DEF 1
RL is the set of values of $s$ for which $1 + KL(s) = 0$ as $k = 0 \rightarrow 0$
The roots of this CE, $1 + KL(s) = 0$, are the poles of the CLTF of a control system

DEF 2
A point $s_0$ in the RL satisfies
$L(s) = -\frac{1}{K} \Rightarrow$ for $k \geq 0$, $L(s)$ is a negative real number

$\Rightarrow \angle L(s_0) = 180^\circ + 360^\circ (p - 1) \forall p \in \mathbb{N}^+$
RECALL \( \angle L(s_0) = \angle \{ \lambda_0 - \lambda_2 \} - \angle \{ \lambda_0 - \psi \} = \angle \psi - \angle \phi_i \)

\[ L(s) = \frac{s+1}{s(s+5)[(s+2)+4]} \]

\( s_0 = \frac{-1+2j}{1+4j} \)

\( p = 0, -5, -2 \pm 2j \)

\( \angle L(s_0) = 180^\circ + 360(l-1) = \frac{-1+2j+1}{(-1+2j+5)[(-1+2j+2)^2+4]} \)

\[ = \angle \psi - (\phi_1 + \phi_2 + \phi_3 + \phi_4) = 90^\circ - 116.6^\circ - 0^\circ - 76^\circ - 26.6^\circ = -128.2^\circ \]

\( \text{NO!} \)
**Rules for Determining Positive Root Locus**

(i.e. $K > 0$)

**R1**

Branches of the RL start at the $n$ poles of $L(s)$ and move towards its $m$ zeros.

- Recall $a(s) + Kb(s) = 0$

  \[
  \begin{align*}
  K = 0 & \quad a(s) = 0 \\
  K = \infty & \quad b(s) = 0 \\
  L(s) &= \frac{1}{K}
  \end{align*}
  \]

**R2**

The loci are on the real axis to the left of an odd # of poles + zeros.

- Contradiction

\[
\Rightarrow L(s) = 0 \quad \neq 180 - 360(k-1)
\]
For large $s \gg K$, $n-m$ branches of the loci are asymptotic to lines at angles $\phi_p$ radiating out of $\phi_0 = \alpha$ where

$$\alpha = \frac{\pi - \frac{2\pi}{n-m}}{2}$$

and

$$\phi_p = \frac{180^\circ + 360^\circ(p-1)}{n-m}$$

Again, as $K \to 0$, $L(s) = -\frac{1}{K} \to 0$

**Case 1: Rule 1**

**Case 2: $s \to \infty$** and $1 + K L(s) = 1 + \frac{K b(s)}{a(s)}$

$$|a(s)| > |b(s)| \Rightarrow L(s) \to 0$$

$$1 + K L(s) \approx 1 + K \frac{1}{(s-\alpha)^{n-m}} \to 0$$
\[ L(s) = \frac{1}{s[(s+4)^2 + 16]} = \frac{1}{s(s+(4+4j))(s+(-4-4j))} \quad \text{when} \quad m = 0 \quad n = 3 \]

\[ \alpha = \frac{\sum p_i - \sum z_i}{n-m} = \frac{-(4+j) - (-4-4j) - 0}{3} \]

\[ \alpha = \frac{-8}{3} \]

\[ \Phi_e = 180 + 360(l-1) \]

\[ = 60 + 120(l-1) \]

\[ = \begin{cases} 60^\circ \\ 180^\circ \\ 300^\circ \ (-60^\circ) \end{cases} \]
The angles of departure of a branch of the locus from a pole are given by:

\[ \phi_{\text{dep}} = \sum \phi_i - \sum \phi_i - 180^\circ - 360^\circ (l-1) \]

\[ \phi_{\text{arr}} = \sum \phi_i - \sum \phi_i + 180^\circ + 360^\circ (l-1) \]

**Multiplicity of the Pole/Zero**

**Imagine**

\[ s_0 \approx \text{Pole} \quad (\text{departure}) \]

\[ s_0 \approx \text{Zero} \quad (\text{arrival}) \]

\[ \phi_{\text{dep}} = "\text{What is seen left}" \]
The locus can have multiple roots at points on the locus to branches will approach a point of \( q \) roots at angles separated by

\[
180^\circ + \frac{360^\circ (\ell - 1)}{q}
\]

\[\text{Ex.}\]

\[L(s) = \frac{1}{5[(s+4)^2 + 16]} = \frac{1}{s(s+(4+4i))(s+(-4-4i))} \quad m = 0\]

\[n = 3\]
DEFINITION OF ROOT LOCUS FOR THE ABOVE CASES, i.e., with only $G(s)$: All points in the s-plane that satisfy the CE form the root locus, i.e., all points for which is $1 + K^*G(s) = 0$, which is the same as all points in the s-plane for which $K^*G(s) = -1$. Since the equation has the complex number $s$, this is a complex equation, i.e., its solution will be complex (see Appendix B in the text for a review of complex variables).

