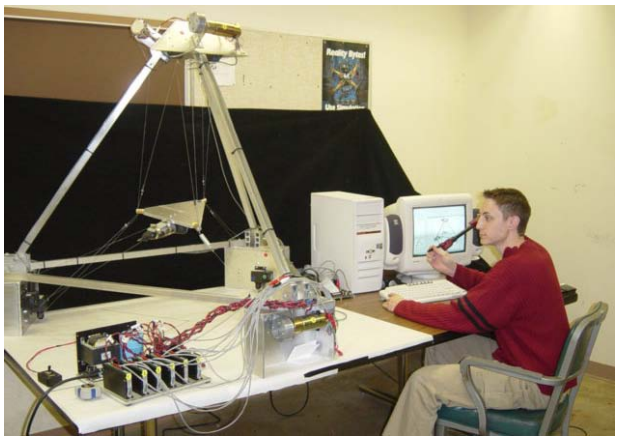
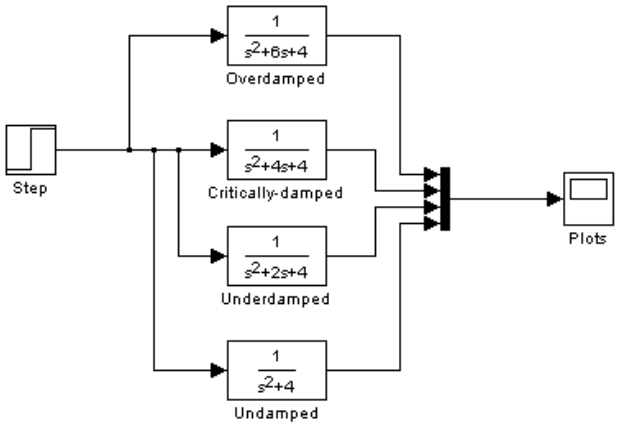
Linear Systems Control for Mechanical Engineers

Dr. Robert L. Williams II
Mechanical Engineering
Ohio University

NotesBook Supplement for:
ME 401 System Analysis and Control

williar4@ohio.edu
www.ohio.edu/people/williar4





These notes supplement the ME 401NotesBook by Dr. Bob

This document presents supplemental notes to accompany the ME 401 NotesBook. The outline given in the Table of Contents on the next page dovetails with and augments the ME 401 NotesBook outline and hence is incomplete here.

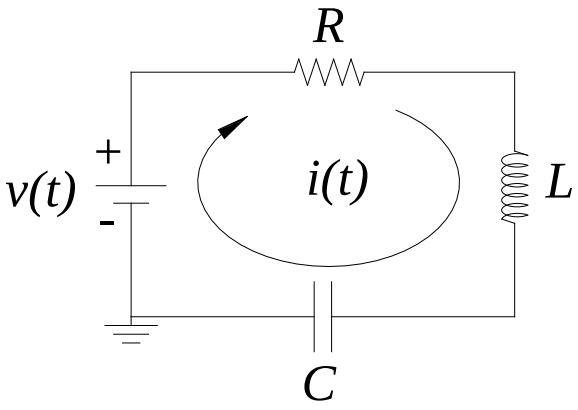
ME 401 Supplement Table of Contents

2. Modeling

2.1 System Modeling

Force-Voltage Analogy

Single series loop with *L*, *R*, *C* elements, voltage input, current output (add a capacitor to the Term Example armature circuit, ignore back-emf). Figure:



Model ODEs: $$L\frac{di(t)}{dt}+Ri(t)+\frac{1}{C}\int i(t)dt=v(t)$$

$$m\frac{dv(t)}{dt}+cv(t)+k\int v(t)dt=f(t)$$

Force-Voltage Analogy

Variable Type	**Translational**	***L*-*R*-*C* Circuit**
input forcing (through)	$f(t)$	$v(t)$ (voltage)
output (across)	$v(t)$ (velocity)	$i(t)$
inertia	m	L
damping	c	R
stiffness	k	$1/C$

3. Linear System Simulation

3.1 Solution of Ordinary Differential Equations

3.1.1 First-Order ODEs

First-Order System Example 2

Same *ck* system and initial condition, but $f(t) = 5\sin 2t$

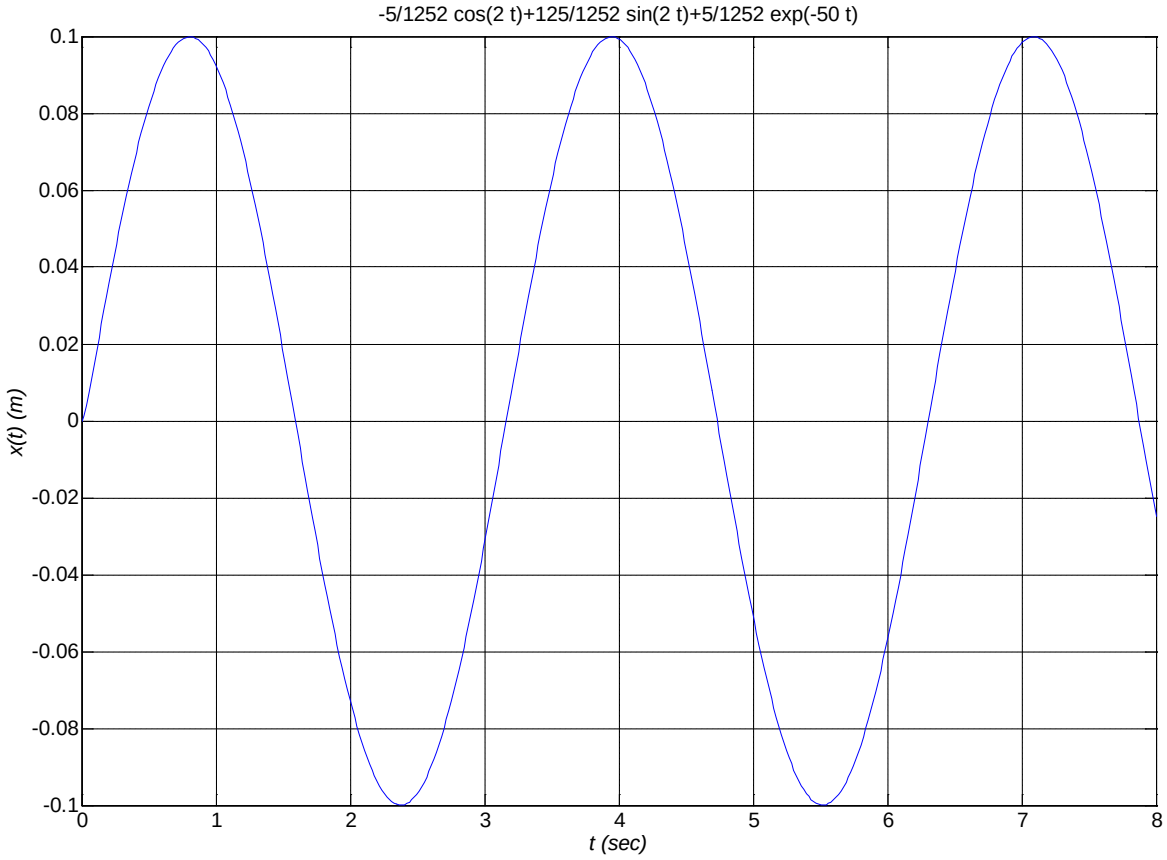
Solve $\dot{x}(t) + 50x(t) = 5\sin 2t$ subject to $x(0) = 0$

Solution:

$$x(t) = \frac{5}{1252}\left(e^{-50t} + 25\sin 2t - \cos 2t\right)$$

equivalent alternate solution form:

$$x(t) = \frac{5}{1252}\left(e^{-50t} + 25.02\sin\left(2t - 0.04\right)\right)$$ using $\sin(a-b) = \sin a\cos b - \cos a\sin b$



- The response $x(t)$ starts at zero as specified by the initial condition.
- The transient solution goes to zero by about t = 0.10 sec.
- Steady-state solution is $x_{ss}(t) = 25.02\sin\left(2t - 0.04\right)$ *m*.
- $\omega = 2$ rad/s $\omega = 2\pi f$ $f = 1/\pi = 1/T$ $T = \pi$ sec

Alternate form for particular solution (Example 2):

Sum-of-angles formulae:

$$\cos(a \pm b) = \cos a \cos b \mp \sin a \sin b$$
$$\sin(a \pm b) = \sin a \cos b \pm c \cos a \sin b$$

$$25\sin 2t - \cos 2t = C\sin(2t - \phi)$$
$$= C\left(\sin 2t \cos\phi - \cos 2t \sin\phi\right)$$

$$\sin 2t : 25 = C\cos\phi$$
$$\cos 2t : -1 = -C\sin\phi$$

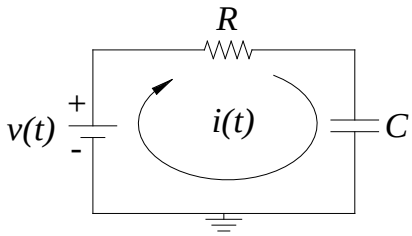
$$C^2\left(\cos^2\phi + \sin^2\phi\right) = 25^2 + 1^2 \qquad C = \sqrt{626} = 25.02$$

$$\frac{\sin\phi}{\cos\phi} = \frac{1}{25} \qquad \phi = \tan^{-1}\left(\frac{1}{25}\right) = 0.04 \ \ rad$$

$$25\sin 2t - \cos 2t = 25.02\sin\left(2t - 0.04\right)$$

In general: $C = \sqrt{B_1^2 + B_2^2}$ $\qquad \phi = \tan^{-1}\left(\frac{-B_2}{B_1}\right)$ $\qquad$ when $\quad x_P(t) = B_1\sin 2t + B_2\cos 2t$
$$= C\sin\left(2t - \phi\right)$$

A Final First-Order ODE example *R*, *C* series electrical circuit (voltage *v*(*t*) input, current *i*(*t*) output):

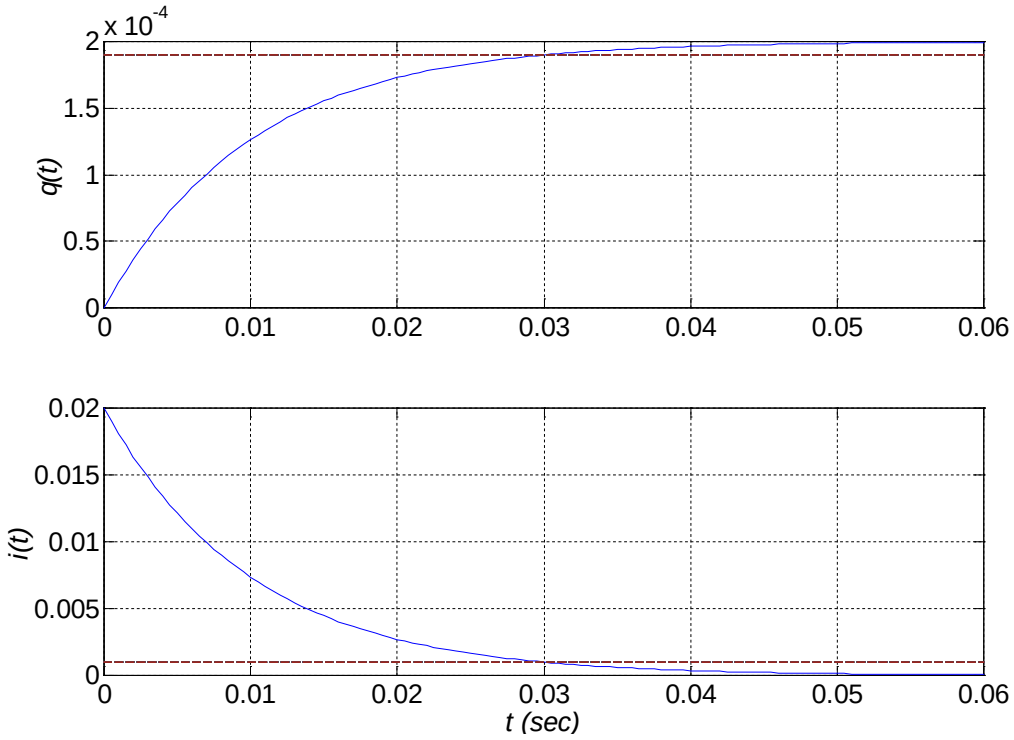


Model (from KVL and electrical circuit element table): $Ri(t)+\frac{1}{C}\int i(t)dt = v(t)$

substitute charge *q*(*t*): $\int i(t)dt = q(t)$ $\quad i(t) = \dot{q}(t)$ $\quad R\dot{q}(t)+\frac{1}{C}q(t) = v(t)$

Given $R = 50\ \Omega$ and $C = 0.2\ mF$, solve: $50\dot{q}(t)+5000q(t)=1$ subject to $q(0)=0$ and a unit step voltage input *v*(*t*). Solution:

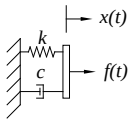
$$q(t) = \frac{1}{5000}\left(1-e^{-100t}\right) \qquad i(t) = \dot{q}(t) = \frac{1}{50}e^{-100t}$$



- As expected from the circuit dynamics, the charge *q*(*t*) in the capacitor builds up to a constant given a constant voltage input.
- Also as expected, the capacitor current *i*(*t*) goes to zero at steady-state.
- The steady state charge value is q_{SS} = 1/5000.
- The time constant is $\tau = RC = 0.01$, so at 3 time constants (*t* = 0.03 *sec*) both the *q*(*t*) and *i*(*t*) values have approached 95% of their respective final values.

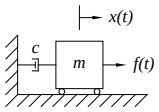
First-Order System Examples

c, *k* massless translational mechanical system
force input $f(t)$ and displacement output $x(t)$



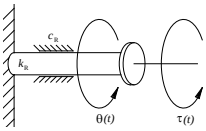
$$c\dot{x}(t) + kx(t) = f(t)$$

m, *c* springless translational mechanical system
force input $f(t)$ and velocity output $v(t)$



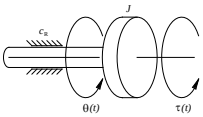
$$m\ddot{x}(t) + c\dot{x}(t) = m\dot{v}(t) + cv(t) = f(t)$$

c_R, k_R massless rotational mechanical system
torque input $\tau(t)$ and angular displacement output $\theta(t)$



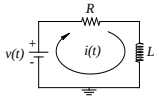
$$c_R\dot{\theta}(t) + k_R\theta(t) = \tau(t)$$

J, c_R springless rotational mechanical system
DC servomotor, torque input $\tau(t)$ and angular velocity output $\omega(t)$



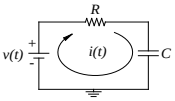
$$J\ddot{\theta}(t) + c_R\dot{\theta}(t) = J\dot{\omega}(t) + c_R\omega(t) = \tau(t)$$

L, *R* series electrical circuit
voltage input $v(t)$ and current output $i(t)$



$$L\frac{di(t)}{dt} + Ri(t) = v(t)$$

R, *C* series electrical circuit
voltage input $v(t)$ and current output $i(t)$, or charge $q(t) = \int i(t)dt$



$$Ri(t) + \frac{1}{C}\int i(t)dt = R\dot{q}(t) + \frac{1}{C}q(t) = v(t)$$

The time constants for these 6 examples are: $\tau \rightarrow \frac{c}{k} \quad \frac{m}{c} \quad \frac{c_R}{k_R} \quad \frac{J}{c_R} \quad \frac{L}{R} \quad \frac{RC}{1}$

All first-order ODEs are solved in the same way as Example 1 (assuming a unit step input), all have the same type of time response graph, all share the same time constant behavior (after three time constants 3τ the output response is within 95% of its final value).

First-order system models are given in:
www.ohio.edu/people/williar4/html/PDF/ModelTFAtlas.pdf.

3.1.2 Second-Order ODEs

Derivation of underdamped homogeneous solution form

In this case we have complex conjugate characteristic polynomial roots: $s_{1,2} = a \pm bi$.

For this example assume $s_{1,2} = -1 \pm 3i$ as in the Section 3.1.2 401 NotesBook example.

Euler's identity must be used:

$$e^{i\theta} = \cos\theta + i\sin\theta$$
$$e^{-i\theta} = \cos(-\theta) + i\sin(-\theta) = \cos(\theta) - i\sin(\theta)$$

$$\begin{aligned} x_H(t) &= A_1 e^{s_1 t} + A_2 e^{s_2 t} \\ &= A_1 e^{(-1+3i)t} + A_2 e^{(-1-3i)t} \\ &= e^{-t}\left(A_1 e^{3it} + A_2 e^{-3it}\right) \\ &= e^{-t}\left(A_1\left(\cos 3t + i\sin 3t\right) + A_2\left(\cos 3t - i\sin 3t\right)\right) \end{aligned}$$

where we used $\theta = 3t$ in Euler's identify. Simplifying by collecting (factoring) terms:

$$x_H(t) = e^{-t}\left(\left(A_1 + A_2\right)\cos 3t + \left(A_1 - A_2\right)i\sin 3t\right)$$

Let
$$B_1 = A_1 + A_2$$
$$B_2 = \left(A_1 - A_2\right)i$$

We can only have real solutions when starting with real ODE coefficients, so $A_1 = A_2^*$ (these constants must be complex conjugates of each other):

$$A_1 = A_{RE} + A_{IM}i \qquad B_1 = 2A_{RE}$$
$$A_2 = A_{RE} - A_{IM}i \qquad B_2 = -2A_{IM}$$

Therefore $x_H(t) = e^{-t}\left(B_1\cos 3t + B_2\sin 3t\right)$

B_i are the real constant unknown homogeneous solution coefficients

And the general underdamped homogeneous solution form given $s_{1,2} = a \pm bi$ is:

$$x_H(t) = e^{at}\left(B_1\cos bt + B_2\sin bt\right)$$

That is, the real part of the poles is placed in the exponential and the imaginary part of the poles is placed as the circular frequency of the *cos* and *sin* functions.

Second-Order System Example 1

$m = 1\ kg,\ c = 7\ Ns/m,\ k = 12\ N/m$

Solve $\ddot{x}(t) + 7\dot{x}(t) + 12x(t) = f(t) = 3u(t)$ subject to $\begin{matrix} x(0) = 0.10m \\ \dot{x}(0) = 0.05m/s \end{matrix}$

This system is overdamped

Real distinct roots, relatively slower response, no overshoot

This solution is left to the interested reader.

Second-Order System Example 2

$m = 1\ kg,\ c = 6\ Ns/m,\ k = 9\ N/m$

Solve $\ddot{x}(t) + 6\dot{x}(t) + 9x(t) = f(t) = 3u(t)$

Subject to $\begin{matrix} x(0) = 0.10m \\ \dot{x}(0) = 0.05m/s \end{matrix}$

1. Homogeneous Solution $\ddot{x}_H(t) + 6\dot{x}_H(t) + 9x_H(t) = 0$

Assume $x_H(t) = Ae^{st}$ $\left(s^2 + 6s + 9\right)Ae^{st} = 0$

Characteristic polynomial: $s^2 + 6s + 9 = \left(s+3\right)^2 = 0$

$s_{1,2} = -3, -3$ Real, repeated roots

Homogeneous solution form $x_H(t) = A_1e^{-3t} + A_2te^{-3t}$

2. Particular Solution $\ddot{x}_P(t) + 6\dot{x}_P(t) + 9x_P(t) = 3$

$x_P(t) = B$ $0 + 6(0) + 9B = 3$ so $x_P(t) = B = 1/3$

3. Total Solution

$$x(t) = x_H(t) + x_P(t) = A_1e^{-3t} + A_2te^{-3t} + 1/3$$
$$\dot{x}(t) = -3A_1e^{-3t} + A_2e^{-3t} - 3A_2te^{-3t}$$

Now apply initial conditions:

$$x(0) = 0.10 = A_1 + A_2(0) + 0.33$$
$$\dot{x}(0) = 0.05 = -3A_1 + A_2$$

$$A_1 = -0.233$$
$$A_2 = -0.65$$

$$x(t) = -\left(0.233 + 0.65t\right)e^{-3t} + 0.33$$

This system is critically-damped

Real, repeated roots – fastest response without overshoot

Check solution

Plug answer $x(t)$ plus its two derivatives into the original ODE. Also check the initial conditions.

Plot – check transient and steady state solutions, plus total solution.

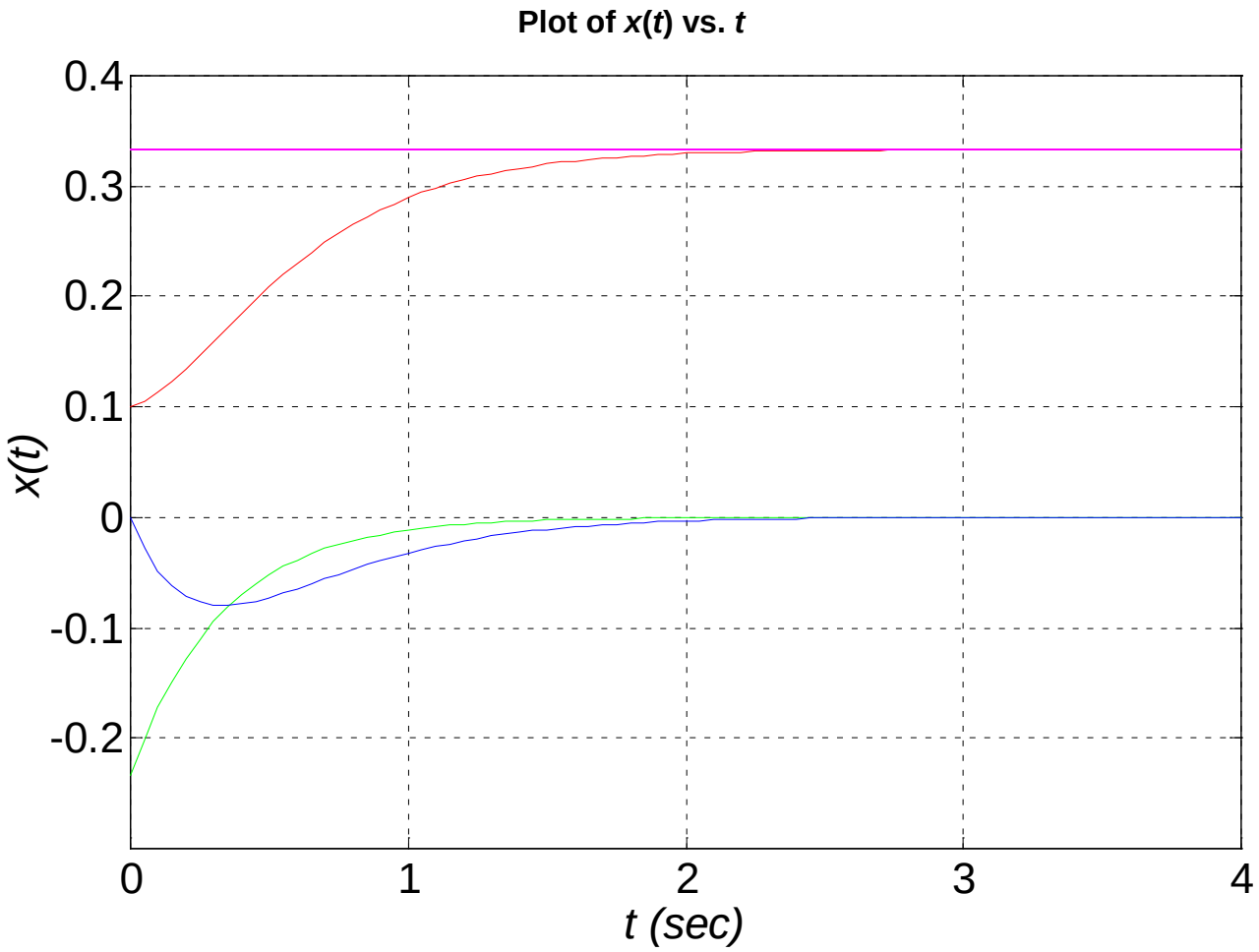
Second-Order System Example 2: Forced *m-c-k* System

Model: $m\ddot{x}(t)+c\dot{x}(t)+kx(t)=f(t)$

$$\ddot{x}(t)+6\dot{x}(t)+9x(t)=3u(t) \qquad \begin{matrix} x(0)=0.10m \\ \dot{x}(0)=0.05m/s \end{matrix}$$

$$s^2+6s+9=(s+3)^2=0$$

Solution: $x(t)=-(0.233+0.65t)e^{-3t}+\frac{1}{3}$



- The total solution $x(t)$ starts at 0.1 *m*, $\dot{x}(t)$ is non-zero, as specified by the initial conditions.
- Transient approaches zero after t = 2.5 *sec*
- Critically damped; –3 root goes to zero slightly faster alone than with ***t***
- Steady-state value is x_{SS} = 1/3 *m*.

Second-Order System Example 4: Forced *m-k* System

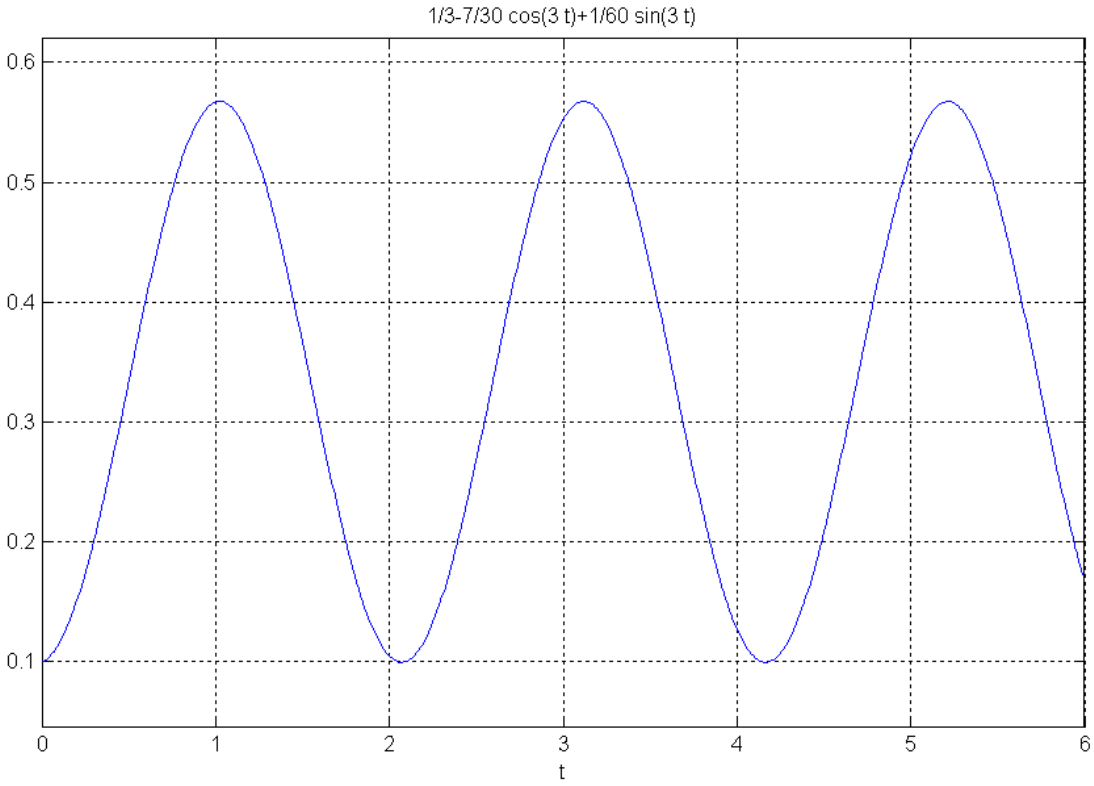
Model: $m\ddot{x}(t) + kx(t) = f(t)$ (no damping)

$$\ddot{x}(t) + 9x(t) = 3u(t) \qquad \begin{matrix} x(0) = 0.10m \\ \dot{x}(0) = 0.05m/s \end{matrix}$$

$$s^2 + 9 = (s - 3i)(s + 3i) = 0$$

Solution: $x(t) = -(7/30)\cos 3t + (1/60)\sin 3t + 1/3$

Plot of *x*(*t*) vs. *t*



- $x(t)$ starts at 0.1 *m*, $\dot{x}(t)$ is non-zero, as specified by the initial conditions.
- Simple harmonic motion
- Undamped; zero viscous damping coefficient
- Transient solution oscillates forever about the particular solution 1/3
- $\omega = 3$ rad/s $\quad \omega = 2\pi f \quad f = 3/2\pi = 1/T \quad T = 2\pi/3 = 2.09$ sec

This system is undamped

Complex-conjugate roots with 0 real part, simple harmonic motion, no damping, theoretically never stops vibrating.

3.2 The Laplace Transform

Partial Laplace Transform Table

	$f(t)$	$F(s)$
1	Dirac delta $\delta(t)$	1
2	unit step $u(t)$	$\frac{1}{s}$
3	unit ramp $r(t) = t$	$\frac{1}{s^2}$
4	t^n	$\frac{n!}{s^{n+1}}$
5	e^{-at}	$\frac{1}{(s+a)}$
6	$1-e^{-at}$	$\frac{a}{s(s+a)}$
7	$\frac{1}{ab}+\frac{e^{-at}}{a(a-b)}+\frac{e^{-bt}}{b(b-a)}$	$\frac{1}{s(s+a)(s+b)}$ $a \neq b$
8	$t^n e^{-at}$	$\frac{n!}{(s+a)^{n+1}}$
9	$\sin \omega t$	$\frac{\omega}{s^2+\omega^2}$
10	$\cos \omega t$	$\frac{s}{s^2+\omega^2}$
11	$e^{-at} \sin \omega t$	$\frac{\omega}{(s+a)^2+\omega^2}$
12	$e^{-at} \cos \omega t$	$\frac{s+a}{(s+a)^2+\omega^2}$

Partial Laplace Transform Table (continued)

	***f*(*t*)**	**F(*s*)**
13	$e^{-at}\left[\cos\omega t+\left(\frac{b-a}{\omega}\right)\sin\omega t\right]$	$\frac{s+b}{(s+a)^2+\omega^2}$
14	$\frac{\sqrt{(b-a)^2+\omega^2}}{\omega}e^{-at}\sin(\omega t+\phi)\quad \phi=\tan^{-1}\frac{\omega}{b-a}$	$\frac{s+b}{(s+a)^2+\omega^2}$
15	$\frac{\omega_n}{\sqrt{1-\xi^2}}e^{-\xi\omega_n t}\sin\omega_d t\,;\quad \omega_d=\omega_n\sqrt{1-\xi^2}\,;\quad \xi<1$	$\frac{\omega_n^2}{s^2+2\xi\omega_n s+\omega_n^2}$
16	$1-\frac{e^{-\xi\omega_n t}}{\sqrt{1-\xi^2}}\sin(\omega_d t+\phi)\,;\quad \phi=\cos^{-1}\xi\,;\quad \xi<1$	$\frac{\omega_n^2}{s(s^2+2\xi\omega_n s+\omega_n^2)}$
17	$\frac{1}{a^2+\omega^2}+\frac{e^{-at}}{\omega\sqrt{a^2+\omega^2}}\sin(\omega t-\phi)\,;\quad \phi=\tan^{-1}\left(\frac{\omega}{-a}\right)$	$\frac{1}{s[(s+a)^2+\omega^2]}$
18	$\frac{b}{a^2+\omega^2}+\frac{1}{\omega}\sqrt{\frac{(b-a)^2+\omega^2}{a^2+\omega^2}}e^{-at}\sin(\omega t+\phi)$ $\phi=\tan^{-1}\left(\frac{\omega}{b-a}\right)-\tan^{-1}\left(\frac{\omega}{-a}\right)$	$\frac{s+b}{s[(s+a)^2+\omega^2]}$

3.2.2 ODE Solution via Laplace Transforms

This subsection presents an alternate solution for the 1st-order ODE problem from the 401 NotesBook™. Again, this ODE is solved using the Laplace Transform method – but now, if we have a Laplace Transform table of sufficient detail, as on the preceding pages, we can skip the partial fraction expansion step and use the table directly.

ODE Solution via Laplace Transforms Examples

First-Order *ck* mechanical system Example 1

Solve $\dot{x}(t) + 50x(t) = 5$ for $x(t)$, subject to $x(0) = 0$ and a step input of magnitude 5.

We already did this ODE solution via the slow ME way and the Laplace Transform method with partial fraction expansion.

