ECE 365

Spring 2006

Matlab Project 2:

Root Locus

John O'Flaherty

1] Problem statement:

It is desired to find the closed loop pole locations for a time constant of 1/3 second for this system:

$$G(s) = \frac{k\left(1+\frac{s}{5}\right)}{s^2\left(1+\frac{s}{12}\right)} = \frac{12k(s+5)}{5s^2(s+12)} \text{ for feed forward,}$$

and $H(s) = 1+\frac{s}{12} = \frac{(s+12)}{12}$ for feedback.

The open loop transfer function is $G(s)H(s) = \frac{k(s+5)}{5s^2}$ (after cancellation).

The closed loop transfer function is $\frac{G(s)}{1+G(s)H(s)} = \frac{12k(s+5)}{5(s+12)\left(s^2+\frac{k}{5}s+k\right)}$.

2] Approach: The entire root locus is plotted by hand and by Matlab, and the desired root locations are found by Matlab and by closed form mathematical solution.

3] Analysis and results:

A: **Root locus construction and sketch:**

1. The expression G(s)H(s) is equated to -1. There are two finite open loop poles at s = 0. There is one finite open loop zero at s = -5. The number of asymptotes is np – nz = 1.

2.The poles and the zero lie on the real axis. A point to the left of the zero at -5 has an odd number (3) of poles and zeros to its right, as do all points from there to – infinity, so all lie on the root locus.

3. The angle of the only asymptote is 180° / 1 = 180°. This asymptote coincides with the real axis.

4. The breakaway and breakin points occur at *dk/ds* = 0. Solving the equation $\frac{k(s+5)}{5s^2}+1=0$ for *k* gives

$k = -\frac{5s^2}{(s+5)}$. Then solving $\frac{dk}{ds} = -\frac{5(s(s+10))}{(s+5)^2} = 0$ gives *s* = 0 and *s* = -10. The points at *s* = 0 are breakaway points coincident with the two poles at zero.

5. Application of Routh's criterion shows no sign changes for positive values of *k*, so there are no imaginary axis crossings except the poles at zero.

6. Two poles to the right of a zero form a circle around the zero. The circle is centered at -5; since the locus includes s = 0, the radius of the circle must be 5. The other side of the circle then coincides with the breakin points at -10.

7. The angles of arrival and departure are determined by the fact that the poles lie on the real axis and also are the breakaway points for the circle. The departure angles must be ± 90°. The arrival angles at the zeros at -5 and - infinity must be 0° and 180°, respectively.

A hand sketch of the root locus based on the foregoing information is attached to the report.

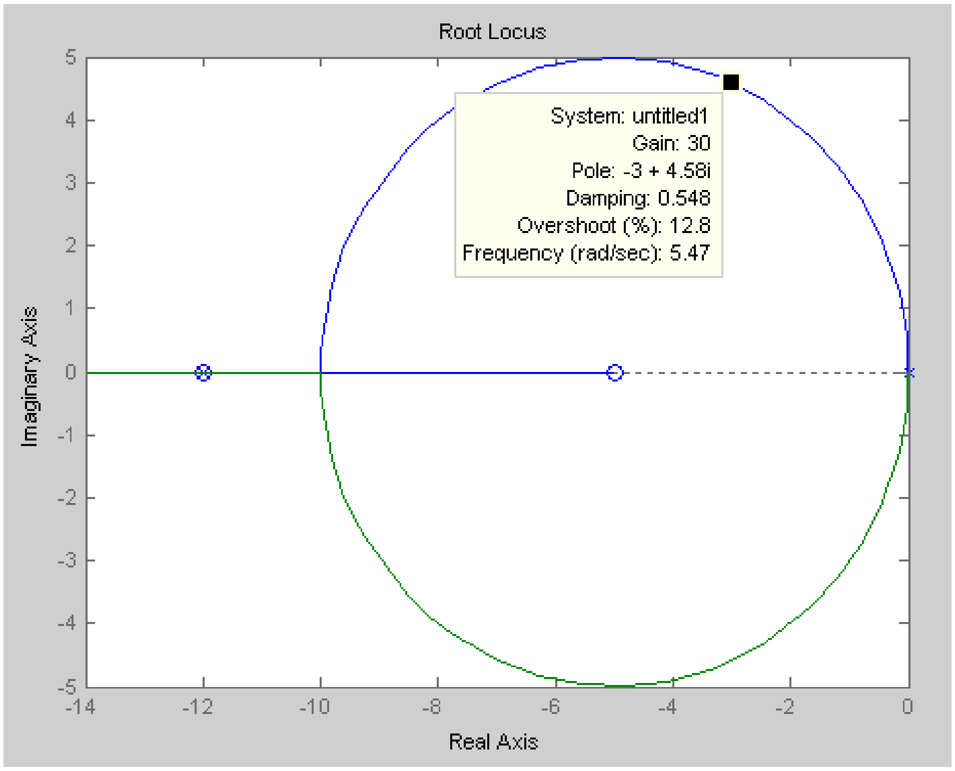
The information provided by the root locus plot is the trajectory of the roots of the closed loop system as the gain (*k*) of the feedback is changed from zero to infinity. The open loop poles are the starting points at *k* = 0, and the zeros are the ending points at *k* = infinity. The gain at any point on the root locus can be calculated as the product of the distances from that point to all poles divided by the product of the distance to all zeros. The location of poles chosen along the root locus determines the transient response of the system. Thus it is possible to choose available points on the root locus for a given system to get the desired transient behavior, and then calculate the gain necessary to position the poles there.