Now, $K^*G(s) = -1$ implies that both the following conditions should hold ‘simultaneously’:

$$|G(s)| = 1/K \text{ (MAGNITUDE CONDITION)}, \text{ and}$$

$$\angle G(s) = 180 + 360(l-1) \text{ for integer values of } l \text{ (ANGLE CONDITION)}$$

What is $\angle G(s)$ for $G(s) = (s+1)$?... and for $G(s) = 1/(s+1)$?

**Note:** Angle of the complex number $(s+a)$ can be computed by drawing a line from $s = -a$ (pole or zero as the case may be, depending on whether the term is in the numerator or denominator) to the point ‘s’ in the complex plane (angles are always measured from the positive real axis counterclockwise). Can you show why this is true?

We use this insight to get $\angle G(s)$ when $G(s)$ has multiple poles and zeros. How?

$$\angle G(s) = \angle(s + z_1)(s + z_2)\ldots /[(s + p_1)(s + p_2)\ldots].$$ Complex algebra shows that $\angle a/b = \angle a - \angle b$. So, we get

$$\angle G(s) = [\angle(s + z_1) + \angle(s + z_2) + \ldots] - [\angle(s + p_1) + \angle(s + p_2) + \ldots] = \sum \psi_i - \sum \phi_i$$

So, to find $\angle G(s)$ at any point $s = s^*$, here is what we can do:

Draw lines from that point $s^*$ to all the zeros and poles, and then measure the angles (from positive real axis direction) to the lines, and substitute in the formula above. In the text, the angles from the point $s^*$ to the $i^{th}$ zero of $G(s)$ is referred to as $\psi_i$, and the angle from point $s^*$ to the $i^{th}$ pole of $G(s)$ is referred to as $\phi_i$. 
So, all that needs to be done to determine whether a point \( s = s^* \) is on the root locus is to test what the phase of \( G(s) \) is \( 180 + 360(l-1) \) for integer values of \( l \), i.e., Using the above development, we check whether
\[
\sum \gamma_i - \sum \phi_i = 180^0 - 360^0 l.
\]
If the angle condition is satisfied, then it is on the root locus, and if it is not satisfied, then it is not on the root locus.

Note: Once the location of the root is determined, i.e., \( s^* \) is located that is on the root locus, the magnitude condition can then be used to determine the gain that will get the root at that location, since \( K = 1/|G(s)| \). This means that if this particular gain is used, the root will be at \( s^* \) (note that this is only one of the total number of roots, i.e., closed loop poles; there will be ‘n’ such roots on the n branches).

This approach is very popular among control engineers, and the rigor of drawing root locus is made much easier now using packages such as MATLAB!

Some insights from plot various root loci – the addition of a zero tends to pull the locus to the LEFT side (more stable!) while the addition of a pole does the opposite, i.e., it tends to ‘bend’ the branches to the right.

Is it not surprising that we can analyze the transient and steady state behavior of the closed loop system (\( KG(s)/(1+KG(s)) \)) using only the open loop TF \( G(s) \)?!!
Definition of Root Locus (GENERAL CASE)

**Definition I:** The root locus is the set of values of $s$ for which $1 + K*P(s) = 0$ is satisfied as the real parameter $K$ varies from 0 to $\infty$ (positive locus; for negative $K$ it is called a negative locus). This implies that

$$|P(s)| = -1/K \text{ (MAGNITUDE CONDITION)}.$$  

Note that typically, $K*P(s) = D(s)G(s)H(s)$ which is the loop transfer function of the system $L(s)$, i.e., we just pull ‘$K$’ out of $L(s)$. As noted earlier, roots on the locus are closed-loop poles of the system for the particular value of $K$.