Take Laplace transform of both sides (don't forget the initial condition – but it is given as zero):

$$\left[sX(s) - x(0)\right] + 50X(s) = \frac{5}{s}$$

$$sX(s) + 50X(s) = \frac{5}{s}$$

Solve for variable of interest, $X(s)$ – the Laplace transform of the answer, $x(t)$:

$$(s+50)X(s) = \frac{5}{s}$$

$$X(s) = \frac{5}{s(s+50)}$$

Up to this point the solution is identical to that in the 401 NotesBook™. But now we can skip the partial fraction expansion if we use the following Laplace Transform table entry:

$$F(s) = \frac{a}{s(s+a)} \qquad \leftrightarrow \qquad f(t) = 1 - e^{-at}$$

We must algebraically modify $X(s)$ so that the same constant a appears in the numerator and denominator, by multiplying by 1 (10/10):

$$X(s) = \frac{10}{10}\left(\frac{5}{s(s+50)}\right) = \frac{1}{10}\left(\frac{50}{s(s+50)}\right)$$

Taking the inverse Laplace transform of $X(s)$ yields the solution $x(t)$:

$$x(t) = L^{-1}\left\{X(s)\right\} = L^{-1}\left\{\frac{1}{10}\left(\frac{50}{s(s+50)}\right)\right\} = \frac{1}{10}L^{-1}\left\{\left(\frac{50}{s(s+50)}\right)\right\}$$

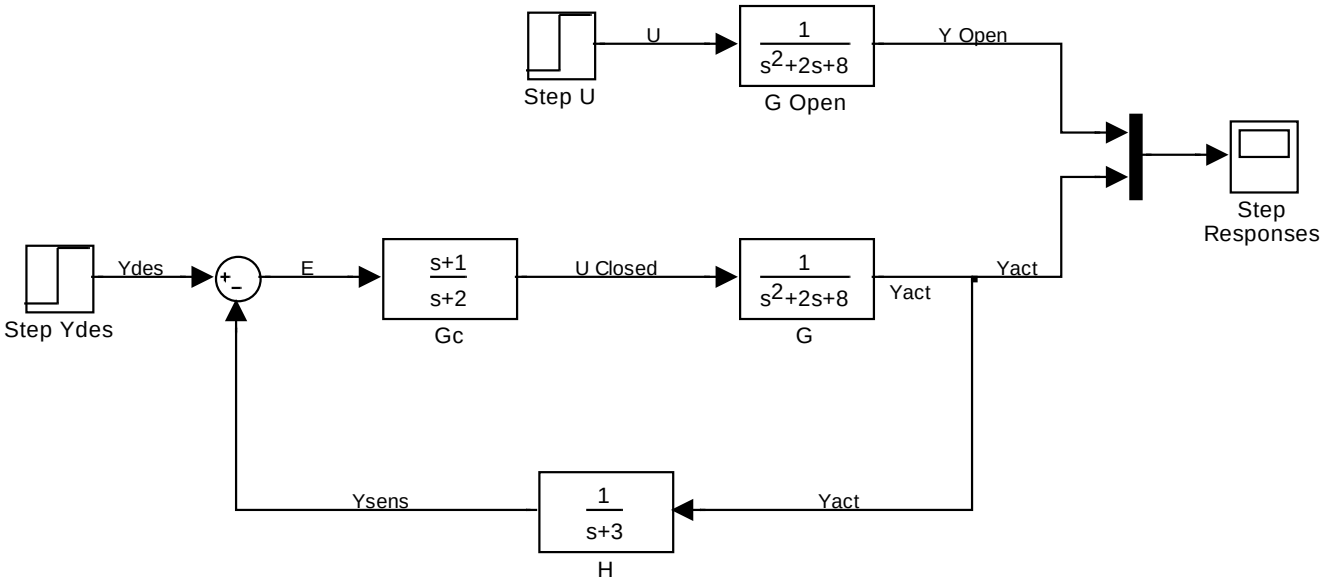
$$x(t) = \frac{1}{10}(1 - e^{-50t})$$

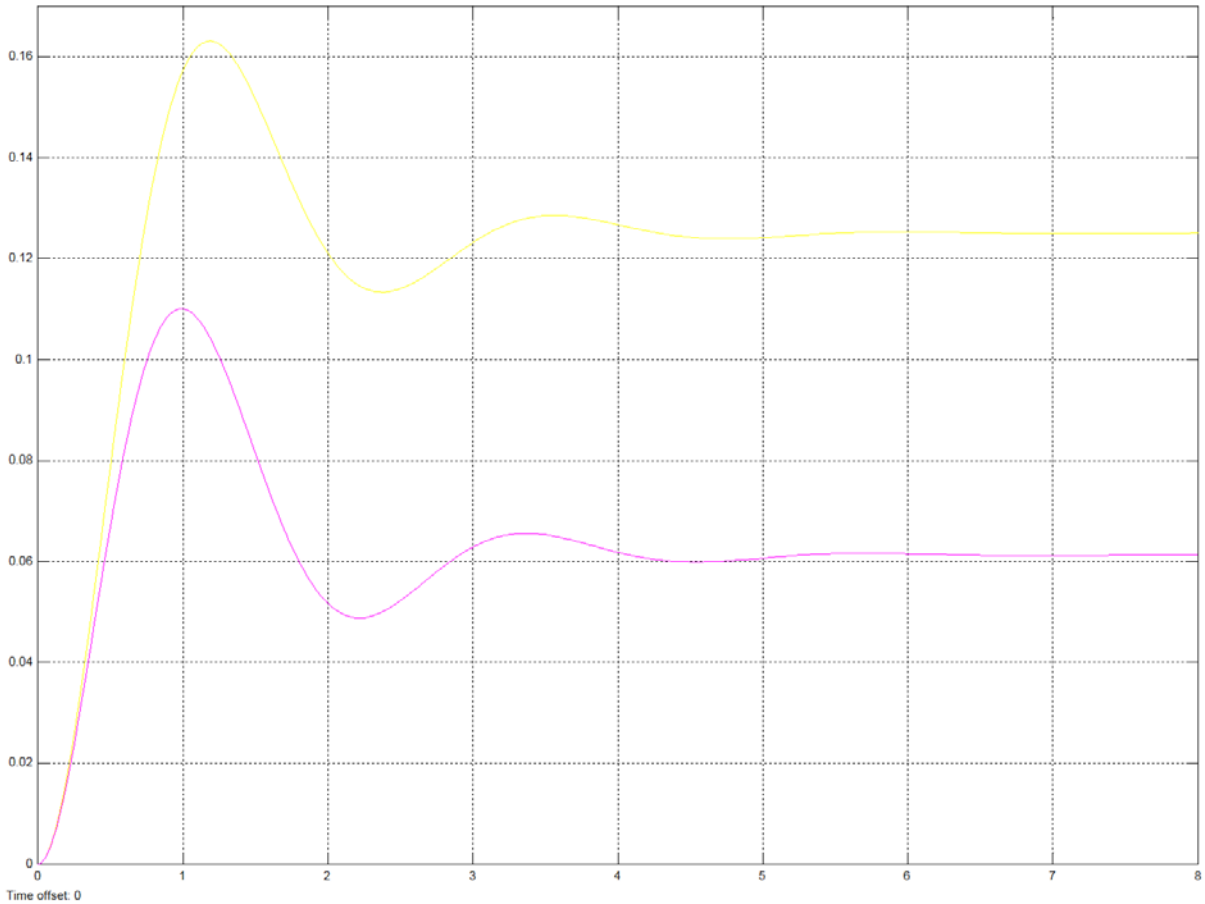
This is the same solution obtained twice previously.

4. Transfer Functions and Block Diagrams

4.2 Block Diagrams

Simulink model and plots for the Example of Section 4.2:





Open-loop Closed-loop

4.4 Feedback

Four reasons for using feedback:

1. To modify the transient response of the system. We can change open-loop poles to closed-loop poles with more desirable behavior to ensure stability and modify the system transient performance.

2. To reduce steady-state error in the system.

3. To decrease the sensitivity of the closed-loop system to variations in the open-loop plant transfer function. Sensitivity is like a derivative $\frac{\Delta T(s)}{\Delta G(s)}$.

4. To reduce the effects of disturbances, unmodeled dynamics, uncertainties, nonlinearities, parameters changing with time, and noise. To increase system robustness.

Feedback is not free. A closed-loop feedback system is more expensive and complex, and thus less reliable, than an open-loop system. Therefore, the engineer must determine if closed-loop feedback control is justified or if the open-loop system can perform adequately.

Fortunately for ME 401, there are a host of real-world dynamic systems which demand closed-loop feedback control.

5. Transient Response

5.1 Second-Order System Damping Conditions

Here is the MATLAB program to create the figure for the example of Section 5.1, comparing the overdamped, critically-damped, underdamped, and undamped second-order system responses to a unit step input.

```
%---------------------------------------------------------------------
%   Over-, Critically, Under-, and Undamped cases using step function
%       Dr. Bob, ME 401
%---------------------------------------------------------------------
clear; clc;

num = [1];
denOVER = [1 6 4];            % Overdamped
denCRIT = [1 4 4];            % Critically-damped
denUNDR = [1 2 4];            % Underdamped
denUN   = [1 0 4];            % Undamped

polesOVER = roots(denOVER);   % Poles for each case
polesCRIT = roots(denCRIT);
polesUNDR = roots(denUNDR);
polesUN   = roots(denUN);

OVER = tf(num,denOVER);
CRIT = tf(num,denCRIT);
UNDR = tf(num,denUNDR);
UN   = tf(num,denUN);

t = [0:0.01:8];
[yOVER,xOVER] = step(OVER,t); % Unit step responses
[yCRIT,xCRIT] = step(CRIT,t);
[yUNDR,xUNDR] = step(UNDR,t);
[yUN,xUN]     = step(UN,t);

figure;
plot(t,yOVER,'r',t,yCRIT,'g',t,yUNDR,'b',t,yUN,'m');  % Plot unit step responses
set(gca,'FontSize',18);
grid; ylabel('\ity(t)'); xlabel('\itt (\itsec)');
legend('Over','Crit','Under','Un');
```

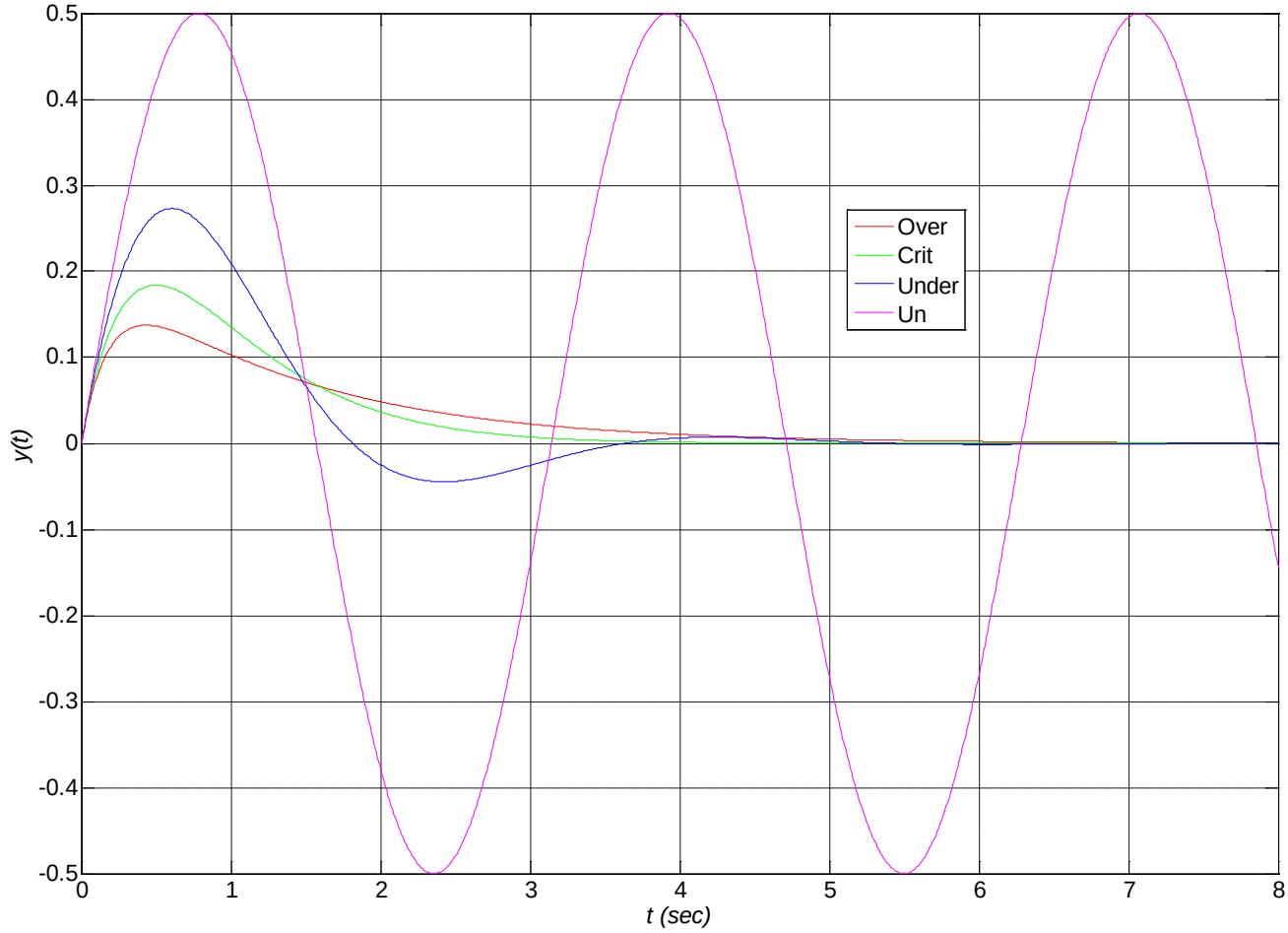
Over-, Critically-, Under-, and Un-damped Examples using `impulse`

Solve:
$$\ddot{y}(t)+6\dot{y}(t)+4y(t)=\delta(t)$$
$$\ddot{y}(t)+4\dot{y}(t)+4y(t)=\delta(t)$$
$$\ddot{y}(t)+2\dot{y}(t)+4y(t)=\delta(t)$$
$$\ddot{y}(t)+4y(t)=\delta(t)$$

for $y(t)$, all subject to zero initial conditions $\begin{matrix} y(0)=0 \\ \dot{y}(0)=0 \end{matrix}$ and an impulse input.

For the impulse responses, solve the same ODEs as the step examples in the 401 NotesBook, subject to zero initial conditions and impulse input $\delta(t)$. Note that though we specified zero initial conditions $y(0)=0$ and $\dot{y}(0)=0$, for the impulse responses only the $y(0)$ initial condition can be satisfied, i.e. all $\dot{y}(0)$ initial velocities take their own value, different from zero as seen below.

Impulse responses plot:



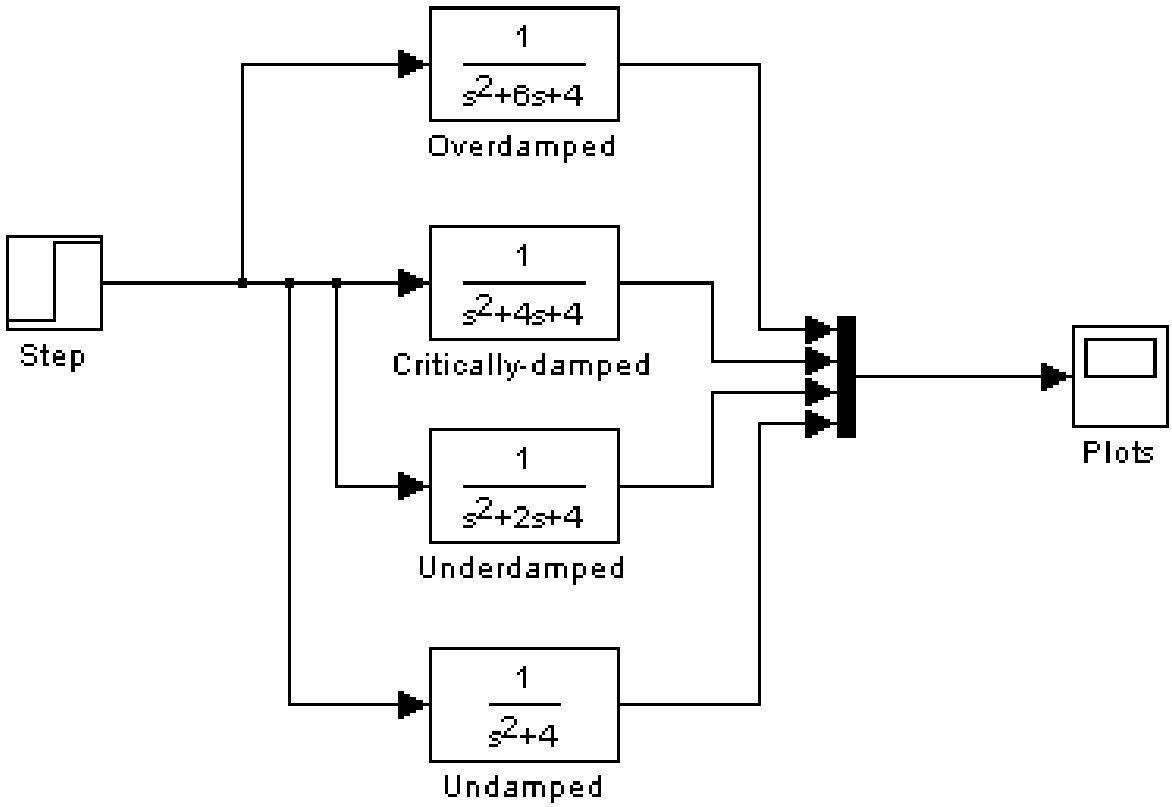
The impulse input final value is 0.

The above figure was generated using the same MATLAB program as given above, but substituting MATLAB function **`impulse`** for **`step`**.

MATLAB Simulink

MATLAB has a powerful graphical user interface (GUI) called **Simulink**. From the MATLAB command window type **`simulink`** and you can create and simulate any control system described by block diagrams and transfer functions. Look around to find the transfer function, step input, and scope (plot) output blocks. Simply drag and drop the desired block to the Simulink workspace. Double-click on any block to change its parameters. Connect blocks with lines using the mouse/cursor. Simply press the play button to run your model. It's fast, fun, and addicting! Make sure you can do the steps by hand, but you are free to use this tool for homework.

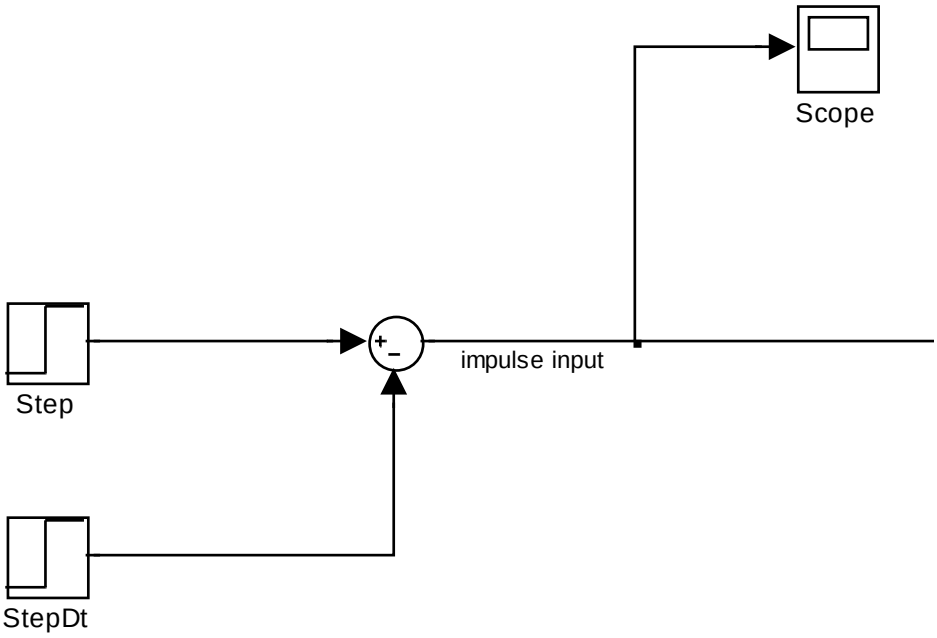
For example, the Simulink diagram below recreates the unit step responses for the overdamped, critically-damped, underdamped, and undamped second-order system example of Section 5.1. All transfer functions are fed by the same **`Step`** input. The mux, or multiplexer, block is used to compare 4 results in one plot, in this case combining 4 scalar signals to one vector signal, to send to the **`Plots`** scope. The resulting plot is identical in form to that shown in the 401 NotesBook, though the Simulink graphics do not look as nice.



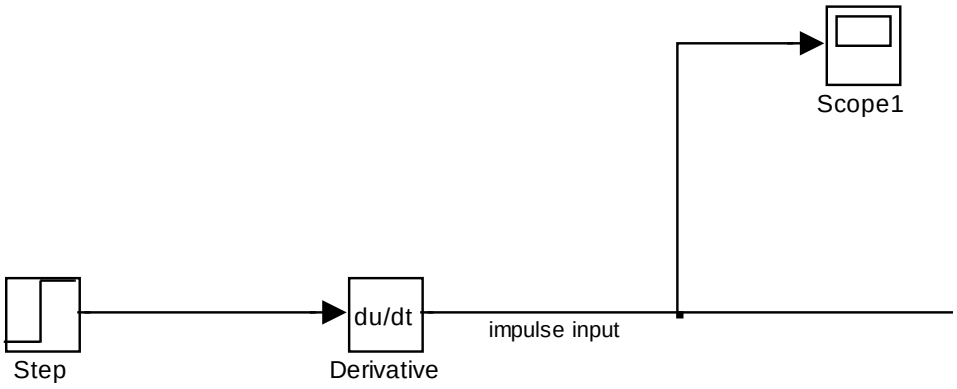
Impulse Input in Simulink

One can simulate the impulse responses in Simulink also. Since there is no **`Impulse`** input block, we must create our own impulse input. There are two possible methods.

1. The Dirac Delta impulse input $\delta(t)$ should have infinite magnitude but infinitesimal duration. It is normalized in the sense that $\int \delta(t) = 1$, i.e. the area under the 'curve' is 1. One way to approximate the impulse input $\delta(t)$ is to turn on a step input at $t = 0$ with a large magnitude M and then subtract from this step input another of equal magnitude M, starting at time $\Delta t = 1/M$ sec. See the figure below. I have had success with $M = 1000$ and 10000, among others. Be sure to scope the resulting impulse input to ensure it is what you intended.



2. We know that $\delta(t) = \frac{du(t)}{dt}$, where $u(t)$ is the unit step input and $\delta(t)$ is the impulse input. See the figure below. Again, scope the resulting impulse input to ensure it is correct. It appears that this approach may not work when starting at $t = 0$ (it does not generate $\delta(t)$ but gives zero instead). If you start the Simulink simulation at $t = 1$, it should work fine. Again, be sure to scope the resulting impulse input to ensure it is what you intended.



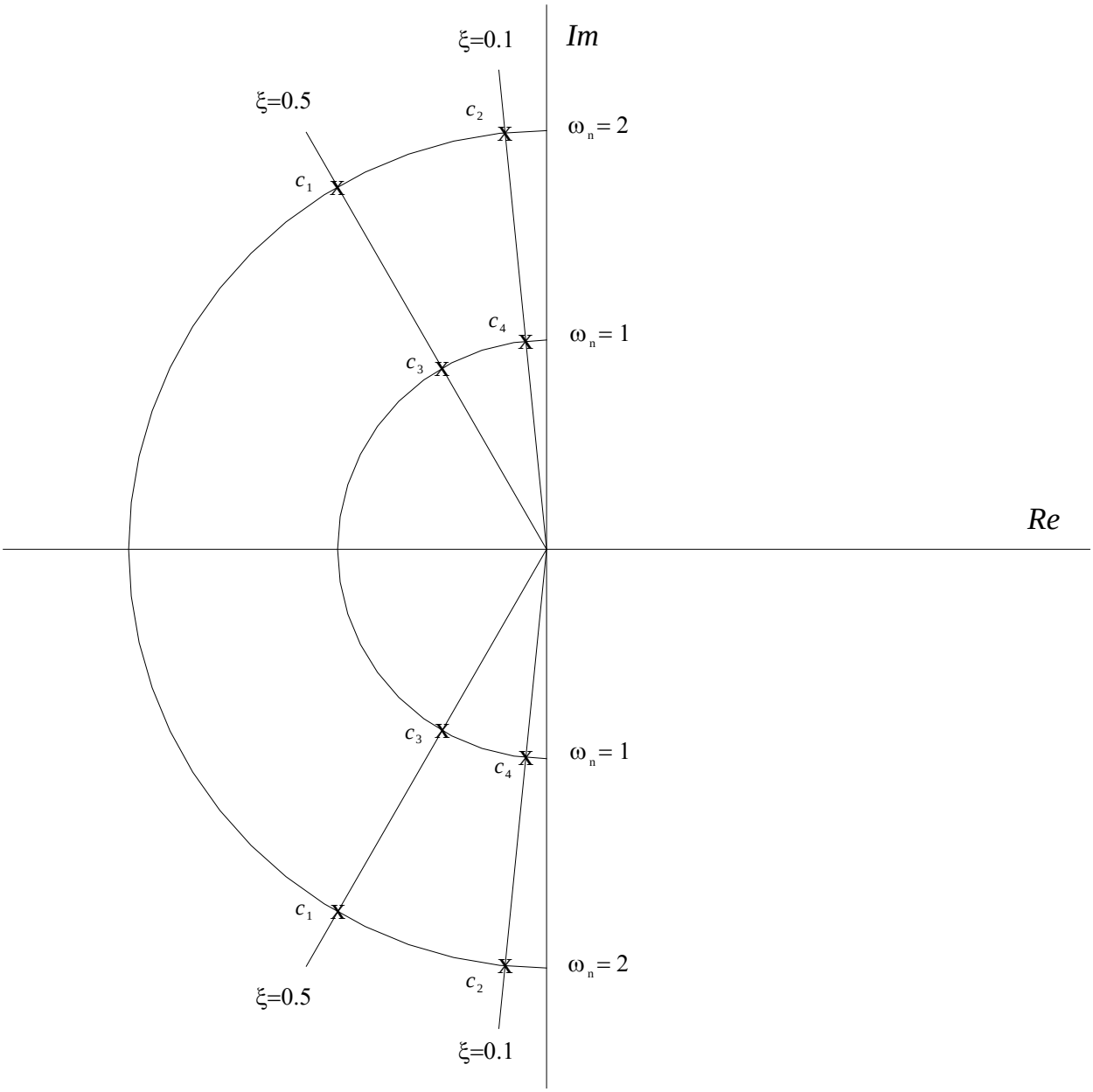
5.3 Open-Loop and Closed-Loop System Examples

Here is the MATLAB program to create the figure for the example of Section 5.3, comparing the open- and closed-loop system responses to a unit step input.

```
%-------------------------------------------
%   Open-loop/Closed-loop Example
%       Dr. Bob, ME 401
%-------------------------------------------
clear; clc;
%   Open-loop unit step response
numo = [4];  deno = [1 2 4];
SysO = tf(numo,deno);
zeroso = roots(numo);  poleso = roots(deno);
[wno,zetao] = damp(deno);
t = [0:0.01:6];
yo = step(SysO,t);  % Open-loop unit step response

%   Closed-loop
K = 10;     % Simple proportional controller
numc = [4*K];  denc = [1 2 4+4*K];
SysC = tf(numc,denc);
zerosc = roots(numc);  polesc = roots(denc);
[wnc,zetac] = damp(denc);
yc = step(SysC,t);  % Closed-loop unit step response

%   Check hand-derivation of T(s)=[numc/denc] via MATLAB
numgc = [K];
dengc = [1];
[numa,dena]   = series(numgc,dengc,numo,deno);
[numc2,denc2] = feedback(numa,dena,[1],[1]);
TCheck = tf(numc2,denc2)      % Display T(s) check results

%   Plot open- and closed-loop step responses
figure;
plot(t,yo,'r',t,yc,'g');
set(gca,'FontSize',18);  legend('Open-loop','Closed-loop');
axis([0 6 0 1.5]);
grid; ylabel('\ity(t)'); xlabel('\itt (\itsec)');

%   Performance specs: open-loop
wdo = wno(1)*sqrt(1-zetao(1)^2);               % Damped natural frequency
tro = (2.16*zetao(1) + 0.60)/wno(1)            % Rise time
tpo = pi/wdo                                   % Peak time
poo = 100*exp(-zetao(1)*pi/sqrt(1-zetao(1)^2))  % Percent overshoot
tso = 4/(zetao(1)*wno(1))                      % Settling time

%   Performance specs: closed-loop
wdc = wnc(1)*sqrt(1-zetac(1)^2);               % Damped natural frequency
trc = (2.16*zetac(1) + 0.60)/wnc(1)            % Percent overshoot
tpc = pi/wdc(1)                                % Peak time
poc = 100*exp(-zetac(1)*pi/sqrt(1-zetac(1)^2))  % Percent overshoot
tsc = 4/(zetac(1)*wnc(1))                      % Settling time

%   Right-click for performance specs: open- and closed-loop
figure;  step(SysO); grid;
figure;  step(SysC); grid;
```

5.4 First- and Second-Order Transient Response Characteristics

The Cartesian representation of the 3x4 subplots (step responses) from the 401 NotesBook Section 5.4 is for convenience only, since that is the way the MATLAB subplot works. For more accuracy, the underdamped cases c. should be represented using polar coordinates (for this example shown to scale below) rather than Cartesian coordinates:



Recall for underdamped poles the polar representation is $r = \omega_n$ and $\theta = \sin^{-1}\xi$; thus the two symmetric angles in the examples above are $\theta = 5.7^\circ$ and $\theta = 30^\circ$.

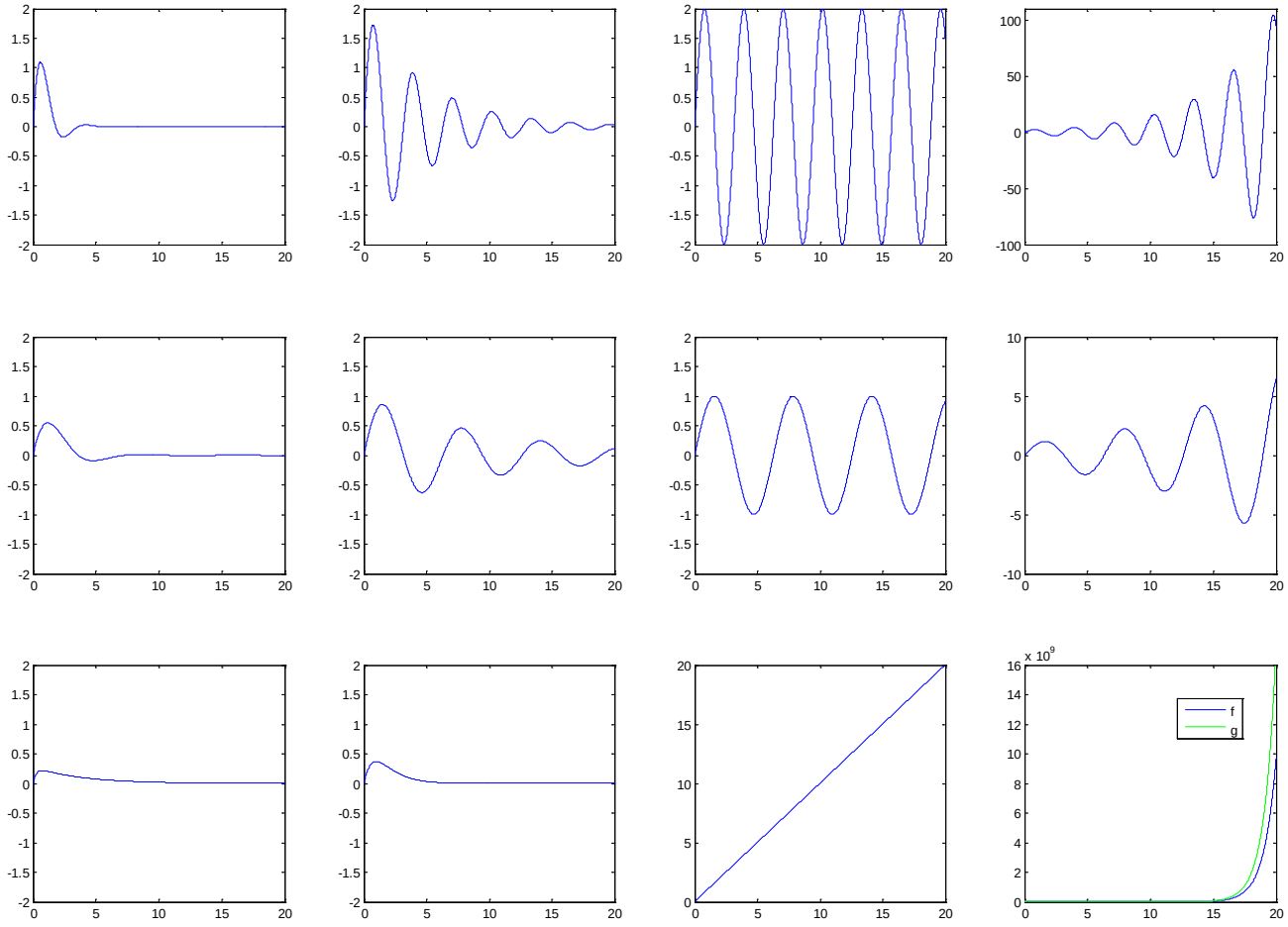
Also, remember all complex conjugates occur in pairs, as shown above (two poles generate one time response plot in all second-order cases, including the non-complex-conjugate cases).

We can also perform the example of Section 5.4 using the **`impulse`** function in place of the **`step`** function; this is shown below.

Transient Response Characteristics vs. *Re-Im*-plane pole locations
(`impulse` responses, generic second-order system)

Figure key:

$\xi = 0.5 \quad \omega_n = 2$ $s_{1,2} = -1 \pm 1.73i$ **c_1. underdamped**	$\xi = 0.1 \quad \omega_n = 2$ $s_{1,2} = -0.2 \pm 1.99i$ **c_2. underdamped**	$\xi = 0 \quad \omega_n = 2$ $s_{1,2} = \pm 2i$ **d_1. undamped**	$\xi = -0.1 \quad \omega_n = 2$ $s_{1,2} = 0.2 \pm 1.99i$ **e_1.**
$\xi = 0.5 \quad \omega_n = 1$ $s_{1,2} = -0.5 \pm 0.866i$ **c_3. underdamped**	$\xi = 0.1 \quad \omega_n = 1$ $s_{1,2} = -0.1 \pm 0.995i$ **c_4. underdamped**	$\xi = 0 \quad \omega_n = 1$ $s_{1,2} = \pm i$ **d_2. undamped**	$\xi = -0.1 \quad \omega_n = 1$ $s_{1,2} = 0.1 \pm 0.995i$ **e_2.**
$\xi = 2 \quad \omega_n = 1$ $s_{1,2} = -3.73, -0.27$ **a. overdamped**	$\xi = 1 \quad \omega_n = 1$ $s_{1,2} = -1$ **b. critically-damped**	$\xi = 0 \quad \omega_n = 0$ $s_{1,2} = 0$ **d_3. special undamped**	$\xi = -1 (-1.005)$ $\omega_n = 1 \; (\omega_n = \sqrt{0.99}\,)$ $s_{1,2} = 1 \; (0.9, 1.1)$ **f. (g.)**



Note: while cases f and g may look identical between the step (401 NotesBook) and impulse (this page) responses, they are different – the impulse responses are going to infinity faster than the step cases.

Here is the MATLAB program to create the figure for the example of Section 5.4, comparing the various second-order system transient responses to a unit step input, based on pole locations. To find the impulse responses of the previous page, simply substitute MATLAB function **`impulse`** for **`step`**.

```
%---------------------------------------------------------------------
%  Generate plots to display transient unit step response characteristics
%       given different dimensionless damping ratios and natural frequencies
%            for the generic second-order system         Dr. Bob, ME 401
%---------------------------------------------------------------------
clc; clear; figure;

dt = 0.01; tf = 20; t = [0:dt:tf];
Xmin = 0; Xmax = tf; Ymin = 0; Ymax = 2;

subplot(3,4,1);
zeta = 0.5;  wn = 2; num = [wn^2]; den = [1 2*zeta*wn wn^2]; y = step(num,den,t);
plot(t,y); axis([Xmin Xmax Ymin Ymax]); r1 = roots(den);

subplot(3,4,2);
zeta = 0.1;  wn = 2; num = [wn^2]; den = [1 2*zeta*wn wn^2]; y = step(num,den,t);
plot(t,y); axis([Xmin Xmax Ymin Ymax]); r2 = roots(den);

subplot(3,4,3);
zeta = 0;    wn = 2; num = [wn^2]; den = [1 2*zeta*wn wn^2]; y = step(num,den,t);
plot(t,y); axis([Xmin Xmax Ymin Ymax]); r3 = roots(den);

subplot(3,4,4);
zeta = -0.1; wn = 2; num = [wn^2]; den = [1 2*zeta*wn wn^2]; y = step(num,den,t);
plot(t,y); axis([Xmin Xmax -50 50]);    r4 = roots(den);

subplot(3,4,5);
zeta = 0.5;  wn = 1; num = [wn^2]; den = [1 2*zeta*wn wn^2]; y = step(num,den,t);
plot(t,y); axis([Xmin Xmax Ymin Ymax]); r5 = roots(den);

subplot(3,4,6);
zeta = 0.1;  wn = 1; num = [wn^2]; den = [1 2*zeta*wn wn^2]; y = step(num,den,t);
plot(t,y); axis([Xmin Xmax Ymin Ymax]); r6 = roots(den);

subplot(3,4,7);
zeta = 0;    wn = 1; num = [wn^2]; den = [1 2*zeta*wn wn^2]; y = step(num,den,t);
plot(t,y); axis([Xmin Xmax Ymin Ymax]); r7 = roots(den);

subplot(3,4,8);
zeta = -0.1; wn = 1; num = [wn^2]; den = [1 2*zeta*wn wn^2]; y = step(num,den,t);
plot(t,y); axis([Xmin Xmax -10 10]);    r8 = roots(den);

subplot(3,4,9);
zeta = 2;    wn = 1; num = [wn^2]; den = [1 2*zeta*wn wn^2]; y = step(num,den,t);
plot(t,y); axis([Xmin Xmax Ymin Ymax]); r9 = roots(den);

subplot(3,4,10);
zeta = 1;    wn = 1; num = [wn^2]; den = [1 2*zeta*wn wn^2]; y = step(num,den,t);
plot(t,y); axis([Xmin Xmax Ymin Ymax]); r10 = roots(den);

subplot(3,4,11);
zeta = 0;    wn = 0; num = [1];    den = [1 0 0];            y = step(num,den,t);
plot(t,y); axis([Xmin Xmax 0 200]);     r11 = roots(den);

subplot(3,4,12);
zeta = -1;   wn = 1; num = [wn^2]; den = [1 2*zeta*wn wn^2]; yf = step(num,den,t);
zeta = -1.005; wn   = sqrt(0.99); num = [wn^2];
den = [1 2*zeta*wn wn^2]; yg = step(num,den,t);
plot(t,yf,'b',t,yg,'g'); legend('f','g'); axis([Xmin Xmax 0 16e09]);
```

5.6 Term Example Open-Loop Transient Response

Here is the MATLAB program to create the figure for the example of Section 5.6, presenting the Term Example open-loop transient responses for load shaft angle and load shaft angular velocity outputs given unit impulse, unit step, and unit ramp inputs.