B. Matlab root locus:

The uncanceled expression is used so the root locus plot will graphically show the overlying pole and zero of the cancellation -

$$G(s)H(s) = \frac{k\left(1+\frac{s}{5}\right)}{s^2\left(1+\frac{s}{12}\right)}, \quad H(s) = 1+\frac{s}{12}$$

The system could be entered into Matlab as `G=tf([1/5 1],[1/12 1 0 0])`, `H=tf([1/12 1],[1])`, but it is simpler to enter `s=tf('s')`, followed by literal entry of the formulas: `g=(1+s/5)/(s^2*(1+s/12)),h=1+s/12`. The resulting transfer functions echoed by Matlab are identical. The root locus is plotted by `'rlocus(g*h)'`:



C. Setting k for t = 1/3 second:

The target time constant is 1/3 second. The inverse of the time constant is $\zeta\omega_n = 3$ rad / s. This will be the abscissa of the final selected pole location. An approximate gain figure can be found by use of the mouse on the root locus plot. Once the approximate gain figure of 30 is obtained, the numerical data can be sharpened by `'[r,k]=rlocus(g*h,29:0.01:31);'`, followed by `'r2=r(2,:);'`. This last selects the column with the needed information and discards the columns for the static pole at -12 and the conjugate of the pole of interest. Then the instruction `'find(abs(real(r2)+3)<0.001)'` gives the result `'100 101 102'`, and choosing the middle value `'r2(101)'` gives the desired pole location, -3.0000 + 4.5826i. The instruction `'k(101)'` gives the gain at this location, 30. The total gain of the system at the desired time constant of 1/3 second is then 30.

Analytical solution by CAS:

The desired gain is provided by the solutions to the equation G(s)H(s) = -1. The angle condition is 180 degrees (or any odd multiple thereof), and the magnitude condition is unity. The open loop transfer function with (-3 + $j\omega$) substituted for s is $\frac{k(-3+j\omega+5)}{5(-3+j\omega)^2} = \frac{k(2+j\omega)}{5(-3+j\omega)^2}$. Solving $\angle \frac{k(2+j\omega)}{5(-3+j\omega)^2} = \pi$

for ω_d with a numeric solver gives 4.582576 (the angle is independent of *k* in this equation). Substituting

this value for ω and numerically solving $\left|\frac{k(2+j\omega)}{5(-3+j\omega)^2}\right| = \left|\frac{k(2+j4.58)}{5(-3+j4.58)^2}\right| = 1$ for *k* gives a gain of 30.

D: Resulting closed loop transfer function:

Substituting k = 30 into the system transfer functions, and extracting and multiplying by the DC gain factor already implied by the transfer functions gives the simplified expression for the closed loop transfer function:

$$\frac{G(s)}{1+G(s)H(s)} = \frac{12\times 30\times(s+5)}{5(s+12)\left(s^2+\frac{k}{5}s+k\right)} = \frac{\mathbf{72(s+5)}}{\mathbf{(s+12)(s^2+6s+30)}}\ .$$