**Definition II:** The root locus of $G(s)$ (note: $G(s)=K*P(s)$ in what follows below) is the set of points in the $s$-plane where the phase of $G(s)$ is $180^0$. What is phase of $G(s_1)$ for a point $s_1$ in the complex plane? From the definition of root locus, all points that satisfy the ANGLE CONDITION $\sum \psi_i - \sum \phi_i = 180^0 - 360^0 l$ lie on the root locus. Here $\psi_i$ the angle from the test point to an OL zero, and $\phi_i$ if the angle from the test point to an OL pole.
Steps to sketch the root locus. Proceed using the example \( G(s) = \frac{1}{s[(s + 4)^2 + 16]} \) as done in text.

Step 0: Let ‘n’ – number of poles \( p_i \), and ‘m’ – number of zeroes \( z_i \). Observations – (i) the number of branches of the root locus = the number of OL poles, and (ii) all branches start at OL poles and end at OL zeroes or at \( \infty \).

Step 1: Draw the axes of the s-plane to a suitable scale and enter an X on this plane for each pole of \( G(s) \) and an O for each zero of \( G(s) \).

Step 2: Find the real axis portions of the locus – locus exists on the real axis to the left of an odd number of real poles plus zeros.

Step 3: Draw \( n-m \) radial asymptotes centered at \( \alpha \) and with angles \( \phi_i \), where

\[
\alpha = \frac{\sum p_i - \sum z_i}{n - m}
\]

where \( p_i \) are the poles, and \( z_i \) are the zeros

\[
\phi_i = \frac{180^0 + 360^0(l - 1)}{n - m}
\]
Step 4: If complex poles exist, then compute departure angles from poles and arrival angles to zeros by searching for points around the pole or zero where the phase of G(s) is 180°, so that

\[ q_{\phi_{\text{dep}}} = \sum \psi_i - \sum \phi_i - 180^0 - 360^0 \]

\[ q_{\phi_{\text{arr}}} = \sum \phi_i - \sum \psi_i + 180^0 + 360^0 \]

where q is the order of the pole or zero, and l takes on q integer values such that the angles are between ± 180°. Refer to the text (pg. 242-244) for definitions of angles and details.

Step 5: Determine imaginary axis intersection – For this, set \( s = j\omega_0 \) in the CE, and compute the points where the locus crosses the imaginary axis for positive values of K. No solution will exist if it does not cross the imag axis.

Step 6: The equation has multiple roots at points on the locus (for which the angle of G(s) is 0° + 360°l) where \( d/ds (1/G) = 0 \).

Step 7: Fill in the locus using these calculations as guides, and the locus is complete.

ADDITIONAL EXAMPLES – MANY IN TEXT: DO ALL OF THESE PRIOR TO ATTEMPTING HW Pbs.
Another one for you to try - For the dc motor example case with K = \( \tau = 1 \), draw the root locus using (i) analytical, (ii) root locus rules, and (iii) the Matlab command.
5.3 SOME MATLAB COMMANDS

**Problem Statement:** Consider the open loop system which has a transfer function

\[ H(s) = \frac{Y(s)}{U(s)} = \frac{s + 7}{s(s + 5)(s + 15)(s + 20)} \]

Design a feedback controller for the system by using the root locus method. Design criteria are 5% overshoot and 1 second rise time.

Step 1. **rlocus** command: Enter the transfer function and the command to plot the root locus.

MATLAB File: rl.m

```matlab
num=tf([1 7],1);
den=tf([1 0],1)*tf([1 5],1)*tf([1 15],1)*tf([1 20],1);
Hs=num/den;
rlocus(Hs)
```
Step 2. **sgrid** command: The plot Fig 1 shows all possible closed-loop pole locations for a pure proportional controller. To determine which part of the locus is available, we can use the command `sgrid(Zeta, Wn)` to plot lines of constant damping ratio and natural frequency. For this problem, overshoot less than 5% and a rise time of 1 second correspond to a damping ratio zeta greater than 0.7 and a natural frequency Wn greater than 1.8. Add following MATLAB command to rl.m to get the plot in Fig 2.

```
zeta=0.7;
Wn=1.8;
sgrid(zeta,Wn)
```

Step 3. **rlocfind** command: To make the overshoot less than 5%, the poles have to be in between the two dotted lines, and to make the rise time shorter than 1 second, the poles have to be outside of the dotted semicircles. From the plot we see that there is a part of the root locus inside the desired region. So in this case we need only a proportional controller to move the poles to the desired region. We can use `rlocfind` command in Matlab to choose the desired poles on the locus.

```
[kd, poles]=rlocfind(Hs)
```

This gives you a cross-hair that you can use to locate any point on the root locus.

