Electronic Controllers using Op-Amp Integrated Circuits

- This script describes the different types of controllers that can be designed with electronic passive components such as resistors and capacitors along with active components, operational amplifiers.
- Each of this controllers will be described in a schematic form as well as their corresponding transfer functions
- After learning about the individual components, we will use them as building blocks to design more complicated controllers, such as a PID controller.
- Lastly, an arbitrary transfer function from a DC motor will be presented and the appropriate PID controller will be designed using MATLAB and then the equivalent Op-Amp circuit will be defined.

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

The Building Blocks of Electronically Designed Controllers

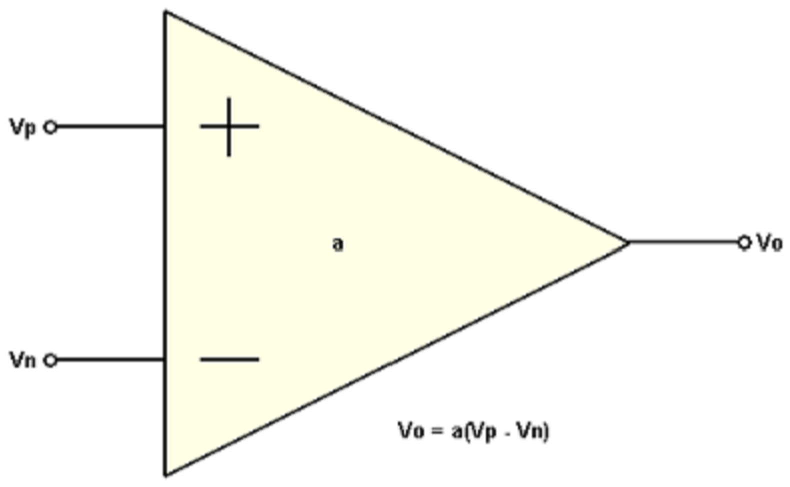
- Circuits are designed using the following concepts:
- *Current I:* The time rate of change of electrical charge, measured in Amperes(A)
- *Voltage V:* The potential difference or the energy needed to move a unit charge from point a to point b, measured in Volts (V)
- *Resistance R:* Passive circuit element that resists the flow of electric current, measured in Ohms (Ω)
- *Capacitance C:* The ratio of the charge on one plate of a capacitor to the voltage difference between the two plates, measured in Farads (F)
- *Impedance Z:* The complex ratio of the phasor voltage **V** to the phasor current **I**, also measured in Ohms (Ω) - where $R = \text{Re } Z$ is the *resistance* and $X = \text{Im } Z$ is the *reactance,* which is associated with the imaginary component j for inductance and $-j$ for capacitance.
- *Operation Amplifier:* An active circuit element designed to perform mathematical operations such as addition, subtraction, multiplication, division, differentiation, and integration depending on the arrangement of passive circuit elements.
- We use analysis methods such as Kirchoff's Voltage and Current Law in order to derive equations for the voltage drop across various components or the current entering or leaving a node in the circuit.
- Using these concepts, we can derive complex electrically designed controllers

Op-Amp Description

- An operational amplifier is a modern implementation of Harold Black's feedback amplifier and is a universal component that is widely used for instrumentation, control, and communication
- Very versatile as any linear system can be implemented by combining operational amplifiers with resistors and capacitors, most notely, analog proportional-integral-derivative controllers

```
clc; close all; clear;
f = imread('OpAmp.png'); imshow(f), title('Non-Inverting Op-Amp')
```

Non-Inverting Op-Amp



The Open-Loop Transfer Function can be described as:

$$A(s) = \frac{A_0}{(1 + s/\omega_1)(1 + s/\omega_2)}$$

The value of A_0 is ideally ∞ but typically within the range of $10^6 - 10^8$ in most practical Op Amp ICs

Transfer Function has two real poles at

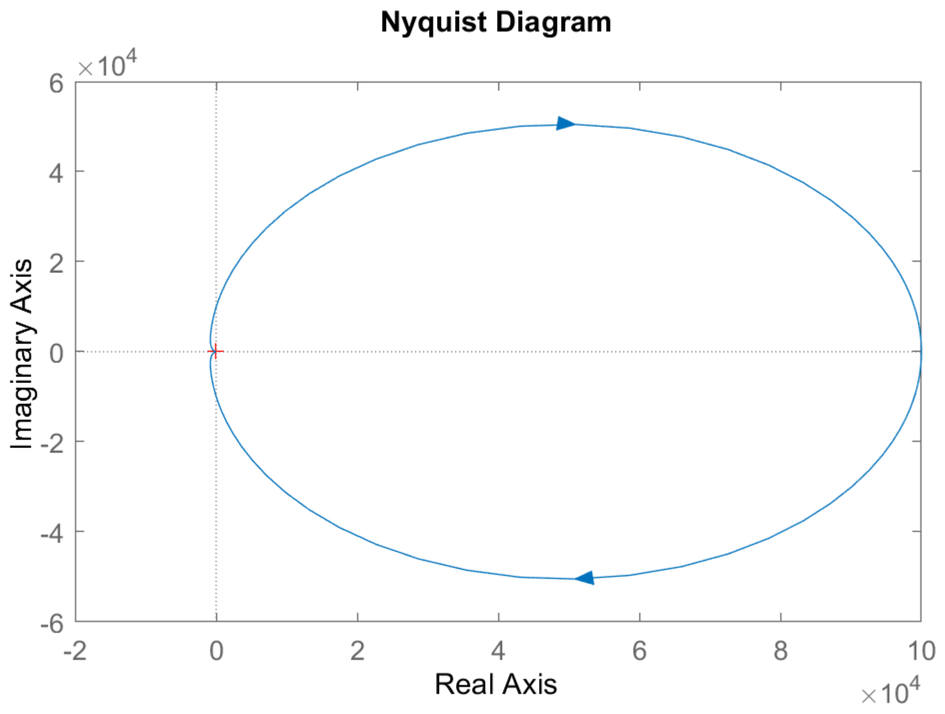
$\omega_1 = 10,000$ and $\omega_2 = 1,000,000$

Assume Values and Create Transfer Function

```
A0 = 1e5; w1 = 1e4; w2 = 1e6;
s = tf('s');
A_open = A0/(1 + s/w1)/(1 + s/w2)
```

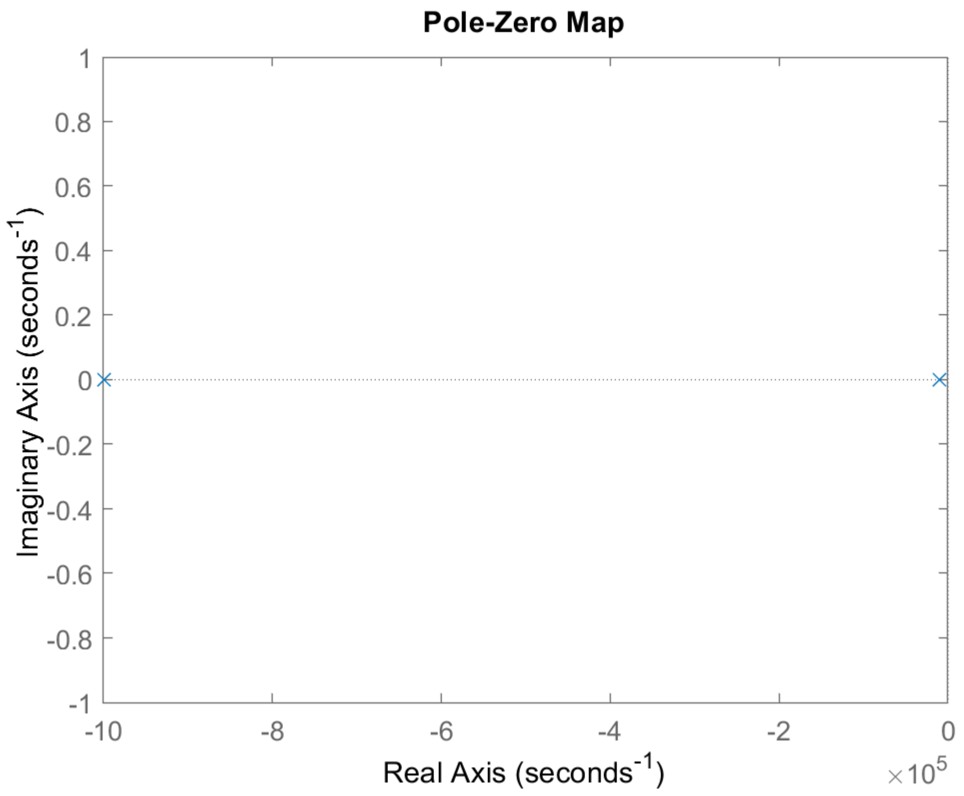
```
A_open =
 
          1e15
  ----------------------
  s^2 + 1.01e06 s + 1e10
 
Continuous-time transfer function.
```

```
nyquist(A_open)
```



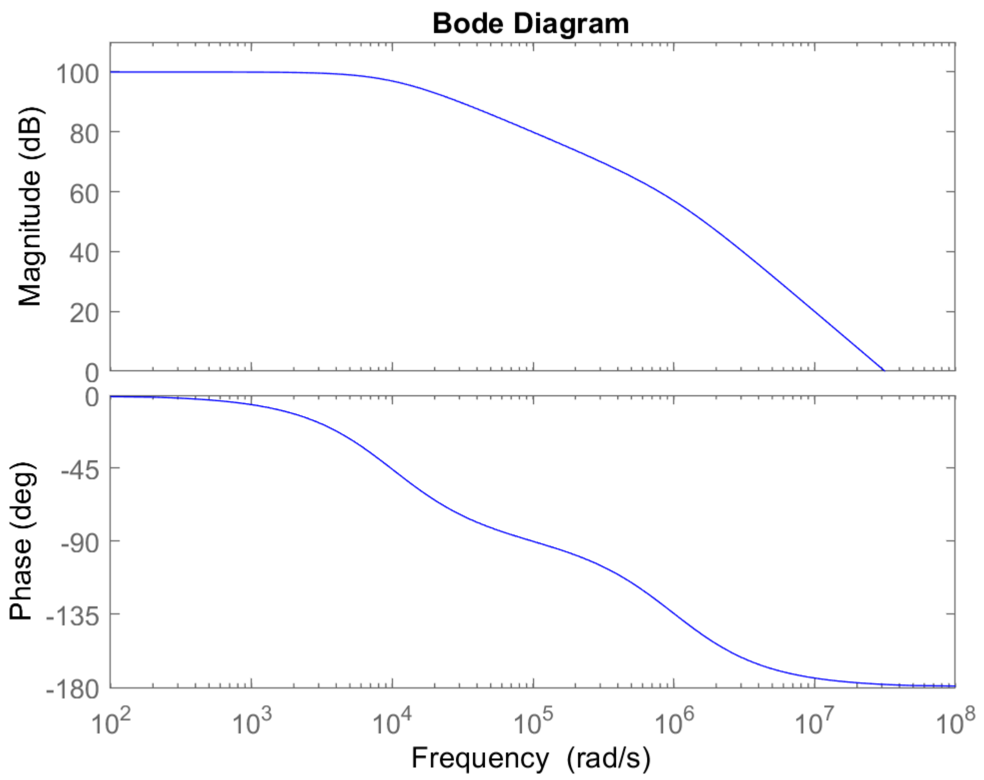
Plot Pole-Zero Map

```
figure, pzmap(A_open)
```



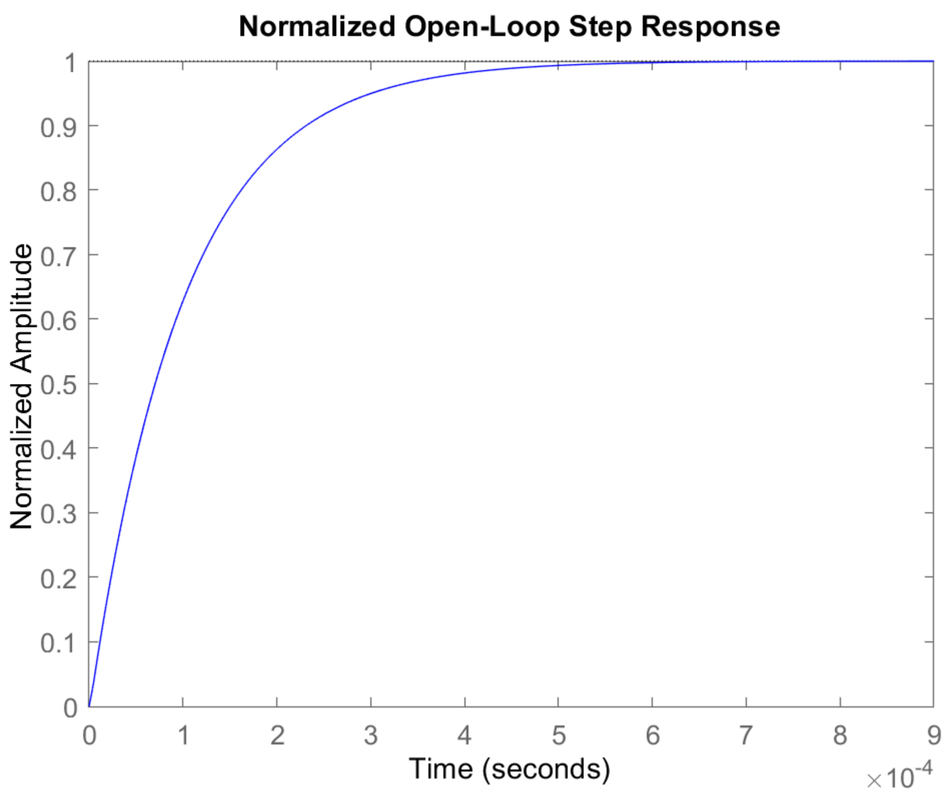
Display Bode Plot for Transfer Function

```
h = bodeplot(A_open,'b');
setoptions(h,'FreqUnits','rad/s','MagUnits','dB','PhaseUnits','deg',...
    'YLimMode','Manual','YLim',{[0,110],[-180,0]});
```