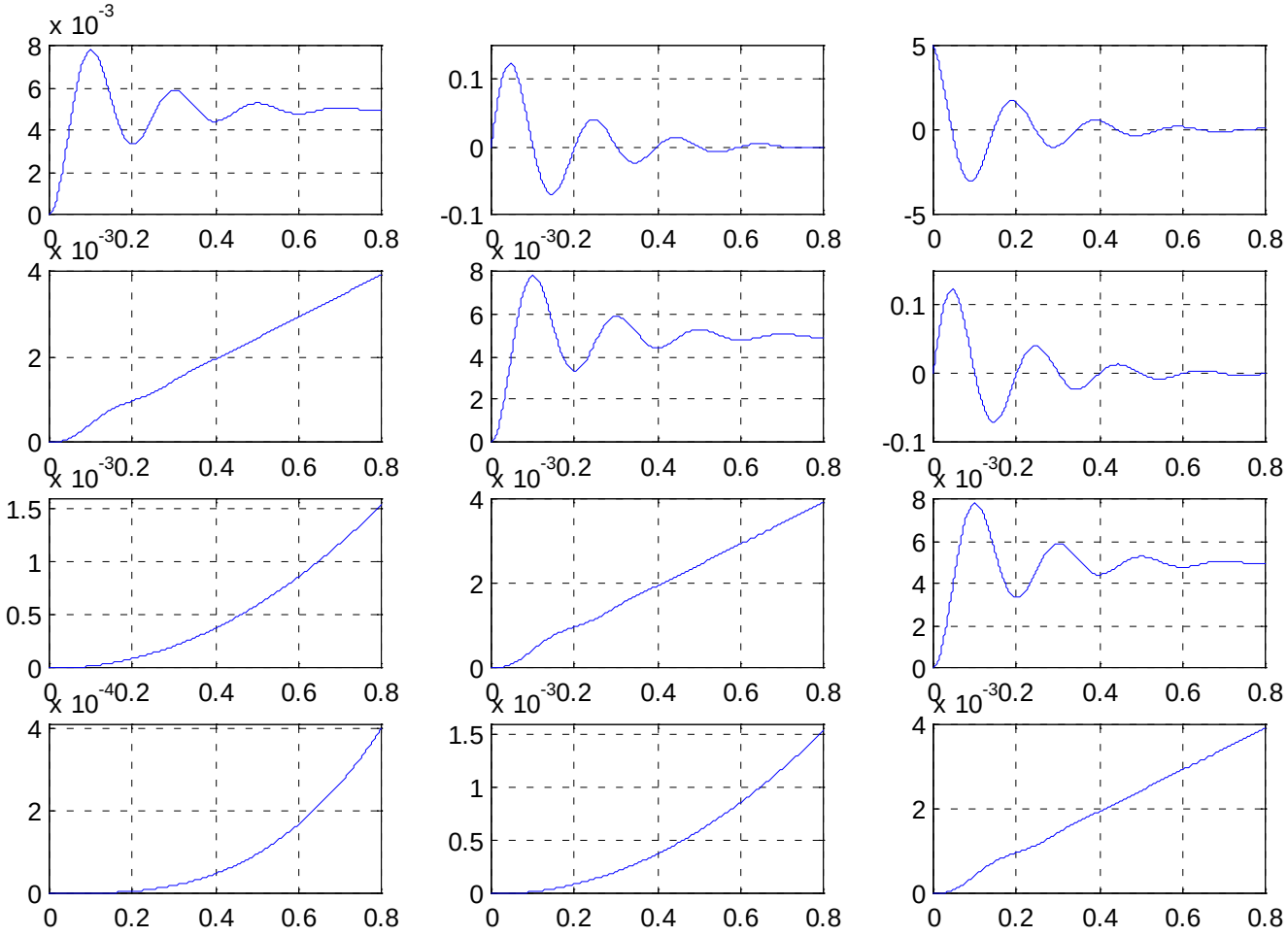
```
%-------------------------------------------------------------------
%  Term example - electromechanical system transient responses
%     Theta and Omega outputs for impulse, step, and ramp inputs
%        Dr. Bob ME 401
%-------------------------------------------------------------------

clc; clear;

numt  = [5];  dent  = [1 11 1010 0];   %  open-loop transfer function  V to ThetaL
Syst = tf(numt,dent);
polet = roots(dent);
numw  = [5];  denw  = [1 11 1010];     %  open-loop transfer function  V to OmegaL
Sysw = tf(numw,denw);
polew = roots(denw);

t0 = 0; dt = 0.005; tf = 0.8;          %  evenly-spaced time array
t  = [t0:dt:tf];

figure;

subplot(321);
[y,x] = impulse(Syst,t);               %  impulse
plot(t,y); grid; axis([0 0.8 0 0.008]);

subplot(322);
[y,x] = impulse(Sysw,t);
plot(t,y); grid; axis([0 0.8 -0.1 0.15]);

subplot(323);                          %  step
[y,x] = step(Syst,t);
plot(t,y); grid; axis([0 0.8 0 0.004]);

subplot(324);
[y,x] = step(Sysw,t);
plot(t,y); grid; axis([0 0.8 0 0.008]);

subplot(325);
[y,x] = lsim(Syst,t,t);                % unit ramp input u(t) = t
plot(t,y); grid; axis([0 0.8 0 0.0016]);

subplot(326);
[y,x] = lsim(Sysw,t,t);
plot(t,y); grid; axis([0 0.8 0 0.004]);
```

Term Example Transient Response, Extended

We can extend the Term Example transient responses example from 401 NotesBook Section 5.6 for one more input (the unit parabola, $p(t) = t^2/2$) and one more output ($\alpha_L(t)$, the angular acceleration of the load shaft).

Figure key:

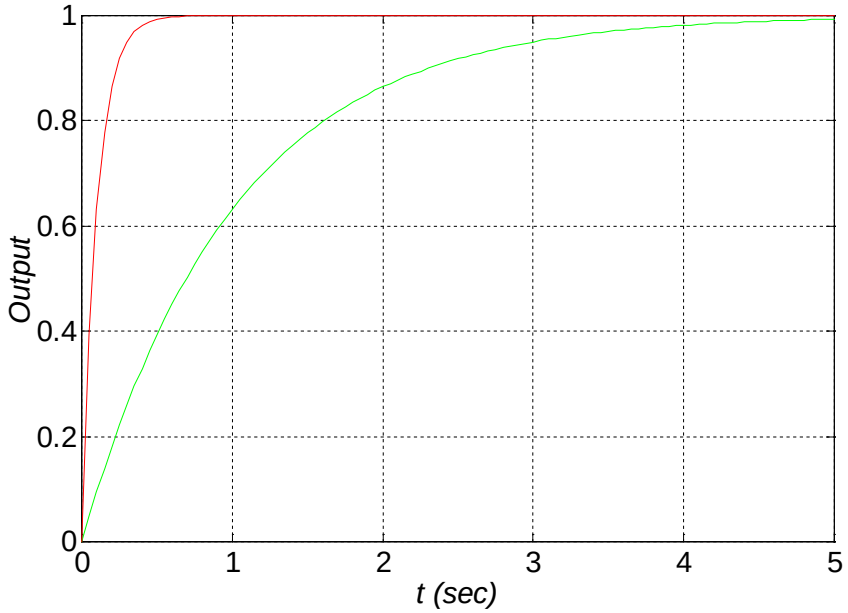
$\theta_L(t)$ impulse	$\omega_L(t)$ impulse	$\alpha_L(t)$ impulse
$\theta_L(t)$ unit step	$\omega_L(t)$ unit step	$\alpha_L(t)$ unit step
$\theta_L(t)$ unit ramp	$\omega_L(t)$ unit ramp	$\alpha_L(t)$ unit ramp
$\theta_L(t)$ unit parabola	$\omega_L(t)$ unit parabola	$\alpha_L(t)$ unit parabola



Electrical vs. Mechanical Rise Time

Usually the electrical system time constant L/R is small relative to the mechanical system time constant J_E/c_E. This means that when voltage $v_A(t)$ is applied to the armature circuit, the armature current $i_A(t)$ rises much faster than the motor shaft angular velocity $\omega_M(t)$ does when $i_A(t)$ is applied to generate motor torque $\tau_M(t)$. See the figure below. Here are the component first-order transfer functions and time constants for the armature circuit and rotational mechanical system dynamics:

$$G_1(s)=\frac{I_A(s)}{V_A(s)-V_B(s)}=\frac{1}{Ls+R} \qquad \frac{L}{R}=0.1 \qquad G_3(s)=\frac{\Omega_M(s)}{\mathrm{T}_M(s)}=\frac{1}{J_E s+c_E} \qquad \frac{J_E}{c_E}\cong 1$$



armature current motor shaft angular velocity

Therefore, the electromechanical system open-loop block diagram transfer function could be simplified as follows. For $V_A(s)$ input, $\Theta_L(s)$ output, the open-loop transfer function is:

$$G_\theta(s)=\frac{K_T}{\left[(Ls+R)(Js+C)+K_M K_B\right]sn}=\frac{\frac{K_T}{R}}{\left[\left(\frac{L}{R}s+1\right)(Js+C)+\frac{K_T K_B}{R}\right]sn}$$

Set $\frac{L}{R}\to 0$ relative to $\frac{J}{c}=\frac{J_E}{c_E}$ since the mechanical system dominates.

$$G_\theta(s)=\frac{\frac{K_T}{R}}{\left[(Js+C)+\frac{K_T K_B}{R}\right]sn}$$

The simplified $V_A(s)$ input, $\Theta_L(s)$ output transfer function is second-order and the simplified $V_A(s)$ input, $\Omega_L(s)$ output transfer function is first-order:

$$G_\theta(s)=\frac{\frac{K_T}{n}}{\left[JRs+(RC+K_T K_B)\right]s} \qquad G_\omega(s)=\frac{\frac{K_T}{n}}{JRs+(RC+K_T K_B)}$$

6. Controller Design

6.1 Controller Design Introduction

Second-order performance specifications revisited: inequalities

For controller design, how should we choose good behavior? One must specify good poles as an input to the controller design process. Thus far we have specified an exact desired percent overshoot (yielding an exact ξ) and an exact settling time (yielding an exact ω_n knowing ξ from *PO*), from which the desired behavior characteristic polynomial is $s^2 + 2\xi\omega_n s + \omega_n^2$, from which the desired closed-loop poles can be found.

A much more general and powerful method for specifying good controller behavior (desired closed-loop poles) is to specify inequalities for performance specifications rather than exact values. In this example we require $t_P \le 2$ sec, $PO \le 5\%$, and $t_S \le 4$ and the problem is to determine and show on the *Re-Im* plane acceptable regions for the closed-loop controller poles given these three simultaneous inequality constraints.

a. From $t_P = \dfrac{\pi}{\omega_n\sqrt{1-\xi^2}} = \dfrac{\pi}{\omega_d} \le 2$

we have $\omega_d \ge \pi/2$ rad/sec (by symmetry, also $-\omega_d \le -\pi/2$ rad/sec).

b. From $PO = 100e^{\left(\frac{-\xi\pi}{\sqrt{1-\xi^2}}\right)} \le 5\%$

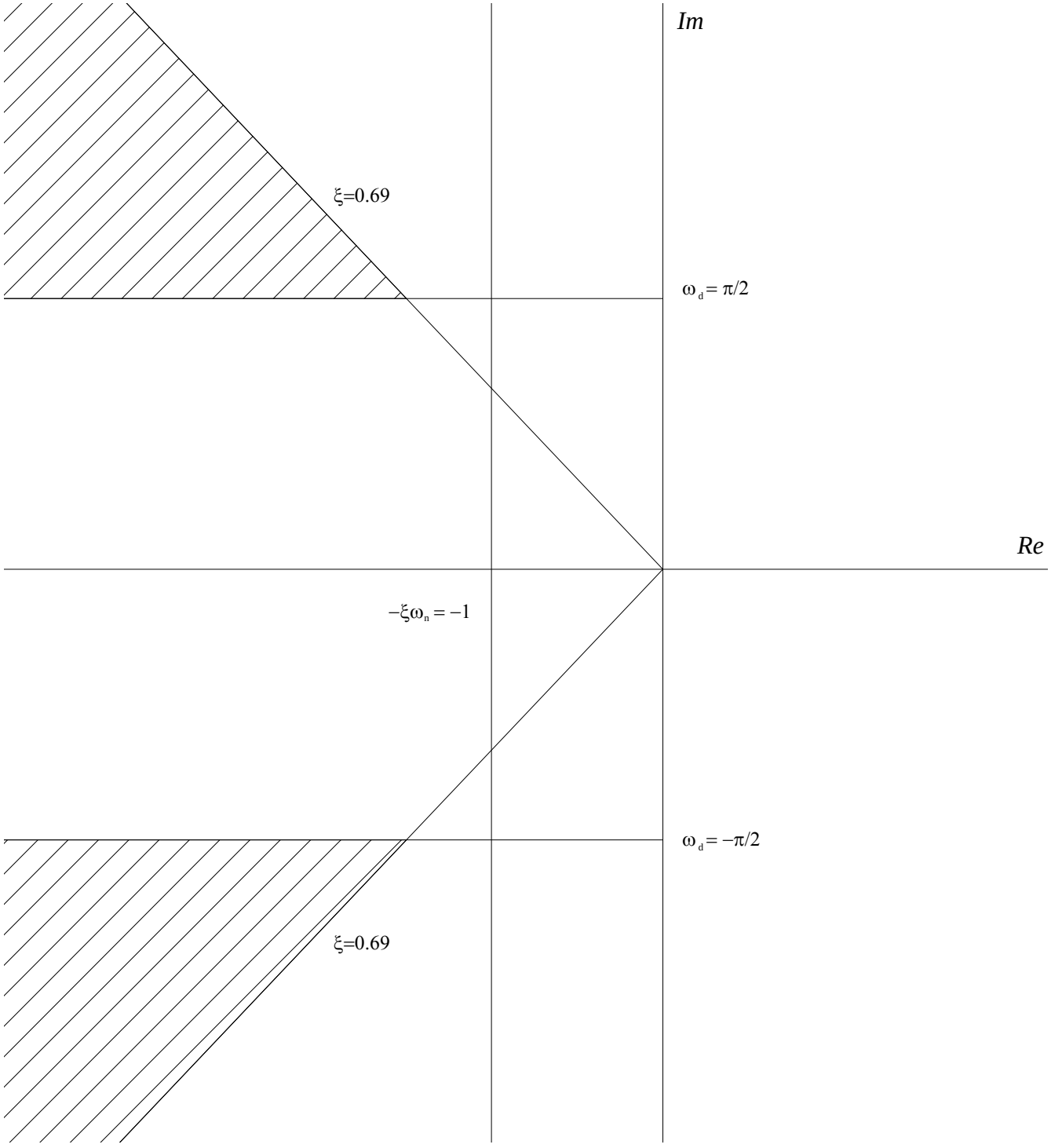
we have $\xi \ge 0.69$, or $\theta \ge 43.6^\circ$ (symmetric)

c. From $t_S \cong \dfrac{4}{\xi\omega_n} \le 4$

we have $\xi\omega_n \ge 1$, that is $-\xi\omega_n \le -1$

Let us plot all three constraints on the *Re-Im* plane; we get three straight-line constraints, two of them symmetric about the *Re* axis. Then we shade the acceptable side of each constraint line.

Plot inequality constraints on Re-Im pole plane:



Acceptable range (cross-hatched) for three performance specification inequalities

To satisfy the inequality requirements on the performance specifications, one may choose two symmetric underdamped complex-conjugate poles anywhere in the shaded regions. The peak time and percent overshoot specifications dominate and the settling time constraint is not active.

***ITAE* Performance Index** Alternate method for choosing good desired closed-loop poles.

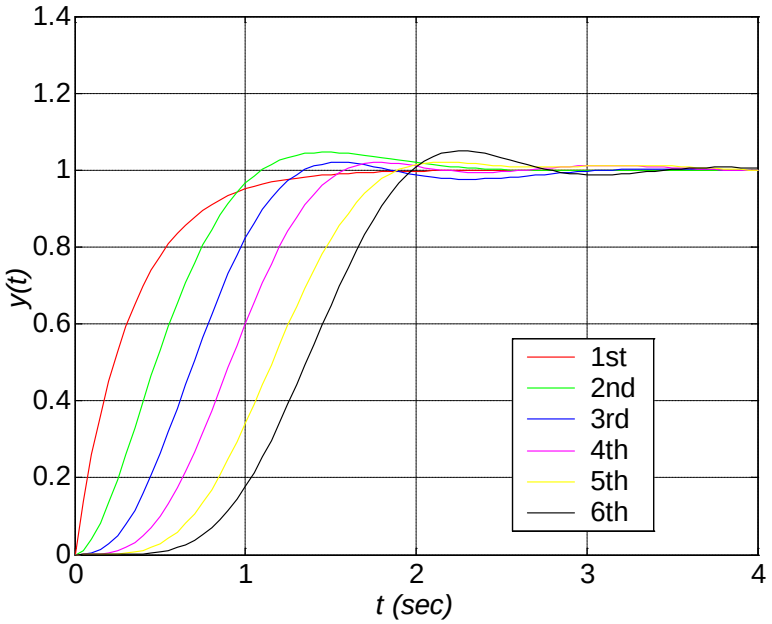
control swiftness of response: change rise and peak time
control error of response: change % overshoot, settling time

These are competing requirements. How should we choose the desired poles for the closed-loop system controller design? Try performance indices – many cases have been solved to tell you the optimal poles given various performance measures (indices). Here we will only consider one: the ***integral of time multiplied by the absolute error* (*ITAE*)**.

$$ITAE = \int_0^\infty t\left|e(t)\right| dt$$

Minimize *ITAE* to simultaneously optimize competing requirements. Minimum *ITAE* means less time and smaller error in balance. For first- through sixth-order systems, the following characteristic polynomials minimize *ITAE*. If we design the feedback controller to meet one of these specifications, the shaping of the dynamic transient response will be optimized according to *ITAE*. The engineer must set the value for ω_n to suit the time nature of the desired closed-loop system. Note for all cases (except the first order system) some overshoot is required to optimize *ITAE*. We simulate with MATLAB to see this (using ω_n = 3 rad/s for all plots below).

Order	Optimal Characteristic Polynomials
1	$s+\omega_n$
2	$s^2+1.4\omega_n s+\omega_n^2$
3	$s^3+1.75\omega_n s^2+2.15\omega_n^2 s+\omega_n^3$
4	$s^4+2.1\omega_n s^3+3.4\omega_n^2 s^2+2.7\omega_n^3 s+\omega_n^4$
5	$s^5+2.8\omega_n s^4+5.0\omega_n^2 s^3+5.5\omega_n^3 s^2+3.4\omega_n^4 s+\omega_n^5$
6	$s^6+3.25\omega_n s^5+6.6\omega_n^2 s^4+8.6\omega_n^3 s^3+7.45\omega_n^4 s^2+3.95\omega_n^5 s+\omega_n^6$



ITAE Desired Unit Step Responses

6.2 Root-Locus Method

Examples 2: Re-create these examples in MATLAB. For each, be sure to interpret what is happening in the root locus plot.

2a. $G(s) = \dfrac{s+2}{s(s+4)}$

2b. $G(s) = \dfrac{s+1}{s(s+2)(s+4)^2} = \dfrac{s+1}{s(s^3+10s^2+32s+32)}$

2c. $G(s) = \dfrac{s+1}{s(s+2)(s+3)} = \dfrac{s+1}{s(s^2+5s+6)}$

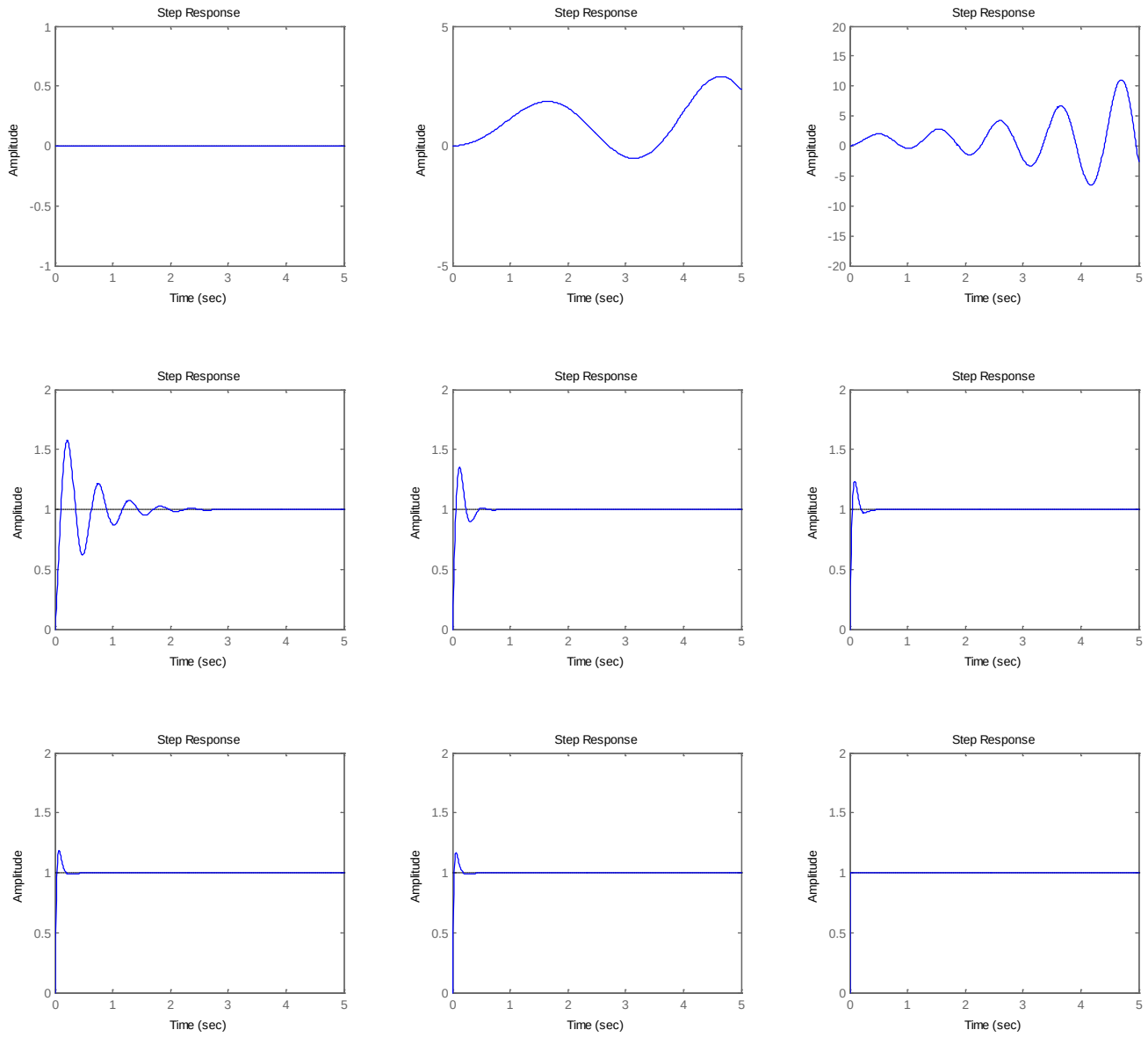
2d. $G(s) = \dfrac{1}{s(s^3+12s^2+64s+128)}$

2e. $G(s) = \dfrac{s^2+14s+96}{s^2(s+14)}$

2f. $G(s) = \dfrac{(s^2+4s+10004)(s^2+12s+90036)}{(s+10)(s^2+2s+2501)(s^2+6s+22509)}$

Example 3 Conditionally stable example (continued)

The unit step responses for Root-Locus Example 3 from the 401 NotesBook are shown below, as K increases. Clearly we need a method for determining K to yield a desirable closed-loop system.



Conditionally-stable example, unit step responses as K increases from 0. K values legend:

0	0.01	0.07
0.40	1.03	1.96
2.73	3.20	6420

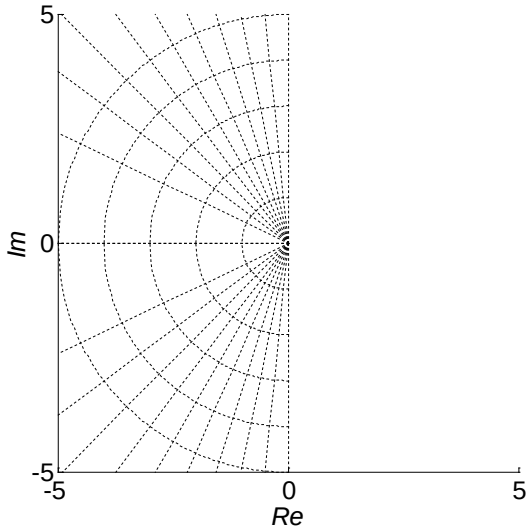
Unstable for $0.001 \le K \le 0.160$ (using **`rlocfind(Sys)`**)

Relate ξ and ω_n to the Re-Im poles plane

Recall the underdamped generic second-order system poles are:

$$s_{1,2} = -\xi\omega_n \pm \omega_n\sqrt{1-\xi^2}\,i = -\xi\omega_n \pm \omega_d i \qquad (0 < \xi < 1)$$

Further recall the graphical representation of these underdamped $s_{1,2}$ pole locations (derived earlier in the 401 NotesBook™, this is symmetric as shown below). We measure θ with the right hand from the positive vertical (by symmetry, measure θ with the left hand from the negative vertical):



The radial distance is ω_n and $\theta = \sin^{-1}\xi$. We can use MATLAB to create ω_n, ξ grids on the root-locus plot. This **`sgrid`** result is shown above and is symmetric about the *Re* axis.

```
dmp = [0:0.1:1];  om  = [1:1:5];  sgrid(dmp,om);
```

Also, we can add vertical lines to the left of zero for lines of constant settling time (i.e. constant $-\xi\omega_n$) and symmetric horizontal lines for lines of constant peak time (i.e. constant $\omega_d = \omega_n\sqrt{1-\xi^2}$).

Put your root-locus plot on the same figure:

```
hold on;
rlocus(Sys);
[K,poles] = rlocfind(Sys)
```

Be sure the root-locus figure is active (if there are other MATLAB graphics figures on the screen). The MATLAB window gives you a cross-hair cursor with which to select a point on the root-locus plot. Choose a point where the root-locus plot intersects a desirable ω_n, ξ or other desirable pole specification. MATLAB responds with the *K* value and associated poles.

For controller design, check out **`rltool`**, very powerful and cool! **`rltool`** has built-in grid tools for two of the generic second-order system performance specifications (settling time and percent overshoot), plus ξ and ω_n: right-click $\rightarrow$ design requirements $\rightarrow$ new.

6.8 Controller Design Example 2

This example uses a similar open-loop transfer function as in Controller Design Example 1, but it is an unstable open-loop plant transfer function (negative damping term):

$$G(s) = \frac{10}{s^2 - s + 10}$$

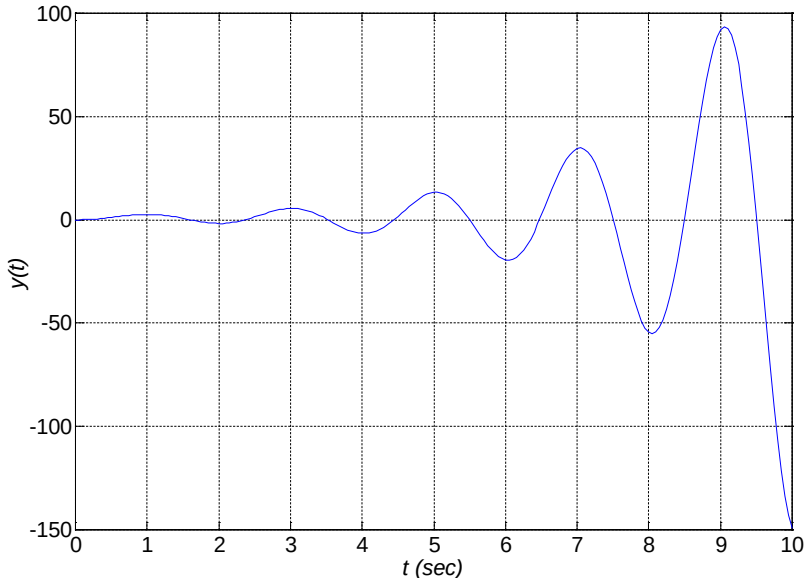
Step 1. Analyze the open-loop system behavior.

There is only one small change in the open-loop transfer function compared to Controller Design Example 1: there is a negative sign on the open-loop *s* term. The negative damping injects energy into the system instead of dissipating energy.

open-loop poles $s_{1,2} = +0.5 \pm 3.12i$ Positive real poles, UNSTABLE!

t_R = 0.08 *s* t_P = 1.01 *s* *PO* = 165.4% t_S = –8 *s*

These transient characteristics are all **BOGUS** since they are calculated based on the underdamped, positive damping, performance specification equations. The open-loop unit step response is:



When we use MATLAB function **`step`** for this problem and right-click for the performance specifications, no useful results are generated.

Step 2. Specify the desired behavior for the closed-loop system.

$$\Delta_{CL_{DES}}(s) = (s + 26.7)(s^2 + 5.33s + 14.93) = s^3 + 32s^2 + 157.2s + 398.2$$

Step 3. Specify the form for the controller transfer function.

PID Controller: $G_C(s) = K_P + \frac{K_I}{s} + K_D s$

Step 4. State the controller design problem to be solved.

Given $G(s) = \dfrac{10}{s^2 - s + 10}$, $H(s) = 1$, $\Delta_{CL_{DES}}(s) = s^3 + 32s^2 + 157.2s + 398.2$, and $G_C(s) = K_P + \dfrac{K_I}{s} + K_D s$

Solve for the PID controller gains K_P, K_I, and K_D. Then evaluate the PID controller performance.

Step 5. Solve for the unknown gains to achieve the desired behavior for the closed-loop system. Derive the closed-loop transfer function as a function of the PID controller gains:

$$T(s) = \frac{G_C(s)G(s)}{1 + G_C(s)G(s)H(s)} = \frac{\dfrac{10\left(K_D s^2 + K_P s + K_I\right)}{s\left(s^2 - s + 10\right)}}{1 + \dfrac{10\left(K_D s^2 + K_P s + K_I\right)}{s\left(s^2 - s + 10\right)}} = \frac{\dfrac{10\left(K_D s^2 + K_P s + K_I\right)}{s\left(s^2 - s + 10\right)}}{\dfrac{s\left(s^2 - s + 10\right) + 10\left(K_D s^2 + K_P s + K_I\right)}{s\left(s^2 - s + 10\right)}}$$

$$T(s) = \frac{10\left(K_D s^2 + K_P s + K_I\right)}{s\left(s^2 - s + 10\right) + 10\left(K_D s^2 + K_P s + K_I\right)}$$

$$T(s) = \frac{10\left(K_D s^2 + K_P s + K_I\right)}{s^3 + \left(-1 + 10K_D\right)s^2 + \left(10 + 10K_P\right)s + \left(10K_I\right)}$$

Match the symbolic form (function of K_P, K_I, K_D) with the numerical desired characteristic polynomial – use the same third-order desired characteristic polynomial from Controller Design Example 1, which is an augmented version of the second-order behavior with 5% overshoot and 1.5 sec settling time.

Denominator parameter matching (fully-decoupled solution):

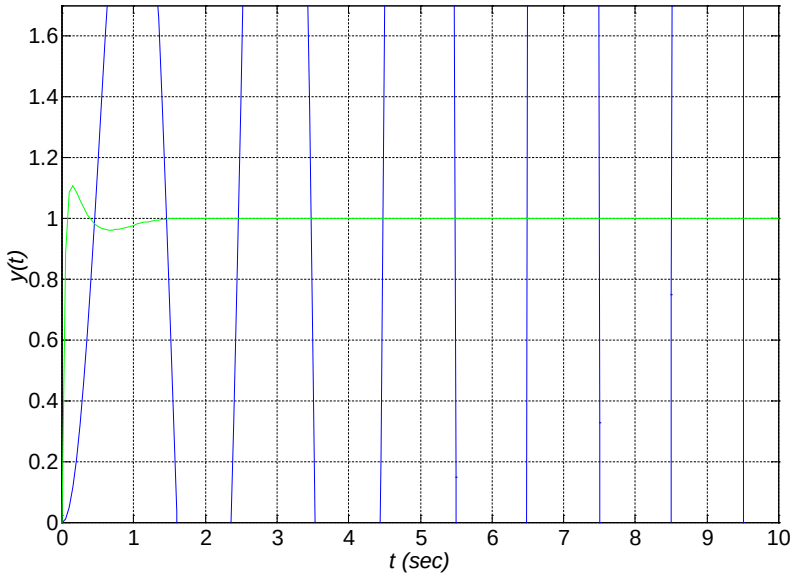
$$s^3 \rightarrow 1 = 1$$
$$s^2 \rightarrow -1 + 10K_D = 32$$
$$s^1 \rightarrow 10 + 10K_P = 157.2$$
$$s^0 \rightarrow 10K_I = 398.2$$

$$K_D = 3.3$$
$$K_P = 14.72$$
$$K_I = 39.82$$

There is only one small change, compared to the stable underdamped system PID controller from Controller Design Example 1: only K_D changed slightly (it was 3.1 before; now it is 3.3 for the unstable system), but K_P and K_I are **identical**.

Step 6. Evaluate the PID controller performance in simulation.

Open- vs. Closed-Loop unit step responses:



Open-Loop		Closed-Loop PID	
t_R	= N/A *sec*	t_R	= 0.05 *sec*
t_P	= ∞ *sec*	t_P	= 0.16 *sec*
PO	= NaN	PO	= 10.7%
t_S	= ∞ *sec*	t_S	= 1.03 *sec*
e_{SS}	= ∞	e_{SS}	= 0%

Step 7. Include an output attenuation correction factor if necessary.

Again, no output attenuation correction factor is required thanks to the K_I term – the effective closed-loop stiffness is the same as the open-loop stiffness. The K_I term ensures zero steady-state error.

Closed-loop transfer function: $$T(s) = \frac{33s^2 + 147.2s + 398.2}{s^3 + 32s^2 + 157.2s + 398.2}$$

We stabilized the unstable system but did not achieve the desired transient characteristics. The above PID closed-loop system response is again too fast and too sharp. The desired poles are matched, but the settling time is approximately 1.0 *sec*, faster than the specified the 1.5 *sec*. The percent overshoot is approximately 10.9%, violating the 5% overshoot specification. Therefore, we need a pre-filter.

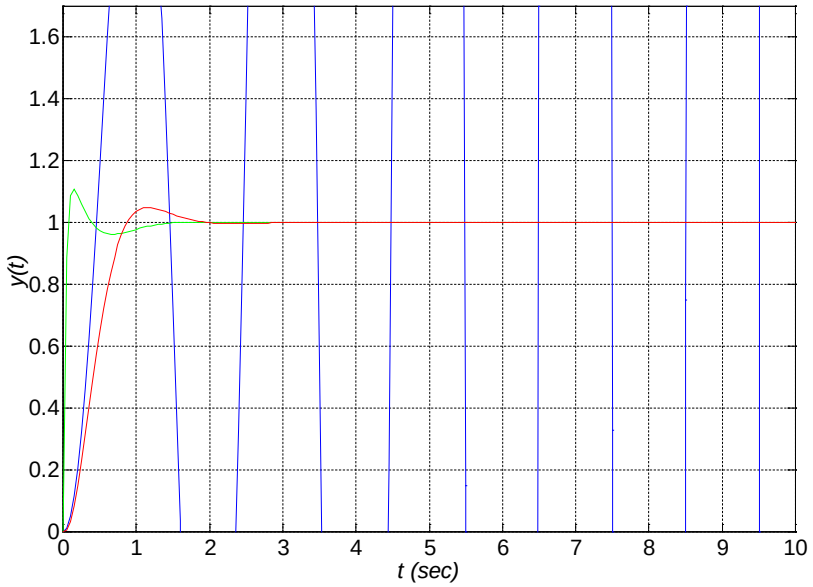
Step 8. Include a pre-filter transfer function $G_P(s)$ if necessary.