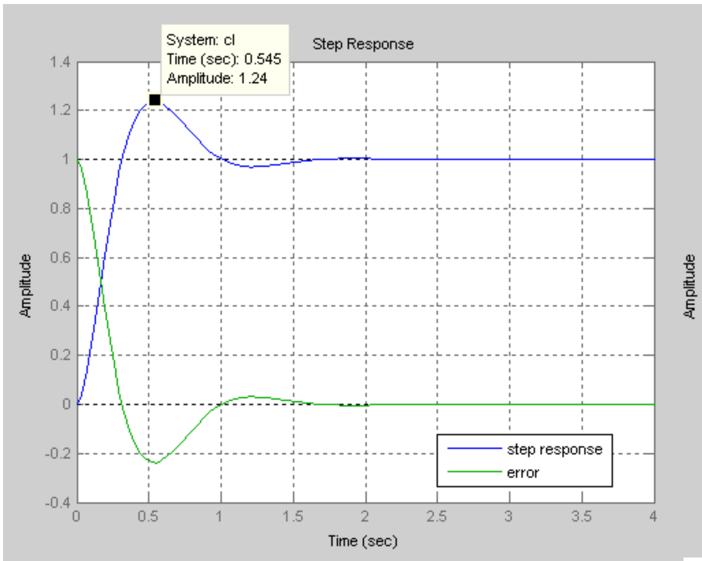
In fully factored form, this is

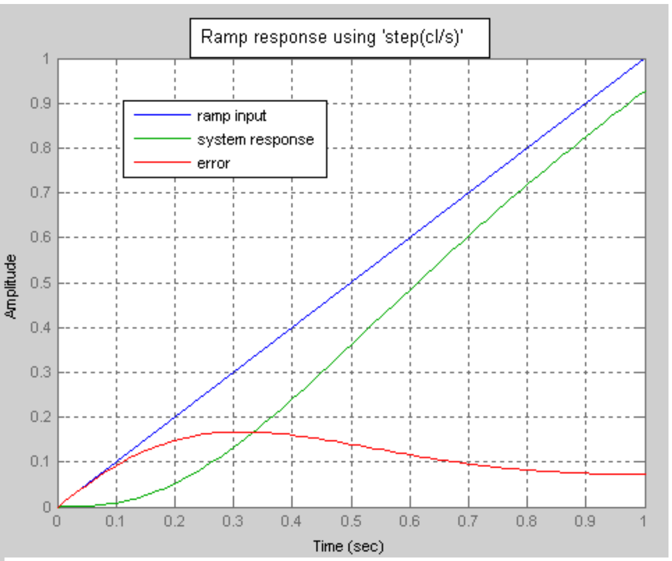
$$\frac{72(s+5)}{(s+12)(s+3+\sqrt{21}j)(s+3-\sqrt{21}j)} = \frac{72(s+5)}{(s+12)(s+3+4.58258j)(s+3-4.58258j)}$$

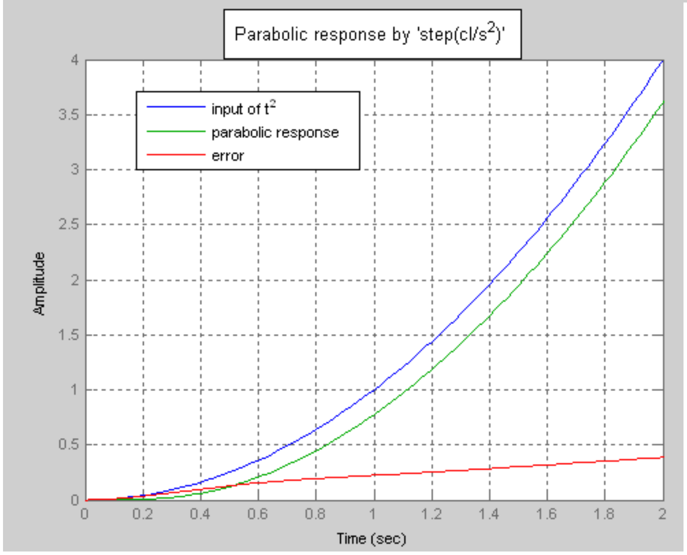
E. Matlab simulations:

The closed loop system can be created either by the instruction `'cl=feedback(30*g,h)'` or by `'cl=30*g/(1+30*g)'`. The gain is applied to G(s) so the scaling of the system isn't distorted. If the gain is applied to the feedback, the dynamics will be the same but a unit step will produce an output of 1 / 30. Following are plots of response of the system to unit step, unit ramp, and parabolic (t^2 / 2) inputs. The graphs for `'step(cl)'`, `'step(cl/s)'`, and `'step(cl/s^2)'` produce the same outputs as the function `'lsim()'` with appropriate inputs, but are simpler to use for these basic functions. The graphs also show the system error, or (input - output); in the case of the ramp and parabolic inputs, the forcing function is also shown.

The first graph is the step response of the system. Note that the overshoot shown in the pop-up box is 24%. The overshoot figure shown in the root locus plot is 12.8% but this figure applies only to systems without zeros in the closed loop transfer functions. There should be a warning in the pop-up box about the limited applicability of its data.