View the Normalized Open-Loop Step Response of System

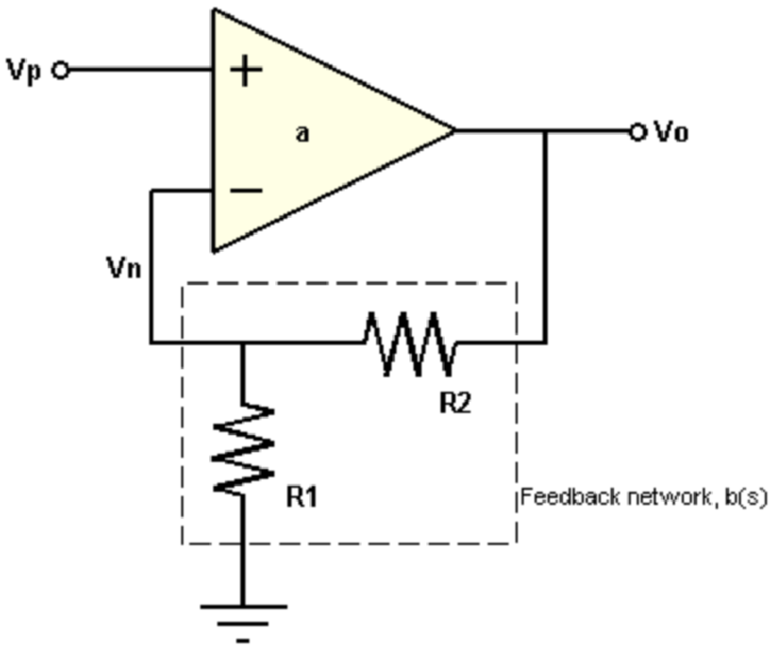
```
A_norm = A_open / dcgain(A_open);
stepplot(A_norm,'b')

title('Normalized Open-Loop Step Response');
ylabel('Normalized Amplitude');
```



Feedback Amplifier

```
f = imread('FeedbackAmplifier.png'); imshow(f), title('Feedback Amplifier')
```

Feedback Amplifier



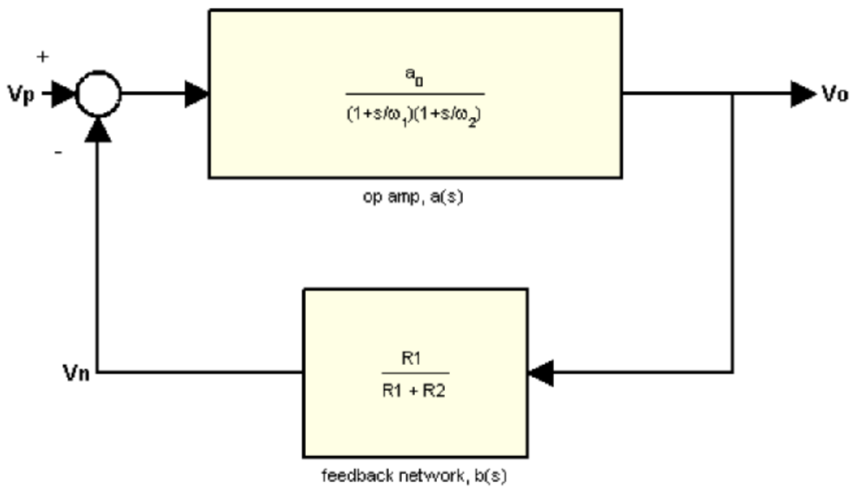
The Feedback network, $B(s)$ is voltage divider between input V_n and output V_o . Solving for this ratio gives the transfer function:

$$B = \frac{V_o}{V_{in}} = \frac{R_1}{R_1 + R_2}$$

The Closed-Loop Diagram:

```
f = imread('ClosedLoopGain.png'); imshow(f), title('Closed-Loop Block Diagram')
```

Closed-Loop Block Diagram



The ratio of the closed-loop gain would then be

$$A = \frac{V_o}{V_p} = \frac{A}{1 + AB}$$

However, if the product, AB, is extremely large, then the gain can be approximated too

$$A = \frac{1}{B}$$

Design Amplifier with DC Gain of 10 with $R_1 = 10k\Omega$. Solve for R_2 :

```
A0 = 10; B = 1/A0;
R1 = 1e4; R2 = R1 * (1/B - 1)
```

```
R2 = 90000
```

Thus, $R_2 = 90k\Omega$ would produce the desired Gain of $A = 10V/V$

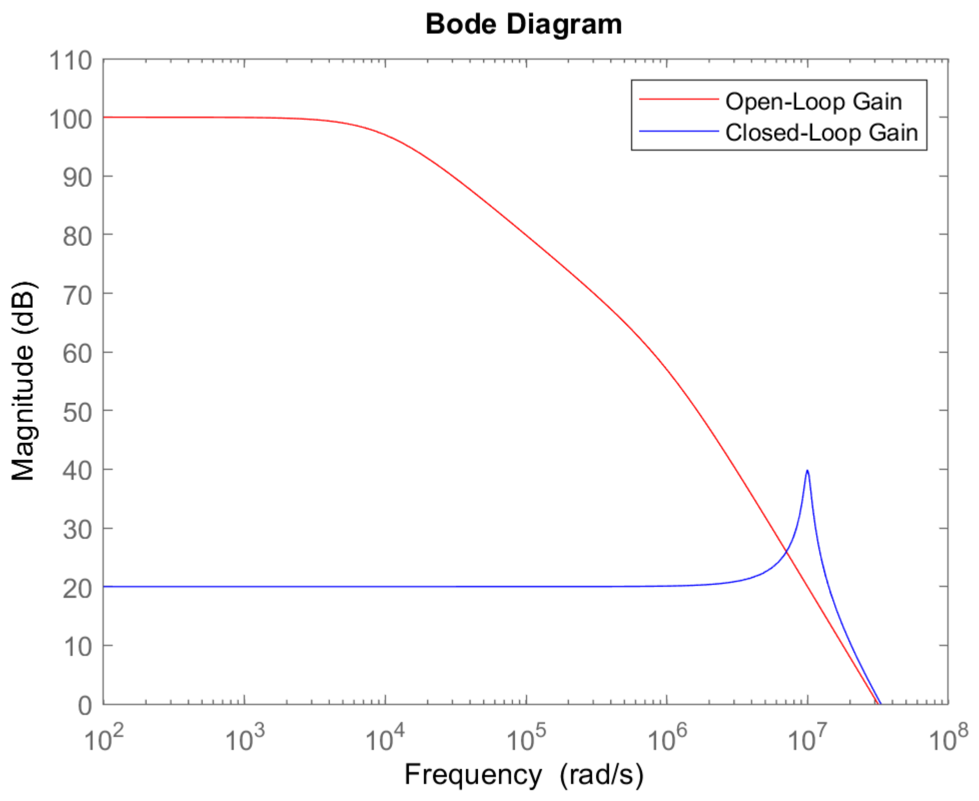
Construct Closed-Loop System using Feedback() Function

```
A_closed = feedback(A_open,B)
```

```
A_closed =
 
            1e15
  ----------------------
  s^2 + 1.01e06 s + 1e14
```

```
Continuous-time transfer function.
```

```
bodemag(A_open,'r',A_closed,'b');
legend('Open-Loop Gain','Closed-Loop Gain')
ylim([0,110]);
```



Observations:

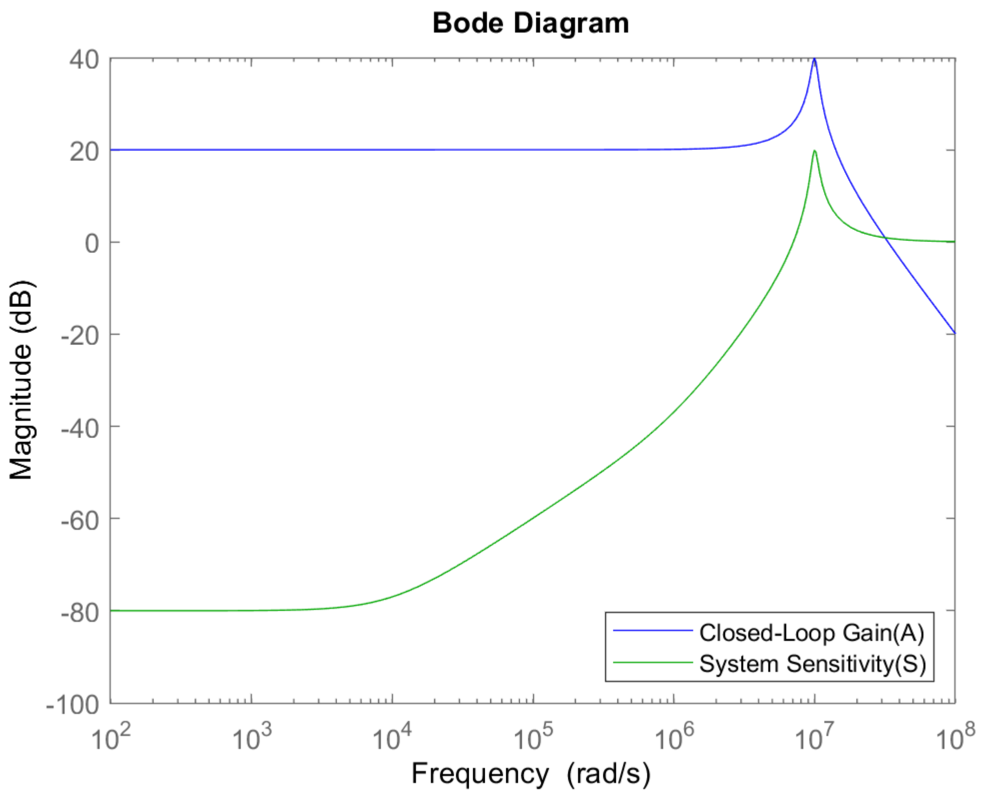
- We can see that the Closed-Loop Gain is much lower for Low Frequencies, but the bandwidth has increased as a trade-off
- This Gain vs Bandwidth trade-off presents a powerful tool when designing feedback circuits
- Presents a drastic reduction in sensitivity that allows to make precise systems from uncertain components
- Feedback can be used to trade high gain and low robustness for low gain and high robustness

Sensitivity of the Gain to Variations

```
% Define Loop gain - total gain a signal experiences traveling around the
% loop
L = A_open*B;
S = 1/(1 + L);
S = feedback(1,L)
```

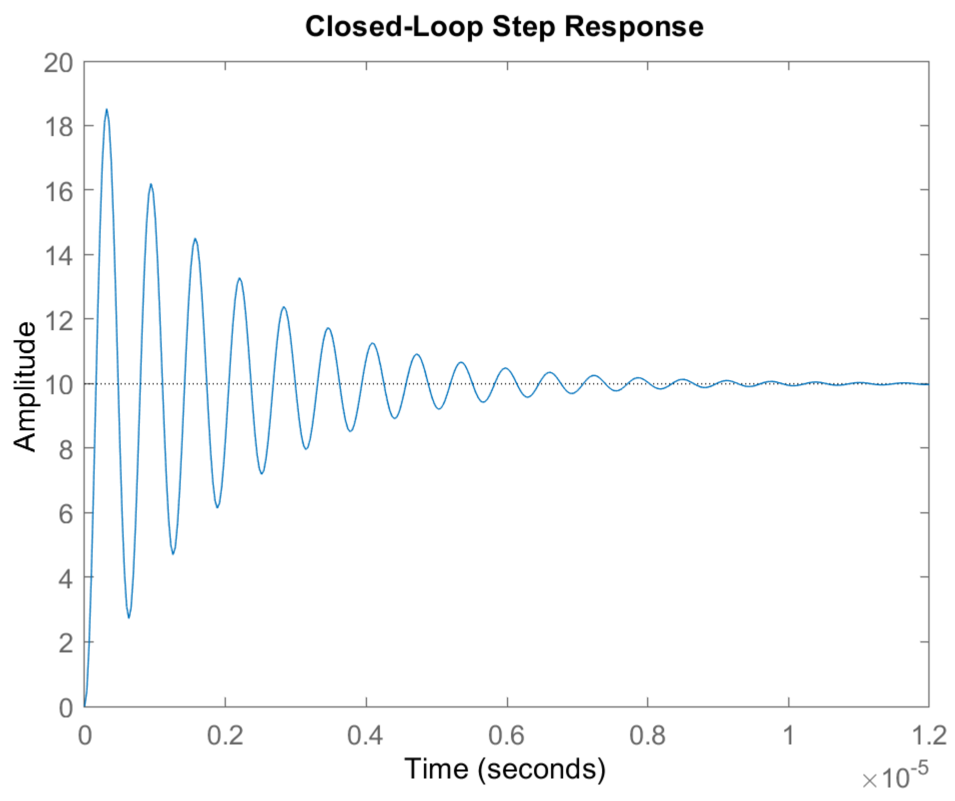
```
S =

  s^2 + 1.01e06 s + 1e10
  ----------------------
  s^2 + 1.01e06 s + 1e14

Continuous-time transfer function.
```

```
bodemag(A_closed,'b',S,'g')
legend('Closed-Loop Gain(A)', 'System Sensitivity(S)','Location','SouthEast')
```



The very small low-frequency sensitivity (-80 dB) indicates a design whose closed-loop gain suffers minimally from open-loop gain variation

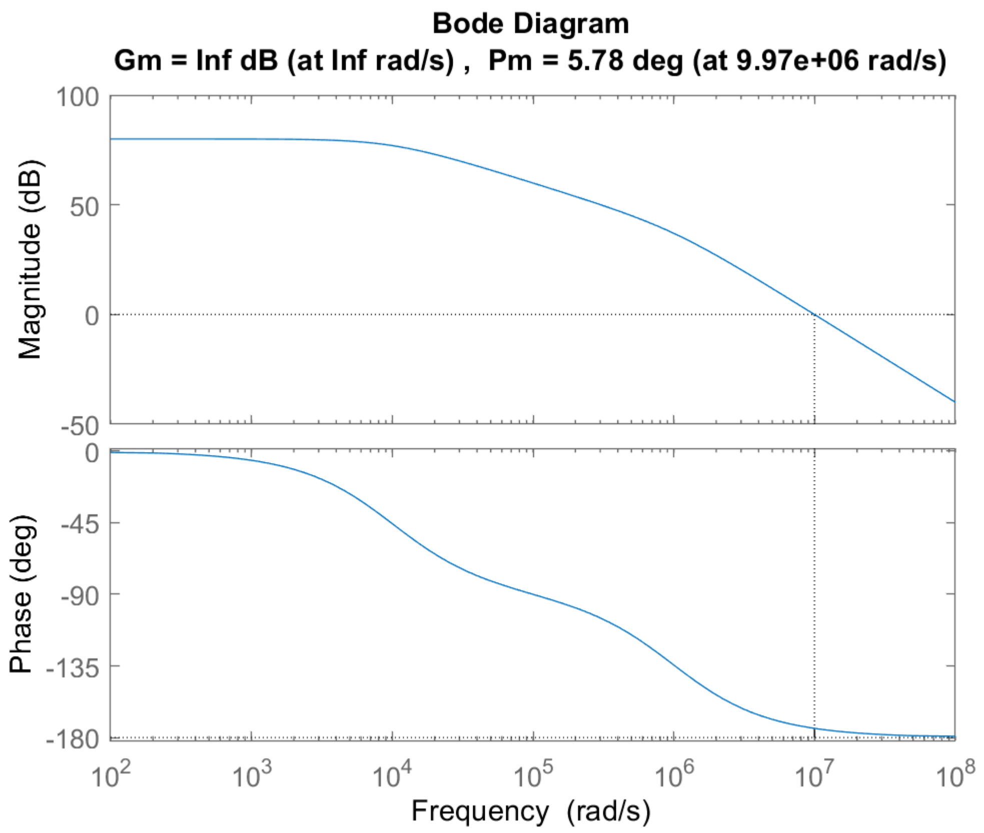
```
stepplot(A_closed), title('Closed-Loop Step Response')
```



Note that the use of feedback has greatly reduced the settling time (by about 98%). However, the step response now displays a large amount of ringing, indicating poor stability margin.