Two unwanted zeros were introduced by the PID controller; these can be cancelled by a pre-filter.

Pre-filter transfer function: $$G_P(s) = \frac{398.2}{33s^2 + 147.2s + 398.2}$$

Note the 398.2 numerator is included since the pre-filter shouldn't attenuate the output. Again plotting the unit step responses, we now see that the performance specifications are met.

Example 2 PID Controller Open-, Closed-, and Closed-loop with pre-filter unit step responses:

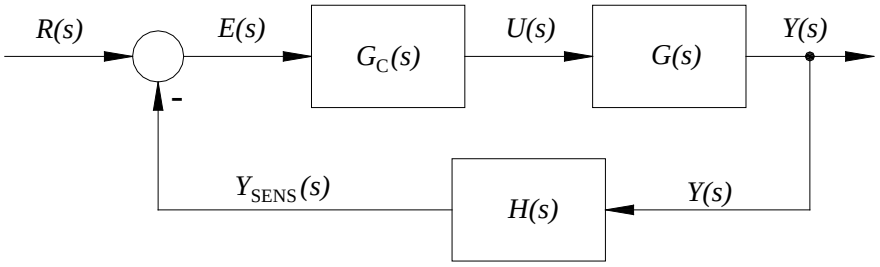


Open-Loop		**Closed-Loop PID**		**Closed-Loop PID with Pre-Filter**	
t_R	= N/A *sec*	t_R	= 0.05 *sec*	t_R	= 0.55 *sec*
t_P	= ∞ *sec*	t_P	= 0.16 *sec*	t_P	= 1.15 *sec*
PO	= NaN	*PO*	= 10.7%	*PO*	= 4.9%
t_S	= ∞ *sec*	t_S	= 1.03 *sec*	t_S	= 1.59 *sec*
e_{SS}	= ∞	e_{SS}	= 0%	e_{SS}	= 0%

The pre-filtered closed-loop unit step response is theoretically identical to that of the Lead and PID Controller Design Example 1 cases.

6.10 Whole T(s) Matching Controller Design: the J-Method

Standard negative feedback closed-loop block diagram:



Summary of controller design via parameter matching:

- **Step 1. Analyze the as-given open-loop system behavior.**
- **Step 2. Specify and evaluate the desired behavior for the closed-loop system.** This step yields a numerical desired characteristic polynomial for the $T(s)$ denominator, called $\Delta_{DES}(s)$.
- **Step 3. Specify the form for the controller transfer function.** Assume a standard form for your controller $G_C(s)$.
- **Step 4. State the controller design problem to be solved.**
- **Step 5. Solve for the unknown gains to achieve the desired behavior for the closed-loop system.** Derive closed-loop transfer function as a function of controller gains and parameters:

$$T(s) = \frac{G_C(s)G(s)}{1 + G_C(s)G(s)H(s)}$$

  - Match the symbolic parameters in the denominator of $T(s)$ with your numerical desired characteristic polynomial – term-by-term according to s powers (make sure both polynomials have the same leading coefficient for the highest s power).
  - Solve for the controller unknowns, determine the correction factor (if necessary), determine the pre-filter transfer function $G_P(s)$ (if necessary), and simulate the resulting performance.
- **Step 6. Evaluate the controller performance in simulation.**
- **Step 7. Include an output attenuation correction factor if necessary.**
- **Step 8. Include a pre-filter transfer function $G_P(s)$ if necessary.**
- **Repeat Step 6.**
- **Step 9. Re-design and re-evaluate the controller if necessary, if the performance specs are not met.**

There are possible drawbacks with this method, the biggest being: the order of the closed-loop system must match the number of controller unknowns. If the number of unknowns is less, there is no solution (overconstrained) and if the number of unknowns is greater, there are infinite solutions (underconstrained). In either of the mismatch cases we can use trial-and-error controller design. This can be frustrating and ineffective since there are ∞^n solutions, where n is the number of controller unknowns.

Further drawbacks include the need for most standard controllers to add a correction factor for output attenuation and a pre-filter transfer function $G_P(s)$ to cancel the unwanted introduced zero(s).

There is a better method, here called the whole $T(s)$ Matching Controller Design, or simply the J-Method. This is named after ME student **Jason Denhart** who, in Spring 2008, posed the question: Why can't we just specify the entire numerical $T(s)$ and solve for the exact $G_C(s)$ necessary to provide that $T(s)$?

The J-Method

- **Step 1. Analyze the as-given open-loop system behavior.**
- **Step 2. Specify and evaluate the desired behavior for the closed-loop system.** This step yields a numerical desired characteristic polynomial for the $T(s)$ denominator (see Note 1 below), plus the desired $T(s)$ numerator.
- **Step 3. State the controller design problem to be solved.** The former Step 3 is skipped, i.e. the J-Method determines the controller form in the solution procedure so there is no need to assume a standard controller form for $G_C(s)$.

  Given: The plant and sensor transfer functions $G(s)$ and $H(s)$ in the context of our standard closed-loop negative feedback diagram.
  The desired closed-loop behavior, expressed by a whole numerical $T(s)$ (as opposed to the desired closed-loop characteristic polynomial only as before).

  Find: The form and the unknowns (gains) of the controller transfer function $G_C(s)$.

- **Step 4. Solve for the unknown controller form including the unknown gains.** Use the standard equation:

$$T(s) = \frac{G_C(s)G(s)}{1 + G_C(s)G(s)H(s)}$$

  Substitute your whole numerical desired $T(s)$, plus the known numerical plant and sensor transfer functions $G(s)$ and $H(s)$, and solve for the unknown controller transfer function $G_C(s)$, without the need to assume a certain controller form:

$$G_C(s) = \frac{T(s)}{G(s)(1 - H(s)T(s))}$$

- **Step 5. Simulate the resulting closed-loop system performance.**
- **Step 6. Re-design and re-evaluate if necessary, if the performance specs are not met.**

Notes:

1. The order of the denominator of your desired $T(s)$ must be equal or greater than the order of the denominator of $G(s)$. If the $G(s)$ denominator is of higher order than the $T(s)$ denominator, your resulting $G_C(s)$ will have a higher order in the numerator than the denominator, which is impossible to implement in MATLAB or Simulink, unless it happens to boil down to a PID form.
2. You do not need a correction factor – this is already loaded into your desired $T(s)$.
3. You do not need a pre-filter transfer function $G_P(s)$ since we are matching the entire desired $T(s)$.
4. The resulting controller $G_C(s)$ will not necessarily be of any recognizable classical controller form.
5. MATLAB was used to help with the algebra (either symbolically or numerically). This should work; however, problems were encountered with large integers and excessive controller numerator and denominator orders.
6. If you follow the requirement for $T(s)$ denominator order and make no algebra mistakes, the performance you obtain will be theoretically identical to a good classical controller with appropriate pre-filter and correction factor.
7. The major drawback of the J-Method – this method does not handle disturbances well, as most other controllers do (presented later).
8. Another major drawback of the J-Method is that it requires perfect knowledge of the open-loop plant, which is impossible. Thus, it is less robust to modeling uncertainties than other controllers.

Controller Design Example 1: The J-Method

Step 1. Analyze the as-given open-loop system behavior.

This step is the same as in Controller Example 1, with $G(s) = \frac{10}{s^2 + s + 10}$.

Step 2. Specify and evaluate the desired behavior for the closed-loop system, to achieve 5% overshoot and 1.5 sec settling time. We have used these performance specifications before; the associated desired whole closed-loop transfer function is (assuming a final value of 1 is desired from a unit step input):

$$T(s) = \frac{14.93}{s^2 + 5.33s + 14.93}$$

Step 3. State the controller design problem to be solved.

Given $G(s) = \frac{10}{s^2 + s + 10}$, $H(s) = 1$, and $T(s) = \frac{14.93}{s^2 + 5.33s + 14.93}$

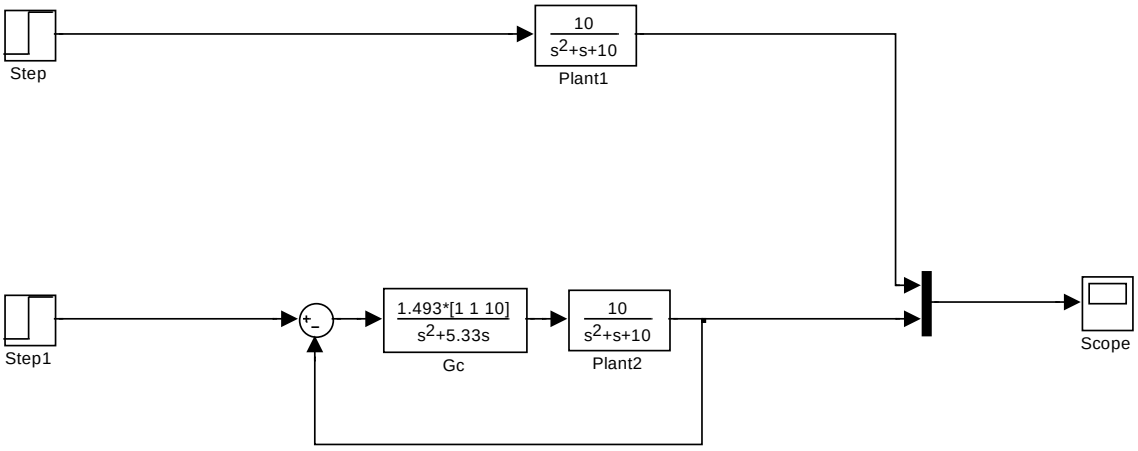
Solve for $G_C(s)$ via whole $T(s)$ matching.

Step 4. Solve for the unknown controller form including the unknown gains.

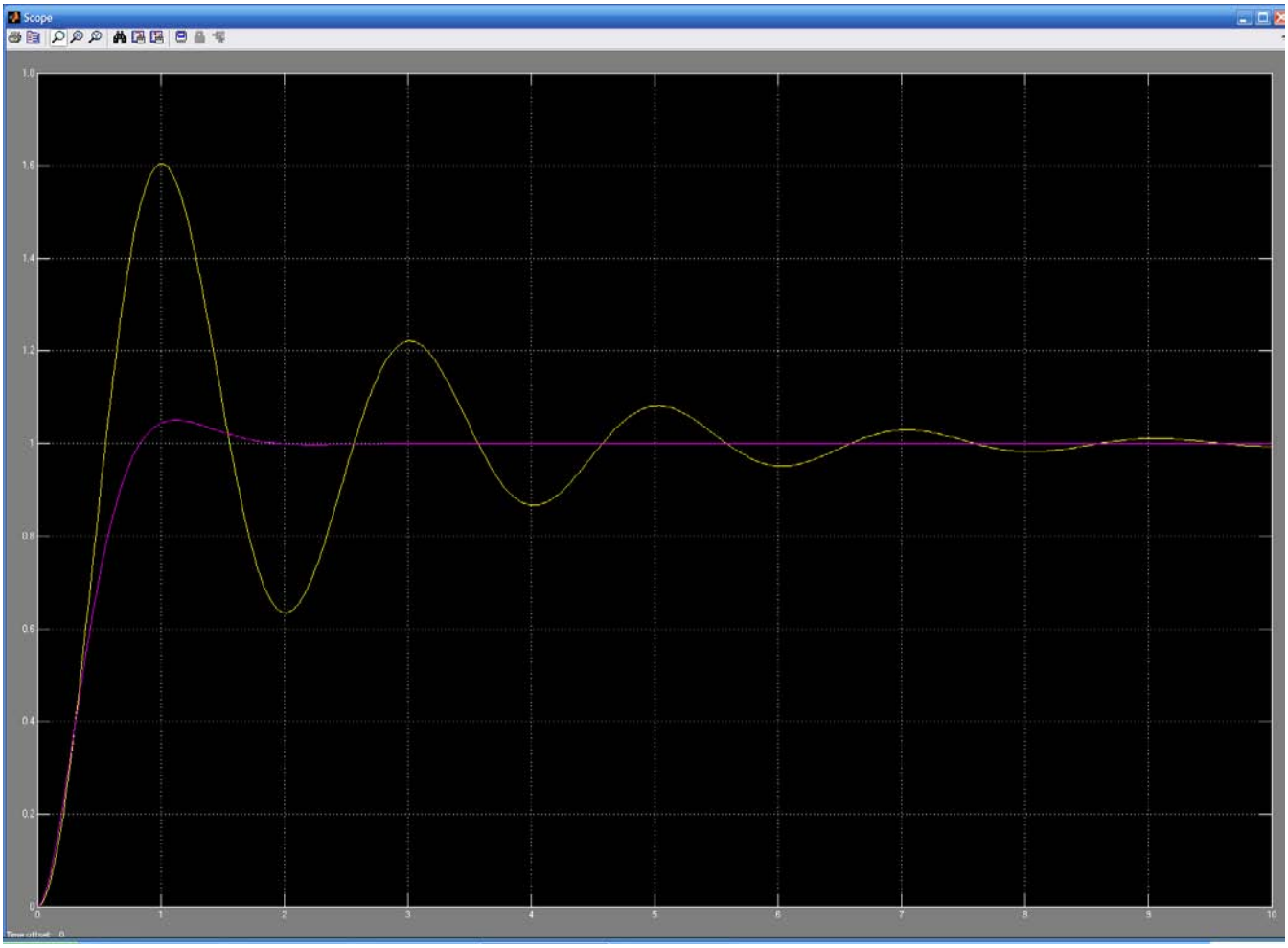
The result is: $G_C(s) = \frac{1.493(s^2 + s + 10)}{s(s + 5.33)}$

What type of controller is this?

<u>Step 5.</u> Simulate the resulting closed-loop system performance.
Simulink Evaluation for the J-Method Controller Design Example 1



Simulink simulation results:



Open-Loop Closed-Loop J-Method

There is no need for a correction factor or pre-filter.

Examples

To rigorously test the J-Method, the following three open-loop transfer functions $G(s)$ and the following three desired whole closed-loop transfer functions $T(s)$ were chosen, to be mixed-and-matched for a total of 9 examples:

***G*(*s*)**	***T*(*s*)**
1. $\frac{1}{s+1}$	a. $\frac{2}{s+2}$
2. $\frac{10}{s^2+s+10}$	b. $\frac{15}{s^2+6s+15}$
3. $\frac{10}{(s+1)(s^2+s+10)}$	c. $\frac{450}{s^3+36s^2+195s+450}$

- The whole desired $T(s)$ in a. is based on a time constant of $\tau = 0.5$ sec.
- The whole desired $T(s)$ in b. started from our familiar specification of 5% overshoot and 1.5 sec settling time (leading to a dominant second-order desired denominator of $s^2+5.33s+14.93$). But I wanted nice integers for the multiple example combinations, so this one corresponds to 2.13% overshoot and 1.38 sec settling time ($\xi = 0.78$, $\omega_n = 3.87$ rad/s, and $s_{1,2} = -3 \pm 2.45i$).
- The whole desired $T(s)$ in c. started from b., augmenting the denominator with a third pole $s_3 = -30$, so the same dominant second-order behavior will appear (2.13% overshoot and 1.38 sec settling time).
- All whole desired $T(s)$ in a., b., and c. artificially obtain a steady state value of 1 give a unit step input function, with selection of the numerator constant to match the denominator 'spring'.

Let's do a few of these examples now. Assume unity negative feedback for all examples.

Example 1a

Given open-loop plant $G(s) = \frac{1}{s+1}$, ideal sensor transfer function $H(s) = 1$, and desired whole $T(s) = \frac{2}{s+2}$, determine $G_C(s)$ via whole $T(s)$ matching:

Answer: $G_C(s) = \frac{2(s+1)}{s}$ (what kind of controller is this?)

Simulink simulation results

Open-Loop Closed-Loop J-Method

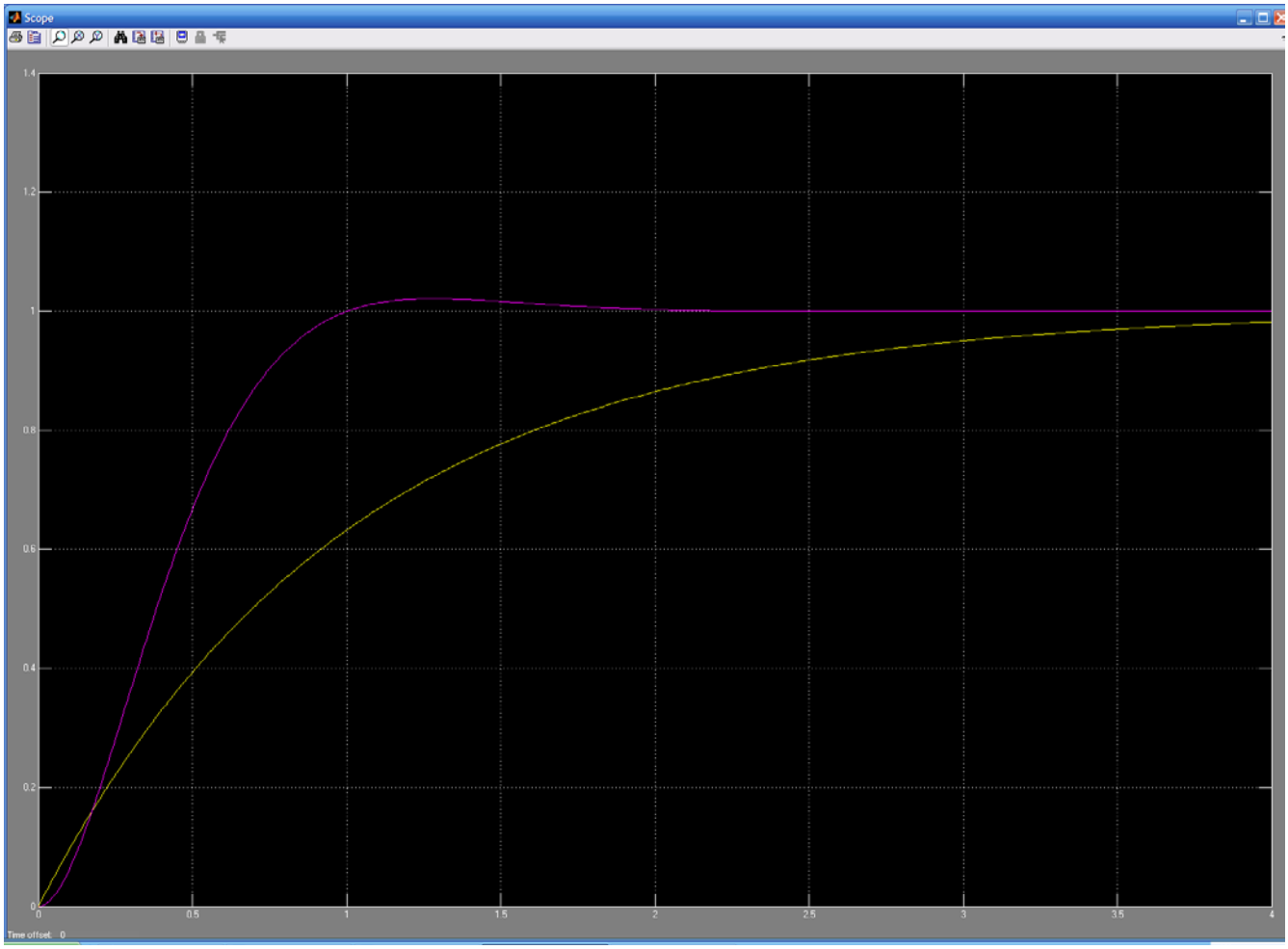
The open-loop time constant of $\tau = 1$ sec is halved to the closed-loop time constant of $\tau = 0.5$ sec, which doubles the speed of the closed-loop system to reach 95% of the final value of 1.

Example 1b

Given open-loop plant $G(s) = \frac{1}{s+1}$, ideal sensor transfer function $H(s) = 1$, and desired whole $T(s) = \frac{15}{s^2 + 6s + 15}$, determine $G_C(s)$ via whole $T(s)$ matching:

Answer: $G_C(s) = \frac{15(s+1)}{s(s+6)}$ (what kind of controller is this?)

Simulink simulation results



Open-Loop Closed-Loop J-Method

The open-loop time constant was τ = 1 sec and the desired closed-loop performance specs of 2.13% overshoot and 1.38 sec settling time are met in simulation.

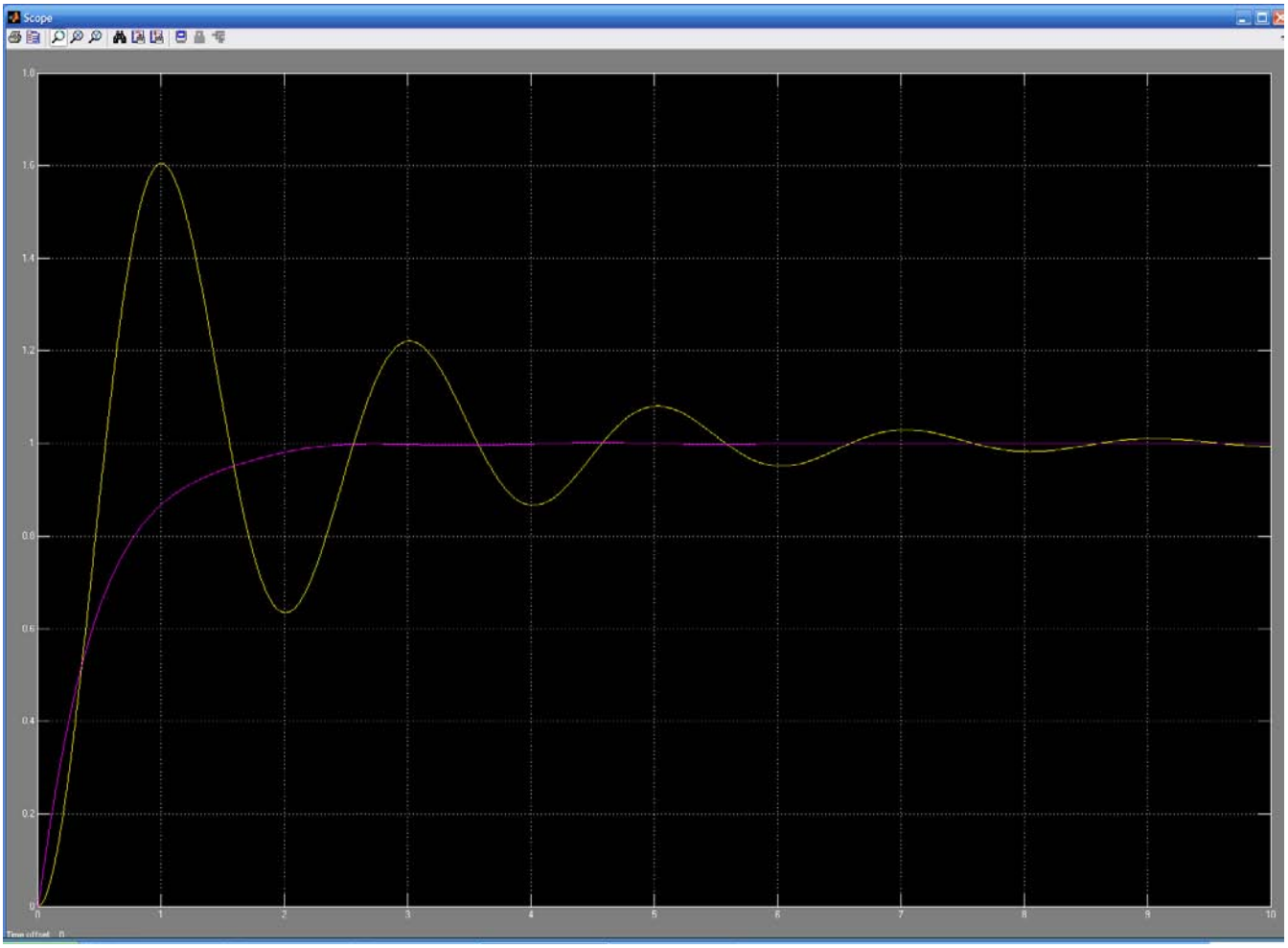
Example 2a

Given open-loop plant $G(s) = \frac{10}{s^2 + s + 10}$, ideal sensor transfer function $H(s)$ = 1, and desired whole $T(s) = \frac{2}{s+2}$, determine $G_C(s)$ via whole $T(s)$ matching:

Answer: $G_C(s) = \frac{s^2 + s + 10}{5s}$ (what kind of controller is this?)

This is an exception to the rule that the $T(s)$ denominator must of order equal or greater than the order of the $G(s)$ denominator, since it is a PID controller.

Simulink simulation results



Open-Loop Closed-Loop J-Method

The open-loop behavior is the familiar stable underdamped system and the closed-loop time constant is $\tau = 0.5$ sec which allows closed-loop system to reach 95% of the final value of 1 in 1.5 sec, thus meeting the design goals in simulation.

Examples Summary Table

For the combinations of examples considered, the table below summarizes the numerator and denominator orders using the J-Method. Also given is the resulting form of the controller transfer function $G_C(s)$ (n.r.f. stands for 'no recognizable form').

		***T*(s)**		
		a. 1st-order	b. 2nd-order	c. 3rd-order
***G*(s)**	1st	$\frac{1^{st}}{1^{st}}$, PI	$\frac{1^{st}}{2^{nd}}$, Lead/I	$\frac{1^{st}}{3^{rd}}$, n.r.f.
	2nd	$\frac{2^{nd}}{1^{st}}$, PID [1]	$\frac{2^{nd}}{2^{nd}}$, n.r.f.	$\frac{2^{nd}}{3^{rd}}$, n.r.f.
	3rd	$\frac{3^{rd}}{1^{st}}$, n.r.f. [2]	$\frac{3^{rd}}{2^{nd}}$, n.r.f. [2]	$\frac{3^{rd}}{3^{rd}}$, n.r.f.

Examples $G_C(s)$ Summary

For the nine example combinations considered, the table below summarizes the resulting controller transfer functions $G_C(s)$ using the J-Method.

		***T*(s)**		
		a. 1st-order	b. 2nd-order	c. 3rd-order
***G*(s)**	1st	$\frac{2(s+1)}{s}$	$\frac{15(s+1)}{s(s+6)}$	$\frac{450(s+1)}{s(s^2+36s+195)}$
	2nd	$\frac{s^2+s+10}{5s}$ [1]	$\frac{1.5(s^2+s+10)}{s(s+6)}$	$\frac{45(s^2+s+10)}{s(s^2+36s+195)}$
	3rd	$\frac{s^3+2s^2+11s+10}{5s}$ [2]	$\frac{1.5(s^3+2s^2+11s+10)}{s(s+6)}$ [2]	$\frac{45(s^3+2s^2+11s+10)}{s(s^2+36s+195)}$

[1]Exception to the rule that the order of the denominator of your desired $T(s)$ must be equal or greater than the order of the denominator of $G(s)$.

[2]Impossible to implement.

In addition to this set of nine examples, the J-Method has been successfully tested with the following (where the a-c cases represent the same desired closed-loop transfer functions as above):

- Integrating open-loop transfer function $G(s)$ (an s can be factored out in the denominator, i.e. there is no 'spring'):

$$G(s) = \frac{10}{s^2 + s} = \frac{10}{s(s+1)}$$

a. $G_C(s) = \frac{s+1}{5}$ PD b. $G_C(s) = \frac{1.5(s+1)}{(s+6)}$ Lead c. $G_C(s) = \frac{45(s+1)}{s^2+36s+195}$ n.r.f.

- Non-ideal sensor transfer function $H(s) \neq 1$:

$$H(s) = \frac{1}{s+4} \qquad \text{for } G(s) = \frac{1}{s+1}$$

a. $G_C(s) = \frac{2(s^2+5s+4)}{s^2+6s+6}$ b. $G_C(s) = \frac{15(s^2+5s+4)}{s^3+10s^2+39s+45}$ c. $G_C(s) = \frac{450(s^2+5s+4)}{s^4+40s^3+339s^2+1230s+1350}$

- Open-loop transfer function $G(s)$ with a zero:

$$G(s) = \frac{10(s+1)}{s^2+s+10}$$

a. $G_C(s) = \frac{s^2+s+10}{5s(s+1)}$ b. $G_C(s) = \frac{1.5(s^2+s+10)}{s(s^2+7s+6)}$ c. $G_C(s) = \frac{45(s^2+s+10)}{s(s^3+37s^2+231s+195)}$

- Unstable open-loop transfer function $G(s)$, e.g. Controller Design Example 2:

$$G(s) = \frac{10}{s^2-s+10}$$

a. $G_C(s) = \frac{s^2-s+10}{5s}$ b. $G_C(s) = \frac{1.5(s^2-s+10)}{s(s+6)}$ c. $G_C(s) = \frac{45(s^2-s+10)}{s(s^2+36s+195)}$

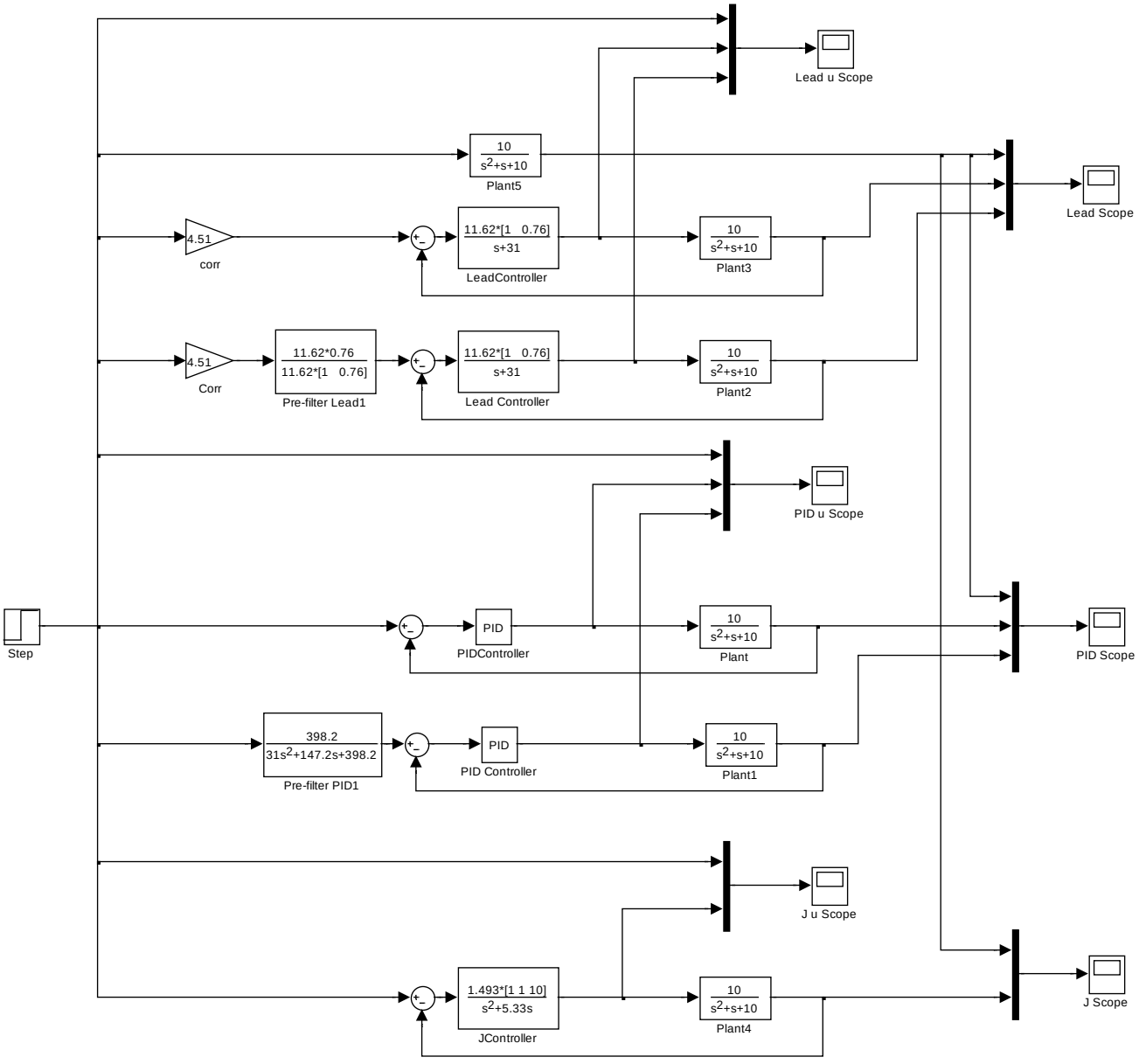
Try all of these in Simulink simulation to evaluate their performance.

All examples work as desired – except for the unstable cases which all have numerical instability problems, with the negative K_P gains – the unstable poles cannot be cancelled perfectly. **So we CANNOT use the J-Method for unstable open-loop systems**.

6.11 Closed-Loop Controller Input Effort

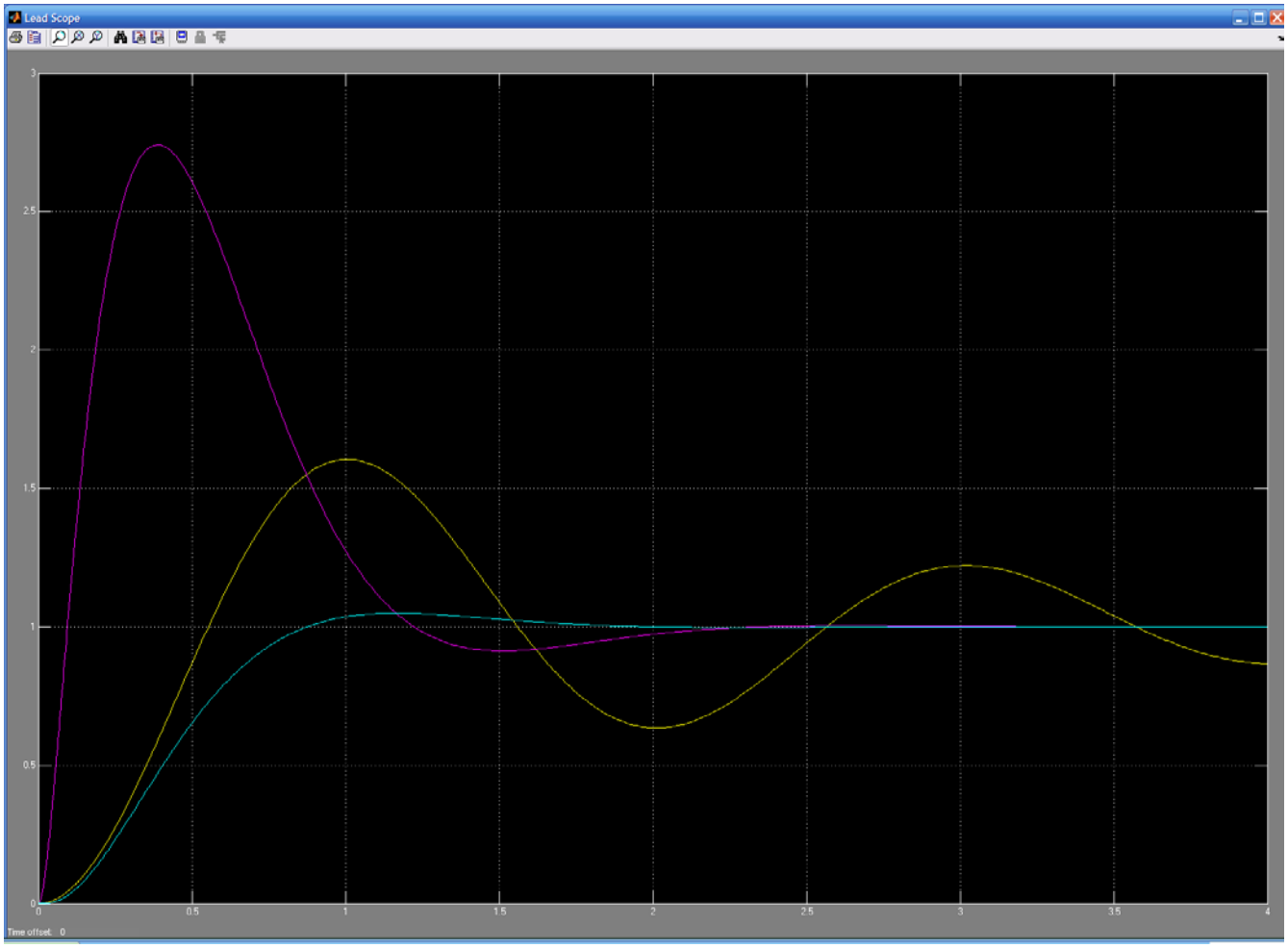
Closed-Loop Controller Input Effort Examples

We now return to the Controller Design Examples 1, with the Lead controller (without and with pre-filter), the PID controller (without and with pre-filter), and the J-Method controller.