Step 4. Click on the plot the point where you want the closed loop pole to be. You may want to select the points indicated in the plot below to satisfy the design criteria. Matlab main window will calculate gain and pole locations when you click on the plot. Note that all the plots selected ("+") are at reasonable positions.

Step 5. Simulate the system and check if the specifications are satisfied. Note that there are 4 poles for the system.
MATLAB Root Locus Example

Objective?
1) Need $\zeta = 0.45$
2) Need to know range of $K$ for stability
How? Use root locus

What is $\zeta$ for the open loop system? Do an open loop simulation to determine this.

```matlab
num=[1 -4 20];
den=conv([1 2],[1 4]);
D=tf(num,den);
step(D)
```

(i) Plot the root locus.
```matlab
rlocus(D);
hold on
```

(ii) Plot the $\zeta = 0.45$ line.
```matlab
zeta=0.45;
sgrid(zeta)
```

(iii) Find Imag. axis crossing and gain.
```matlab
rlocf(D)
```

You will see a crosshair appear in the root locus plot. Move the crosshair to the Imag. crossing point and click the button. The corresponding gain $K$ will be shown on the screen.

(iv) For breakaway and breakin points, again use crosshairs as in step (iii).

(v) Using crosshairs, you will find that the system is stable for $0 < K < 1.5$. 

18
5.4 DESIGN USING DYNAMIC COMPENSATION

What are deficiencies of a proportional-only controller?

Assume that a customer comes to you and wants you the smart MU graduate to improve her system characteristics so that it has a settling time of 1.2 sec, and rise time of 0.26 sec.

Your first step is to determine a model for the system. Assume you did open loop tests and determined the OLTF to be 

\[ G(s) = \frac{1}{s(s+1)}. \]

What do you do next?
What are deficiencies of a proportional-only controller?
Assume that a customer comes to you and wants you the smart MU graduate to improve her system characteristics so that it has a settling time of 1.2 sec, and rise time of 0.26 sec.
Your first step is to determine a model for the system. Assume you did open loop tests and determined the OLT F to be $G(s) = \frac{1}{s(s+1)}$. What do you to next?

Add the LOWEST cost controller – proportional controller, $K$, and see if it can do the job. What do you think?
5.4 DESIGN USING DYNAMIC COMPENSATION
What are deficiencies of a proportional-only controller?
Assume that a customer comes to you and wants you the smart MU graduate to improve her system characteristics so
that it has a settling time of 1.2 sec, and rise time of 0.26 sec.
Your first step is to determine a model for the system. Assume you did open loop tests and determined the OLTF to be
\[ G(s) = \frac{1}{s(s+1)}. \]
What do you to next?

Add the LOWEST cost controller – proportional controller, \( K \), and see if it can do the job. What do you think?

Does it give you the design you want?
5.4 DESIGN USING DYNAMIC COMPENSATION

PI vs. LAG (APPROXIMATE PI) and PD vs. LEAD (APPROXIMATE PD)

- **PI compensation**
The primary reason for integral control is to reduce or eliminate constant steady-state error

\[ D(s) = K \frac{T_Is + 1}{T_Is} \]

PI feedback control cannot be realized using passive elements.

- **Lag compensation (approximate PI)**

\[ D(s) = K \frac{s + z}{s + p} \quad (z > p) \]

Lead compensation can be realized using passive elements.
- **PD compensation**
  Derivative feedback is used in conjunction with proportional feedback to improve transient characteristics.

  \[ D(s) = K(1 + T_D s) \]

  PD control amplifies noise.

- **PID compensation**

  \[ D(s) = K\left(1 + \frac{1}{T_I s} + T_D s\right) \]

  PID control eliminates steady state and transient errors.

- **Lead compensation (approximate PD)**

  \[ D(s) = K\frac{s + z}{s + p} \quad (z < p) \]

  Lead compensation does not amplify noise due to the presence of the additional pole.

- **Lead- Lag compensation**

  \[ D(s) = K\left(\frac{s + z_1}{s + p_1}\right)\left(\frac{s + z_2}{s + p_2}\right) \]

  Lead-lag compensation approximates PID controller.
RECALL -- for the Unity Feedback case, if \( D(s)G(s) \) has \( n \) free \( s \) in the denominator, then the system is of type \( n \)

E.g., \( D(s)G(s) = 1/[s(s+1)] \) is a type 1 system and will have a zero ss error for a step input and a constant steady state error for a ramp input.