You can analyze the stability margin by plotting the loop gain, L(s)

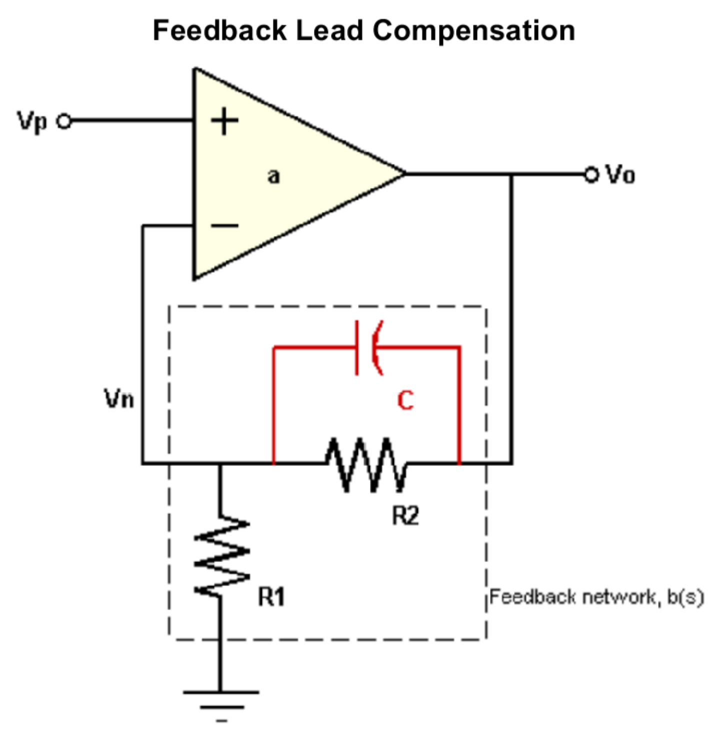
```
margin(L)
```



The resulting plot indicates a phase margin of less than 6 degrees. You will need to compensate this amplifier in order to raise the phase margin to an acceptable level (generally 45 deg or more), thus reducing excessive overshoot and ringing.

Feedback Lead Compensation

```
f = imread('FeedbackCompensation.png'); imshow(f), title('Feedback Lead Compensation')
```

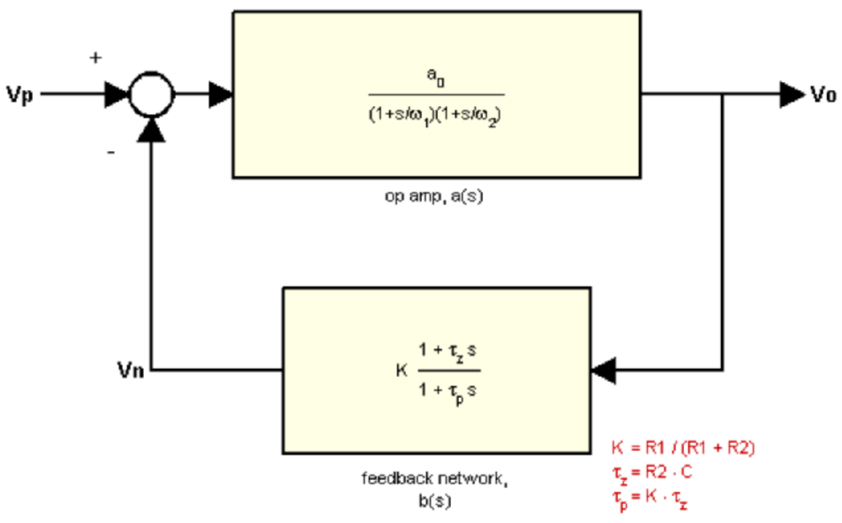


A commonly used method of compensation in this type of circuit is "feedback lead compensation". This technique modifies b(s) by adding a capacitor, C, in parallel with the feedback resistor, R2

The capacitor value is chosen so as to introduce a phase lead to b(s) near the crossover frequency, thus increasing the amplifier's phase margin.

```
f = imread('FeedbackCompensationBlockDiagram.png'); imshow(f), title('Closed-Loop Diagram')
```

Closed-Loop Diagram


+
Vp
-
$\frac{a_0}{(1+s/\omega_1)(1+s/\omega_2)}$
op amp, a(s)
Vo
Vn
$K \frac{1+\tau_z s}{1+\tau_p s}$
feedback network, b(s)
$K = R1 / (R1 + R2)$
$\tau_z = R2 \cdot C$
$\tau_p = K \cdot \tau_z$

You can approximate a value for C by placing the zero of b(s) at the 0dB crossover frequency of L(s):

```
[Gm,Pm,Wcg,Wcp] = margin(L);
C = 1/(R2*Wcp)
```

```
C = 1.1139e-12
```

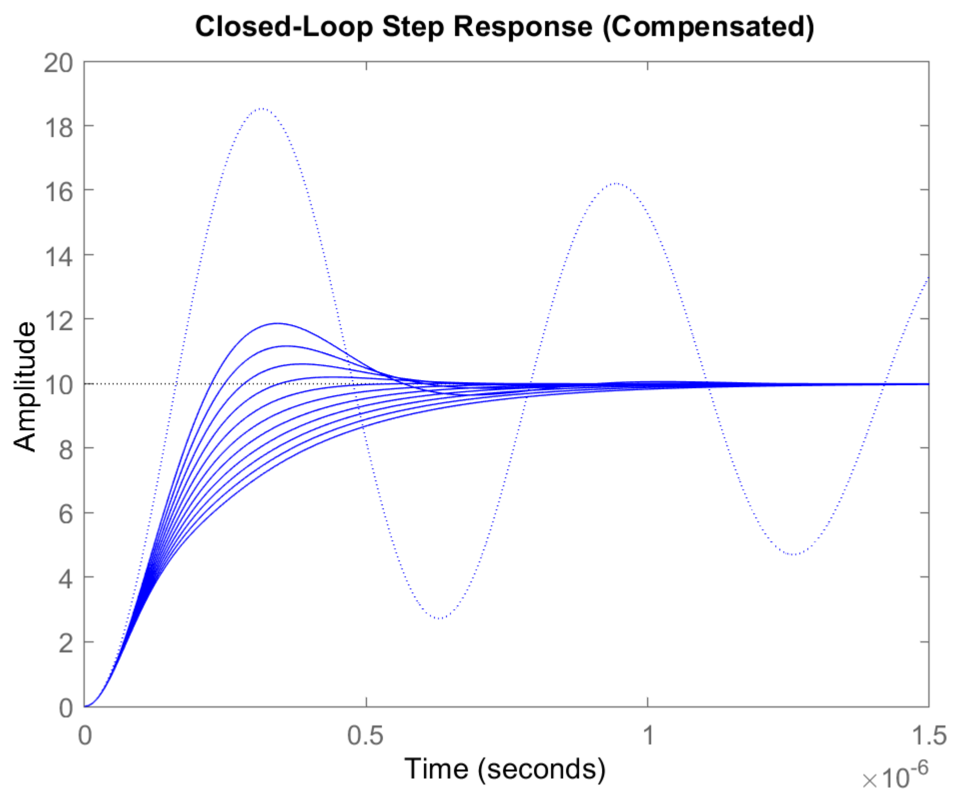
```

K = R1/(R1+R2);
C = [1:1/5:3]*1e-12;
for n = 1:length(C)
    B_array(:,:,n) = tf([K*R2*C(n) K],[K*R2*C(n) 1]);
end
```

Now you can create LTI arrays for A(s) and L(s):

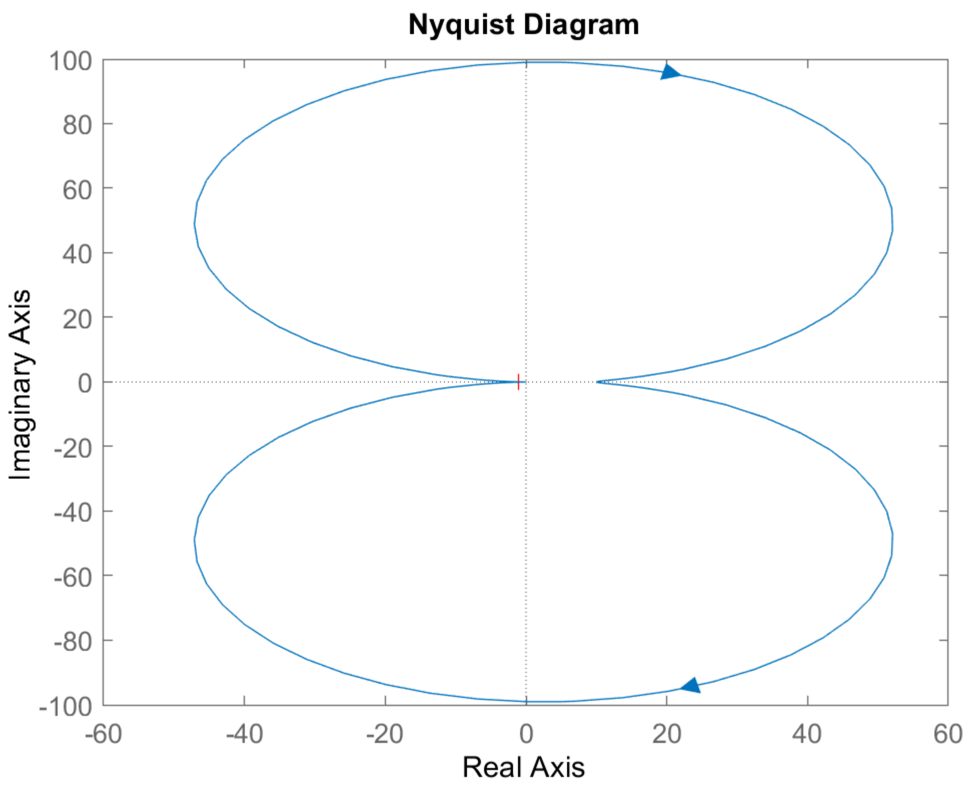
```
A_array = feedback(A_open,B_array);
L_array = A_open*B_array;
```

Plot the step response for multiple values of the Capacitor:

```
stepplot(A_closed,'b:',A_array,'b',[0:.005:1]*1.5e-6);
title('Closed-Loop Step Response (Compensated)');
```

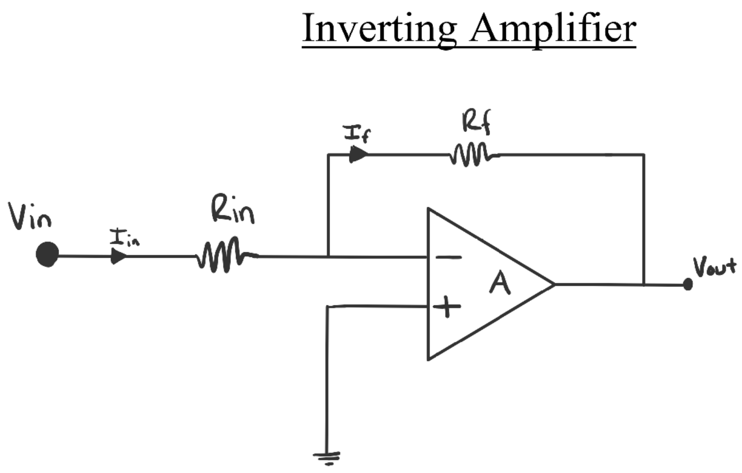


```
nyquist(A_closed)
```



The Inverting Amplifier

- Using only resistors and an op amp component we can design a proportional gain controller



- Property of the Op Amp is that it has infinite input impedance, thus no current will flow through the + or - terminals, and will result in zero voltage drop. After performing some KCL to solve for the transfer function of the system, we find:

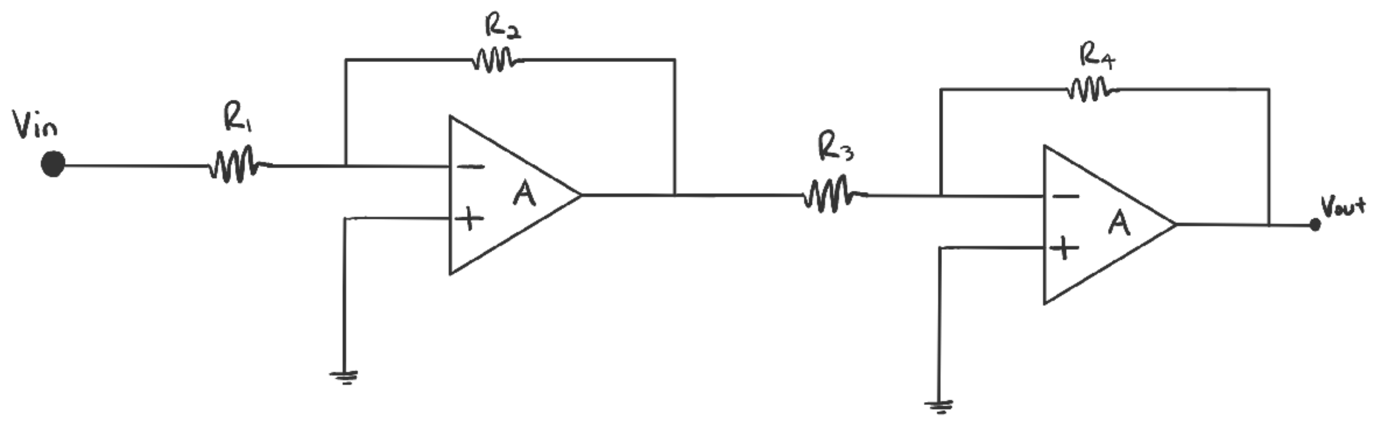
$$V_{\text{out}} = -\frac{R_f}{R_{\text{in}}} V_{\text{in}} \leftrightarrow H(s) = -\frac{R_f}{R_{\text{in}}}$$

- Thus, this arrangement of components provides a negative amplitude gain to any signal that is passed through this circuit. If we cascade a second inverting amplifier with equivalent resistors, we get a Proportional Controller.