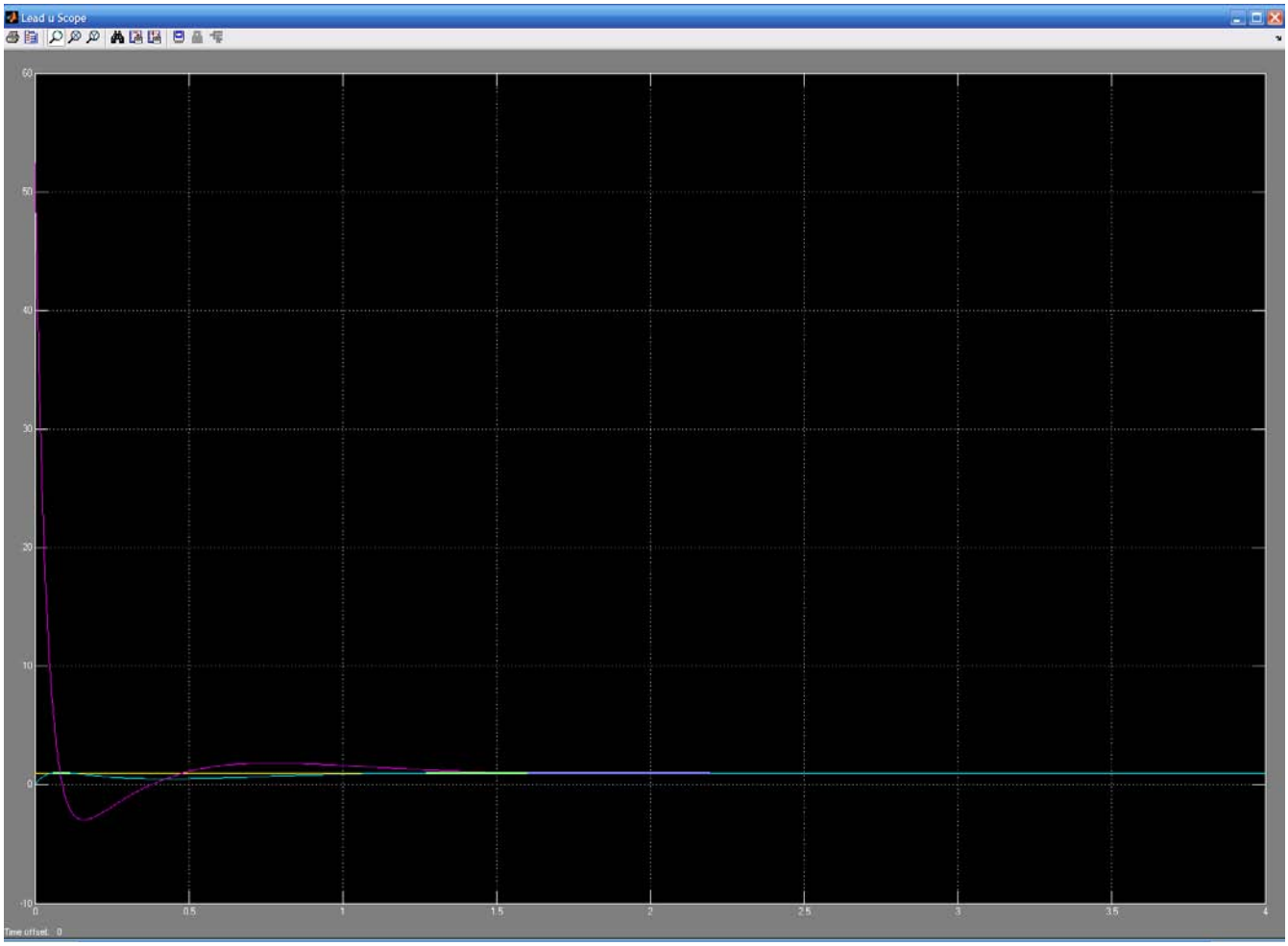
Simulink Model for open-loop, Lead, PID, and J-Method Input Effort Simulation

Lead Controller



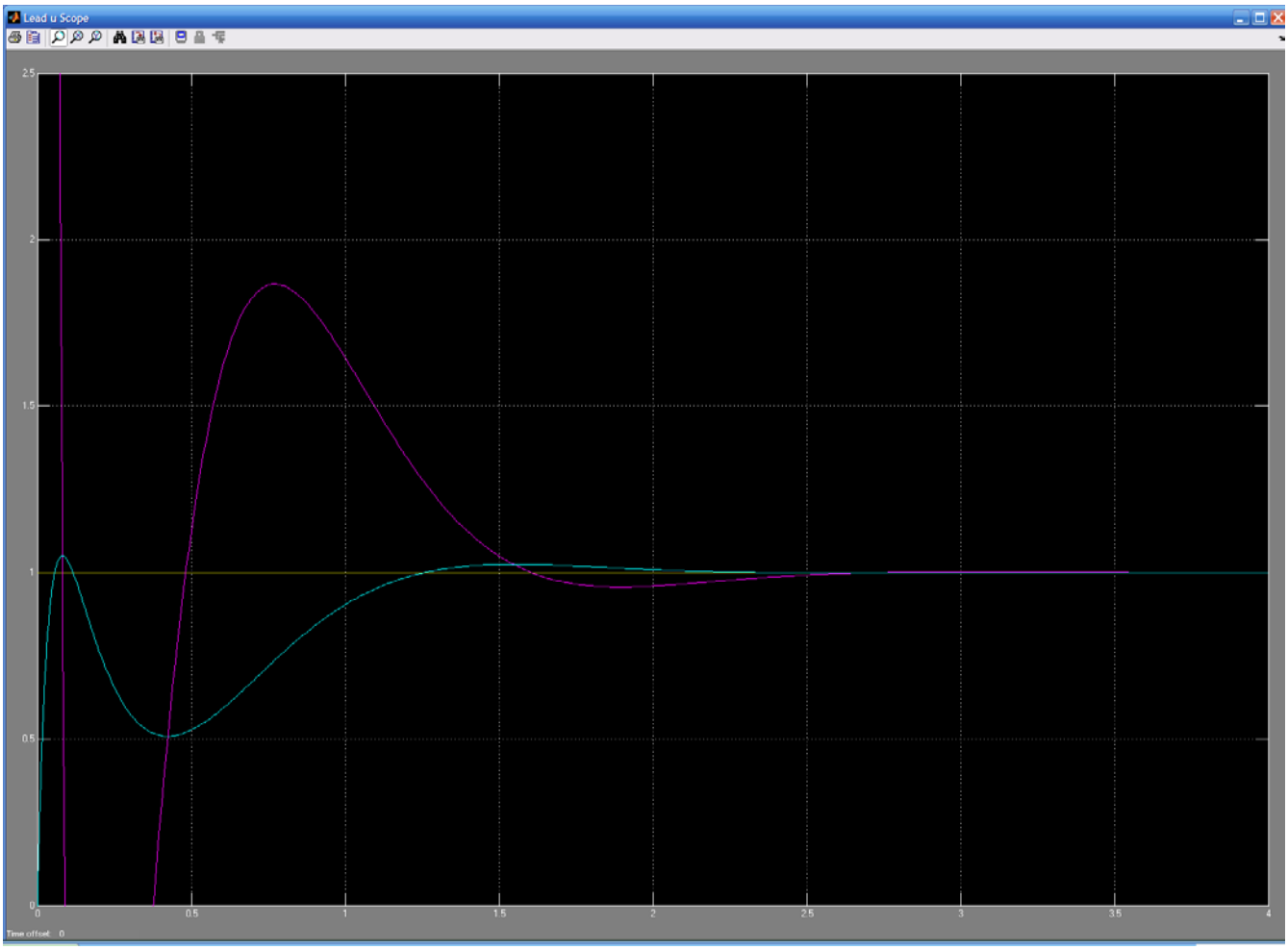
Output Unit Step Responses

Open-loop, non-pre-filtered Lead, pre-filtered Lead



Lead Controller Input Effort (entire scale)

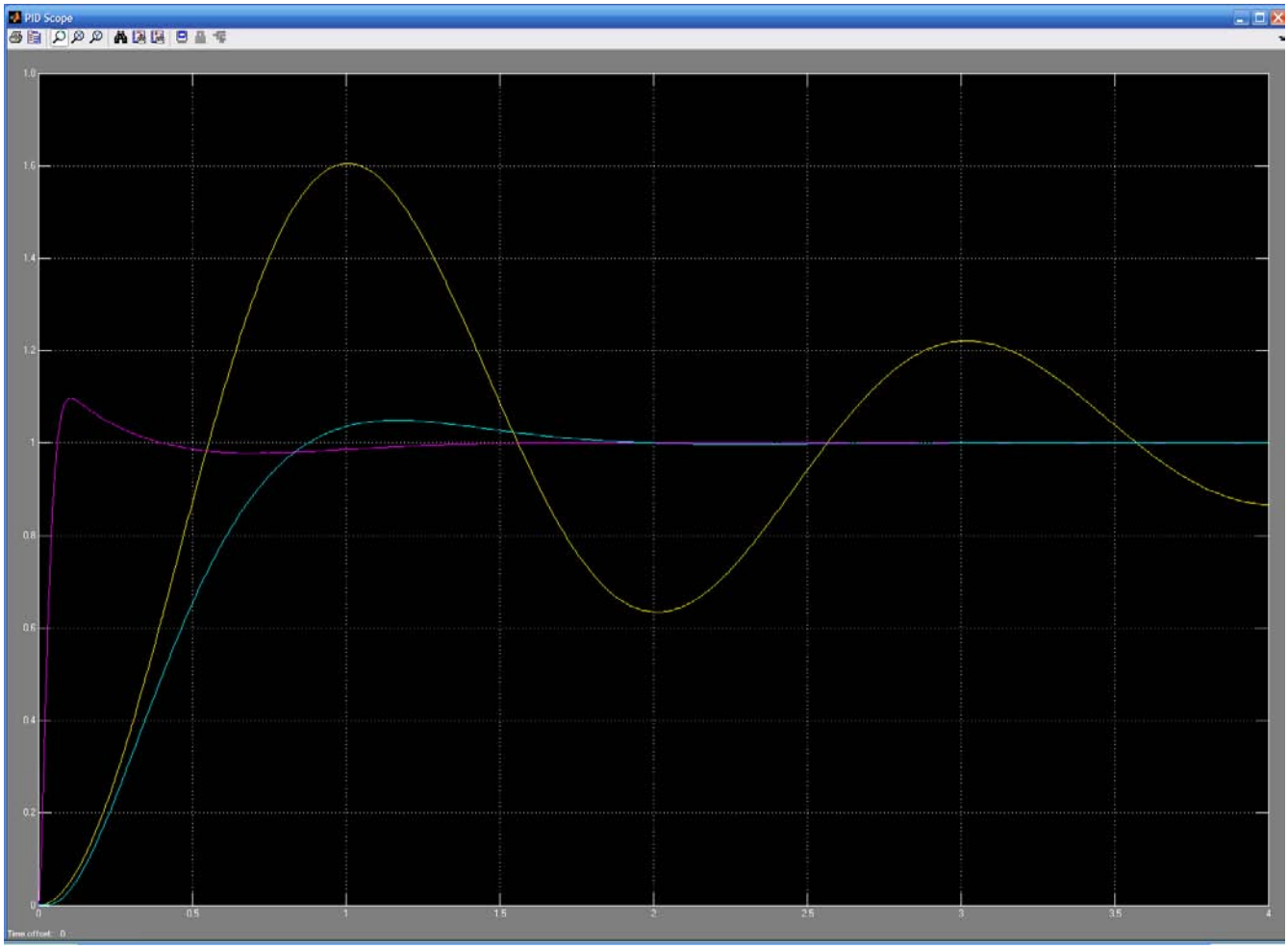
Open-loop (1), non-pre-filtered Lead (52.4), pre-filtered Lead (1.1)



Lead Controller Input Effort (zoomed in)

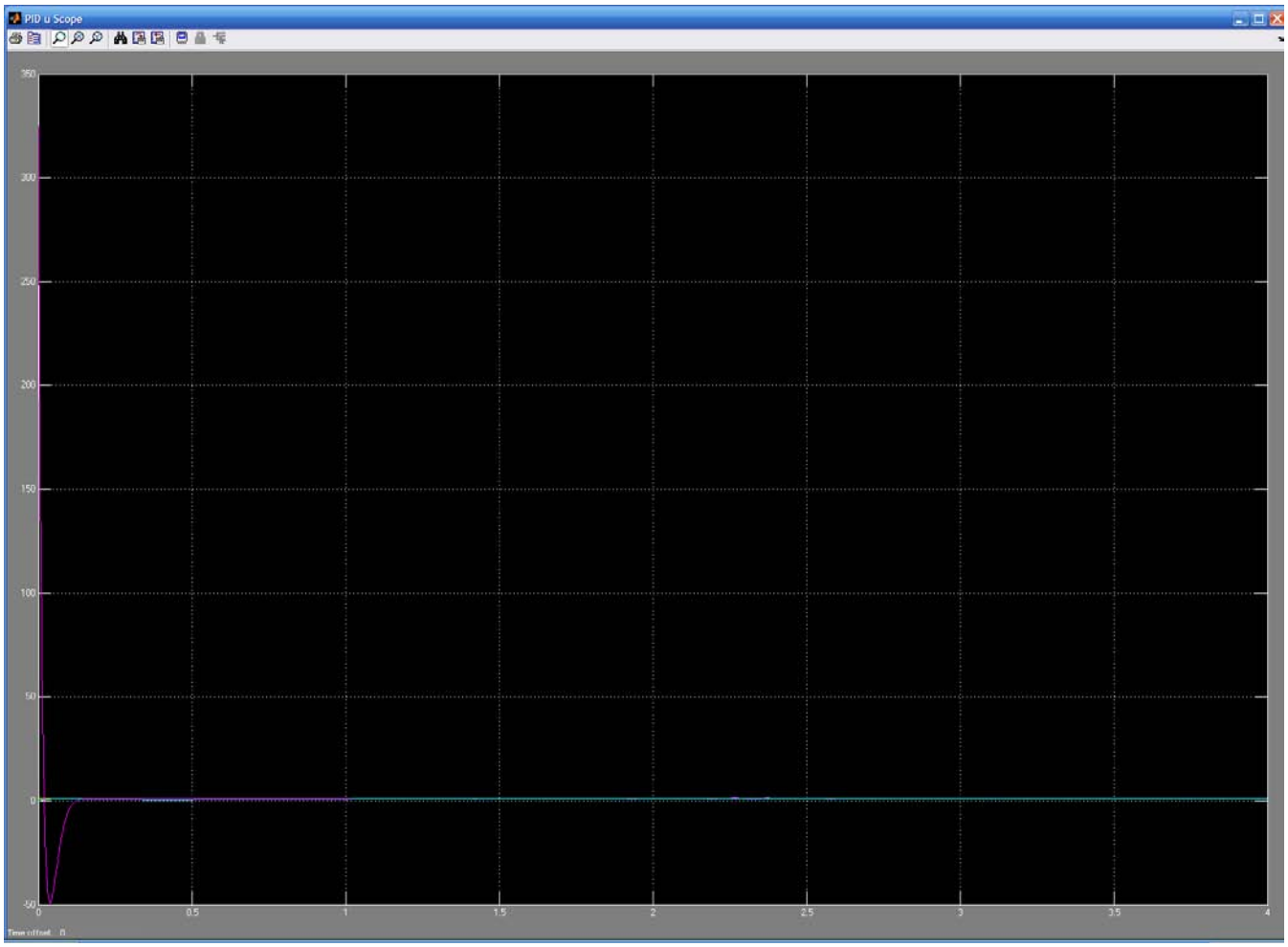
Open-loop (1), non-pre-filtered Lead (52.4), pre-filtered Lead (1.1)

PID Controller



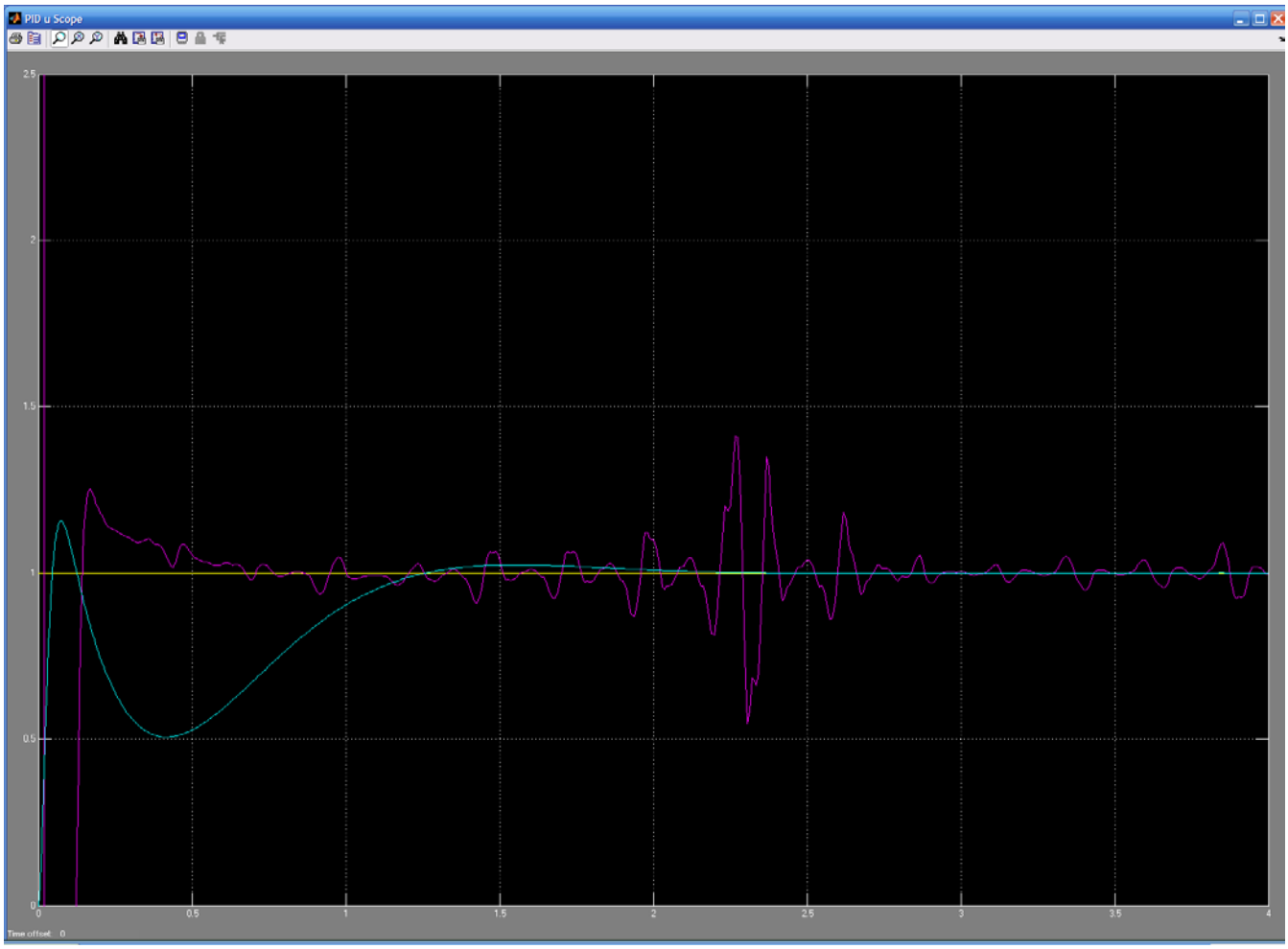
Output Unit Step Responses

Open-loop, **non-pre-filtered PID**, **pre-filtered PID**



PID Controller Input Effort (entire scale)

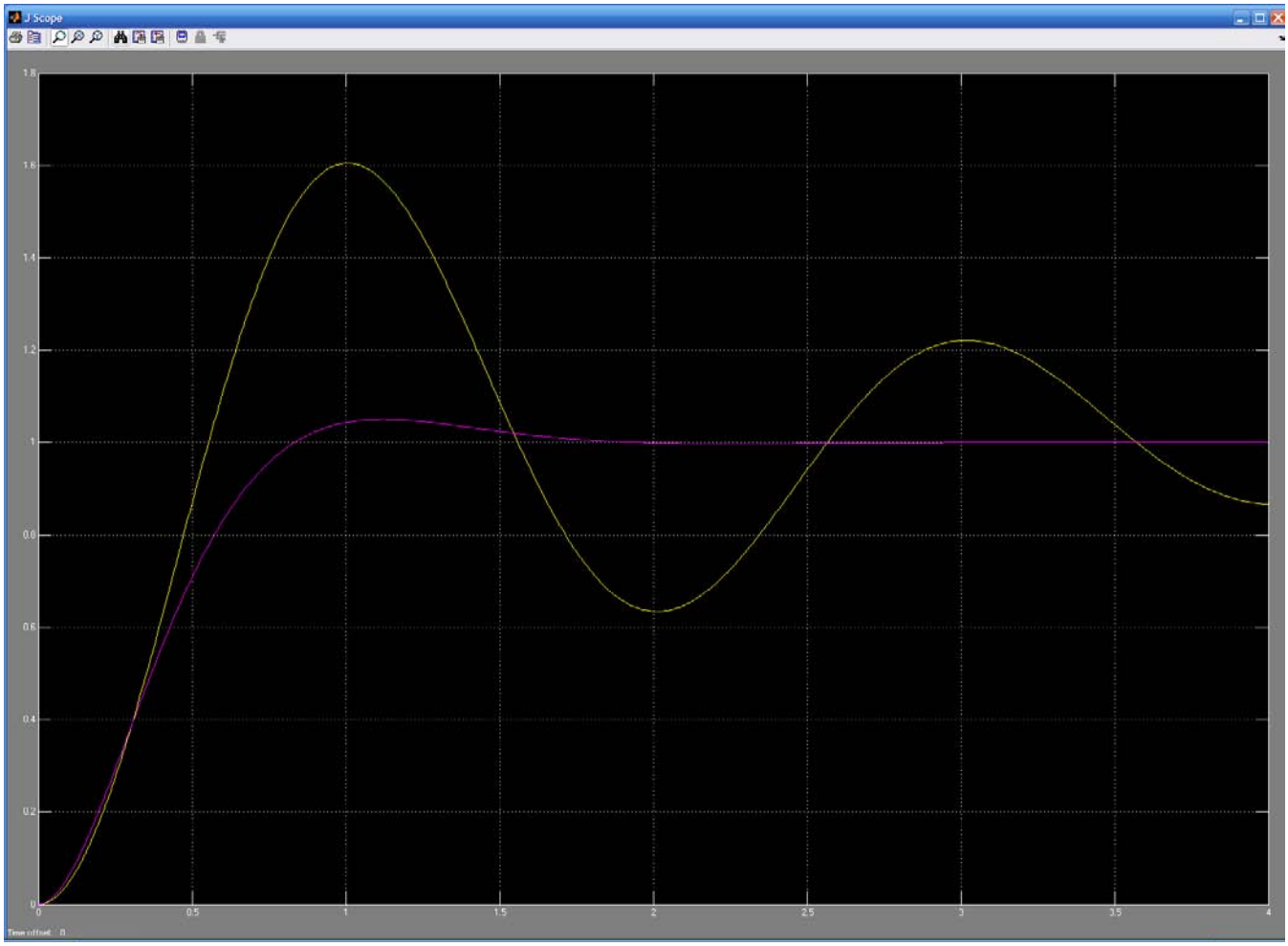
Open-loop (1), non-pre-filtered PID (324.7), pre-filtered PID (1.2)



PID Controller Input Effort (zoomed in)

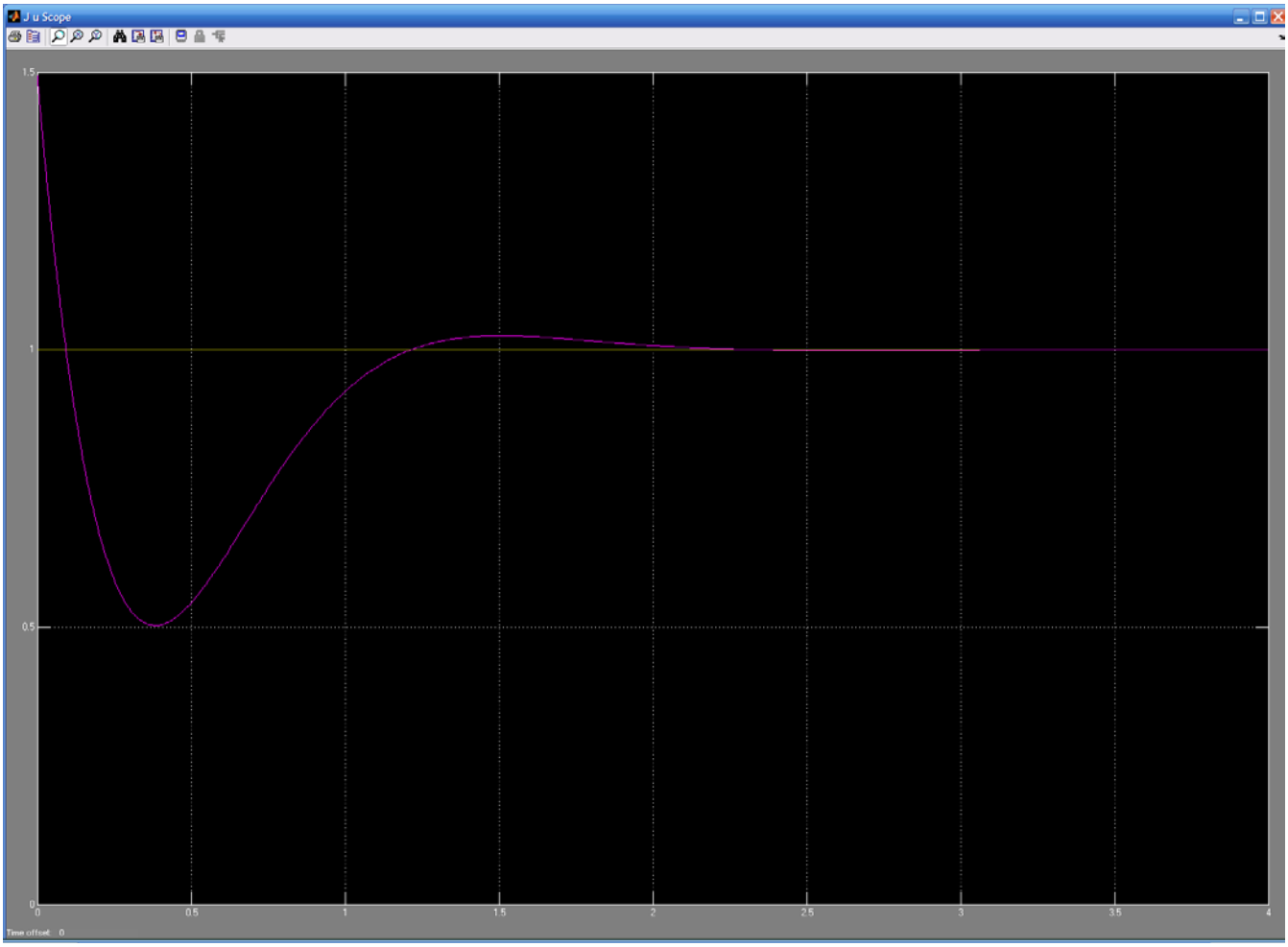
Open-loop (1), non-pre-filtered PID (324.7), pre-filtered PID (1.2)

J Controller



Output Unit Step Responses

Open-loop, J-Method Controller



J-Controller Input Effort

Open-loop (1), J-Method Controller (1.5)

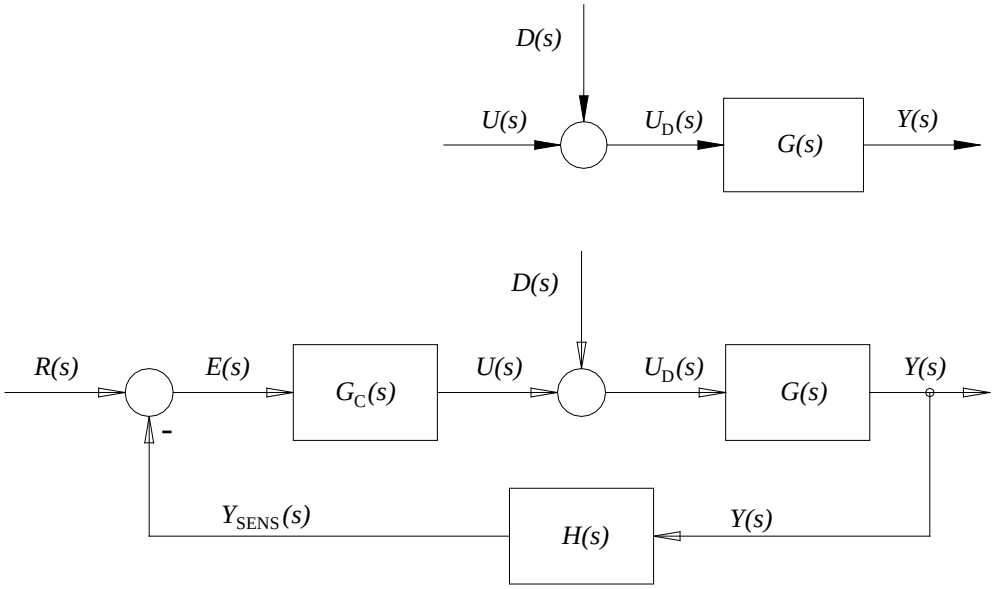
Input effort discussion

All open-loop input efforts are simply unit step inputs, shown for comparison purposes. Without pre-filtering, the input effort required for the Lead and PID controllers is huge (peaks of greater than 50 and greater than 300, respectively), so large we cannot see the details for the other curves on those plots. The pre-filtered Lead and PID controllers require maximum input efforts of just over 1 and nearly 1.2, respectively, a dramatic difference! The J-Method requires no pre-filter – its maximum input effort requirement is 1.5 actuator input units.

Similarly, the non-pre-filtered Lead and PID controllers require very fast changing of the input actuator values, which may exceed physical capabilities of the real-world actuator. The pre-filtered Lead and PID, plus the J-Method controllers do not require these radical rates of change for the actuator input efforts.

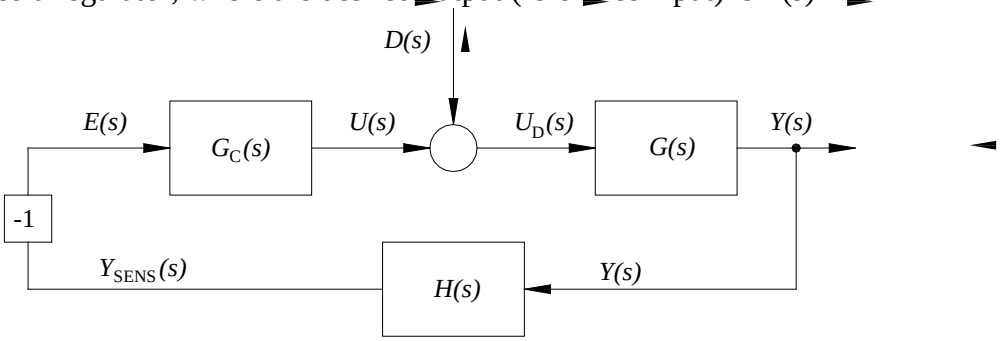
6.12 Disturbance Evaluation after Controller Design

General disturbance diagrams (open- and closed-loop):

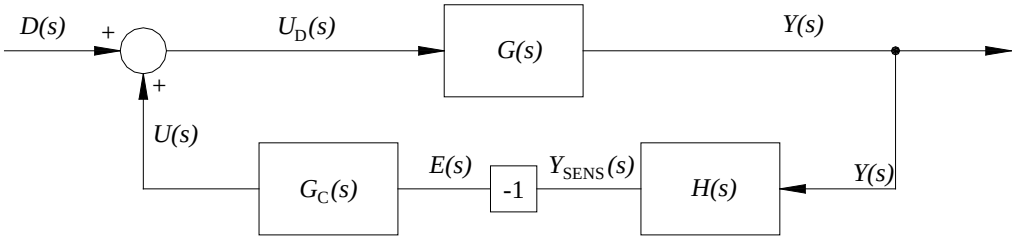


Disturbance diagram with zero reference input

This is called a regulator, where the desired output (reference input) is $R(s) = 0$.



Change this into a diagram that looks more similar to the standard closed-loop feedback diagram:



The closed loop transfer function between the disturbance input $D(s)$ and $Y(s)$ output is:

$$T_D(s) = \frac{Y(s)}{D(s)} = \frac{G(s)}{1 + G_C(s)G(s)H(s)}$$

This is for MATLAB implementation, to simulate the effects of the disturbance input separately. Then use linear superposition in MATLAB to find the total solution as the sum of the reference input response (with zero disturbance) plus the disturbance input (with zero reference input).

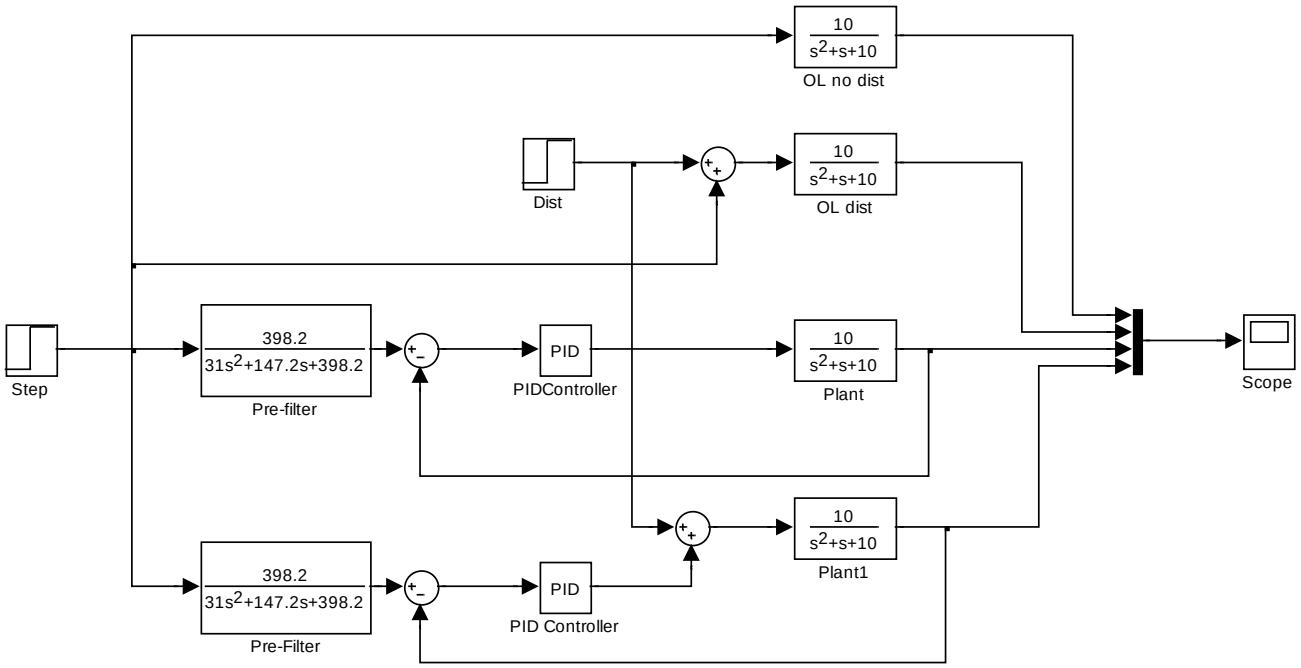
Controller Design with Disturbances (one possible strategy – for use in ME 401)

1. Design the controller as before with no disturbances – then evaluate in simulation.
2. For the same controller(s), now apply a disturbance and evaluate how well the controller rejects the disturbance vs. the open-loop system, both with zero reference input.
3. Repeat 2, apply a disturbance and evaluate how well the controller rejects the disturbance vs. the open-loop system, now with non-zero reference input.