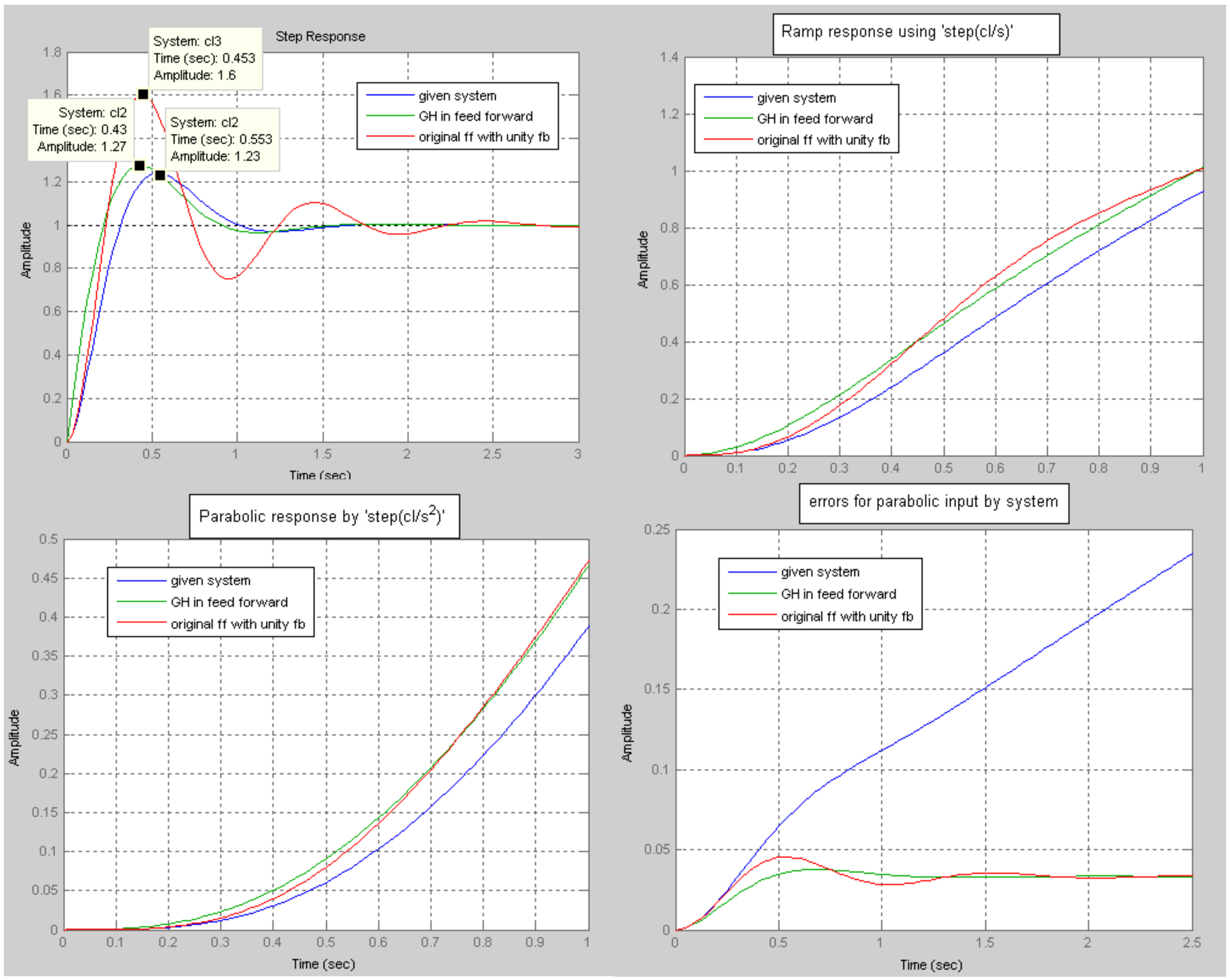






In the last graph, it is interesting that the parabolic response shows a continually increasing error. Experimentation shows that this error increases without limit. It seems to be due to the 1 + s / 12 term in the feedback. While this cancels the like term in the feed forward function as far as the open loop gain is concerned, the conclusion is inevitable that this particular feedback has converted the system from Type 2 to Type 1. If the system is rearranged to put G and H in series with unity feedback, it is restored to a Type 2 system and the acceleration error becomes a constant. The dynamics of the system are changed a little by this maneuver even though the root locus, based on the open loop transfer function G(s)H(s), is the same.