Proportional Controller

- Using two op amps and two pairs of resistors, we can form a porportional controller that provides a positive amplitude gain to any signal that is passed through this circuit.

Proportional Controller



- Since this circuit arrangement is the cascaded result of two inverting amplifiers, the corresponding transfer function is defined as:

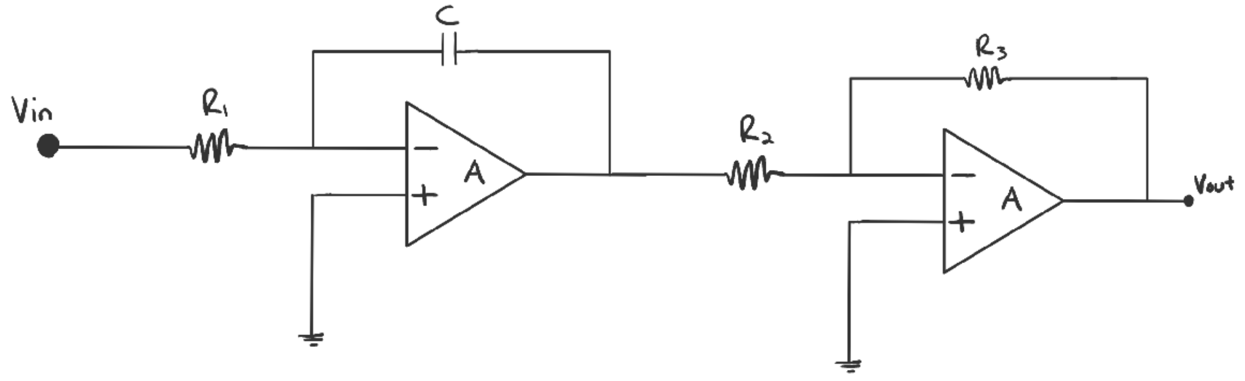
$$V_o = \frac{R_2}{R_1}\frac{R_4}{R_3}V_{\text{in}} \leftrightarrow H(s) = \frac{R_2}{R_1}\frac{R_4}{R_3}$$

- In practice, the values of $R_3 \text{ and } R_4$ are chosen to be equivalent such that the ratio between $R_1 \text{ and } R_2$ are designed to provide the positive amplitude gain factor.

Integrator Controller

- Using two op amps, two resistors and two capacitors, we can form an integrator controller which can be used for changing the waveform of the input, for example, a square wave input will be integrated during the high and low periods of its wave-shape and result in a triangle waveform in the output. This serves as a way of controlling the waveshape of the signals.

Integrator Controller

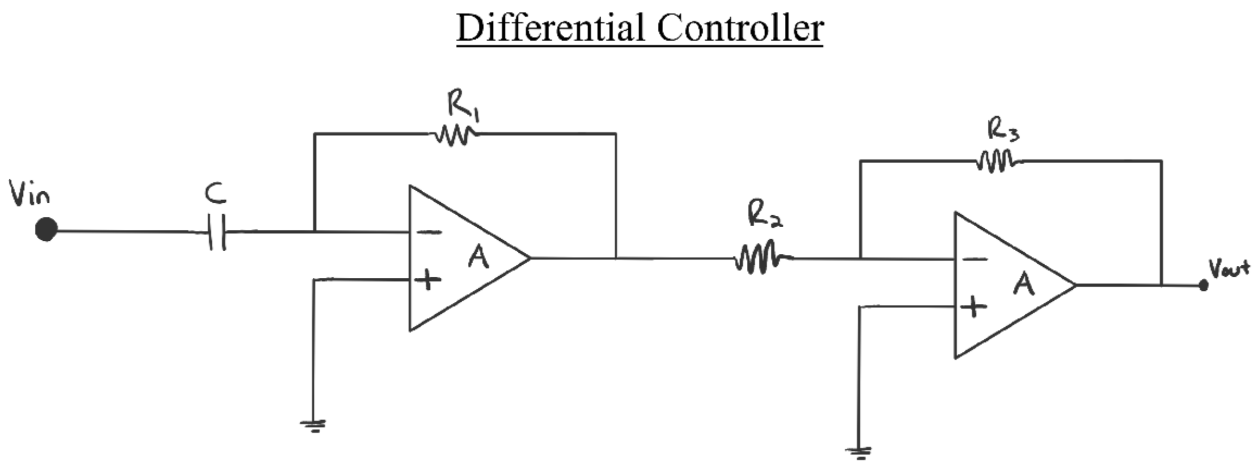


- This controller utilizes two inverting amplifiers that serve to either prove an amplitude boost or unity gain depending on the values of R_2 and R_3. The corresponding transfer function can be defined as:

$$V_o = \left(-\frac{1}{R_1 C}\int_0^t v_{in}(\tau)d\tau\right)\left(-\frac{R_3}{R_2}\right)V_i \leftrightarrow H(s) = \frac{R_3}{sCR_1R_2}$$

Differential Controller

- Again, using two op amps, two resistors and two capacitors, we can form a differential controller by swopping the locations of the capacitor and first resistor in the Integrator design. This controller can also be used for changing the waveform of the input, for example, a triangle waveform input will be transformed into a square-wave as the slope of the triangle remains constant for the rising and falling in the amplitude.

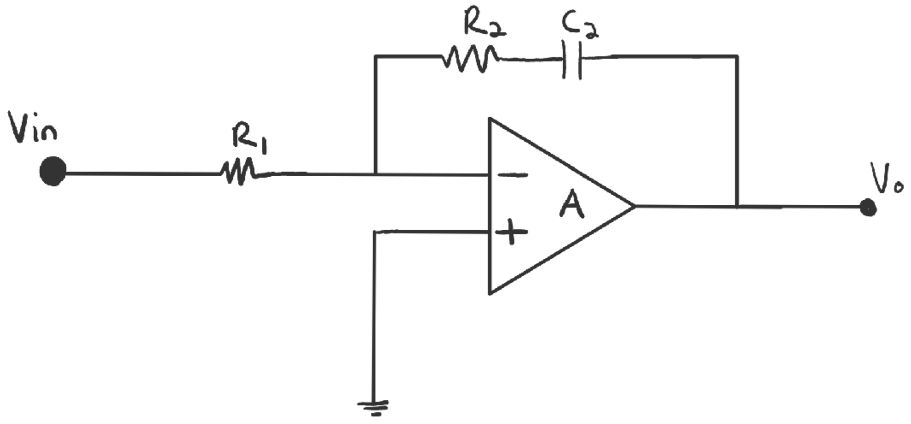


- This controller again utilizes two inverting amplifiers that serve to either prove an amplitude boost or unity gain depending on the values of R_2 and R_3. The corresponding transfer function can be defined as:

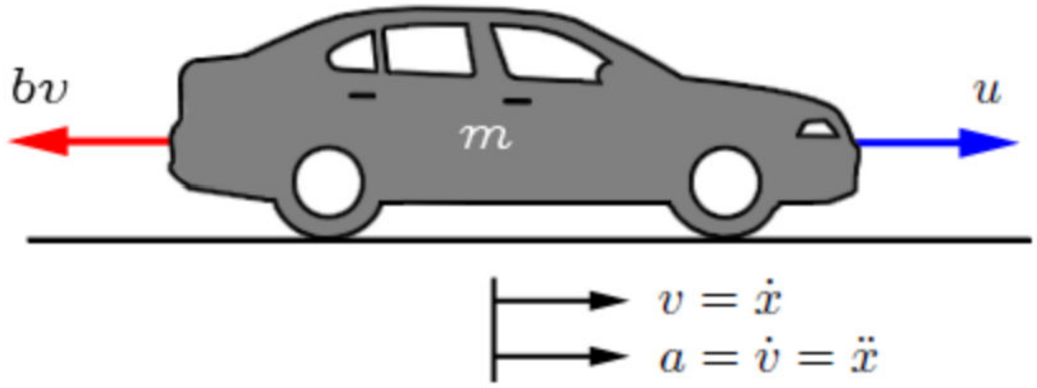
$$V_o = \left(-R_1 C\frac{dv_{in}(t)}{dt}\right)\left(-\frac{R_3}{R_2}\right)V_i \leftrightarrow H(s) = \frac{sCR_1R_3}{R_2}$$

Proportional-Integral Controller

PI Controller



Cruise Control Circuit using a PI Controller



This image describes the physcial equations governing a car that is in motion, with parameters:

- Velocity $v = \frac{dv}{dt}$ being the displacement in location over time
- Damping coefficient b due to the drag of the car
- Acceleration $a = \frac{d^2v}{dt^2}$ being the 2nd derivative of the displacement over time
- Force u which is generated between the road/tires to propel the car forward
- Mass m that describes the physcial weight of the vehicle

After applying Newton's 2nd law, we find that the governing equation for this system is:

$$m\dot{v} = u - bv$$

After rearranging the equation for like terms on both sides:

$$m\dot{v} + bv = u$$

which describes a 1st order differential equation with $y = v$.

Here, we apply some arbitrary parameters to derive the open-loop transfer function of this system.

- $m = 1302\,\mathrm{kg}$ - which is the average weight of a car
- $b = 50\,\frac{N \cdot \mathrm{sec}}{m}$ - how quickly the car will return to rest, or drag
- $u = 650\,N$ - nominal control force that the engine needs to sustain constant speed

Now we derive the open-loop transfer function of this system:

$$m\dot{v} + bv = u \leftrightarrow msV(s) + bV(s) = U(s)$$

$$V(s)(\mathrm{ms} + b) = U(s)$$

$$P(s) = \frac{V(s)}{U(s)} = \frac{1}{ms + b}$$

$$P(s) = \frac{1}{1302s + 50}$$

which describes a transfer function of 1st order with no zeros and a real pole at $p = -\frac{50}{1302} = -0.038$

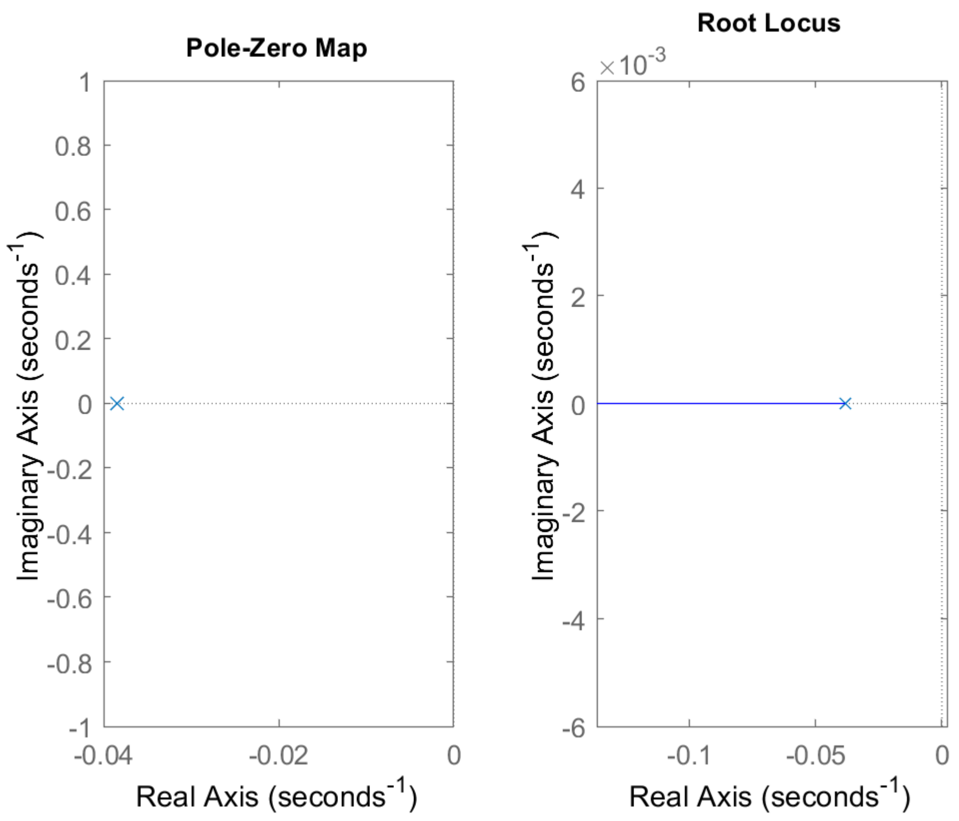
We'll now derive a PI controller that will satisfy the following requirements:

- Rise time < 5 s
- Overshoot < 10%
- Steady-state error < 2%

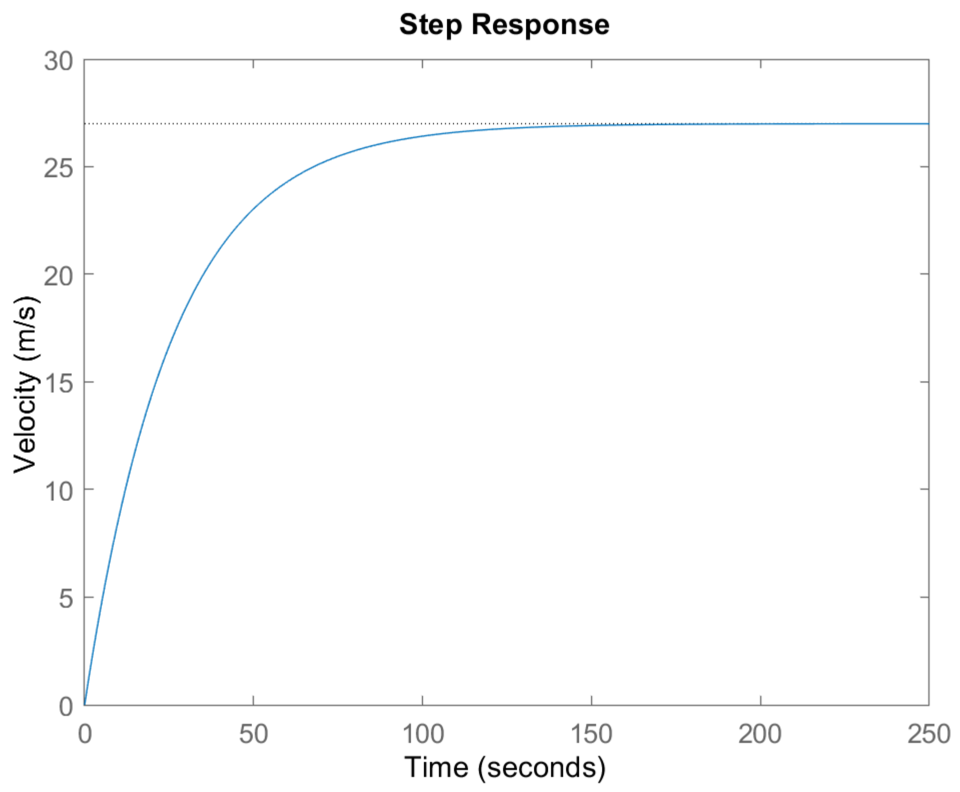
```
% System Parameters
m = 1302; % weight of car in kilograms
b = 50; % damping coefficient in Nsec/m
u = 1350; % Nominal control force to sustain constant speed
s = tf('s');
P_car = 1/(m*s + b)
```

```
P_car =
 
       1
  -----------
  1302 s + 50
 
Continuous-time transfer function.
```

```
figure
subplot(1,2,1),pzmap(P_car)
subplot(1,2,2),rlocus(P_car)
```



```
figure, step(u*P_car), ylabel('Velocity (m/s)')
```



```
P_stable = isstable(P_car)
```

```
P_stable = logical
   1
```

We observe from the Step Response of the system that it takes about 150 secs to reach a steady-state of 27 m/s or about 60 mph. Which is far too slow of a rise time to reach a constant velocity when driving on a highway. However, the system is stable because is only has one real-valued pole that approaches in the direction of $-\infty$ when a large gain is applied to the open-loop response. However, we can do better by designing a controller that shortens the rise time and provides a realistic performance you'd expect of a cruise control module in a car.