One can test open- vs. closed-loop disturbance rejection behavior with various disturbances of different functions (impulse, step, ramp, sine wave, random, etc.), magnitudes, and times of application. Choose whatever makes the best sense for your real-world application.

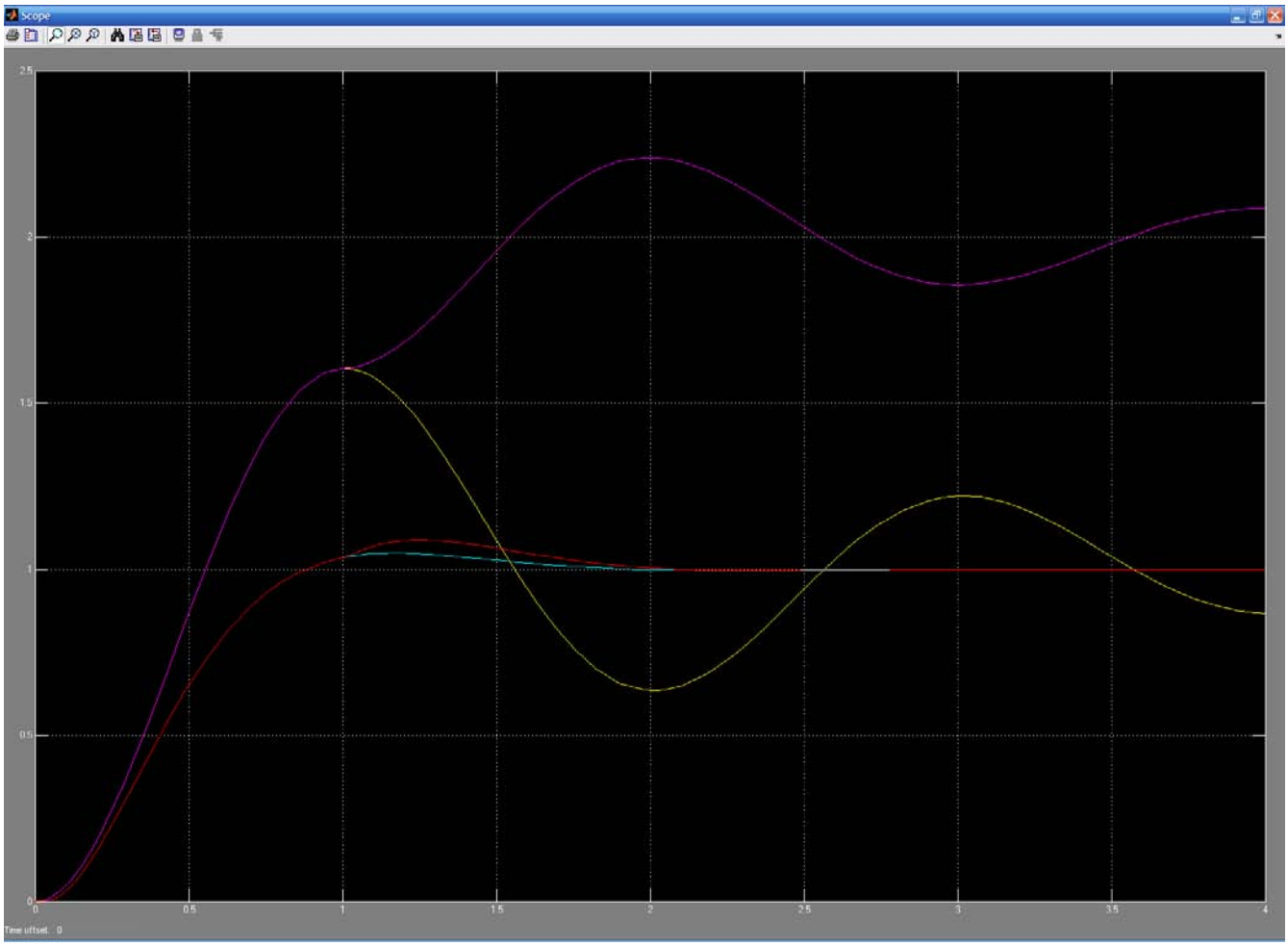
Pre-filtered PID Controller Example 1 with Disturbances

For open- and closed-loop disturbance rejection evaluation we now use the model and PID controller design results for Controller Example 1, presented earlier. The desired behavior was designed for 5% overshoot and 1.5 sec settling time, augmented for a third-order desired characteristic polynomial. The PID controller was designed using denominator polynomial matching and a pre-filter was required but no correction factor, thanks to the I term.



Simulink Model for open-loop, disturbed open-loop, pre-filtered PID, and disturbed pre-filtered PID

For this simulation we specify a unit step disturbance that adds to the actuator input, turned on at t = 1 sec, whereas the basic open- and closed-loop step input turned on a t = 0 sec.

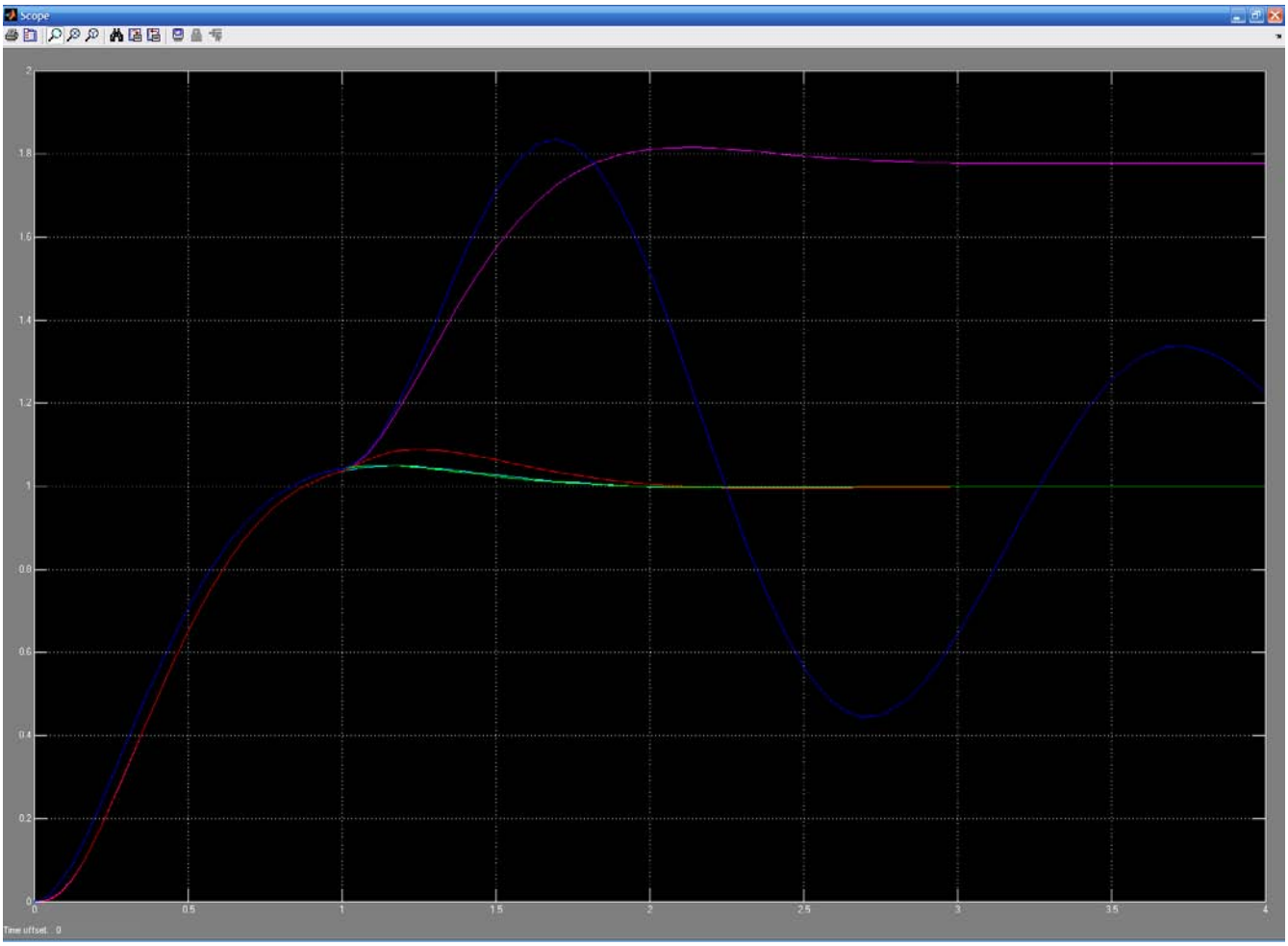


Open-loop, disturbed open-loop, pre-filtered PID, disturbed pre-filtered PID

This Simulink simulation shows that the open-loop system cannot deal with the disturbance at t = 1 sec since the open-loop transient response phase shifts and the steady state error is 100%. In contrast, the closed-loop PID controller is also disturbed at t = 1 sec, but the closed-loop output relatively quickly returns to the designed trajectory. Thus, the PID controller disturbance rejection is much better than that of the open-loop system.

Pre-filtered Lead, pre-filtered PID, and J-Method Controller Example 1 with Disturbances

To compare the closed-loop disturbance rejection of different controllers, we now use the same model and Lead, PID, and J-Method controller design results from Controller Example 1, all presented earlier.

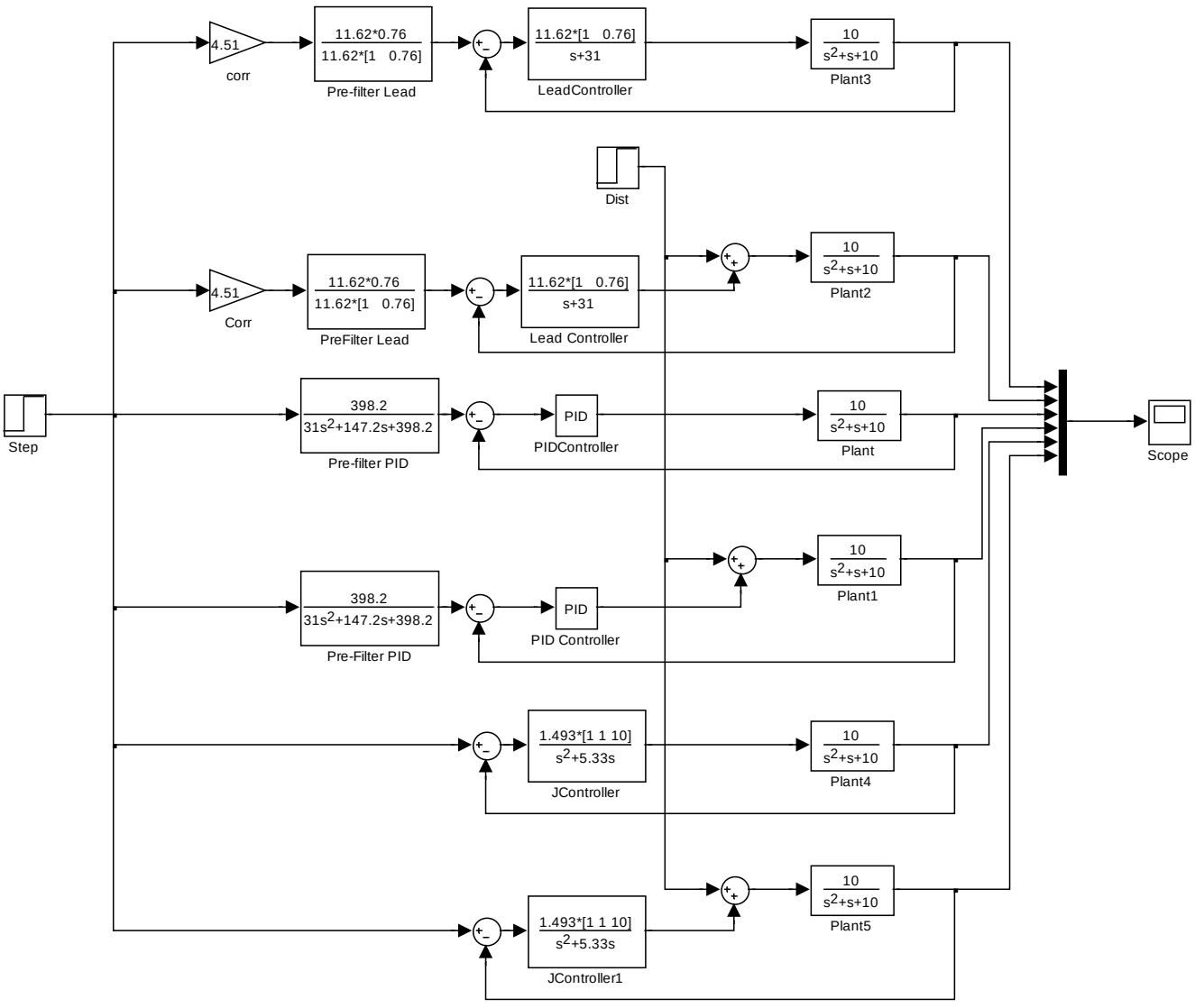


Lead, disturbed Lead, PID, disturbed PID, J-Method, disturbed J-Method

Here the output responses of the pre-filtered closed-loop Lead and PID controllers are identical. Since they are designed based on the augmented third-order system, there is a little error and the J-method controller (blue, the green is covered for the first second) is more accurate for the desired 5% overshoot and 1.5 sec settling time.

After the unit step disturbance is turned on at t = 1 sec, we see that only the PID controller can handle it well. The Lead controller (magenta) has good transient dynamics under the disturbance but nearly 80% steady-state error. The disturbed J-Method controller (blue) overshoots even higher, with worse transient dynamics; it is heading towards zero steady-state error, however, given enough time.

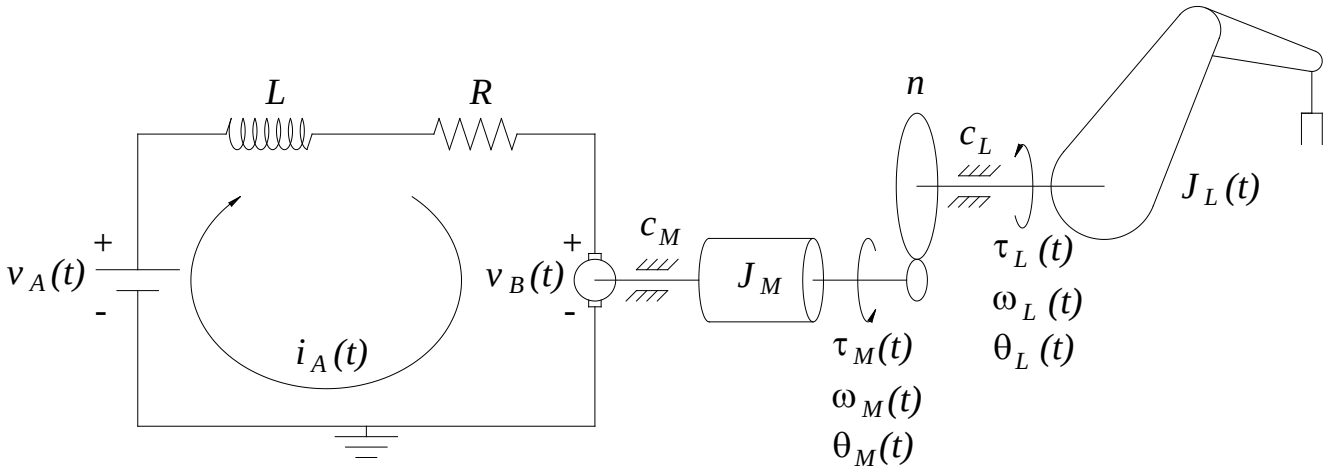
Thus, the PID controller disturbance rejection is much better than that of the Lead or J-Method controllers in this example. The PID controller was not designed for disturbance rejection, it just handles the disturbances better.



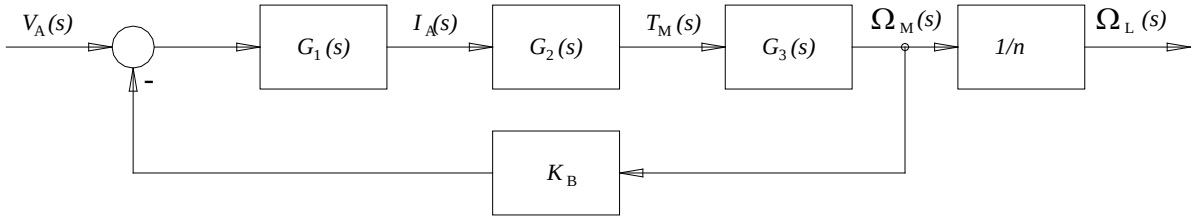
Simulink Model for pre-filtered Lead, pre-filtered PID,
and non-pre-filtered J-Method Controllers, without and with Disturbances

6.13 Term Example Controller Design

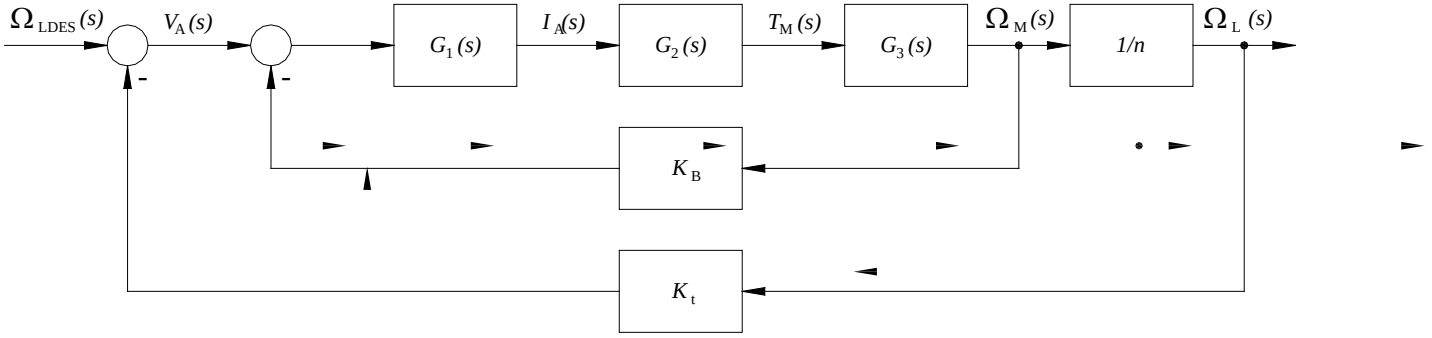
Open-loop system physical diagram



We will design a controller to control the load angular velocity $\omega_L(t)$. Assume a perfect tachometer sensor, $H(s) = K_t = 1$, so there is negative unity feedback. Open- and closed-loop feedback control system diagrams:



Term Example Open-Loop Diagram



Term Example Closed-Loop Diagram

where $G_1(s) = \dfrac{1}{Ls + R}$ $\quad G_2(s) = K_T$ $\quad G_3(s) = \dfrac{1}{J_E s + c_E}$

Step 1. Analyze the as-given open-loop system behavior.

Earlier the open-loop transfer function was derived and the real-world parameters for the NASA ARMII robot shoulder joint were substituted:

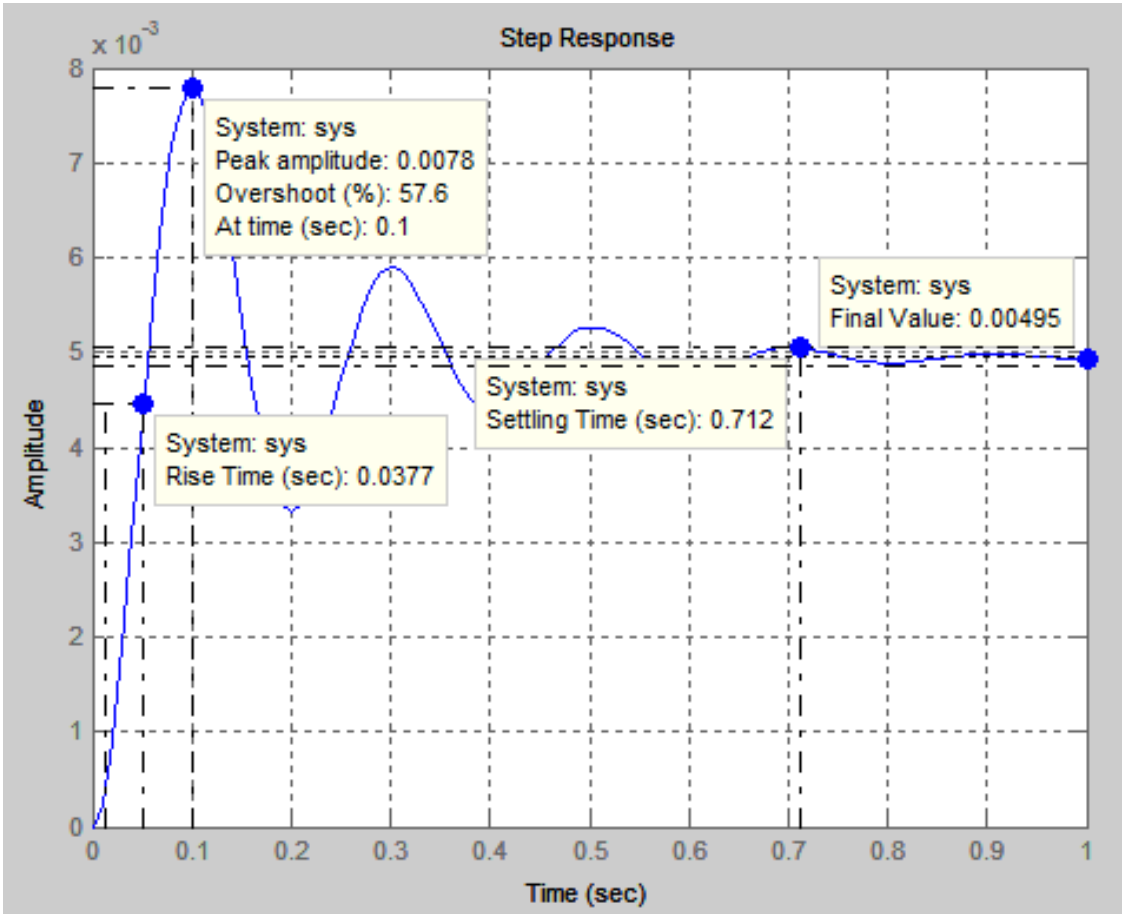
$$G_\omega(s) = \frac{\Omega_L(s)}{V_A(s)} = \frac{\frac{K_T}{n}}{(Ls+R)(J_E s + c_E) + K_T K_B} \cong \frac{5}{s^2 + 11s + 1010}$$

$$\Delta_{OL}(s) = s^2 + 2\xi\omega_n s + \omega_n^2 = s^2 + 11s + 1010$$

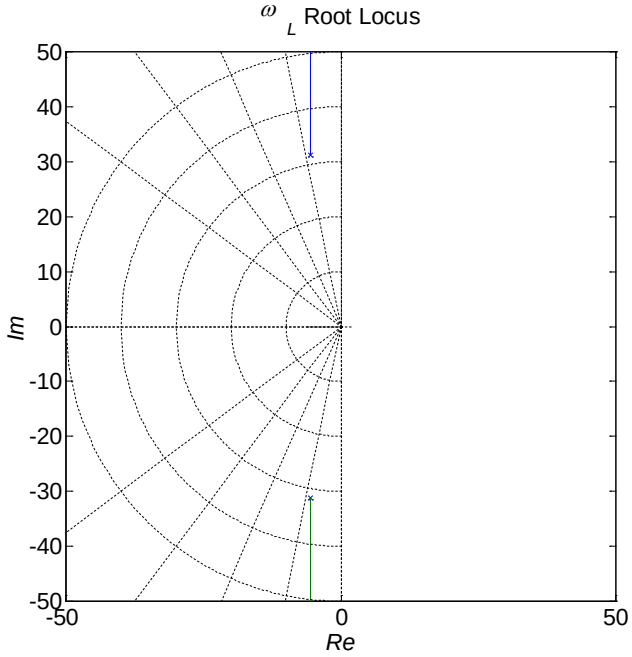
$$\omega_n = \sqrt{1010} = 31.8 \ rad/sec \qquad \xi = \frac{11}{2\omega_n} = \frac{11}{2\sqrt{1010}} = 0.173 \qquad s_{1,2} = -5.5 \pm 31.3i$$

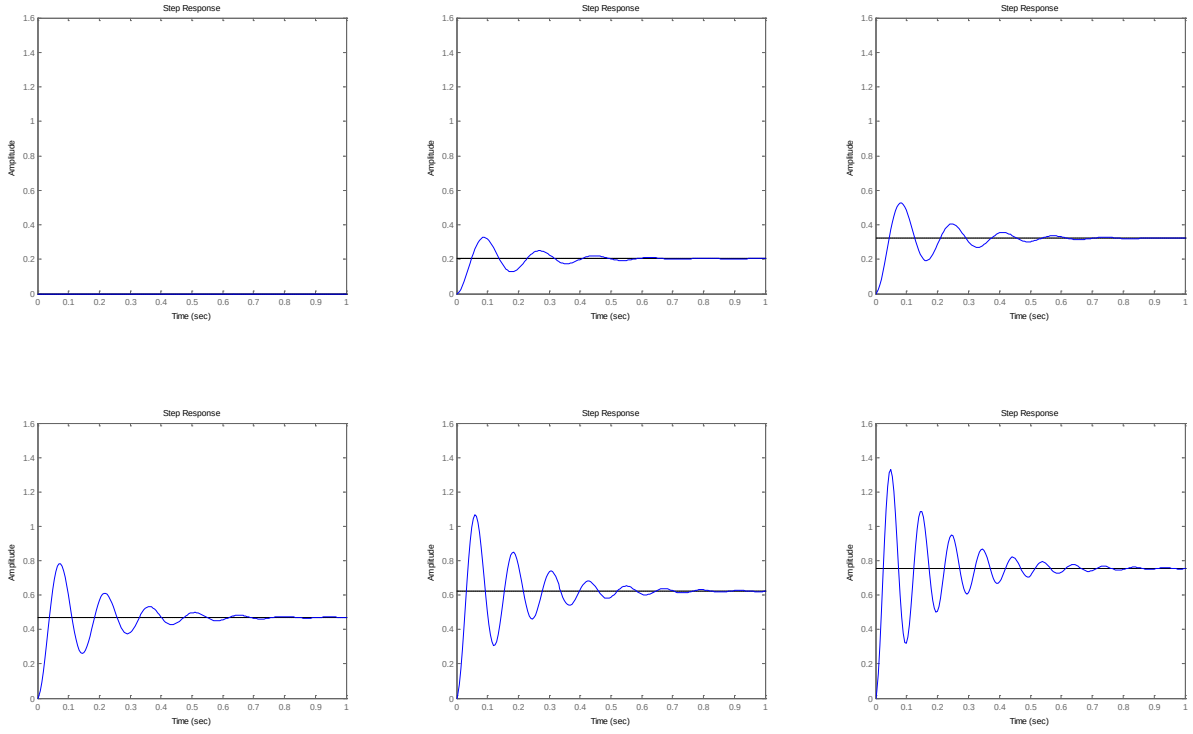
$$t_R = 0.04 \ sec \qquad t_P = 0.10 \ sec \qquad PO = 57.6\% \qquad t_S = 0.71 \ sec$$

This open-loop system has a different time scale than Controller Design Example 1; its output $\omega_L(t)$ rises much faster in response to a unit step input $v_A(t)$. Also, the steady-state value is 5/1010, whereas it was 1.0 for Controller Design Example 1.



First, take a look at the root-locus plots and unit step responses, for the proportional controller $G_C(s) = K$, first for angular velocity output:

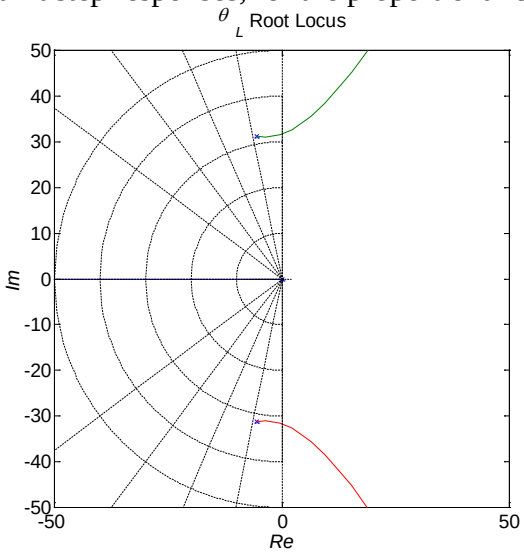


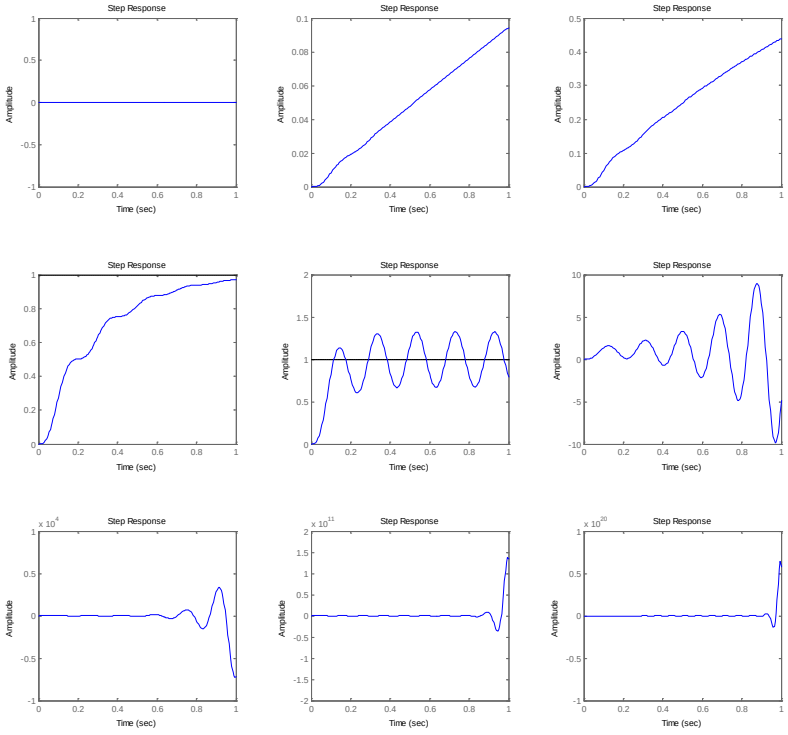


Term Example, $\omega_L(t)$ control, unit step responses as K increases from 0. K values legend:

0	52	96
179	334	622

Here are the root-locus plot and unit step responses, for the proportional controller $G_C(s) = K$, for angle:





Term Example, $\theta_L(t)$ control, unit step responses as K increases from 0. K values legend:

0	20	117
681	2220	3955
9529	55309	220190

The closed-loop system becomes unstable for $K \geq 2222$! Clearly the simple proportional controller will not work, since the angular velocity root-locus plot shows that the closed-loop dimensionless damping ratio must decrease as K increases.

Step 2. Specify and evaluate the desired behavior for the closed-loop system.

To determine the numerical desired characteristic polynomial, let's try something different: ITAE third-order. Arbitrarily let $\omega_n = 5$:

$$\Delta_{DES}(s) = s^3 + 1.75\omega_n s^2 + 2.15\omega_n^2 s + \omega_n^3 = s^3 + 8.75s^2 + 53.75s + 125$$

The associated desired closed-loop poles from third-order ITAE are: $-2.60 \pm 5.34i, -3.54$.

Step 3. Specify the form for the controller transfer function.

PID Controller: $$G_C(s) = K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s}$$

Step 4. State the controller design problem to be solved.

Given $G(s) = \dfrac{5}{s^2 + 11s + 1010}$, $H(s) = 1$, $\Delta_{CL_{DES}}(s) = s^3 + 8.75s^2 + 53.75s + 125$, and $G_C(s) = K_P + \dfrac{K_I}{s} + K_D s$, Solve for the PID controller gains K_P, K_I, and K_D. Then evaluate the PID controller performance.

Step 5. Solve for the unknown gains to achieve the desired behavior for the closed-loop system.

Derive the closed-loop transfer function as a function of the PID controller gains:

$$T(s) = \frac{G_C(s)G(s)}{1 + G_C(s)G(s)} = \frac{\dfrac{5\left(K_D s^2 + K_P s + K_I\right)}{s\left(s^2 + 11s + 1010\right)}}{1 + \dfrac{5\left(K_D s^2 + K_P s + K_I\right)}{s\left(s^2 + 11s + 1010\right)}} = \frac{\dfrac{5\left(K_D s^2 + K_P s + K_I\right)}{s\left(s^2 + 11s + 1010\right)}}{\dfrac{s\left(s^2 + 11s + 1010\right) + 5\left(K_D s^2 + K_P s + K_I\right)}{s\left(s^2 + 11s + 1010\right)}}$$

$$T(s) = \frac{5\left(K_D s^2 + K_P s + K_I\right)}{s\left(s^2 + 11s + 1010\right) + 5\left(K_D s^2 + K_P s + K_I\right)}$$

$$T(s) = \frac{5\left(K_D s^2 + K_P s + K_I\right)}{s^3 + \left(11 + 5K_D\right)s^2 + \left(1010 + 5K_P\right)s + \left(5K_I\right)}$$

Match the symbolic form (function of K_P, K_I, K_D) with the numerical ITAE desired characteristic polynomial.

Denominator parameter matching (fully-decoupled solution):

$s^3 \rightarrow 1 = 1$

$s^2 \rightarrow 11 + 5K_D = 8.75$

$s^1 \rightarrow 1010 + 5K_P = 53.75$

$s^0 \rightarrow 5K_I = 125$

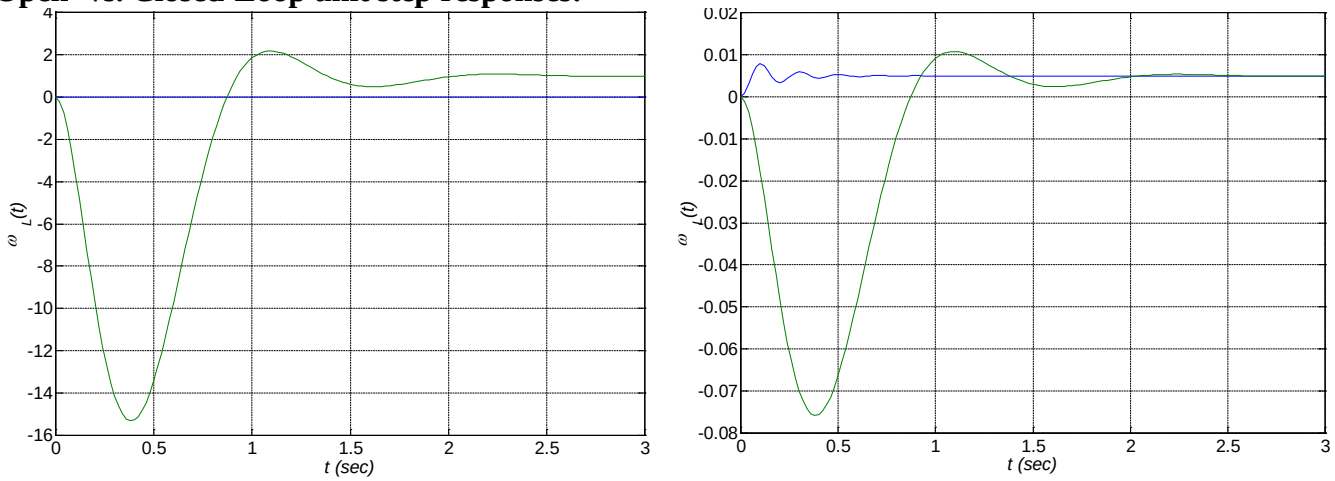
$K_D = -0.45$

$K_P = -191.25$ K_P and K_D are negative!