**How do we calculate ss error for such systems?**

Note that for UFB systems only, the error constants are as follows:

- Position error constant \( K_p = \lim_{s \to 0} D(s)G(s) \);
- Velocity error constant \( K_v = \lim_{s \to 0} sD(s)G(s) \); and
- Acceleration error constant \( K_a = \lim_{s \to 0} s^2D(s)G(s) \) [these terms are used in industry]

**Types?** Type 0 – the system has a constant error to a step; Type 1 – system has a constant error to a ramp input,......and so on.

So, for type 0 systems, ss error to a step input \( r(t) = 1(t) \) is \( e_{ss} = \frac{1}{1 + K_p} \)

For type 1 systems, ss error to a ramp input \( r(t) = t \cdot 1(t) \) is \( e_{ss} = \frac{1}{K_v} \)

For type 2 systems, ss error to parabolic input \( r(t) = t^2 \cdot 1(t) \) is \( e_{ss} = \frac{1}{K_a} \)

Find the ss error to the appropriate inputs for the following systems:

\( H(s) = 1/(s+5) \);  \( H(s) = 1/[s(s+7)] \);  \( H(s) = (s+4)/(s(s+10)) \);  \( H(s) = (s+2)/(s(s+3)(s+12)) \)
5.5 LEAD COMPENSATOR DESIGN

Example: Given an uncompensated system as follows (see page 269)

\[ R(s) + E(s) \rightarrow \frac{K}{s(s + 1)} \rightarrow C(s) \]

Objectives: Customer requires settling time \(< 1.31 \text{ sec}\), and rise time \(< 0.26 \text{ sec}\) (note: no ss specs).
Convert specs to freq. domain specs: Need \(\zeta \geq 0.5\), \(\omega_n \geq 7 \text{ rad/s}\)

MATLAB Solution:

Step 1 Input the system and specs
\[ G=tf([1],[1 1 0]); \text{zetan}=0.5; \text{wn}=7; \]

Step 2 Assume the zero \(z\) of the lead compensator
\[ z=input('Input the zero of the lead compensator: '); \]
In general the zero is placed in the neighborhood of the CL \(\omega_n\), as determined by rise-time or settling-time requirements.

Step 3 Assume the pole \(p\) of the lead compensator
\[ p=input('Input the pole of the lead compensator: '); \]
Usually, the pole is located at a distance 3 to 20 times the value of the zero location.
Note: given the zero location, the pole location that will allow the root locus to pass through a desired point can be determined exactly by geometry.
Step 4 Plot the root locus of the compensated system DG

\[ D = \text{tf}([1 \ z], [1 \ p]); \]
\[ DG = D * G; \]
\[ rlocus(DG); \]

Step 5 Draw the \( \omega_n = 7 \) arc and \( \zeta = 0.5 \) line on the s-plane

\[ \text{sgrid(wn, wn)} \]
\[ \text{pause;} \]

PAUSE causes a procedure to stop and wait for the user to
strike any key before continuing.

Step 6 Find the gain \( K \)

\[ [K, \text{poles}] = \text{rlocfind}(DG) \]

A crosshair will show up on the plot. Move the crosshair to the place where you want to put the dominant poles and click the button. The corresponding gain value will appear on the screen.

Step 7 Simulate the system to see if all requirements have been met.

\[ T = \text{feedback}(K * DG, 1); \]
\[ \text{step}(T); \]

Step 8 Redesign if the simulation shows that requirements have not been met, i.e., try different \( z \) and \( p \) and complete the design.
The consolidated code for the design is listed below:

```matlab
G = tf([1],[1 1 0]);
zeta = 0.5; wn = 7;
z=input('Input the zero of the lead compensator: ');
p=input('Input the pole of the lead compensator: ');
D=tf([1 z],[1,p]);
DG = D*G;
rlocus(DG);
sgrid(zeta,wn);
pause;
[k,poles]=rlocfind(DG)
T=feedback(k*DG,1);
figure(2);
step(T);
```
5.6 LAG COMPENSATOR DESIGN

Consider a system which is lead compensated (i.e., after the previous design) as shown below:

\[ G'(s) = 127 \frac{(s + 5.4)}{(s + 20)} \frac{1}{s(s + 1)}. \]

Objectives: Although the previous design fixed the transient characteristics, we still need to fix the steady state characteristics. Assume that, based on customer specs, we need to reduce the steady state error by a factor of 3.