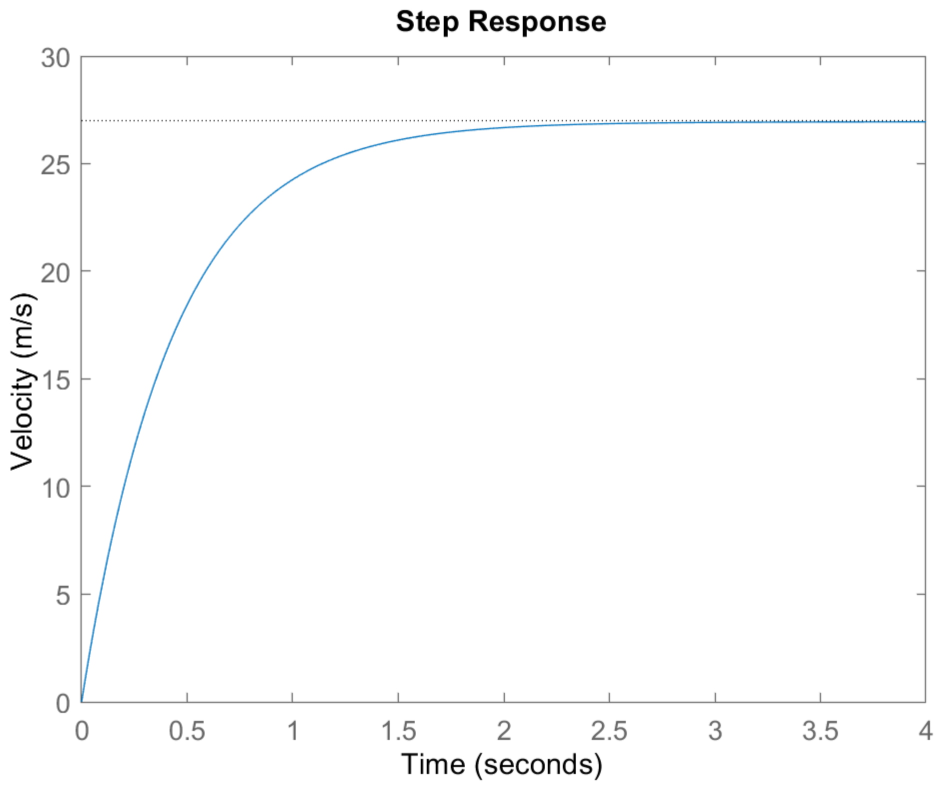
```
% Design PI Controller that satisfies above requirements
Kp = 3000;
Ki = 100;
r = 27; % m/s or 60 mph
C_cruise = pid(Kp,Ki,0)
```

```
C_cruise =
 
             1  
  Kp + Ki * ---
             s  

  with Kp = 3e+03, Ki = 100
 
Continuous-time PI controller in parallel form.
```

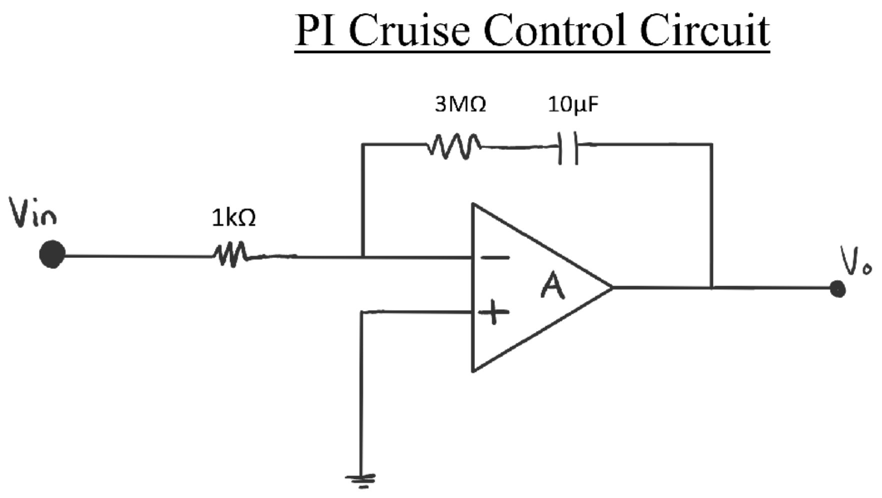
```
Y = feedback(C_cruise*P_car,1)
```

```
Y =
 
       3000 s + 100
  -----------------------
  1302 s^2 + 3050 s + 100
 
Continuous-time transfer function.
```

```
figure, step(r*Y), ylabel('Velocity (m/s)')
```

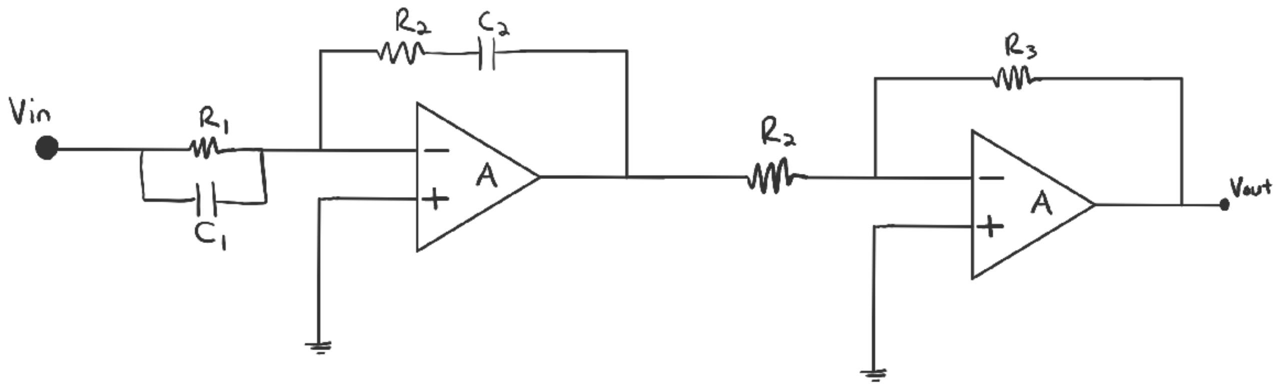


The step response after implementing the controller now shows that the rise time occurs in about 3 secs to reach the desired speed of 13 m/s or 60 mph, which is around 3% of the original rise time

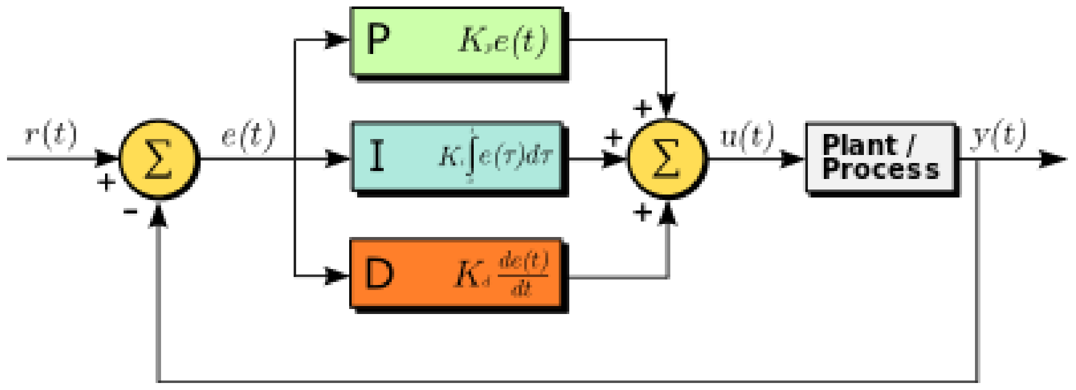


Proportional-Integral-Derivative Controller

PID Controller



Feedback Diagram of PID Controller



The PI controller can be expanded upon with the implementation of differential component at the locations of the input impedance with the intrduction of a capacitor in parallel with the input resistor. The figure above shows this arrangement with the addition of an inverting amplifier that makes it so that the output is positive. The transfer function of this circuit can be derived through inspection of the impedances:

$$Z_1(s) = \frac{R_1}{1 + R_1C_1s},\ Z_2(s) = R_2 + \frac{1}{\mathrm{s}C_s}$$

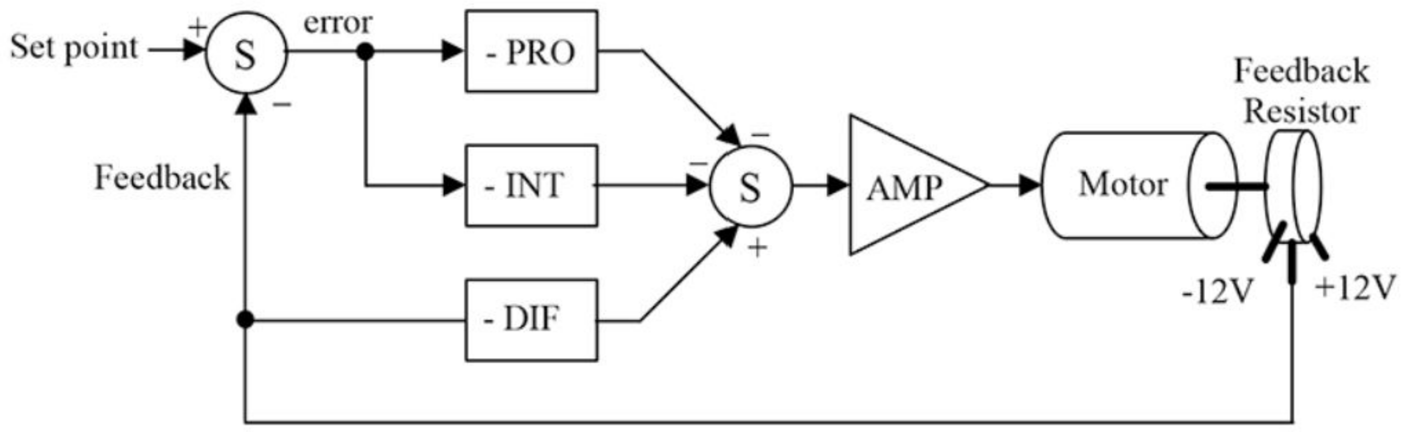
Then the transfer function for this controller is derived from:

$$H(s) = \frac{Z_2(s)}{Z_1(s)} = \frac{(1 + R_1C_1s)(1 + R_2C_2s)}{R_1C_2s}$$

with

$$K_p = \frac{R_1C_1 + R_2C_2}{R_1C_2},\ K_i = R_1C_1 + R_2C_2, \text{ and } K_d = \frac{R_1R_2C_1C_2}{R_1C_1 + R_2C_2}$$

DC Motor Speed Control using Op Amp PID Controller



This diagram displays all the necessary buidling blocks that are needed to create a PID controller using a Op Amp circuits, where a Propoprtional, Intregral, and Derivative controller circuits are summed together and then inverted to counter the DC Motor's transfer function. The result applied to the DC Motor is returned in a feedback loop that is tied to a summing amplifier where the desired signal is being fed, and then the error signal serves as an input into the proportional and integral controllers of the PID.

Physical Parameters and Governing Equations for Electric Motors

- Electric Motor is a transducer where it converts between mechanical energy and electrical energy
- Consists of a an inductor with a magnet that produces a current from an applied voltage across it terminals. This current then gives way to torque on the shaft of the motor which can be used to power rotation loads like wheels, propellors, etc...
- Energy can be converted from mechanical into electrical, where external torque produces a current
- Energy can also be converted from electrical to mechanical, by applying some V_{emf} to produce a desired torque on the shaft, which suggests that we can control a motor's speed by providing a desired V_{emf}
- Thus, an electrical motor's angular velocity can be controlled through the design of electrical circuits that apply this desired voltage, which confirms we can design Operational Amplifier circuits that are analogous for PID control.

Relationship between Current and Torque

$$T \propto K_T \times I$$

- A known motor constant multiplied by the current is proportional to the torque

Relationship between EMF and Angular Velocity

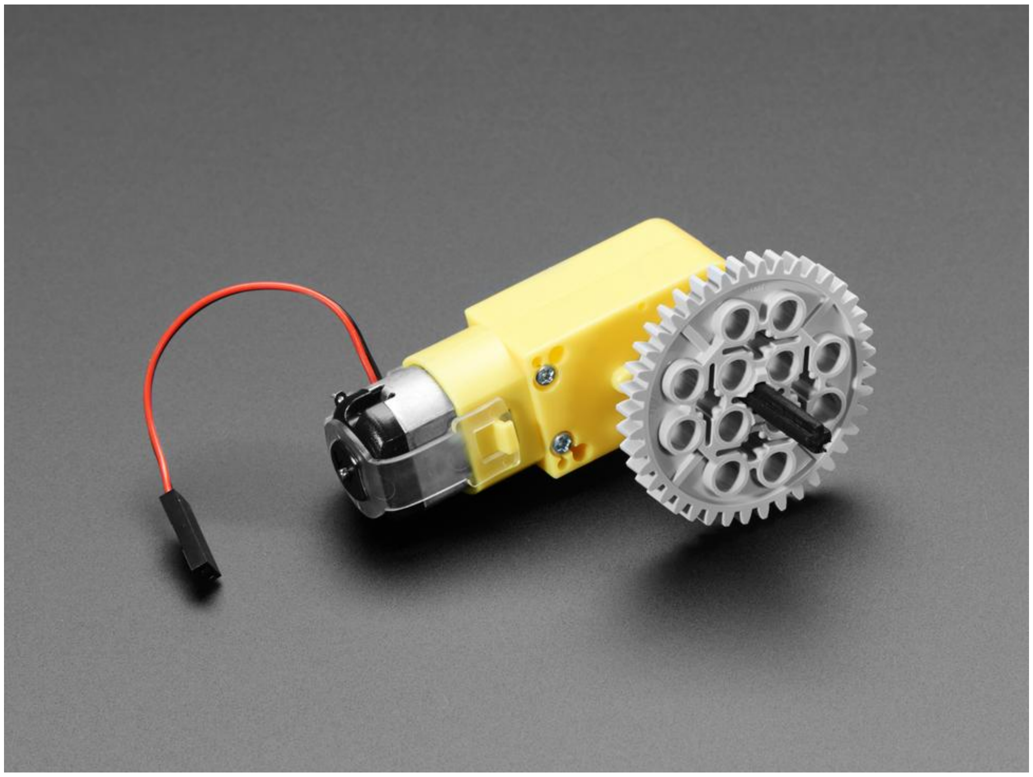
$$V_{\text{emf}} \propto K_\omega \times \omega$$

- For all brushed motors $K_T = K_\omega$ and can thus represent this value as the motor constant K_M

These equations to apply control theory approaches for desired values of the torque T or the angular velocity ω

Since we are using Op-Amp circuits, we are dealing with relatively small operational voltage values, within the range of operating capacities of a Microcontroller or small battery voltage ranges. So not high powered or high voltage circuits.