$K_I = 25$

Step 6. Evaluate the PID controller performance in simulation.

Open- vs. Closed-Loop unit step responses:



Open-Loop Closed-Loop PID

Step 7. Include an output attenuation correction factor if necessary.

Here the K_I term forces the steady-state output to 1.0 (zero steady-state error for unit step input); however, the original system output was not 1.0 (it was 5/1010, we can't even see it on the graph), so we need a correction factor to drive the closed-loop system to the same steady-state value (right plot):

$$R_{corr}(s) = R(s)(corr) = R(s)\frac{y_{sso}}{y_{ssc}} = R(s)\frac{5/1010}{1} = 0.00495R(s)$$

Corrected closed-loop PID transfer function: $$T(s) = \frac{5}{1010}\left(\frac{-2.25s^2 + -956.25s + 125}{s^3 + 8.75s^2 + 53.75s + 125}\right)$$

Again: **Controller design can easily make matters worse, if not done properly!** With negative K_P, the closed-loop step response initially goes down while open-loop goes up. The is a huge (negative) overshoot, and the transient response is slower than the open-loop case.

Step 9. Re-design and re-evaluate the controller.

Perform design iteration until the closed-loop performance specifications are met in simulation. We have no pre-filter yet (**Step 8**), but we have a bigger problem in the negative K_P and K_D.

Step 2. Specify and evaluate the desired behavior for the closed-loop system.

So let's choose a higher ω_n in the third-order ITAE desired closed-loop numerical characteristic polynomial for faster response and positive K_P, K_D. K_P is the limiting case, i.e. if we force it to be positive, K_D will be positive also:

$$11 + 5K_D = 1.75\omega_n$$

@ $K_D = 0$, $\omega_n = 6.29$, but then K_P will still be negative ($K_P = -185$); therefore let us instead set ω_n so that K_P will be positive:

$$1010 + 5K_P = 2.15\omega_n^2 \qquad @\ K_P = 0,\ \omega_n = 21.7$$

So try $\omega_n = 30$: $$\Delta_{DES}(s) = s^3 + 52.5s^2 + 1935s + 27000$$

The new associated desired closed-loop poles from third-order ITAE are: $-15.63 \pm 32.04i, -21.24$

Step 5. Solve for the unknown gains to achieve the desired behavior for the closed-loop system.

Denominator parameter matching (fully-decoupled):

$s^3 \rightarrow 1 = 1$

$s^2 \rightarrow 11 + 5K_D = 52.5$

$s^1 \rightarrow 1010 + 5K_P = 1935$

$s^0 \rightarrow 5K_I = 27000$

$K_D = 8.3$

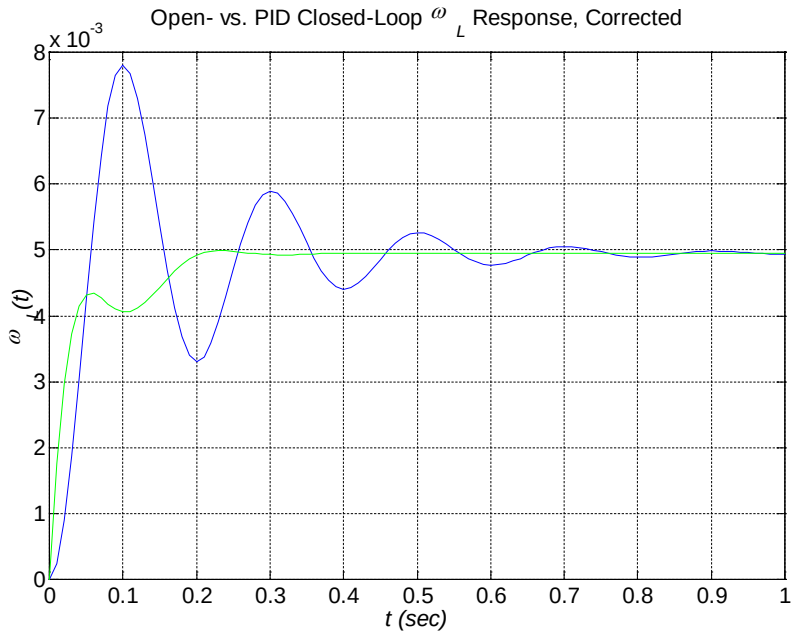
$K_P = 185$

$K_I = 5400$

K_P and K_D are both positive as planned!

Step 6. Evaluate the PID controller performance in simulation.

Open- vs. Closed-Loop unit step responses:



Open-Loop Closed-Loop PID

Closed-loop transfer function:

$$T(s) = \frac{5}{1010}\left(\frac{42s^2 + 925s + 27000}{s^3 + 52.5s^2 + 1935s + 27000}\right)$$

Step 7. Include an output attenuation correction factor if necessary.

We include the same correction factor as above.

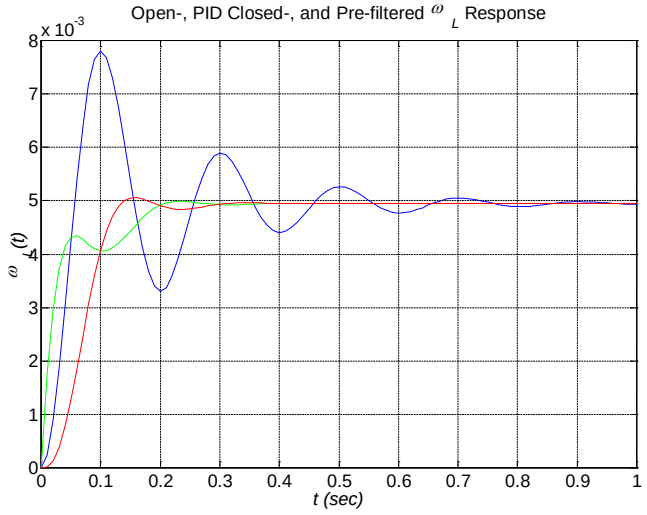
The controller response is much better with the positive K_P and K_D, but the third-order bend in the closed-loop response is not as desired. The desired ITAE response is different, i.e. it doesn't look like this closed-loop response. Therefore, we need a pre-filter.

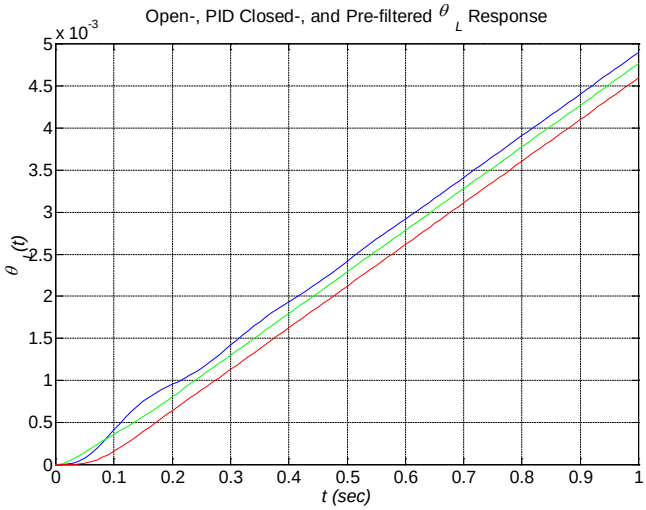
Step 8. Include a pre-filter transfer function if necessary.

Two unwanted zeros were introduced by the PID controller; these can be cancelled by a pre-filter.

Pre-filter transfer function: $$G_P(s) = \frac{27000}{42s^2 + 925s + 27000}$$

Note we specify the 27000 pre-filter transfer function numerator since the pre-filter shouldn't attenuate the output. When we plot the resulting unit step responses we see in the left plot below that the controller design is now successful:





Open-Loop Closed-Loop PID Closed-Loop PID with Pre-Filter

The pre-filtered closed-loop transfer function is:

$$T(s) = \frac{5}{1010}\left(\frac{27000}{s^3 + 52.5s^2 + 1935s + 27000}\right)$$

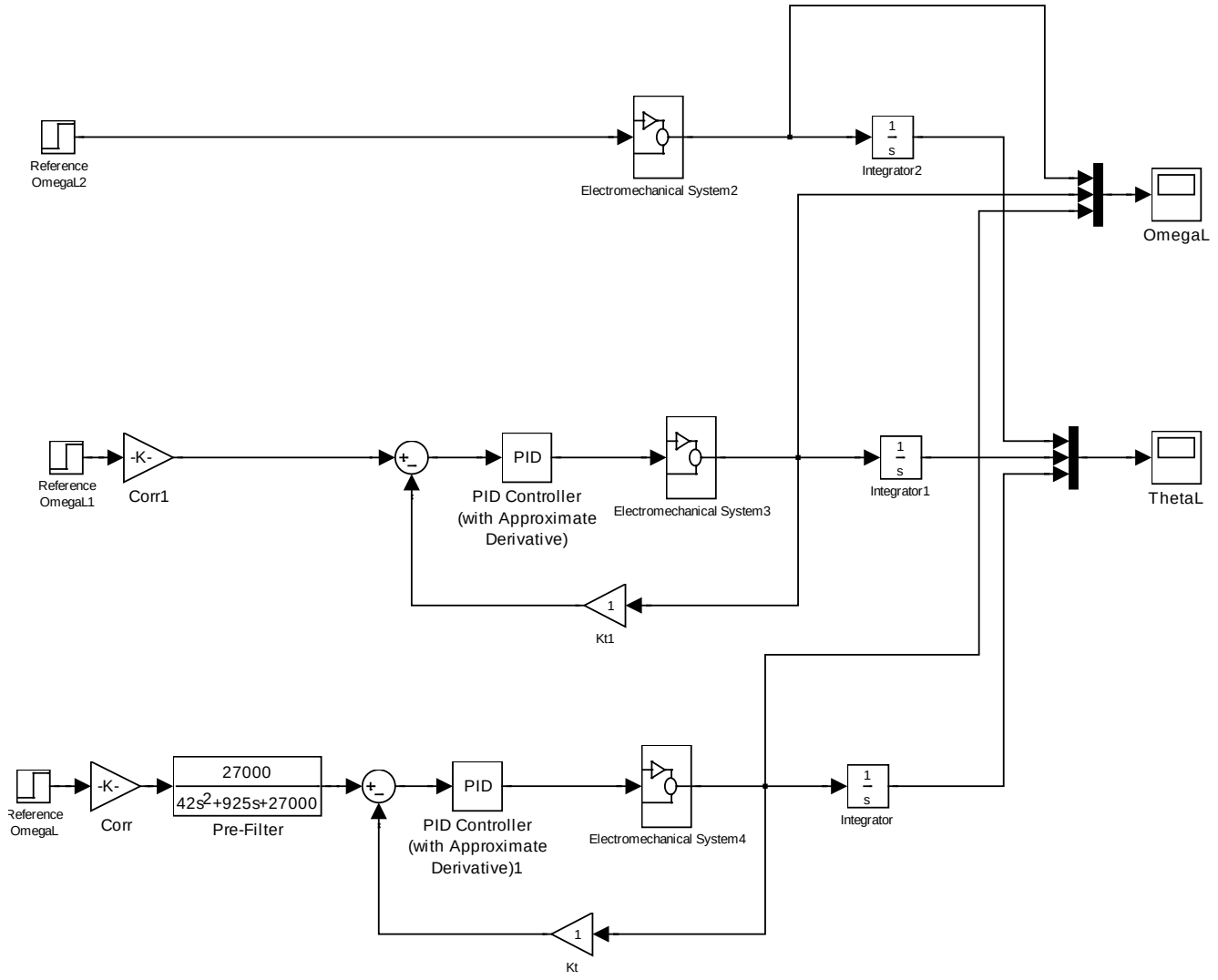
The final closed-loop unit step response is now theoretically identical to the specified third-order ITAE behavior. Note with the ITAE specification we didn't need to specify percent overshoot, settling time, or peak time.

For this controller, the angle $\theta_L(t)$ unit step response in time is given in the right plot above (note the steady-state error since the closed-loop PID lags the open-loop and the closed-loop PID with pre-filter lags even further). We just integrated the $\omega_L(t)$ response to get these results, i.e. we included another *s* factor in the closed-loop system denominator.

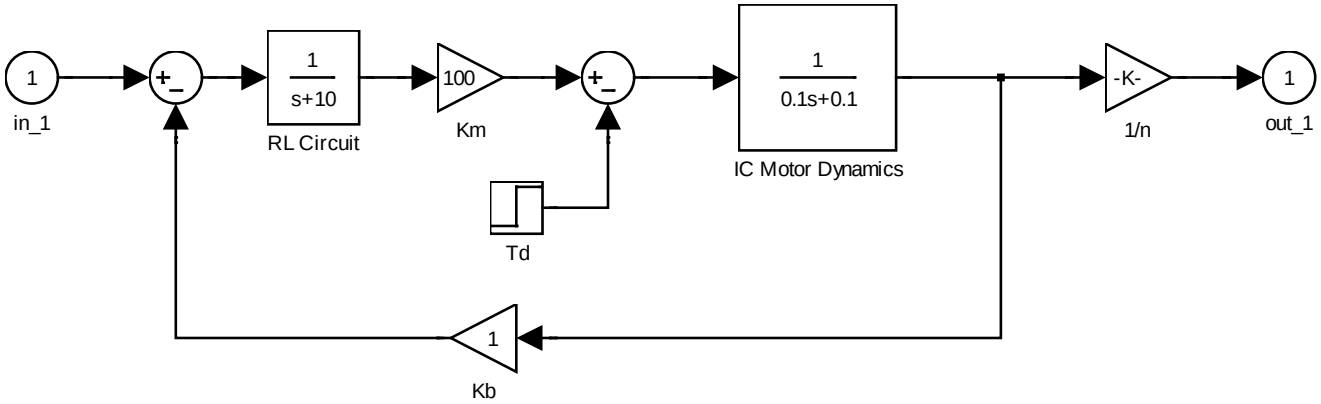
This Term Example controller design section is continued in the on-line 401 NotesBook Supplement, including Simulink model, and J-Method controller design and evaluation.

Step 6. Evaluate the PID controller performance in simulation.

Simulink Diagrams



The Electromechanical System blocks above are Simulink masks for the open-loop plant:



J-Method Controller

Alternate design method for Term Example $\omega_L(t)$ controller design.

Step 2. Specify the desired behavior for the closed-loop system We specify the entire desired closed-loop $T(s)$, not just the denominator. To determine the numerical desired entire closed-loop $T(s)$ transfer function, let's try something different: ITAE second-order.

$$\Delta_{DES}(s) = s^2 + 1.4\omega_n s + \omega_n^2$$
$$\Delta_{DES}(s) = s^2 + 42s + 900$$

ω_n = 30 (from successful PID controller design)

The associated desired closed-loop poles from third-order ITAE are: $-21 \pm 21.4i$.

$$\therefore T(s) = \frac{\frac{5}{1010}(900)}{s^2 + 42s + 900}$$

Step 4. Solve for the unknown controller form including the unknown gains. J-Method:

$$G_C(s) = \frac{T(s)}{G(s)\big(1 - H(s)T(s)\big)}$$

with $G(s) = \dfrac{5}{s^2 + 11s + 1010}$ and $H(s) = 1$

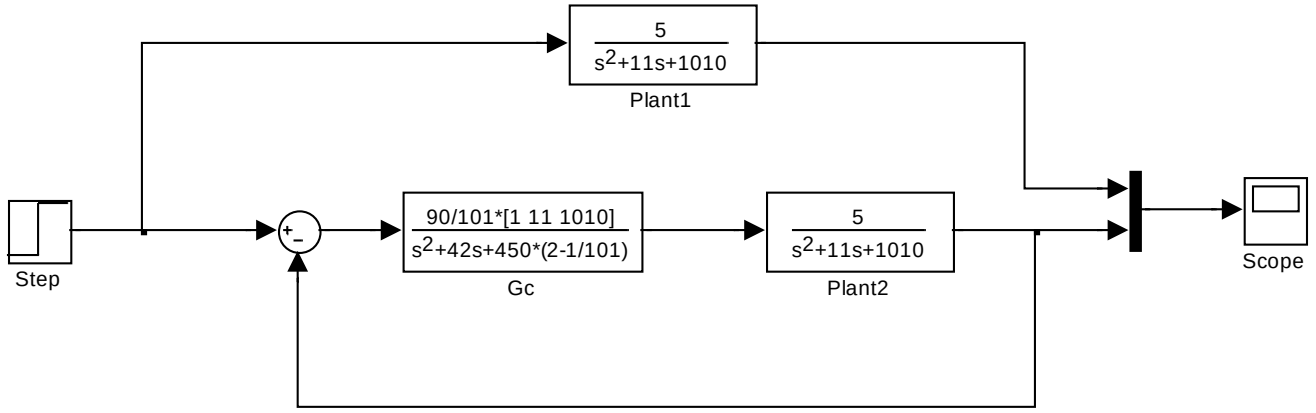
$$G_C(s) = \frac{\dfrac{\frac{5}{1010}(900)}{s^2 + 42s + 900}}{\left(\dfrac{5}{s^2 + 11s + 1010}\right)\left(1 - (1)\dfrac{\frac{5}{1010}(900)}{s^2 + 42s + 900}\right)}$$

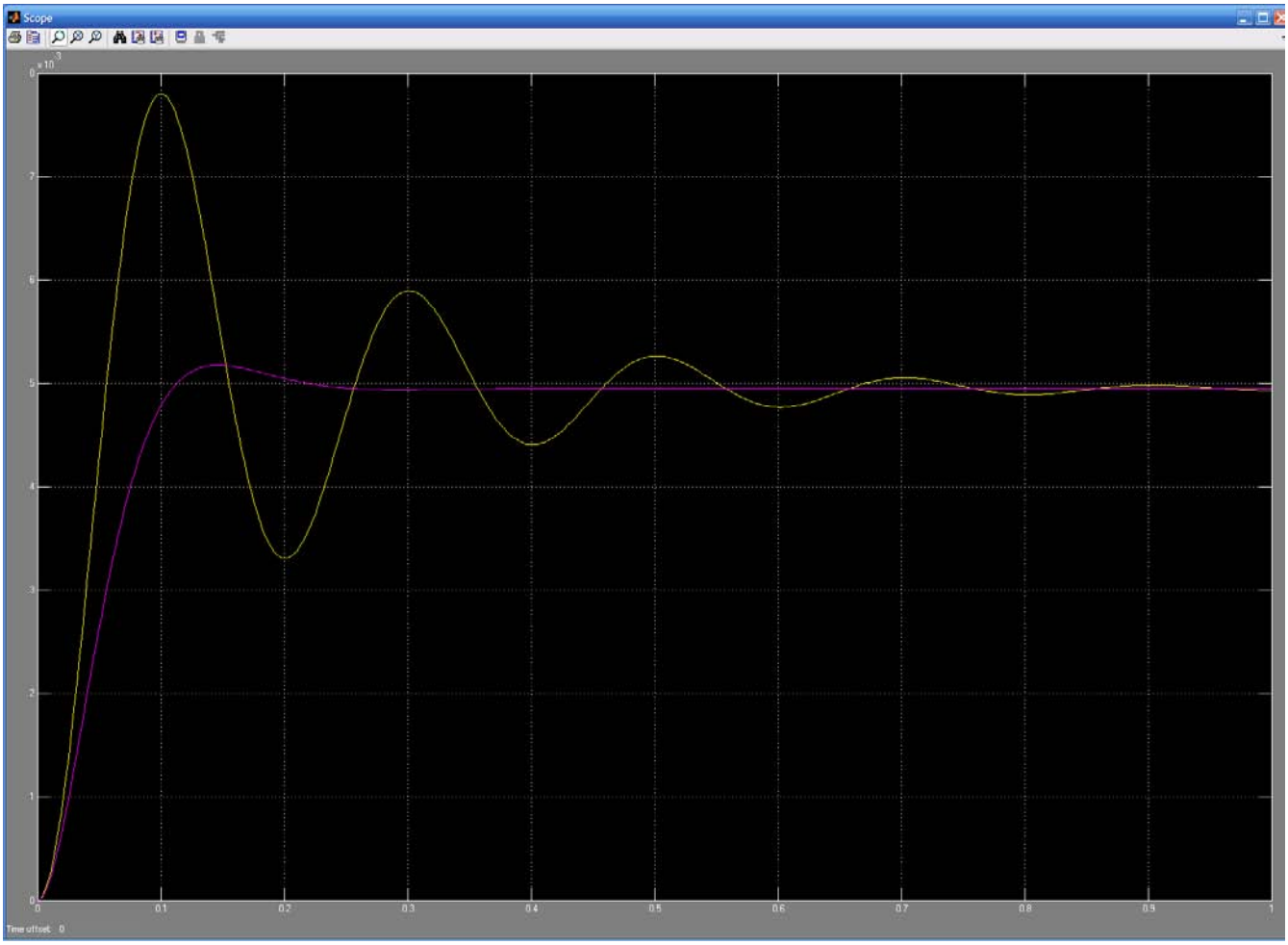
$$G_C(s) = \frac{\left(\dfrac{\frac{450}{101}}{s^2 + 42s + 900}\right)\left(\dfrac{s^2 + 11s + 1010}{5}\right)}{\left(\dfrac{s^2 + 42s + 900 - \frac{450}{101}}{s^2 + 42s + 900}\right)}$$

$$G_C(s) = \frac{\frac{90}{101}\left(s^2 + 11s + 1010\right)}{s^2 + 42s + 450\left(2 - \frac{1}{101}\right)}$$

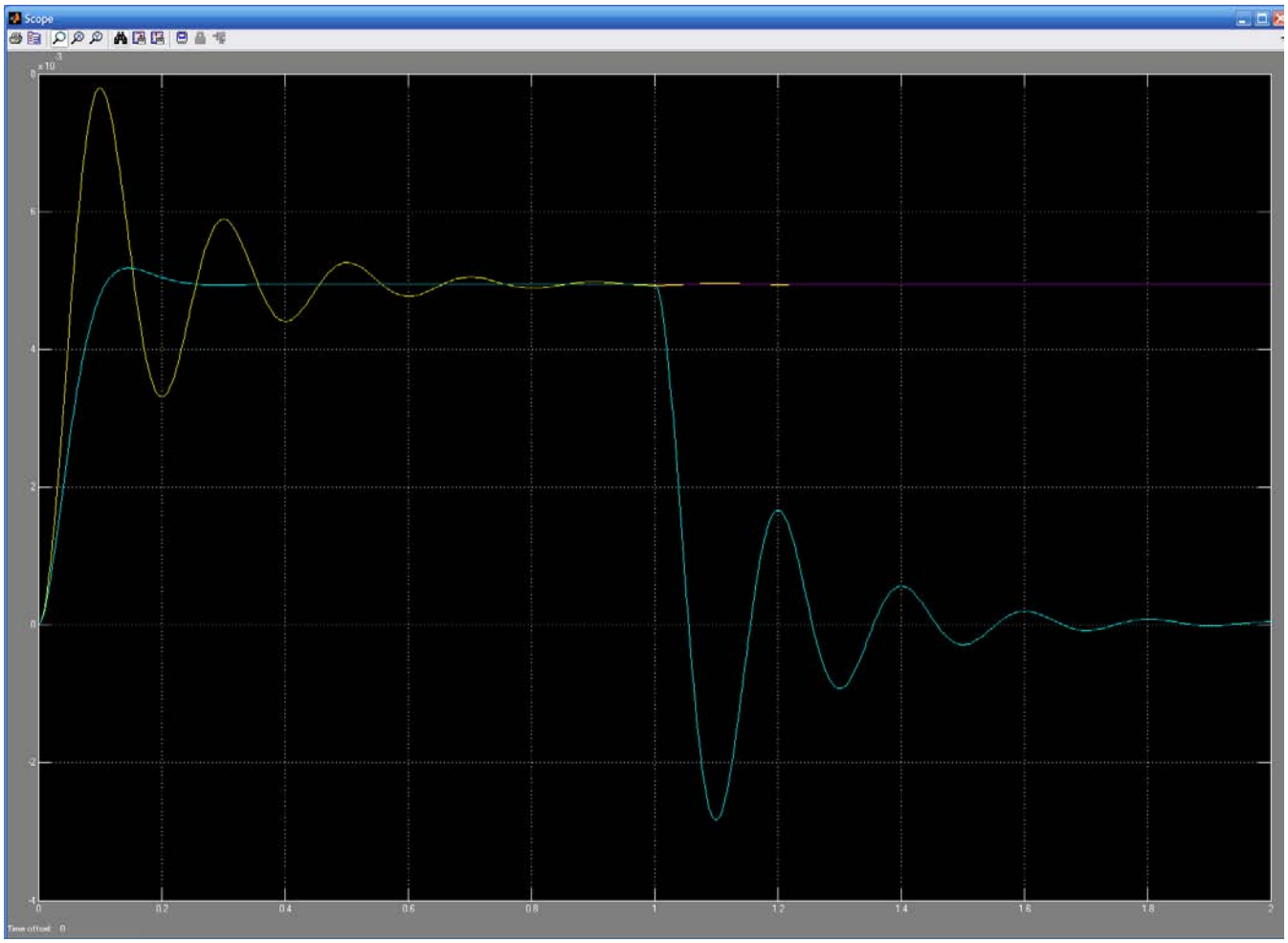
<u>Step 5.</u> Simulate the resulting closed-loop system performance.

Simulink Evaluation for the J-Method Term Example Controller Design, Angular Velocity





Open-Loop Closed-Loop J-Method



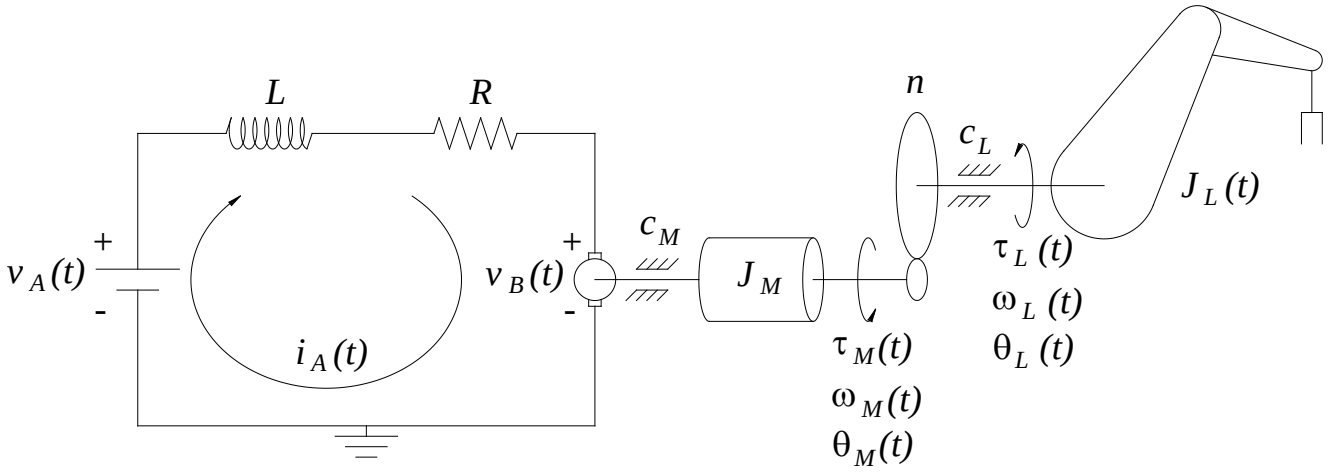
Open-Loop Closed-Loop J-Method Disturbed Closed-Loop J-Method

No need for a correction factor or pre-filter.

This J-Method controller for the Term Example load shaft angular velocity $\omega_L(t)$ control was successful as shown in the left plot above. But how does this controller handle disturbances?

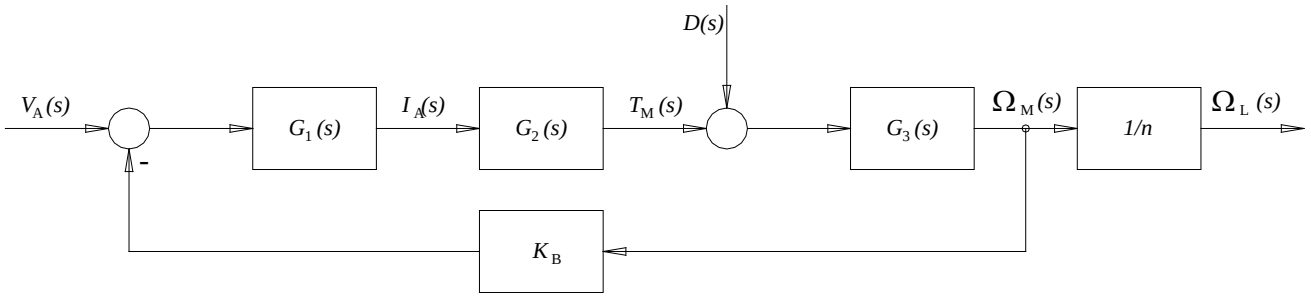
As we have seen before, the J-Method controller is very poor at handling disturbances, in this case a unit step disturbance in voltage, subtracted at t = 1 sec. The disturbed J-Method controller output response, shown in the right figure above, has unacceptable transient response and unacceptable steady-state error.

6.14 Term Example Disturbances and Steady-State Error



Though we have not presented it, it is possible to design controllers specifically to reject disturbances. This last example, the conclusion of the Term Example, discusses controller design to reject disturbances, specifically for achieving lower steady-state error.

Term Example open-loop diagram with disturbance



where $V_A(s)$ is the armature voltage input, $\Omega_L(s)$ is the load shaft angular velocity output, and $D(s)$ is the disturbance modeled at the actuator input level (disturbance in torque, Nm).

$$\text{where } G_1(s) = \frac{1}{Ls + R} \qquad G_2(s) = K_T \qquad G_3(s) = \frac{1}{J_E s + c_E}$$

Let $V_A(s) = 0$ and monitor changes in $\Omega_L(s)$ given a disturbance input. Since $V_A(s) = 0$, any $\Omega_L(s)$ is the steady-state error, that is, we wish as low an output as possible for the open-loop system.

Derive the open-loop disturbance transfer function $G_D(s) = \dfrac{\Omega_L(s)}{D(s)}$:

$$\frac{\Omega_M(s)}{T_M(s) - D(s)} = \frac{1}{J_E s + c_E}$$

Since $V(s) = 0$, $\dfrac{T_M(s)}{-K_B\Omega_M(s)} = \dfrac{K_T}{Ls+R}$ $\qquad \Omega_M(s) = n\Omega_L(s)$

$$\frac{n\Omega_L(s)}{\dfrac{-K_BK_Mn\Omega_L(s)}{Ls+R} - D(s)} = \frac{1}{J_Es+c_E} \qquad n\Omega_L(s)\left(J_Es+c_E\right) = \frac{-K_TK_Bn\Omega_L(s) - \left(Ls+R\right)D(s)}{Ls+R}$$

$$\left[n\left(J_Es+c_E\right)\left(Ls+R\right) + nK_TK_B\right]\Omega_L(s) = -\left(Ls+R\right)D(s)$$

$$G_D(s) = \frac{\Omega_L(s)}{D(s)} = \frac{-\left(Ls+R\right)}{n\left[\left(J_Es+c_E\right)\left(Ls+R\right) + K_TK_B\right]}$$

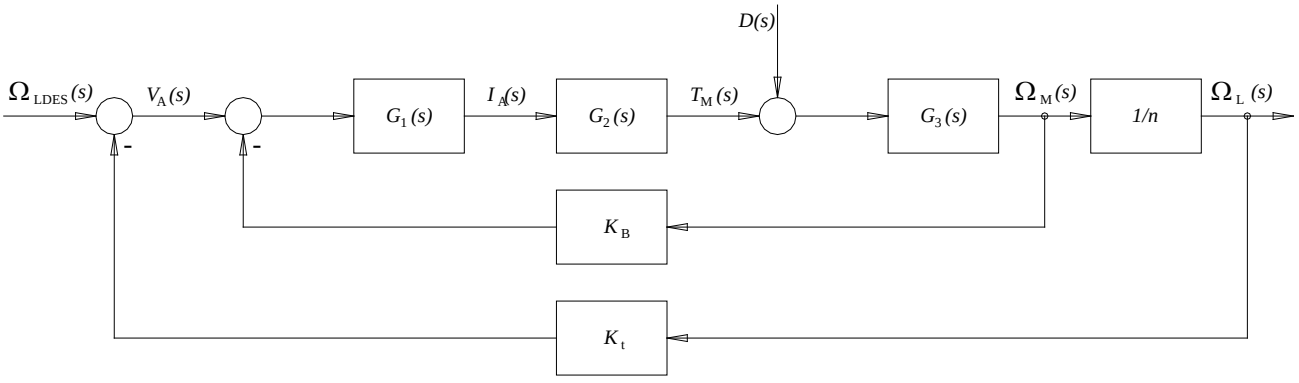
Open-loop steady-state error for a step disturbance of magnitude D: $D(s) = \dfrac{D}{s}$

$$E_{OL}(s) = \Omega_{LDES}(s) - \Omega_L(s) = -\Omega_L(s) = \frac{\left(Ls+R\right)}{n\left[\left(J_Es+c_E\right)\left(Ls+R\right) + K_TK_B\right]}D(s)$$

$$e_{ssOL} = \lim_{t\to\infty} e_{OL}(t) = \lim_{s\to 0} sE_{OL}(s) = \lim_{s\to 0} s\frac{\left(Ls+R\right)}{n\left[\left(J_Es+c_E\right)\left(Ls+R\right) + K_TK_B\right]}\frac{D}{s}$$

$$e_{ssOL} = \frac{RD}{n\left[RC_E + K_TK_B\right]}$$

Closed-loop diagram with disturbance, tachometer, and P controller



where $\Omega_{LDES}(s)$ is the desired load shaft angular velocity output, and the other terms have been previously defined.

Let $\Omega_{LDES}(s) = 0$ and check changes in $\Omega_L(s)$ given a disturbance input. Since $\Omega_{LDES}(s) = 0$, any $\Omega_L(s)$ is the steady-state error, that is, we wish as low an output as possible for the closed-loop system.