MATLAB Solution:

Step 1 Input the system

\[ G = \text{tf}([127 \ 685.8], [1 \ 21 \ 20 \ 0]); \]

Step 2 Draw the \( \omega_n = 7 \) arc and \( \zeta = 0.5 \) line on the s-plane

\[ \text{sgrid(zeta, wn)} \]

Step 3 Assume the pole P1 of the lag compensator

\[ \text{P1=input('Input the pole of the lag compensator: ')}; \]

Usually, for the lag compensator the pole is picked so that it is very close to the origin, e.g., \( P1 = 0.01 \)

Step 4 Assume the zero Z1 of the lag compensator

\[ \text{Z1=input('Input the zero of the lag compensator: ')}; \]

Place the zero such that the \( Z1/P1 \) ratio = Factor by which steady state error should be reduced which in this case is 3. Therefore if \( P1 = 0.01 \) then \( Z1 = 0.03 \)
Step 5 Plot the root locus of the compensated system DG

\[
D=\text{tf}([1 \ z1],[1 \ P1]);
\]

\[
DG=D*G;
\]

\[
rlocus(DG);
\]

\[
pause;
\]

PAUSE causes a procedure to stop and wait for the user to strike any key before continuing.

Step 6 Simulate the system to see if all requirements have been met.

\[
T=\text{feedback}(DG,1);
\]

\[
figure(2);
\]

\[
step(T);
\]

Step 8 Redesign if the simulation shows that requirements have not been met, i.e., iterate with different z and p values. For instance, if the transient response has changed then repeat the above by selecting z and p closer to origin.
Code for the entire design is given below:

```matlab
G = tf([127 685.8], [1 21 20 0]);
zeta = 0.5; wn = 7;
pl=input('Input the pole of the lag compensator: ');
zl=input('Input the zero of the lag compensator: ');
D=tf([1 zl], [1,pl]);
DG = D*G
rlocus(DG);
sgrid(0.5,7);
pause;
T=feedback(DG,1);
figure(2);
step(T);
```
5.7 LEAD-LAG COMPENSATOR DESIGN

Now consider solving the entire problem as one design, i.e., take the original System, and design a lead-lag compensator to achieve the objectives below. \( R(s) + \frac{E(s)}{K \frac{s(s+1)}{s}} \rightarrow C(s) \)

Objectives: Need settling time \(< 1.31\) sec, and rise time \(< 0.26\) sec + ss error. Convert specs to: Need \( \zeta \geq 0.5, \omega_n \geq 7\) rad/s, plus reduce the steady state error.

MATLAB Solution:

Step 1 Input the system and specs

\[ G = \text{tf}([1],[1 1 0]); \quad \text{zeta} = 0.5; \quad \text{wn} = 7; \]

Step 2 Draw the \( \omega_n = 7 \) arc and \( \zeta = 0.5 \) line on the s-plane

\[ \text{sgrid(zeta,wn)} \]

Step 3 Assume the zero \( z \) of the lead compensator

\[ z = \text{input}('\text{Input the zero of the lead compensator:}') \]

In general the zero is placed in the neighborhood of the CL wn, as determined by rise-time or settling-time requirements.
Step 4 Assume the pole P of the lead compensator

```
P=input('Input the pole of the lead compensator: ');
```

Usually, the pole is located at a distance 3 to 20 times the value of the zero location.
Note: given the zero location, the pole location that will allow the root locus to pass through a desired point can be determined exactly by geometry.

Step 5 Find the transfer function of the compensated system DG

```
D=tf([1 z],[1 P]);
DG=D*G;
rlocus(DG);
pause;

Find Gain
[K,poles]=rlocfind(DG)
```

A crosshair will show up on the plot. Move the crosshair to the place where you want to put the dominant poles and click the button. The corresponding gain value will appear on the screen. Now the OL system becomes as follows:

```
KD = K*DG
```

Now simulate the system and determine the steady state error. Determine the factor by which you want to reduce the steady state error.

Step 6 Assume the pole P of the lag compensator

```
Pl=input('Input the pole of the lag compensator: ');
```

Usually, for the lag compensator the pole must be located at a point that is close to the origin typically 0.01
Step 7 Now Assume the zero $Z_1$ of the lag compensator

\[ Z_1 = \text{input('Input the zero of the lag compensator: ')}; \]

Zero is selected so that the $Z_1/P_1$ ratio is equal to the factor by which steady state error needs to be reduced.