Lego Gear Motor



- Used for some electronics circuits and projects
- Operational voltage between 3 - 9V DC
- low-friction and used for high speed design
- its motor constant was derived via MIT Robotics Seminar

Derive Equations Using Parameters of Motor

Internal Resitance of Motor = $25\,\Omega$, then current

$$I = \left(\frac{9V}{25\,\Omega}\right)$$

$$K_T = 0.25\ \frac{N \cdot m}{A}$$

Then we can solve for the torque

$$T = K_T \cdot I$$

$$T = \left(\frac{9}{25}\right)(0.25) = 0.089\ N \cdot m$$

Double check that both motor constants are equivalent

$$V_{\text{emf}} = K_\omega \cdot \omega$$

$$9\ V = K_\omega\left(35.6\ \frac{\text{rad}}{\text{sec}}\right)$$

$$K_\omega = 0.2528 \approx 0.25$$

Thus, $K_T = K_\omega$, and shows that if we can achieve a desired angular speed of the motor by applying V_{emf} that produces a current that sets our desired torque.

Apply Newton's 2nd Law and Kirchoff's Voltage Law to obtain governing equations

The Current and Torque equation is

$$J\ddot{\theta} + b\dot{\theta} = K_M \cdot I$$

where

- $J - \text{resistance to angular acceleration}$
- $b - \text{resistnace to angular velocity}$
- $I - \text{current through the motor}$
- $K_M - \text{motor constant}$

The EMF Voltage and Angular Velocity equation is

$$L\frac{di(t)}{dt} + Ri(t) = V_{emf} - K_M \cdot \dot{\theta}$$

After making the subsitution of $\omega = \dot{\theta}$, we get the two equations:

$$J\dot{\omega} + b\omega = K_M \cdot i$$

$$L\frac{di(t)}{dt} + Ri(t) = V_{emf} - K_M \cdot \omega$$

Laplace Transform of Equations

$$J\dot{\omega} + b\omega = K_M \cdot i \leftrightarrow sJ\Omega(s) + b\Omega(s) = K_M I(s)$$

$$\Omega(s)(Js + b) = K_M I(s)$$

$$L\frac{di(t)}{dt} + Ri(t) = V_{emf} - K_M \cdot \omega \ \leftrightarrow sLI(s) + RI(s) = V(s) - K_M\Omega(s)$$

$$I(s)(Ls + R) = V(s) - K_M\Omega(s)$$

We can subsitute in the equivalence of $I(s)$ to derive the transfer function:

$$\frac{\Omega(s)}{V(s)} = \frac{K}{(Js + b)(Ls + R) + K^2}$$

where

- K – motor constant
- J – resistance to angular acceleration
- b – resistance to angular velocity
- R – internal resistance
- L – inductance

Use MATLAB to experimentally derive PID controller

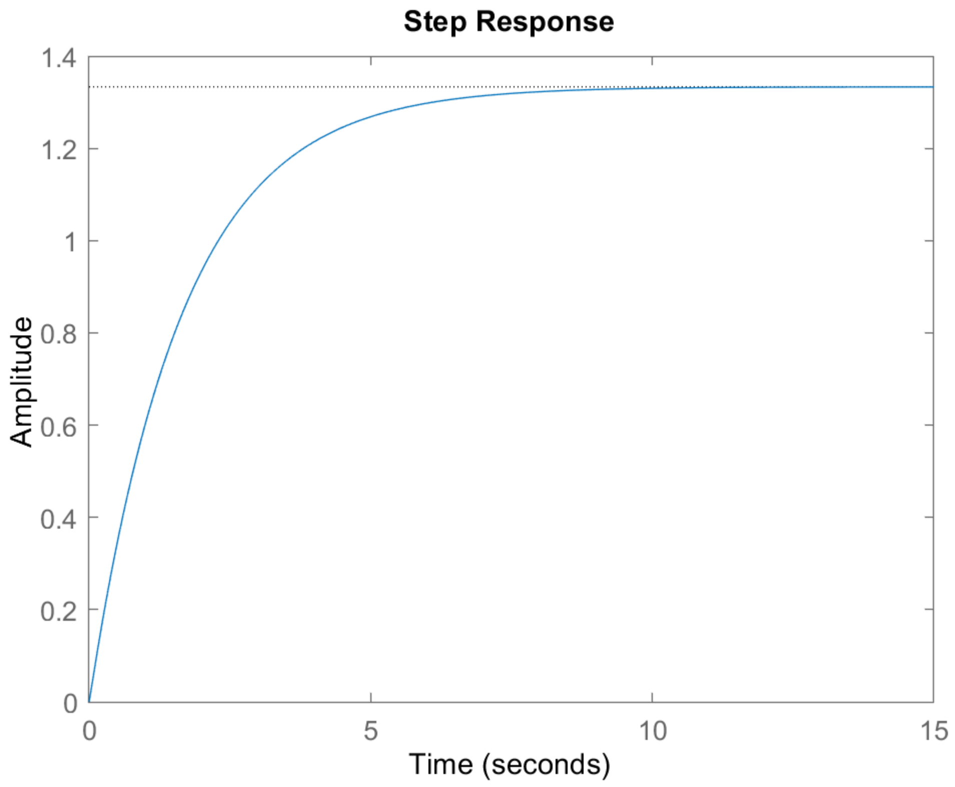
```
s = tf('s');
J = 0.0124; % kg m^2
b = 0.005; % N m s
K = 0.25; % N m / A
R = 25; % ohms
L = 0.1; % H
P = K/((J*s+b)*(L*s+R)+K^2)
```

```
P =
 
               0.25
  -------------------------------
  0.00124 s^2 + 0.3105 s + 0.1875
 
Continuous-time transfer function.
```

```
p = roots([0.00124 0.3105 0.1875])
```

```
p = 2×1
 -249.7979
   -0.6053
```

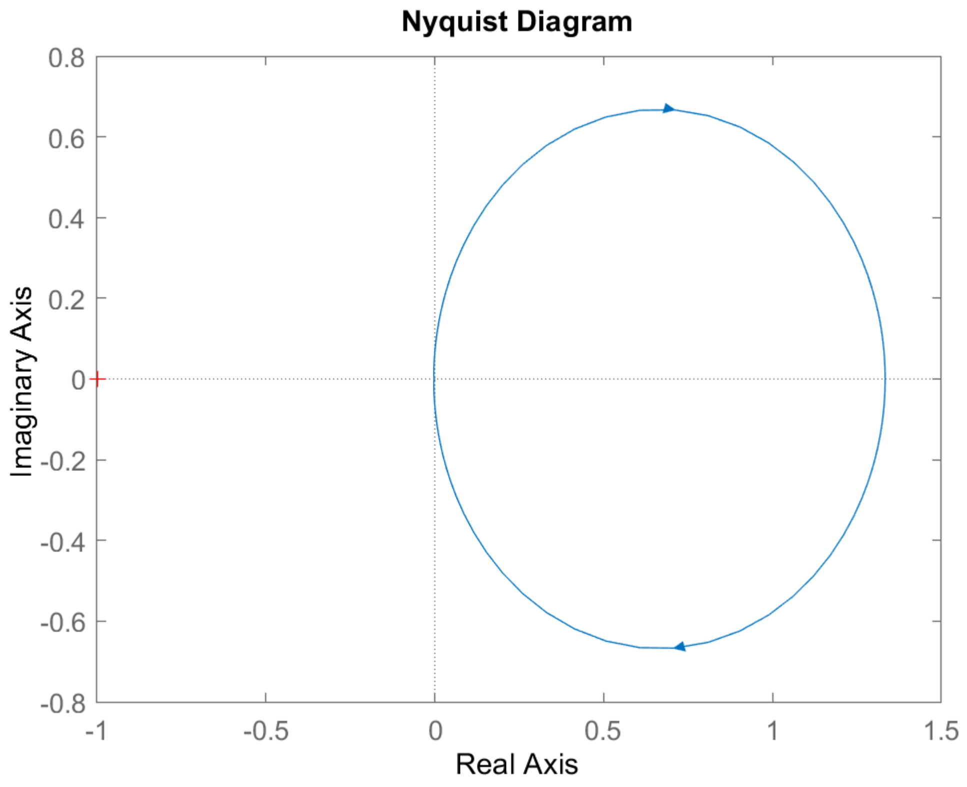
```
% Step Response of Motor
step(P)
```



```
y = step(P);
sserror=abs(1-y(end))
```

```
sserror = 0.3332
```

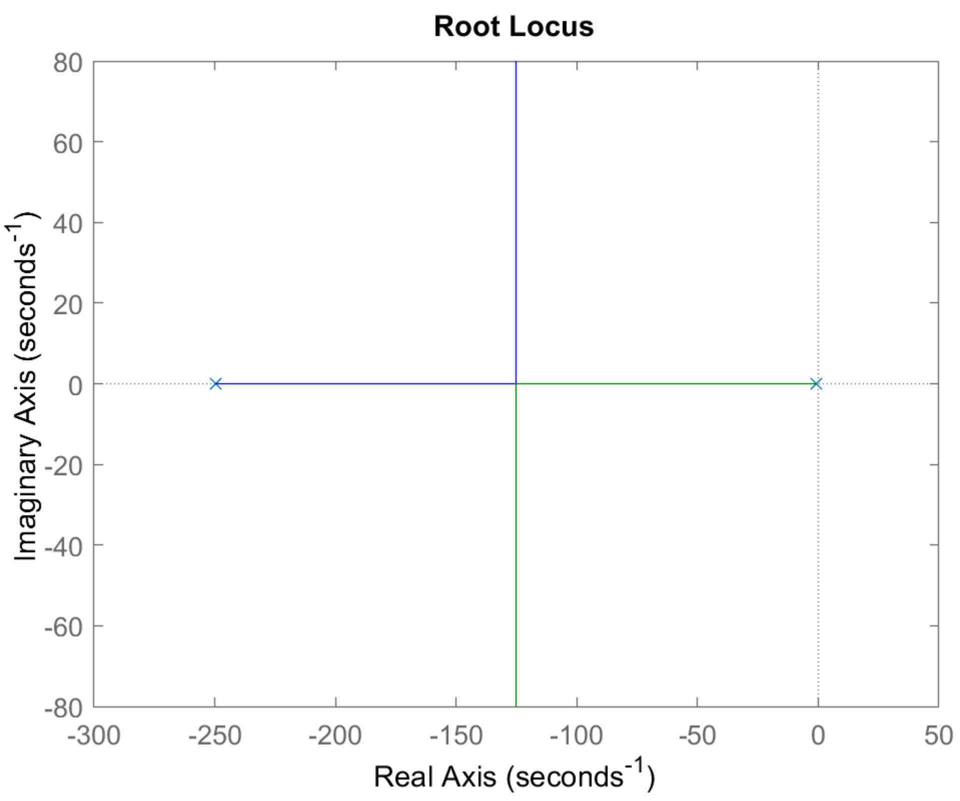
```
nyquist(P)
```



```
SystemStable = isstable(P)
```

```
SystemStable = logical
   1
```

```
% Root Locus of Plant
rlocus(P)
```



```
%[k, poles] = rlocfind(P)
```

Effects of Proportional Control on Motor

- Proportional Control provides amplification to the error term by some proportional constant $-K_p$, which results in a faster response by the system

$$u(t) = K_p e(t)$$

Taking the Laplace Transform,

$$U(s) = K_p E(s)$$

$$\frac{U(s)}{E(s)} = K_p$$

Closing the Loop produces the transfer function

$$\frac{Y(s)}{R(s)} = \frac{C(s)P(s)}{1 + C(s)P(s)}$$

```
t = 0:.0001:.05;
Kp = 100
```