Following is a series of graphs comparing the outputs of (1) the original system (G / (1 + GH)), (2) the system with H put into the feed forward (GH / (1 + GH)), and (3) the original feed forward function with unity feedback (G / (1 + G)). The first graph in this series shows the step response to a system that doesn't cancel the 1 + s / 12 term in G. This system has 60 % overshoot. The system as originally given has 24 % overshoot, while the system modified to put H into feed forward has 27 % overshoot, and a slightly faster response. The last graph shows that putting H into the feed forward makes the error for acceleration inputs constant, and so restores the system to Type 2 status. If the accuracy for acceleration inputs is important in the application, the system with GH as feed forward seems preferable.



4] Conclusions:

Root locus is an interesting tool for understanding dynamic systems. Matlab is a very efficient and powerful means of exploring transfer functions, root loci, and time domain responses of systems. The loss of Type 2 status for the given system would not have been apparent, at least to this student, without use of Matlab.

5] Appendix:

Matlab code used to produce the root locus and the system response to step, ramp and parabolic inputs.

```
s=tf('s')
g=(1+s/5)/(s^2*(1+s/12))
h=1+s/12
ol=g*h
figure
rlocus(ol)
cl=feedback(30*g,h)

figure
step(cl,1-cl,1)
grid
% Ornamentation…
hold on
legend({'system response','error'},...
  'FontSize',8,'Position',[0.2352 0.6744 0.2612 0.1302]);

figure
step(1/s,cl/s,1/s-cl/s,1)
grid
% Ornamentation…
hold on
legend({'ramp input','system response','error'},...
  'FontSize',8,'Position',[0.1879 0.6934 0.2536 0.1302]);
annotation('textbox',...
  'Position',[0.2804 0.8786 0.4161 0.07857],...
  'BackgroundColor',[1 1 1],'FitHeightToText','off',...
  'String',{'Ramp response using ''step(cl/s)'''});

figure
step(1/s^2,cl/s^2,1/s^2-cl/s^2,2)
grid
% Ornamentation…
hold on
legend({'input of t^2','parabolic response','error'},...
  'FontSize',8,'Position',[0.2174 0.6863 0.2916 0.1349]);
annotation('textbox',...
  'Position',[0.3339 0.8778 0.3875 0.08413],...
  'BackgroundColor',[1 1 1],'String',{'Parabolic response by ''step(cl/s^2)'''},...
  'FitHeightToText','on');
```

Matlab code used to produce the comparative system responses to step, ramp and parabolic inputs.

```
s=tf('s'); g=(1+s/5)/(s^2*(1+s/12)); h=1+s/12

cl=feedback(30*g,h) % original system
cl2=feedback(30*g*h,1)  % system with h shifted to feed forward
cl3=feedback(30*g,1) % original feed forward with unity feedback

figure                  % STEP
hold on
step(cl,3)
step(cl2,3)
step(cl3,3)
grid
legend({'given system','GH in feed forward','original ff with unity fb'},...
  'FontSize',8,'Position',[0.2352 0.6744 0.2612 0.1302]);

figure                  % RAMP
hold on
step(cl/s,1)
step(cl2/s,1)
step(cl3/s,1)
grid
legend({'given system','GH in feed forward','original ff with unity fb'},...
  'FontSize',8,'Position',[0.1879 0.6934 0.2536 0.1302]);
annotation('textbox',...
  'Position',[0.2804 0.8786 0.4161 0.07857],...
  'BackgroundColor',[1 1 1],'FitHeightToText','off',...
  'String',{'Ramp response using ''step(cl/s)'''});

figure                  % PARABOLIC
hold on
step(cl/s^2,1)
step(cl2/s^2,1)
step(cl3/s^2,1)
grid
legend({'given system','GH in feed forward','original ff with unity fb'},...
  'FontSize',8,'Position',[0.2174 0.6863 0.2916 0.1349]);
annotation('textbox',...
  'Position',[0.3339 0.8778 0.3875 0.08413],...
  'BackgroundColor',[1 1 1],'String',{'Parabolic response by ''step(cl/s^2)'''},...
  'FitHeightToText','on');

figure                  % PARABOLIC ERRORS
hold on
step(1/s^2-cl/s^2,2.5)
step(1/s^2-cl2/s^2,2.5)
step(1/s^2-cl3/s^2,2.5)
grid
legend({'given system','GH in feed forward','original ff with unity fb'},...
  'FontSize',8,'Position',[0.2174 0.6863 0.2916 0.1349]);
annotation('textbox',...
  'Position',[0.3339 0.8778 0.3875 0.08413],...
  'BackgroundColor',[1 1 1],'String',{'errors for parabolic input by system'},...
  'FitHeightToText','on');
```