Step 8 Find the transfer function of the compensated system $DGL$

\[ L = \text{tf([1 z1],[1 P1])}; \]
\[ KDGL = KDG \times L; \]

Step 9 Plot the root locus of the compensated system $KDGL$

\[ \text{rlocus(KDGL)}; \]
\[ \text{pause}; \]

PAUSE causes a procedure to stop and wait for the user to strike any key before continuing.
Step 10 Simulate the system to see if all requirements have been met.

```matlab
T=feedback(KDGL,1);
figure(2);
step(T);
```

Step 11 Redesign if the simulation shows that requirements have not been met, i.e., try different z and p and complete the design. For instance if the transient response has changed then repeat the above by picking value which is still lower.

The Entire code put together is given below:

```matlab
G = tf([1],[1 1 0]);
zeta = 0.5; wn = 7;
z=input('Input the zero of the lead compensator: ');
p=input('Input the pole of the lead compensator: ');
D=tf([1 z],[1,p]);
DG = D*G;
rlocus(DG);
pause;
[k,poles]=rlocfind(DG);
KD = k*DG
pl=input('Input the pole of the lag compensator: ');
zl=input('Input the zero of the lag compensator: ');
L=tf([1 zl],[1,pl]);
KDGL=KD*L;
rlocus(KDGL);
pause;
T=feedback(KDGL,1);
figure(2);
step(T);
```
5.9 IMPLEMENTATION OF LEAD, LAG AND NOTCH FILTERS: PASSIVE NETWORKS

Note that lead and lag compensation have the general transfer function form of

\[ D(s) = K \frac{s + z}{s + p} \]  \hspace{2cm} (5.1)

Where \( K \) = Gain, \( z = \) zero and \( p = \) pole. Also, the general feedback system with compensation is shown at Figure 1. Consider lead, lag and notch network implementations using passive networks.

**Lead network**

The transfer function of lead circuit shown at Figure 2 is given at below.

\[ \frac{u}{\varepsilon}(s) = \frac{CR_1 R_2 s + R_2}{CR_1 R_2 s + R_2 + R_1} \]  \hspace{2cm} (5.2)

Therefore, the value of gain \( K \), zero \( z \) and pole \( p \) can be determined by setting Equation (5.1) and Equation (5.2) equal to each other.

\[ K = CR_1 R_2 \quad z = \frac{1}{CR_1} \quad p = \frac{R_1 + R_2}{CR_1 R_2} \]
**Lag network**

The transfer function of lead circuit shown at Figure 3 is given at below.

\[
\frac{u}{e}(s) = \frac{CR_2s + 1}{C(R_1 + R_2)s + 1}
\]  

(5.3)

Therefore, the value of gain \(K\), zero \(z\) and pole \(p\) can be determined by setting Equation (5.1) and Equation (5.3) equal to each other.

\[
K = \frac{R_2}{R_1 + R_2}, \quad z = \frac{1}{CR_2}, \quad p = \frac{1}{C(R_1 + R_2)}
\]

**Notch network**

The transfer function of notch circuit shown at Figure 4 is given at below.

\[
\frac{u}{e}(s) = A + B + 1
\]

\[
A = \frac{-(3R^2s^2C_2C_3 + 2R(2C_2 + C_3)s + 2)}{R^3C_1C_2C_3s^5 + R^2(2C_1C_2 + C_1C_3 + 3C_2C_3)s^2 + R(C_1 + 5C_2 + 2C_3)s + 2}
\]

\[
B = \frac{-(C_1C_3R^2s^2 - 2)}{R^3C_1C_2C_3s^5 + R^2(2C_1C_2 + C_1C_3 + 3C_2C_3)s^2 + R(C_1 + 5C_2 + 2C_3)s + 2}
\]
5.10 IMPLEMENTATION OF LEAD, LAG AND NOTCH FILTERS: ACTIVE NETWORKS

Lead network
Lead compensation can be implemented by using analog electronics. The common way is to use an operational amplifier. This circuit is shown at Figure 5.

\[ D(s) = -K_D \frac{T_1 s + 1}{\alpha T_1 s + 1} \]

\[ K_D = \frac{R_F}{R_1 + R_2} = 1 \quad \text{if} \quad R_F = R_1 + R_2 \]

![Figure 5. Lead Circuit](image)