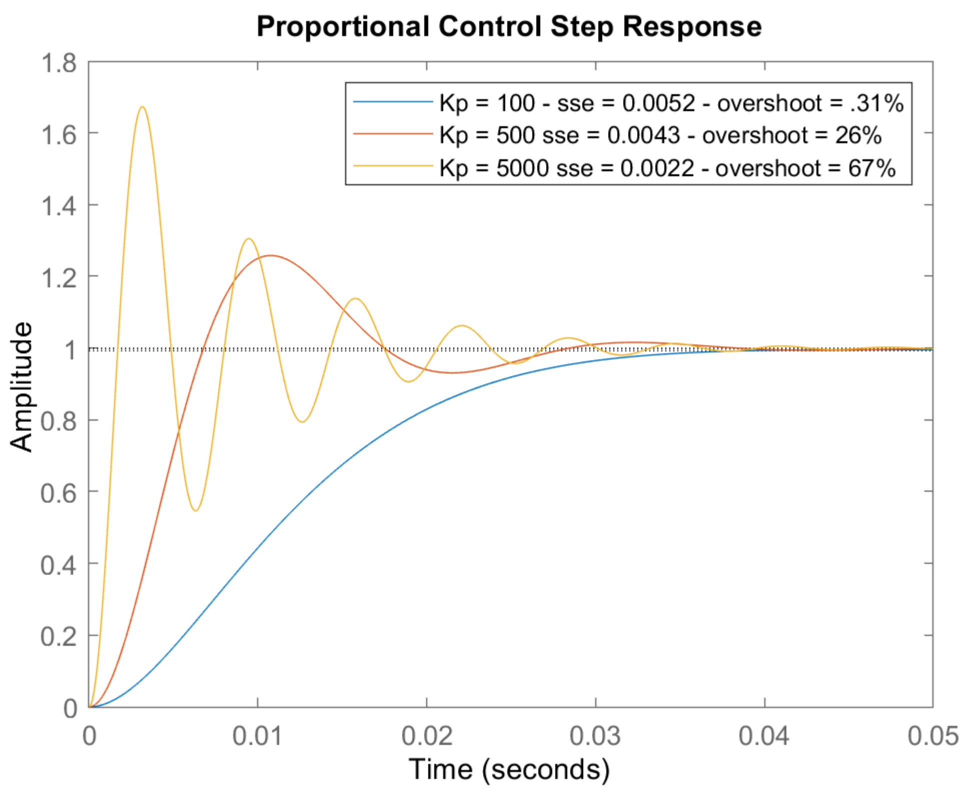
```
Kp = 100
```

```
C = pid(Kp);
```

```
Y = feedback(C*P,1)
```

```
Y =

                25
  ------------------------------
  0.00124 s^2 + 0.3105 s + 25.19

Continuous-time transfer function.
```

```
figure
step(Y,t), hold on
y = step(Y);
sserror100=abs(1-y(end));
S1 = stepinfo(Y); overshoot1 = S1.Overshoot;
Kp = 500;
C = pid(Kp);
Y = feedback(C*P,1);
step(Y,t), hold on
y = step(Y);
sserror1000=abs(1-y(end));
S2 = stepinfo(Y); overshoot2 = S2.Overshoot;
Kp = 5000;
C = pid(Kp);
Y = feedback(C*P,1);
step(Y,t), hold off
y = step(Y);
sserror5000=abs(1-y(end));
S3 = stepinfo(Y); overshoot3 = S3.Overshoot;
title('Proportional Control Step Response')
legend('Kp = 100 - sse = 0.0052 - overshoot = .31%','Kp = 500 sse = 0.0043 - overshoot = 26%',
```



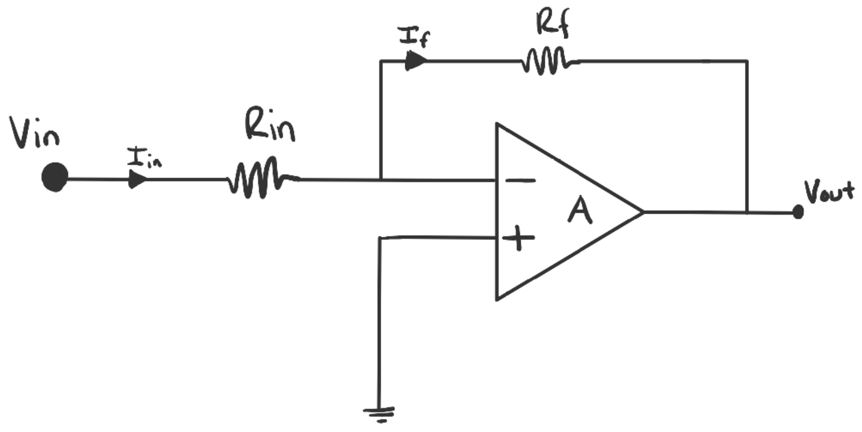
Observations:

- System approaches steady-state quicker with proportional control, but there is a trade-off that introduces overshoot % the faster the rise-time.
- Introduces underdamped system the higher the porportional gain becomes

Op - Amp Proportional Controller - Inverting Amplifier

- Using only resistors and an op amp component we can design a proportional gain controller that provides desired V_{emf} which provides desired angular velocity of motor

Inverting Amplifier



- Property of the Op Amp is that it has infinite input impedance, thus no current will flow through the + or - terminals, and will result in zero voltage drop. After performing some KCL to solve for the transfer function of the system, we find:

$$V_{\text{out}} = -\frac{R_f}{R_{\text{in}}} V_{\text{in}} \leftrightarrow H(s) = -\frac{R_f}{R_{\text{in}}}$$

where

$$K_p = -\frac{R_f}{R_{\text{in}}}$$

- Can use this circuit and the motor input in combination with a summing amplifier to get a single voltage reading for V_{emf}

Effects of Proportional-Integral Control on Motor

- Addition of K_i term to multiply the integral of the error term

$$u(t) = K_p e(t) + K_i \int e(t)\,dt$$

Taking the Laplace Transform,

$$U(s) = K_p E(s) + K_i \frac{E(s)}{s}$$

$$U(s) = E(s)\left(K_p + \frac{K_i}{s}\right)$$

$$\frac{U(s)}{E(s)} = K_p + \frac{K_i}{s}$$

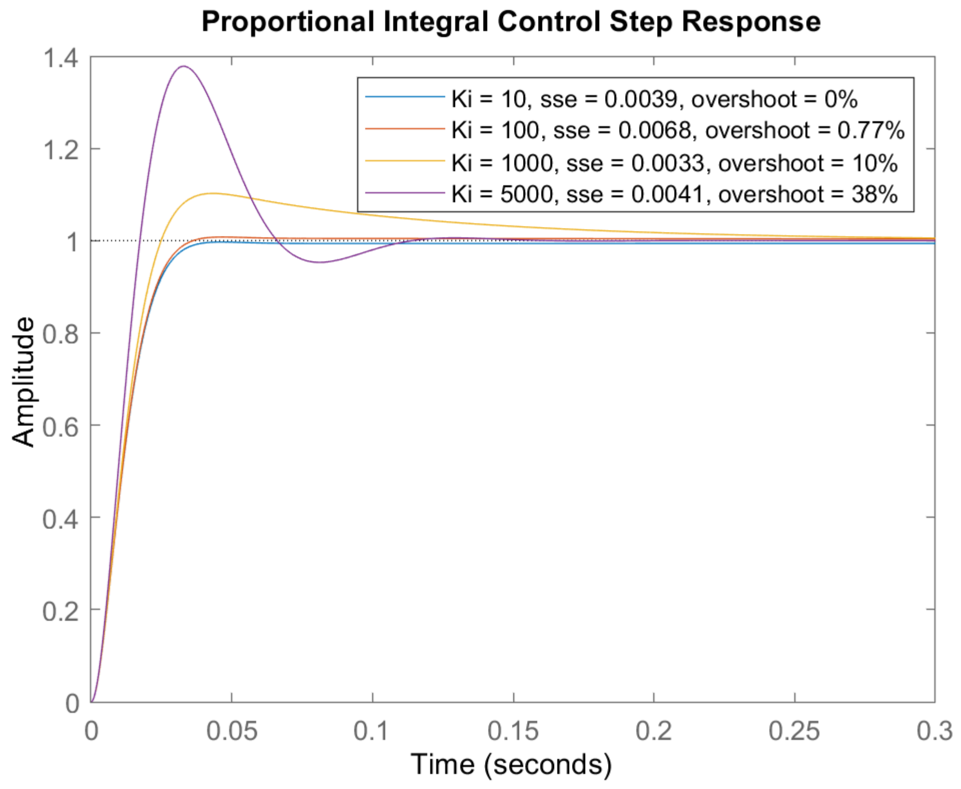
```
% Keep Kp constant and observe effects of varying Ki
t = 0:.0001:0.3;
Kp = 100, Ki = 10
```

```
Kp = 100
Ki = 10
```

```
C = pid(Kp,Ki);
Y = feedback(C*P,1)
```

```
Y =
 
                 25 s + 2.5
  ----------------------------------------
  0.00124 s^3 + 0.3105 s^2 + 25.19 s + 2.5
 
Continuous-time transfer function.
```

```
figure
step(Y,t), hold on
y = step(Y);
sserror10=abs(1-y(end));
S1 = stepinfo(Y); overshoot1 = S1.Overshoot;
Ki = 100;
C = pid(Kp,Ki);
Y = feedback(C*P,1);
step(Y,t), hold on
y = step(Y);
sserror100=abs(1-y(end));
S2 = stepinfo(Y); overshoot2 = S2.Overshoot;
Ki = 1000;
C = pid(Kp,Ki);
Y = feedback(C*P,1);
step(Y,t), hold on
y = step(Y);
sserror1000=abs(1-y(end));
S3 = stepinfo(Y); overshoot3 = S3.Overshoot;
Ki = 5000;
C = pid(Kp,Ki);
Y = feedback(C*P,1);
step(Y,t), hold off
y = step(Y);
sserror5000=abs(1-y(end));
S4 = stepinfo(Y); overshoot4 = S4.Overshoot;
title('Proportional Integral Control Step Response')
legend('Ki = 10, sse = 0.0039, overshoot = 0%','Ki = 100, sse = 0.0068, overshoot = 0.77%','Ki
```



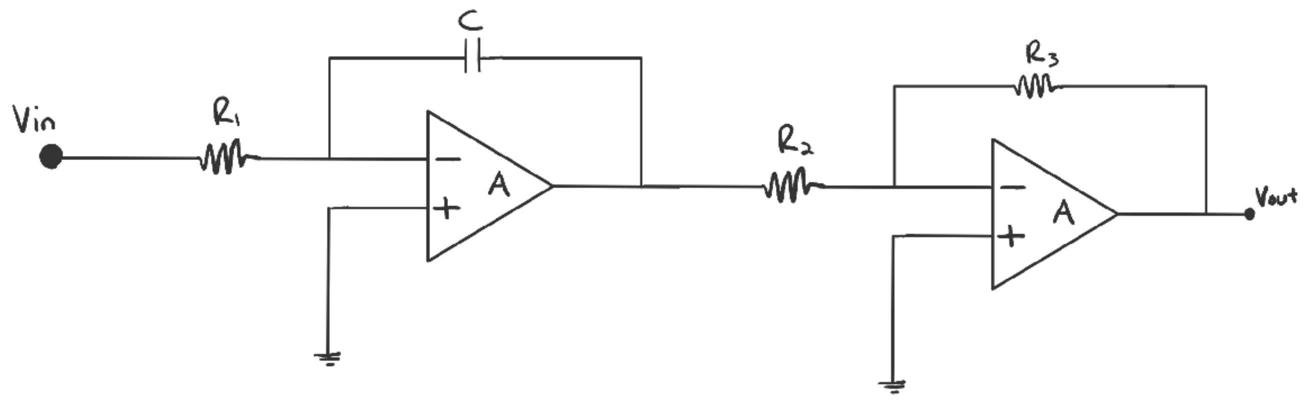
Observations:

- Increasing value of K_i results in a decrease in steady-state error, but there is an increase in overshoot %, which solidifies that there is an inverse relationship between these system specifications.
- This phenomena occurs because if a steady-state error exists, the integration will continue to grow larger and larger over time. It takes the control signal longer to increase enough to eliminate the error

Op - Amp PI Controller - Integrator Controller

- Using two op amps, two resistors and two capacitors, we can form an integrator controller which can be used for changing the waveform of the input, for example, a square wave input will be integrated during the high and low periods of its wave-shape and result in a triangle waveform in the output. This serves as a way of controlling the waveshape of the signals.

Integrator Controller



- This controller utilizes two inverting amplifiers that serve to either prove an amplitude boost or unity gain depending on the values of R_2 and R_3. The corresponding transfer function can be defined as:

$$V_o = \left(-\frac{1}{R_1C}\int_0^t v_{in}(\tau)d\tau\right)\left(-\frac{R_3}{R_2}\right)V_i \leftrightarrow H(s) = \frac{R_3}{sCR_1R_2}$$

where

$$K_i = -\frac{1}{R_1C}$$

Effects of Proportional-Derivative Control on Motor

- Add Derivative term K_D to proportional control instead of integrator term

$$u(t) = K_pe(t) + K_D\frac{de(t)}{dt}$$

Taking the Laplace Transform,

$$U(s) = K_pE(s) + K_D\,sE(s)$$

$$U(s) = E(s)\,(K_p + K_Ds)$$

$$\frac{U(s)}{E(s)} = K_p + \ \ K_Ds$$

```
% Keep Kp constant and observe effects of varying Ki
t = 0:.0001:0.1;
Kp = 100, Ki = 0, Kd = 10
```

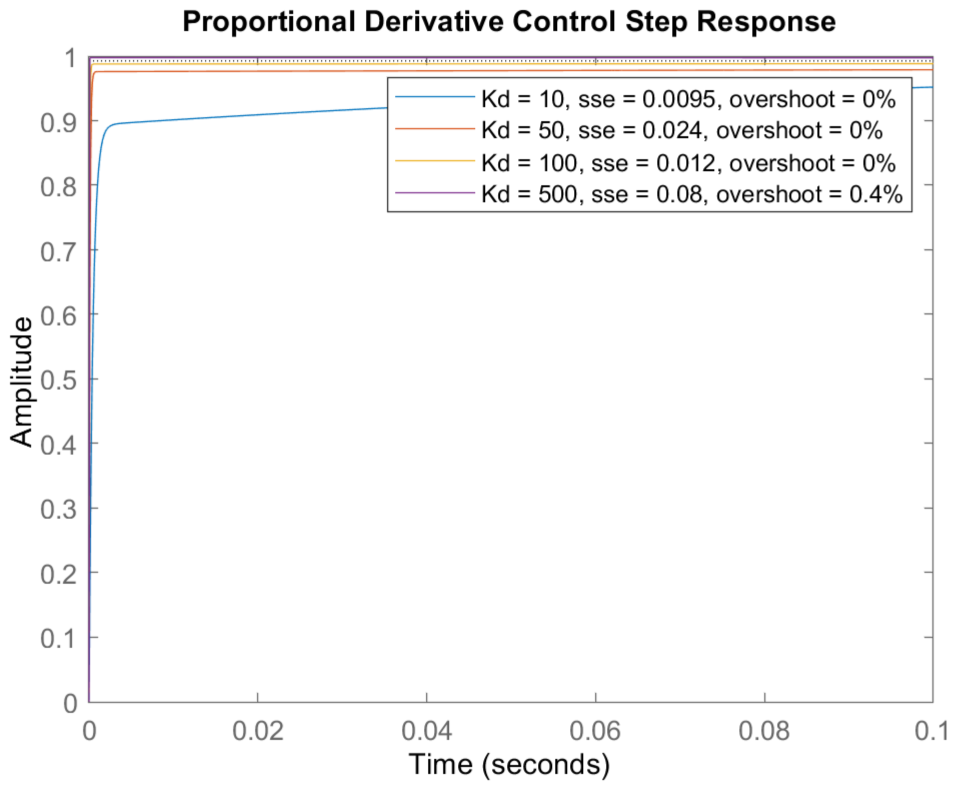
```
Kp = 100
Ki = 0
```

```
Kd = 10
```

```
C = pid(Kp,Ki,Kd);
Y = feedback(C*P,1)
```

```
Y =
 
          2.5 s + 25
  -----------------------------
  0.00124 s^2 + 2.811 s + 25.19
 
Continuous-time transfer function.
```

```
figure
step(Y,t), hold on
y = step(Y);
sserror10=abs(1-y(end));
S1 = stepinfo(Y); overshoot1 = S1.Overshoot;
Kd = 50;
C = pid(Kp,Ki,Kd);
Y = feedback(C*P,1);
step(Y,t), hold on
y = step(Y);
sserror50=abs(1-y(end));
S2 = stepinfo(Y); overshoot2 = S2.Overshoot;
Kd = 100;
C = pid(Kp,Ki,Kd);
Y = feedback(C*P,1);
step(Y,t), hold on
y = step(Y);
sserror100=abs(1-y(end));
S3 = stepinfo(Y); overshoot3 = S3.Overshoot;
Kd = 500;
C = pid(Kp,Ki,Kd);
Y = feedback(C*P,1);
step(Y,t), hold off
y = step(Y);
sserror500=abs(1-y(end));
S4 = stepinfo(Y); overshoot4 = S4.Overshoot;
title('Proportional Derivative Control Step Response')
legend('Kd = 10, sse = 0.0095, overshoot = 0%','Kd = 50, sse = 0.024, overshoot = 0%','Kd = 10
```



