design parameters: damping: 0.800000. wn: 5.500000. K: NaN
Resulting z and p: z: 3298.900000. p: 3298.900000
Step Response performance: T_s: NaN. PO: NaN. Ess: NaN
```

Plot step response

```
fig = figure;
step(closed_loop_tf);
uiwait(fig);
```

```
Error using roots (line 27)
Input to ROOTS must not contain NaN or Inf.

Error in ltipack.tfdata/pole (line 19)
   p = roots(D.den{1});

Error in resppack.ltisource/isstable (line 18)
      p = pole(D,'fast');

Error in resppack.ltisource/getFinalValue (line 13)
      Stable = isstable(this,ModelIndex);  % Note: will update DC gain
 value
```

```
Error in resppack.TimeFinalValueData/update (line 14)
   cd.FinalValue =
   real(getFinalValue(r.DataSrc,find(r.Data==cd.Parent),r.Context));

Error in wavepack.waveform/addchar>LocalUpdateData (line 62)
      update(c.Data(ct),wf)

Error in wavepack.wavechar/draw (line 12)
   feval(this.DataFcn{:});

Error in wavepack.waveform/draw (line 48)
      draw(c,varargin{:})

Error in wrfc.plot/draw (line 17)
      draw(wf)

Error in wrfc.plot/init_listeners>LocalRefreshPlot (line 79)
   draw(this)
```

Calculate ramp response performance

```
stepInfo = stepinfo(closed_loop_tf / tf([1,0], [1]));

[y,t] = step(closed_loop_tf / tf([1,0], [1]));

Ess = abs(t(end)-y(end));

disp(sprintf('Ramp Response performance: T_s: %f. PO: %f. Ess: %f',
 stepInfo.SettlingTime, stepInfo.Overshoot, Ess));
```

```
Ramp Response performance: T_s: NaN. PO: NaN. Ess: NaN
```

Plot ramp response

```
fig = figure;
step(closed_loop_tf / tf([1,0], [1]));
title('Ramp Response');
hold on;
plot(t, t, 'r--');
uiwait(fig);
```

```
Index exceeds the number of array elements (48).

Error in MATLAB_Assignment_DP10_1_COPY_2 (line 76)
title('Ramp Response');
```

Plot zeros and poles

```
fig = figure;
pzmap(closed_loop_tf);
uiwait(fig);
```

Published with MATLAB® R2019a