Notice that the zero is located as \( z = -1/T_1 \) and the pole is located at \( p = -1/\alpha T_1 \), so the parameter \( \alpha \) sets the separation distance between pole and zero, typically a factor of 3 to 20.

\[ T_1 = R_1 C \quad \alpha = \frac{R_2}{R_1 + R_2} \]
### 5.11 OVERVIEW OF CASCADE COMPENSATOR TECHNIQUES

<table>
<thead>
<tr>
<th>$D(z)$</th>
<th>TYPE</th>
<th>ASSUME</th>
<th>HOW?</th>
<th>REMARKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K(1 + T_D s)$</td>
<td>PD</td>
<td>*Steady state error is OK. *Active Network</td>
<td>*Usually increasing $\zeta_\omega$.</td>
<td>*Root locus is changed to go through desired C.L. pole locations. Pole-zero cancellation usually occurs. *Pure D not physically realizable. *Amplify high freq. noise.</td>
</tr>
</tbody>
</table>
| $K \left( \frac{T_s + 1}{\alpha T s + 1} \right) \alpha < 1$ | Lead | *Steady state error is OK, but transient response bad. *Passive Network. | **Root Locus Method**  
*Select desired C.L. pole location.  
*Select compensator zero $-1/T$ and find the pole $-1/\alpha T$ by geometry. In general, the zero is placed in the neighborhood of the C.L. $\omega_n$ and the pole is located at the distance 3 to 20 times the value of the zero location.  
*Justify second order approximation (closest poles are 4 times away), consider effect of zeros.  
**Frequency Response Method**  
**CASE 1:** Design for low-freq gain ($\alpha$, specific.)  
Determine O.L. gain $K$ to meet error constant $(K_p, K_v$ and so forth)  
**CASE 2:** Design for C.L. bandwidth  
*Determine O.L. $\omega_n$ to be $\omega_n/2$.  
*Evaluate PM of the uncompst sys using $K$.  
*Allow for 5° to 12° extra margin, and $\phi_{\text{max}} = \alpha$.  
*Determine $\alpha$ from Eq. 6.37 or Fig. 6.52.  
*Set new gain $\omega_n$ at $\omega_n_{\text{new}}$ and determine the corner freq. by trial-and-error  
*Draw the compts freq. respectively, check PM, and iterate on the compensator if necessary.  
*Iterate on the design. Add an additional lead compensator if necessary.  
| Pole close to zero implies compensation not effective, pole too far implies magnification of noise leading to overheating of actuator by noise energy. |
| $K \left( \frac{T_s + 1}{T_s + 1} \right)$ | PI | *Transient response is OK, but ss is bad. *Active network | *Increasing system type by 1/2. | $T_D$ + 1 ensures that transients does not change if $1/T_D$ is close to the origin (recall root locus design). |
| $K \left( \frac{T_s + 1}{\alpha T s + 1} \right) \alpha > 1$ | Lag | *Transient is OK, but SS error too high. *Passive network. | **Root Locus Method**  
*Find by what factor $K$ (static error constant) needs to be increased. Select $\alpha$ to be that factor.  
*Select both poles & zeros very close so as not to change the C.L. root location.  
**Frequency Response Method**  
*Determine O.L. gain $K$ to meet PM reqmnt.  
*Draw Bode plot of the uncompst sys from $\omega_n = 1$, and evaluate low-freq gain.  
*Determine $\alpha$ to meet the low-freq gain.  
*Choose corner freq $\omega = 1/T$.  
*The other corner freq is then $\omega = 1/\alpha T$.  
*Iterate on the design. | $\alpha$ can be increased together while ensuring that the compensator pole and zero are close only by selecting them close to origin.  
*The decay of the transient due to the C.L. pole near origin may be slow. So the p-z combination should be placed at as high a freq. as possible.  
*The TF from a plant disturbance to error won't have zero and so disturbance transients can be very large. |
| $K \left( \frac{1}{T_i s} + \frac{1}{T_D s} \right)$ | PID | *Need improvement of both transient and SS responses. *Active network. | *SS more important, do PD first then PI.  
*TR more important, do PI first then PD.  
*Check for validity of 2nd order approximation. | |
| $K \left( \frac{T_s + 1}{\alpha T s + 1} \right) \left( \frac{T_s + 1}{\alpha T s + 1} \right)$ | Lead-Lag | *Need improvement of both transient and SS responses. *Active network. | *SS more important, do Lead first then Lag.  
*TR more important, do Lag first then Lead.  
*Check for validity of 2nd order approximation. | |