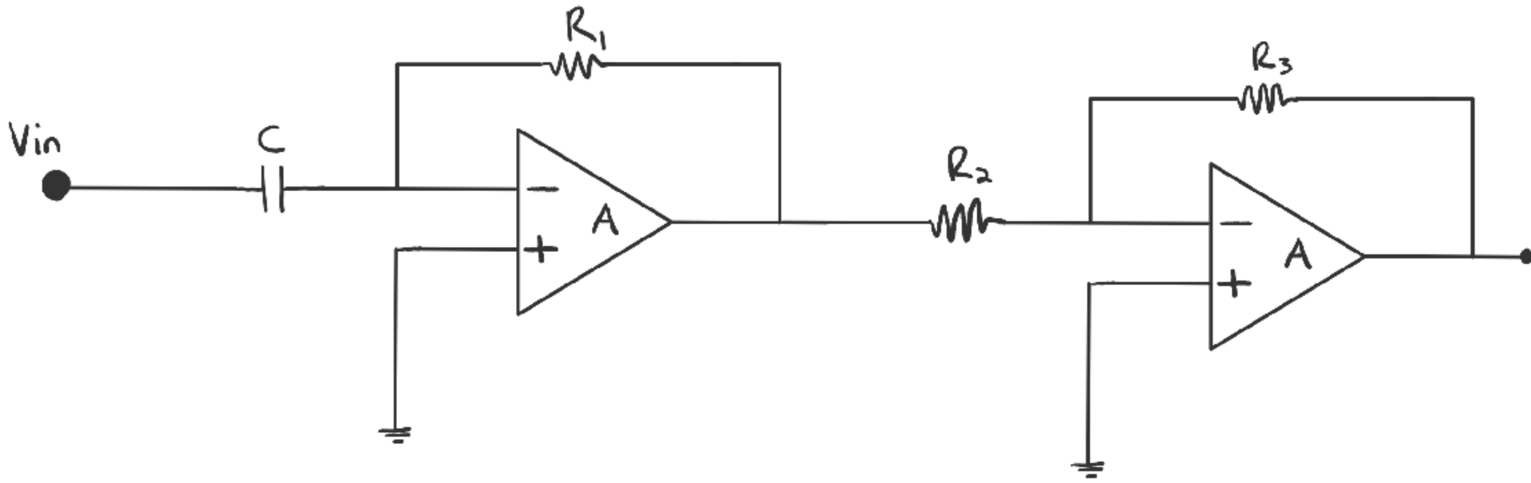
Observations:

- PD control doesn't decrease the steady-state error and the higher the K_D term the larger the steady-state error becomes.
- Once the error term stops changing, then the derivative of the errror is no longer significant to affect the control signal.
- The PD control can cause overdamping to the system, though the system has faster rise time and settling time ,i.e., reaches steady-state more rapidly.

Op - Amp PD Controller - Differential Controller

- Again, using two op amps, two resistors and two capacitors, we can form a differential controller by swopping the locations of the capacitor and first resistor in the Integrator design. This controller can also be used for changing the waveform of the input, for example, a triangle waveform input will be transformed into a square-wave as the slope of the triangle remains constant for the rising and falling in the amplitude.

Differential Controller



- This controller again utilizes two inverting amplifiers that serve to either prove an amplitude boost or unity gain depending on the values of R_2 and R_3. The corresponding transfer function can be defined as:

$$V_o = \left(-R_1 C\,\frac{dv_{in}(t)}{dt}\right)\left(-\frac{R_3}{R_2}\right)V_i \leftrightarrow H(s) = \frac{sCR_1R_3}{R_2}$$

where

$$K_D = -\,R_1 C$$

Effects of Proportional-Integral-Derivative Control on Motor

- Combine all of the previous concepts into one expression for the transfer function

$$u(t) = K_p e(t) + K_i \int e(t)\,dt + K_D \frac{de(t)}{dt}$$

Taking the Laplace Transform,

$$U(s) = K_p E(s) + K_i\,E(s)/s + K_D s E(s)$$

$$U(s) = E(s)\,(K_p + K_i/s + K_D s)$$

$$\frac{U(s)}{E(s)} = \; K_p + K_i/s + K_D s$$

```
% After some experimenting found acceptable value to be
Kp = 6000, Ki = 3600, Kd = 100
```

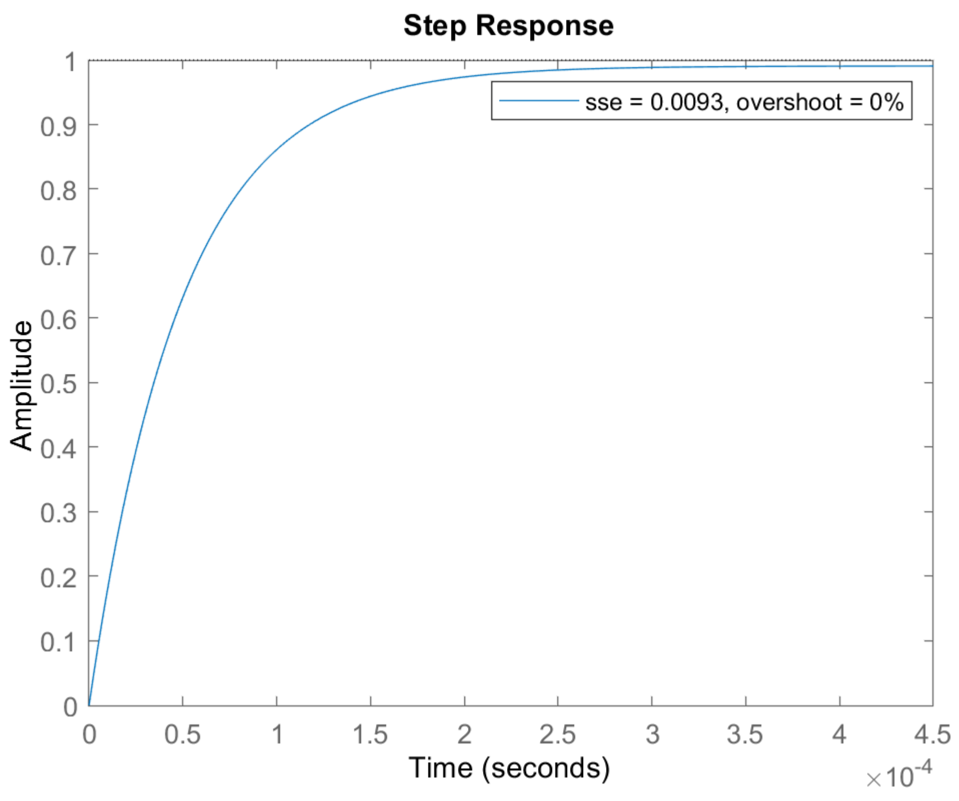
```
Kp = 6000
```

```
Ki = 3600
Kd = 100
```

```
t = 0:.0001:0.2;
C = pid(Kp,Ki,Kd);
Y = feedback(C*P,1)
```

```
Y =

          25 s^2 + 1500 s + 900
  --------------------------------------
  0.00124 s^3 + 25.31 s^2 + 1500 s + 900

Continuous-time transfer function.
```

```
figure
step(Y), y = step(Y);
sserror=abs(1-y(end));
S = stepinfo(Y); overshoot = S.Overshoot;
legend('sse = 0.0093, overshoot = 0%')
```



Observations:

This design includes all of the benefits from the previously described controllers:

- Fast reaction of proportional controller
- Reduction in steady-state error from the PI controller
- Dampening effect of the PD controller

The closed-loop system approaches steady-state in about $25\ \text{msecs}$, has a steady-state error less than .009, and have 0% overshoot.

The Closed-Loop Transfer Function is:

$$\frac{Y(s)}{R(s)} = \frac{25s^2 + 1500s + 900}{0.00124s^3 + 25.31s^2 + 1500s + 900}$$

Op - Amp PID Controller

Here, in order to realize the necessary values for the PID constants, appropriate values for the resistors and capacitors need to be chosen such that the Op-Amp circuits are analogous to the generated MATLAB transfer functions.

The values for the PID constants were:

- $K_p = 6000 = \frac{R_2}{R_1}$
- $K_i = 3500 = -\frac{1}{R_3C_1}$
- $K_D = 100 = -R_4C_2$

After using some common values for resistors and capacitors and solving for unknown variables, I derived the passive components of this PID controller to be:

- $K_p = 6000 = \frac{600k\Omega}{1k\Omega}$
- $K_i = 3500 = -\frac{1}{(1k\Omega)(.278\mu F)}$
- $K_D = 100 = -(1M\Omega)(100\mu F)\backslash$

where resistor values are:

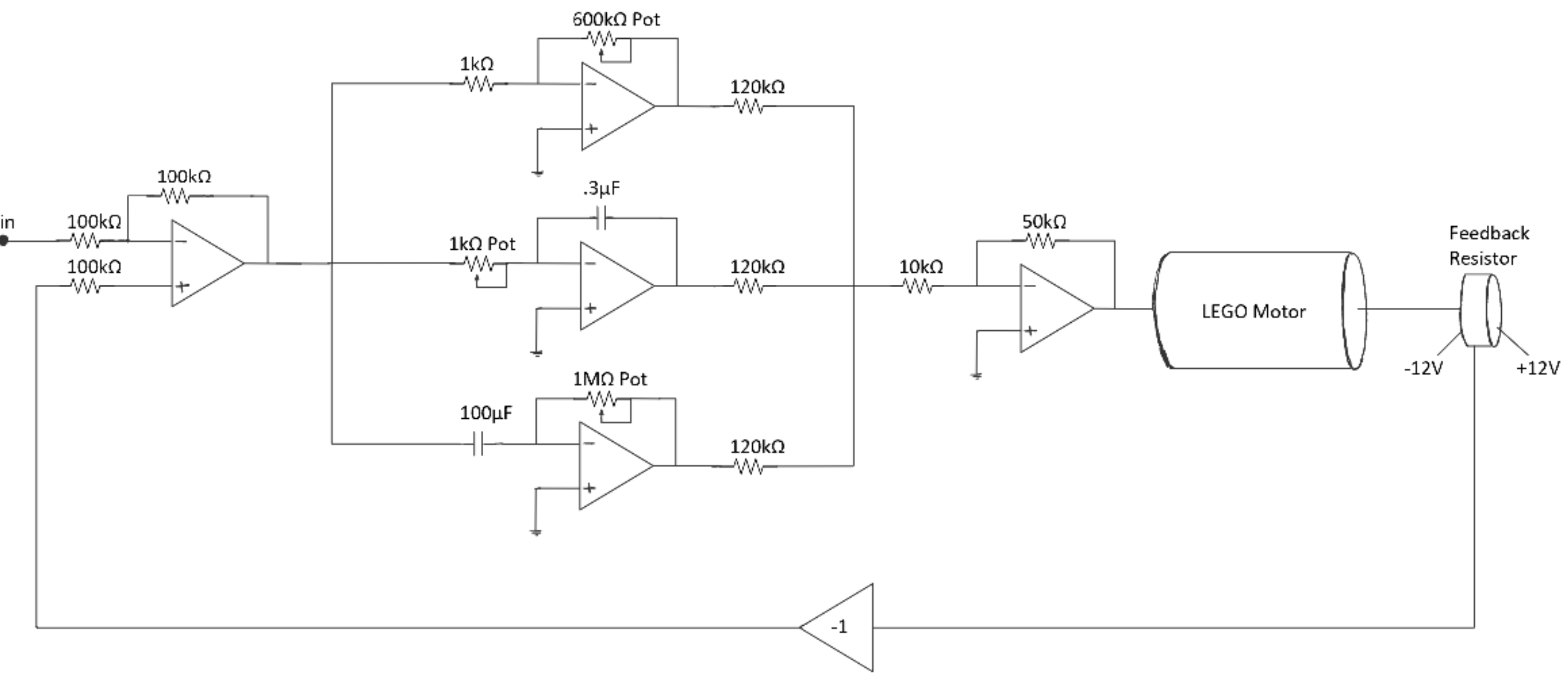
$$R_1 = 1k\Omega,\ R_2 = 600k\Omega,\ R_3 = 1k\Omega,\ R_4 = 1M\Omega$$

and capacitor values are:

$$C_1 = .278\mu F \approx .3\mu F$$

$$C_2 = 100\mu F$$

Analog PID Controller



References

[1] Åström Karl J., & Murray, R. M. (2021). *Feedback systems: An introduction for scientists and Engineers*. Princeton University Press.

[2] Alexander, C. K., & O., S. M. N. (2021). *Fundamentals of Electric Circuits*. McGraw-Hill.

[3] *Introduction: PID Controller Design*. Control Tutorials for MATLAB and Simulink - Introduction: PID Controller Design. (n.d.). Retrieved April 25, 2022, from https://ctms.engin.umich.edu/CTMS/index.php?example=Introduction§ion=ControlPID

[4] “Emf Equation of a DC Generator.” Circuit Globe, November 16, 2015. https://circuitglobe.com/emf-equation-of-dc-generator.html

[5] MIT . (n.d.). *Random Hall Lego Robotics Seminar*. Retrieved from http://web.mit.edu/sp.742/www/motor.html

[6] *Feedback Amplifier Design*. Feedback Amplifier Design - MATLAB & Simulink Example. (n.d.). Retrieved April 25, 2022, from https://www.mathworks.com/help/control/ug/feedback-amplifier-design.html