Derive the closed-loop disturbance transfer function $T_D(s) = \dfrac{\Omega_L(s)}{\Omega_{LDES}(s)}$:

$$\frac{\Omega_M(s)}{T_M(s) - T_D(s)} = \frac{1}{J_E s + c_E} \qquad \frac{T_M(s)}{A(s)} = \frac{K_T}{Ls + R} \qquad \Omega_M(s) = n\Omega_L(s)$$

$$A(s) = V_A(s) - K_B\Omega_M(s)$$
$$V_A(s) = KE_{CL}(s) = K\left(\Omega_{LDES}(s) - K_t\Omega_L(s)\right) = -KK_t\Omega_L(s)$$
$$A(s) = -KK_t\Omega_L(s) - nK_B\Omega_L(s) = -\left(KK_t + nK_B\right)\Omega_L(s)$$

$$T_M(s) = \frac{-K_T\left(KK_t + nK_B\right)\Omega_L(s)}{Ls + R}$$

$$\frac{n\Omega_L(s)}{\dfrac{-K_T\left(KK_t + nK_B\right)\Omega_L(s)}{Ls + R} - D(s)} = \frac{1}{J_E s + c_E}$$

$$n\Omega_L(s)\left(J_E s + c_E\right) = \frac{-K_T\left(KK_t + nK_B\right)\Omega_L(s) - \left(Ls + R\right)D(s)}{Ls + R}$$

$$\left[n\left(J_E s + c_E\right)\left(Ls + R\right) + K_T\left(KK_t + nK_B\right)\right]\Omega_L(s) = -\left(Ls + R\right)D(s)$$

$$T_D(s) = \frac{\Omega_L(s)}{\Omega_{LDES}(s)} = \frac{-\left(Ls + R\right)}{\left[n\left(J_E s + c_E\right)\left(Ls + R\right) + K_T\left(KK_t + nK_B\right)\right]}$$

Closed-loop steady-state error for a step disturbance of magnitude D: $D(s) = \dfrac{D}{s}$

$$E_{CL}(s) = \Omega_{LDES}(s) - \Omega_L(s) = -\Omega_L(s)$$

$$E_{CL}(s) = \frac{\left(Ls + R\right)}{\left[n\left(J_E s + c_E\right)\left(Ls + R\right) + K_T\left(KK_t + nK_B\right)\right]}D(s)$$

$$e_{ssCL} = \lim_{t\to\infty} e_{CL}(t) = \lim_{s\to 0} sE_{CL}(s)$$

$$e_{ssCL} = \lim_{s\to 0} s\frac{\left(Ls + R\right)}{\left[n\left(J_E s + c_E\right)\left(Ls + R\right) + K_T\left(KK_t + nK_B\right)\right]}\frac{D}{s}$$

$$e_{ssCL} = \frac{RD}{nRc_E + K_T\left(KK_t + nK_B\right)}$$

For disturbance-rejecting controller design, ensure the closed-loop steady-state error is less than the open-loop steady-state error.

$$\frac{e_{ssCL}}{e_{ssOL}} = \frac{RD}{nRc_E + K_T\left(KK_t + nK_B\right)} \frac{n\left[Rc_E + K_T K_B\right]}{RD}$$

$$\frac{e_{ssCL}}{e_{ssOL}} = \frac{n\left[Rc_E + K_T K_B\right]}{nRc_E + K_T\left(KK_t + nK_B\right)}$$

Design so $\frac{e_{ssCL}}{e_{ssOL}} << 1$

Numerical example from Term Example

$$e_{ssOL} = \frac{RD}{n\left[Rc_E + K_T K_B\right]} = \frac{10D}{200\left[1+100\right]} = \frac{10D}{20200}$$

$$e_{ssCL} = \frac{RD}{nRc_E + K_T\left(KK_t + nK_B\right)} = \frac{10D}{200+100\left(KK_t+200\right)} = \frac{10D}{20200+100KK_t}$$

$$\frac{e_{ssCL}}{e_{ssOL}} = \frac{10D}{20200+100KK_t}\frac{20200}{10D} = \frac{20200}{20200+100KK_t}$$

$$\frac{e_{ssCL}}{e_{ssOL}} = \frac{20200}{20200+100KK_t} < 1$$

$$20200+100KK_t > 20200$$
$$100KK_t > 0$$
$$KK_t >> 0$$

K_t tachometer selection
K proportional controller gain

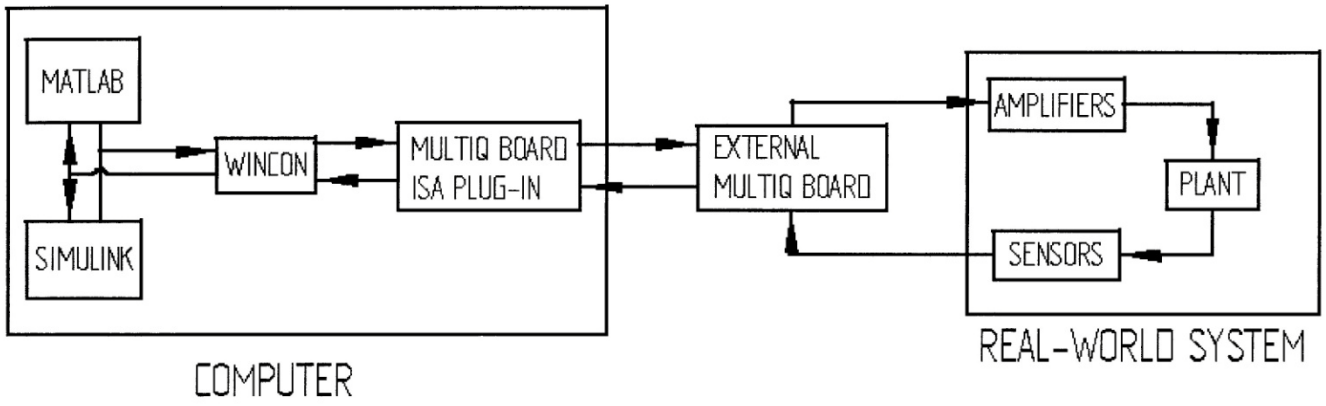
Assume the tachometer gain is fixed. Then the only way to reduce the closed-loop steady-state error is to increase proportional controller gain *K* as much as possible.

Now, this proportional controller for disturbance rejection is not very satisfying – we already know from the root-locus plot that the higher we make *K*, the worse the transient response will be (stable but more highly underdamped as *K* increases). So this demonstrates another tradeoff in controller design – to ensure lower steady-state errors due to unknown, unwanted disturbances, we must accept worse transient response performance.

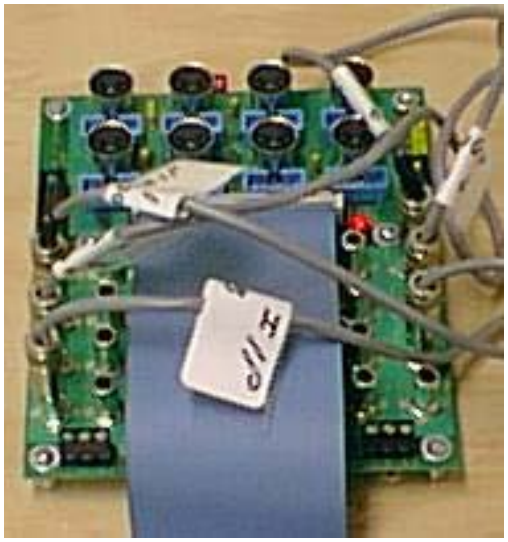
7. Hardware Control in the Ohio University Robotics Lab

7.1 Quanser/Simulink Controller Architecture

Quanser/Simulink Controller Diagram



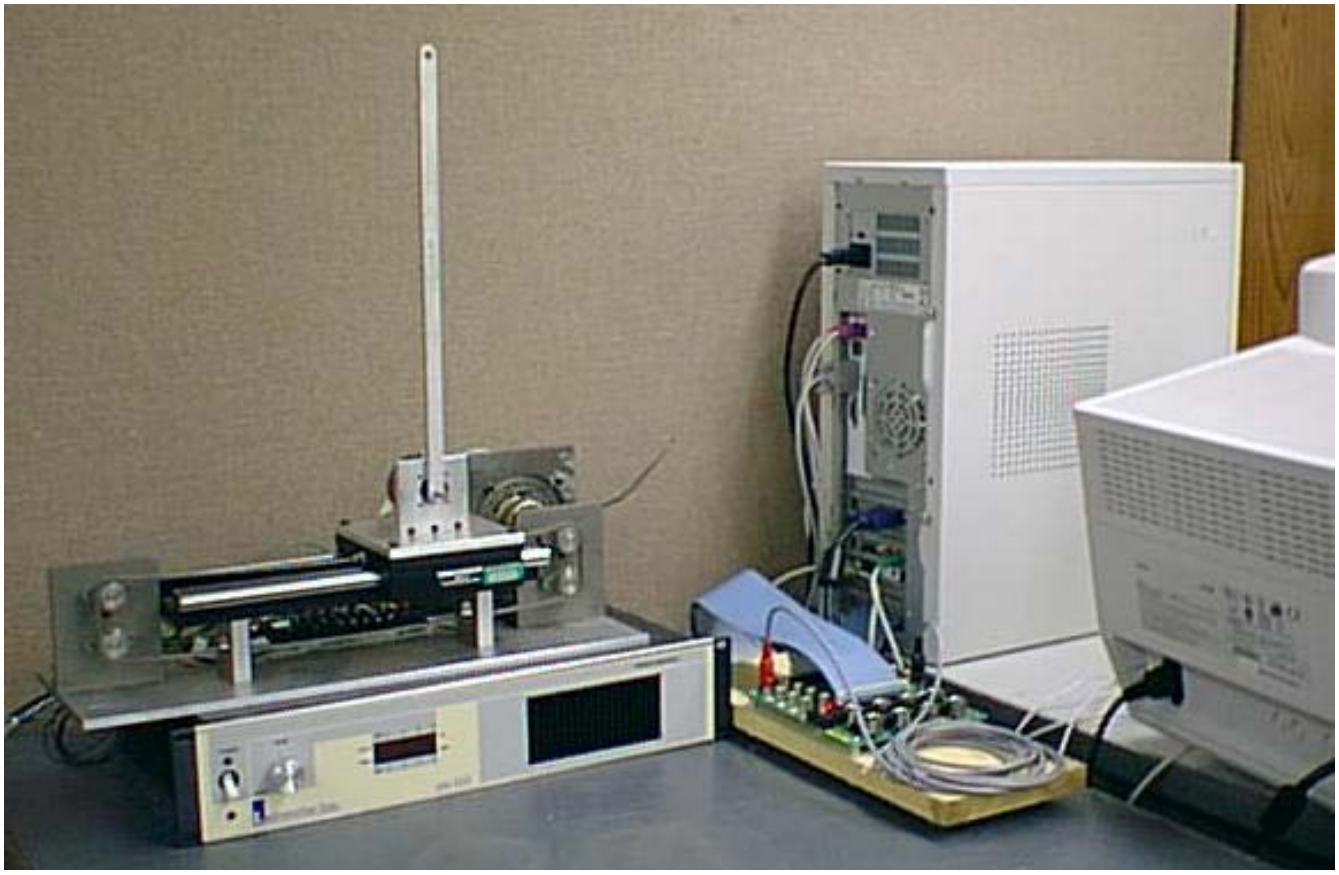
Quanser/Simulink Controller Hardware



External I/O Board

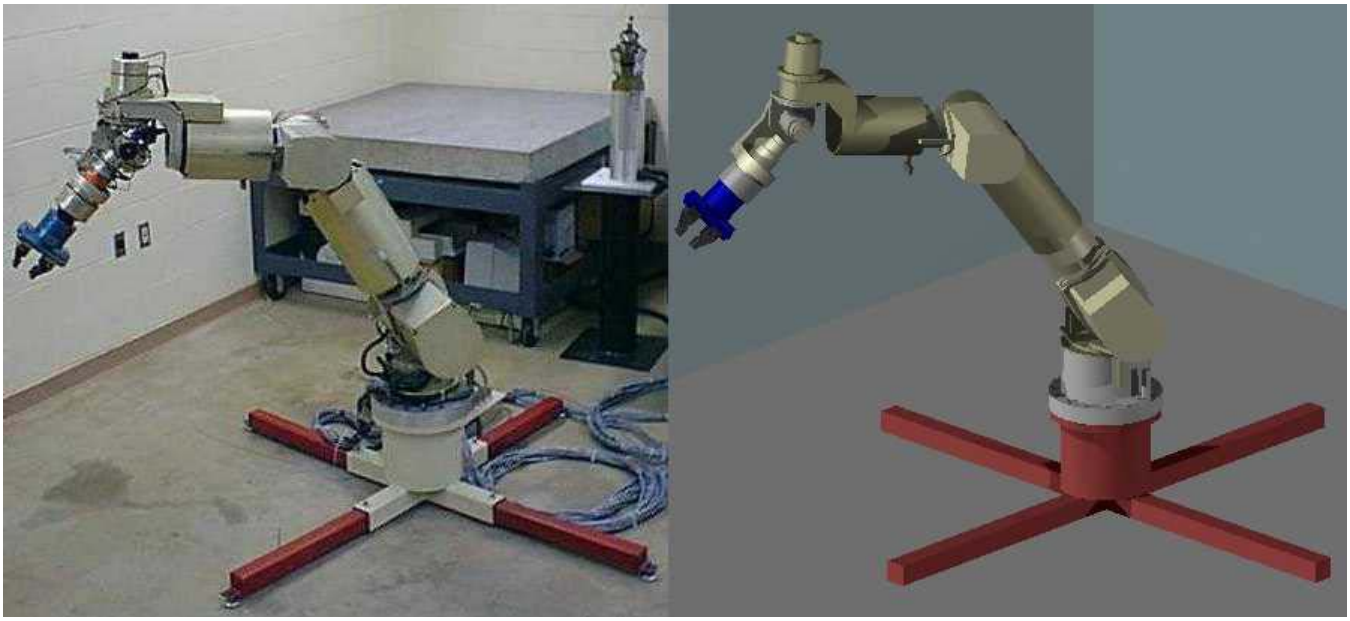
Power Supply

Controlled via Quanser/Simulink

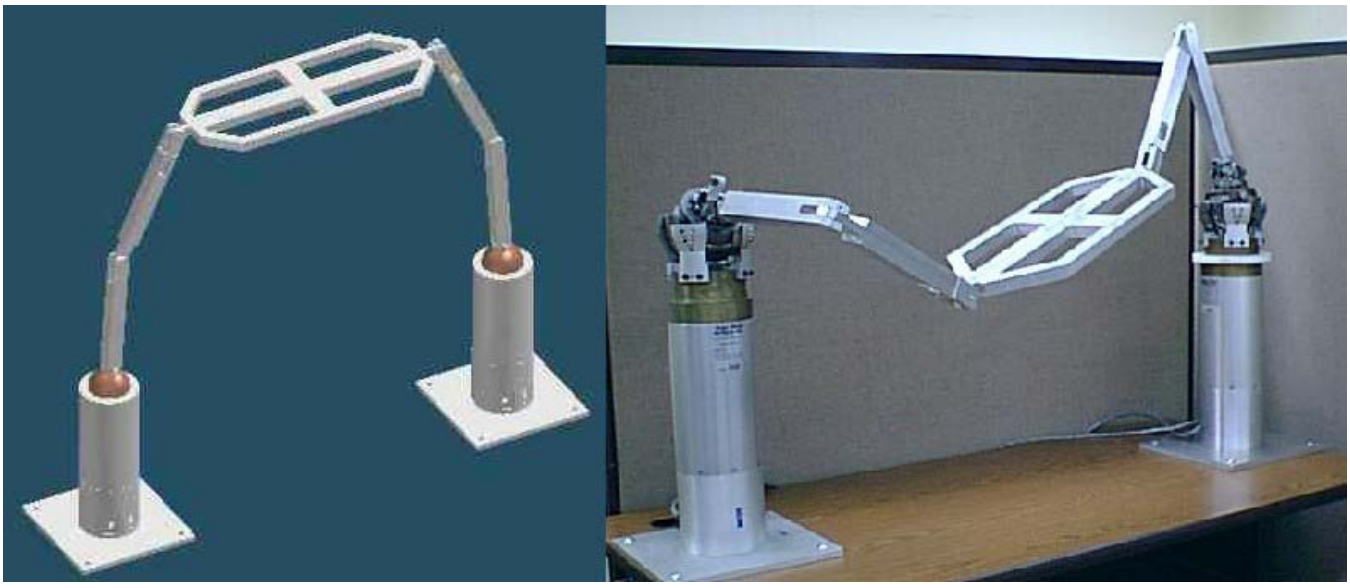


Inverted Pendulum

Controlled via Quanser/Simulink (continued)

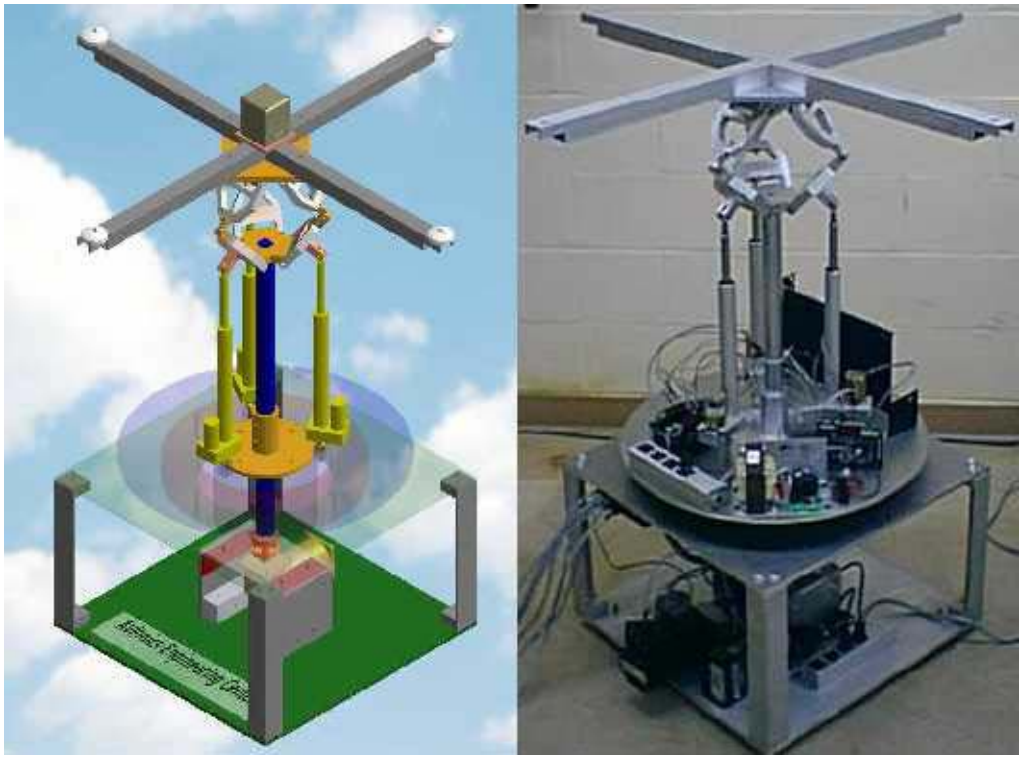


8-axis NASA ARMII (Advanced Research Manipulator II)

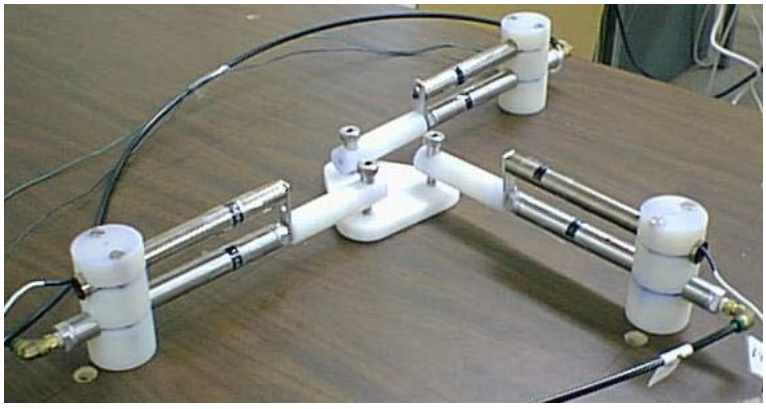


SPAM (SPherically-Actuated platform Manipulator)

Controlled via Quanser/Simulink (continued)

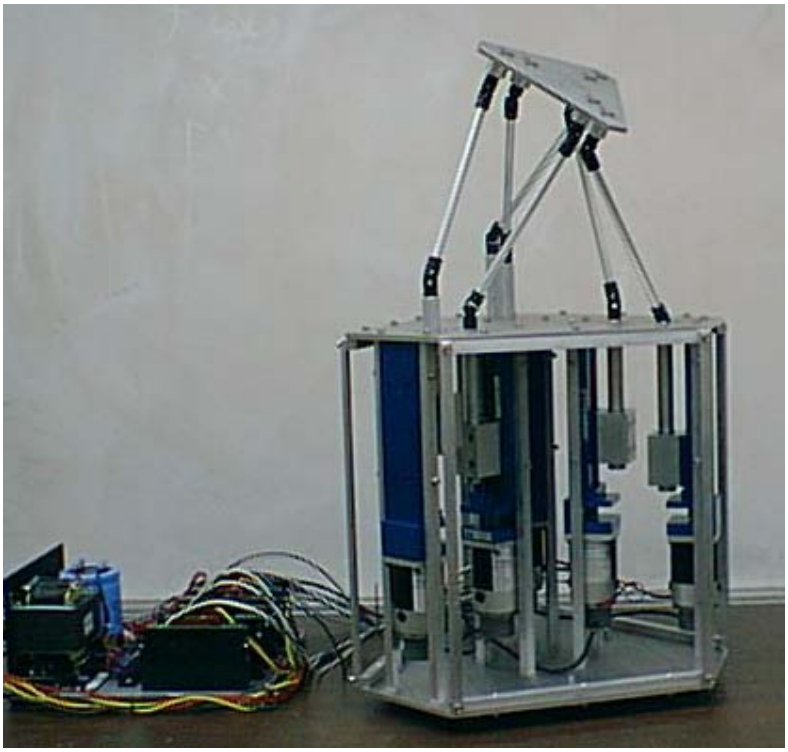


4-dof GPS/IMU Calibration Platform

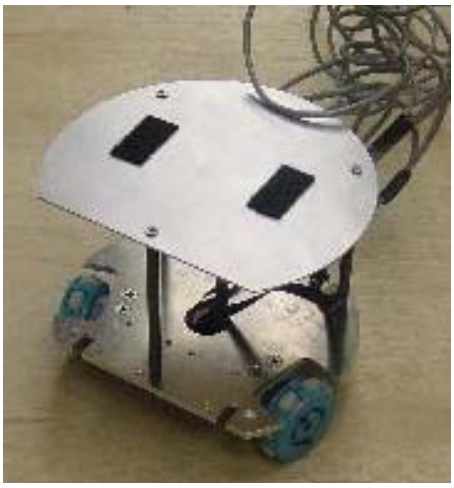


3-dof Planar 3-RPR Parallel Robot

Controlled via Quanser/Simulink (continued)



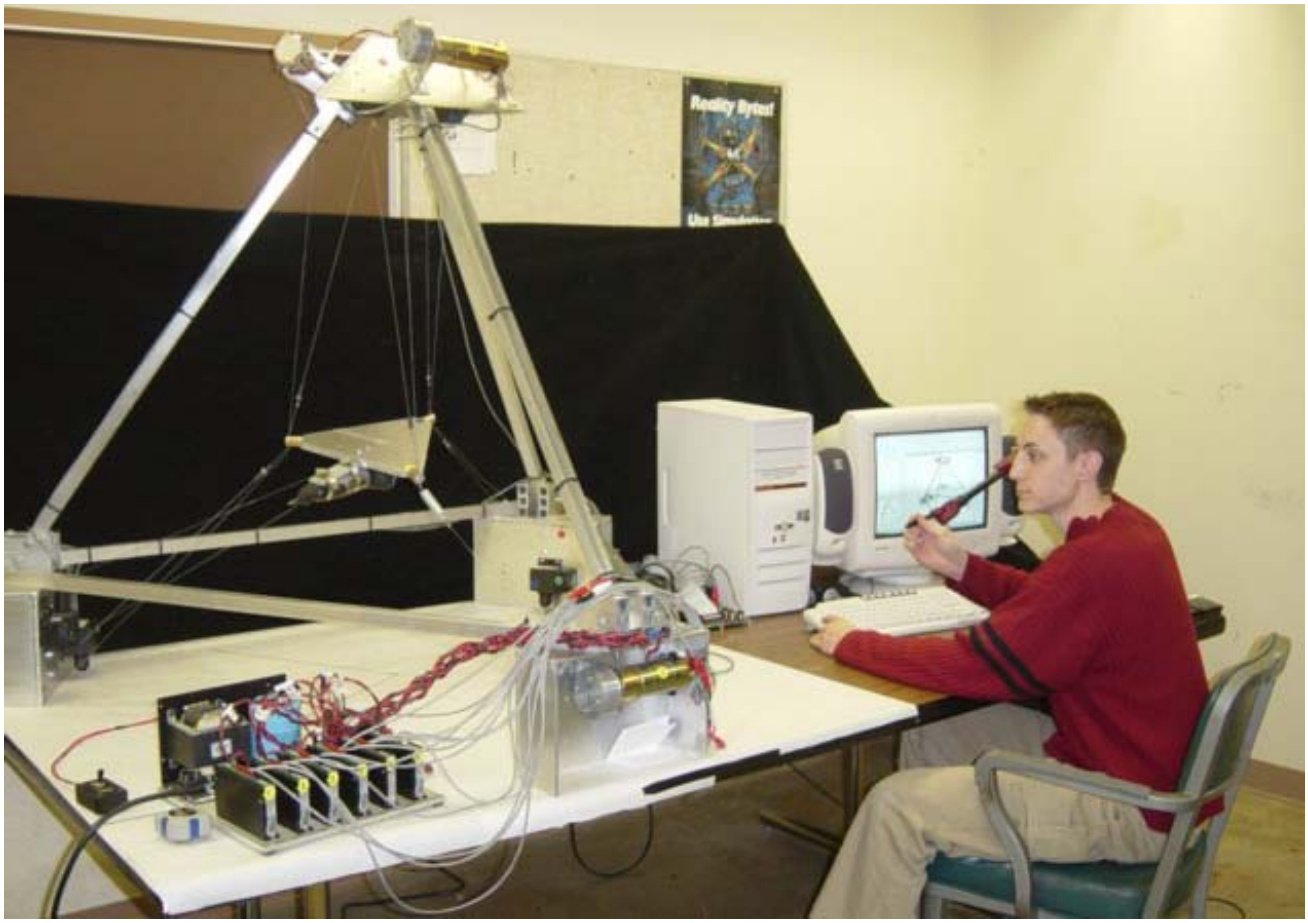
6-dof Spatial 6-PSU Platform



RoboCup Player

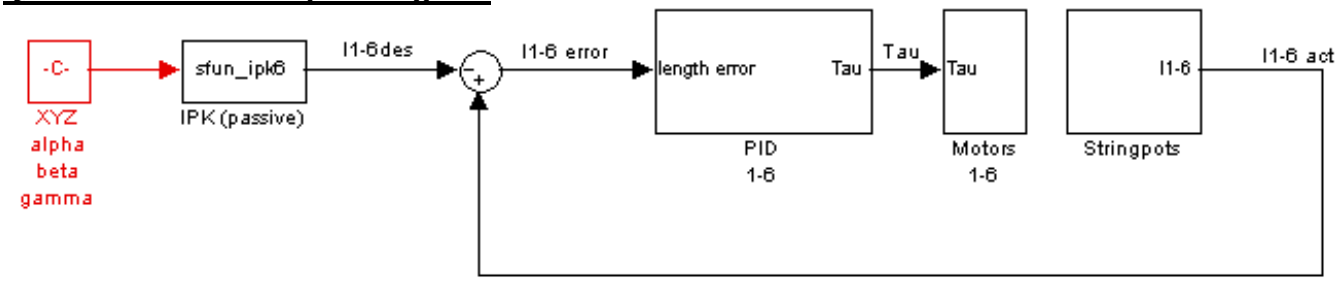
Cable-Suspended Haptic Interface

Controlled via Quanser/Simulink (concluded)



7-Cable Robot with Amplifiers

Quanser/Simulink Split Diagram

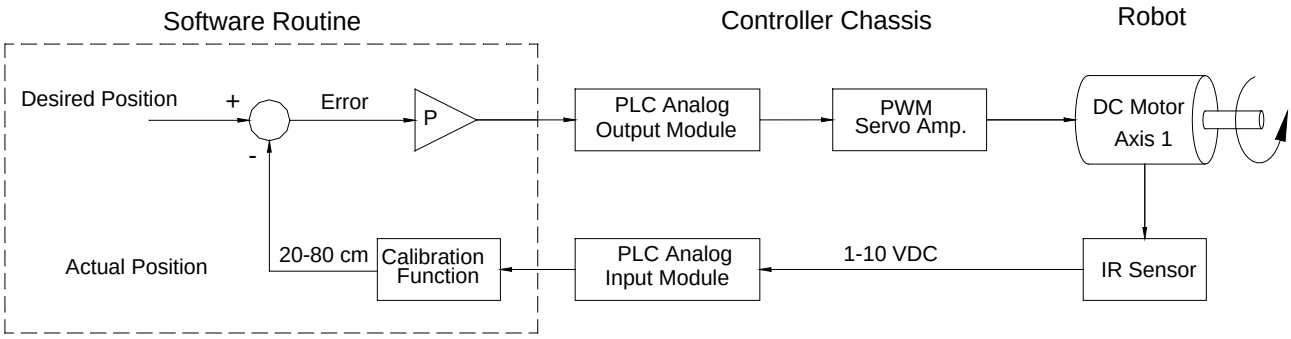


Here the Simulink closed-loop feedback diagram for the 7-cable robot appears to be open (split) and incomplete; the plant is missing. However, the real-world machine under control serves as the plant and no model of this plant is required since the real-world behaves as it will, much better than any model we can construct. So the gap (split) in this diagram is where the real-world plant is (motors outputs to and sensors readings (stringpots in this example) from the real-world).

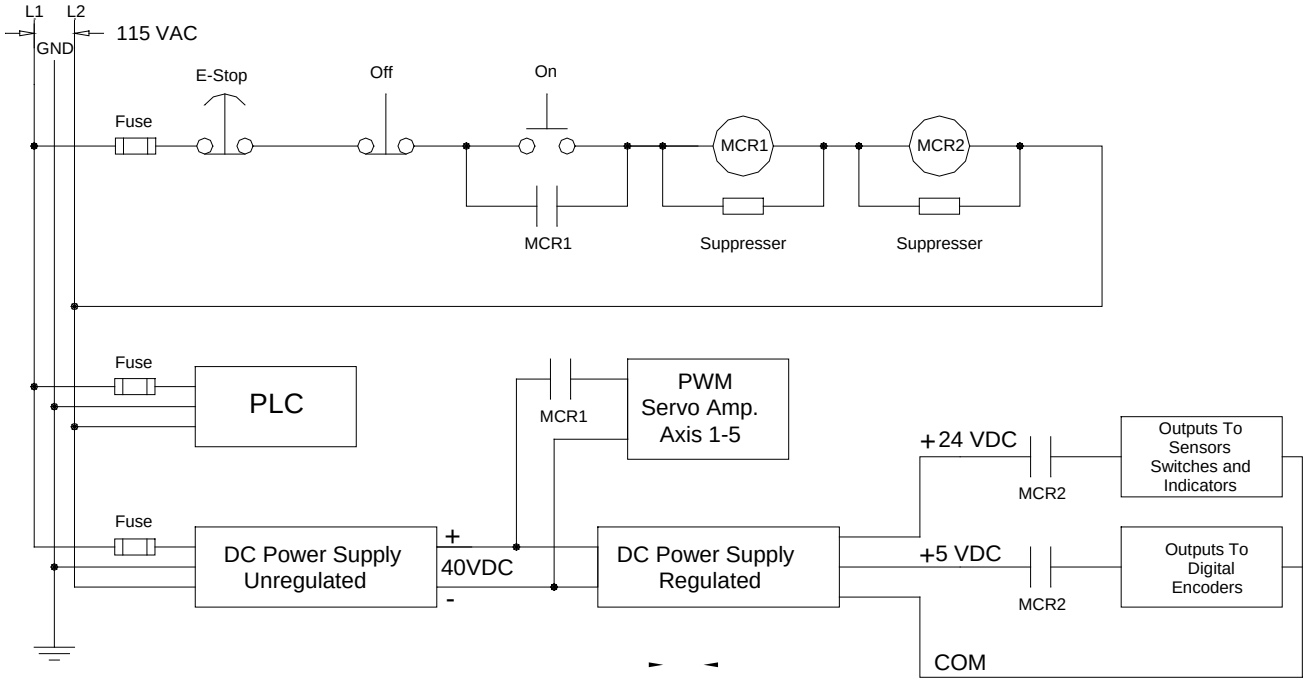
The seventh motor is controlled in a different way for tensioning the cables and entire end-effector.

7.2 Programmable Logic Controller (PLC) Architecture

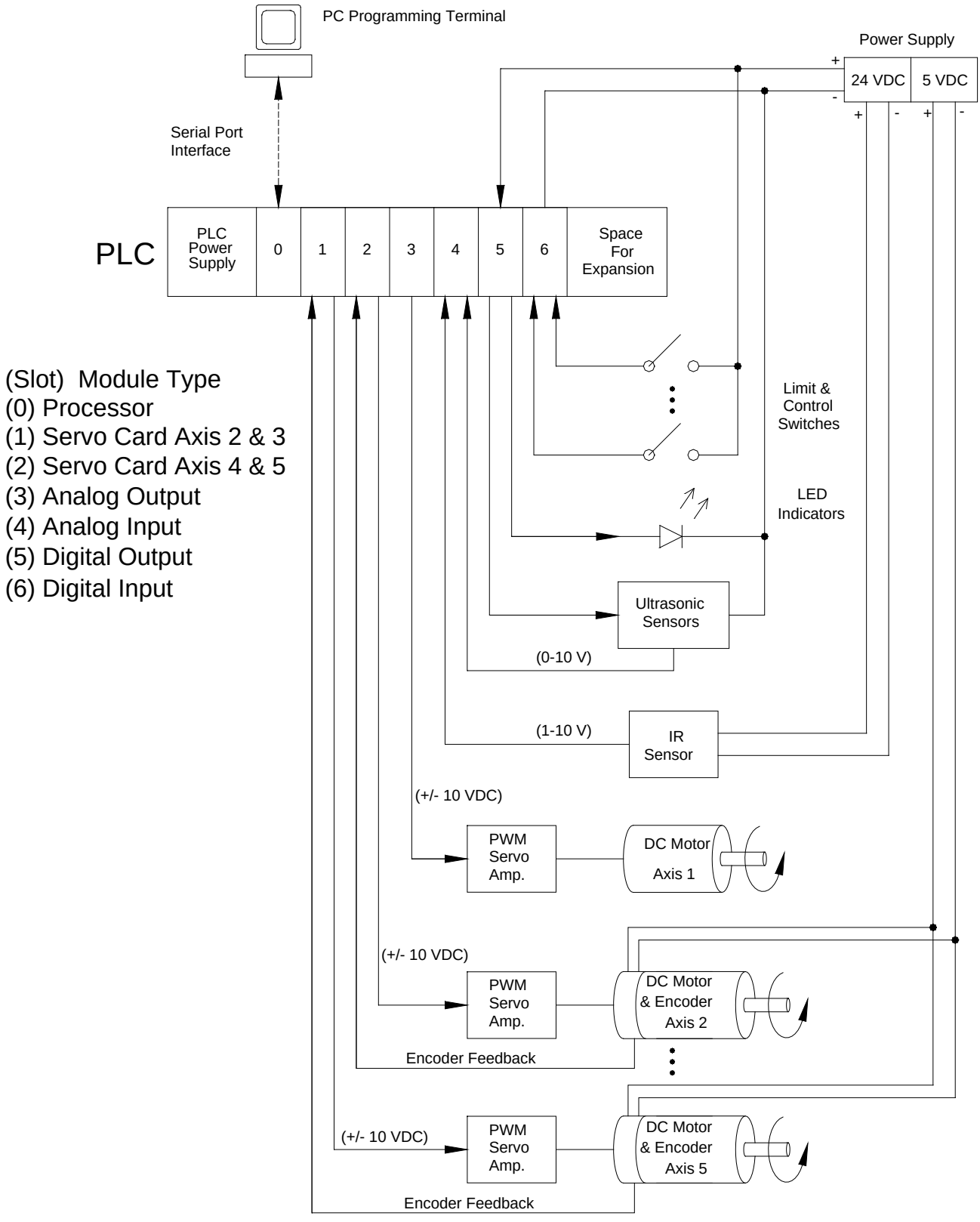
PLC Controller Diagrams



Closed-Loop Position Control of Axis 1

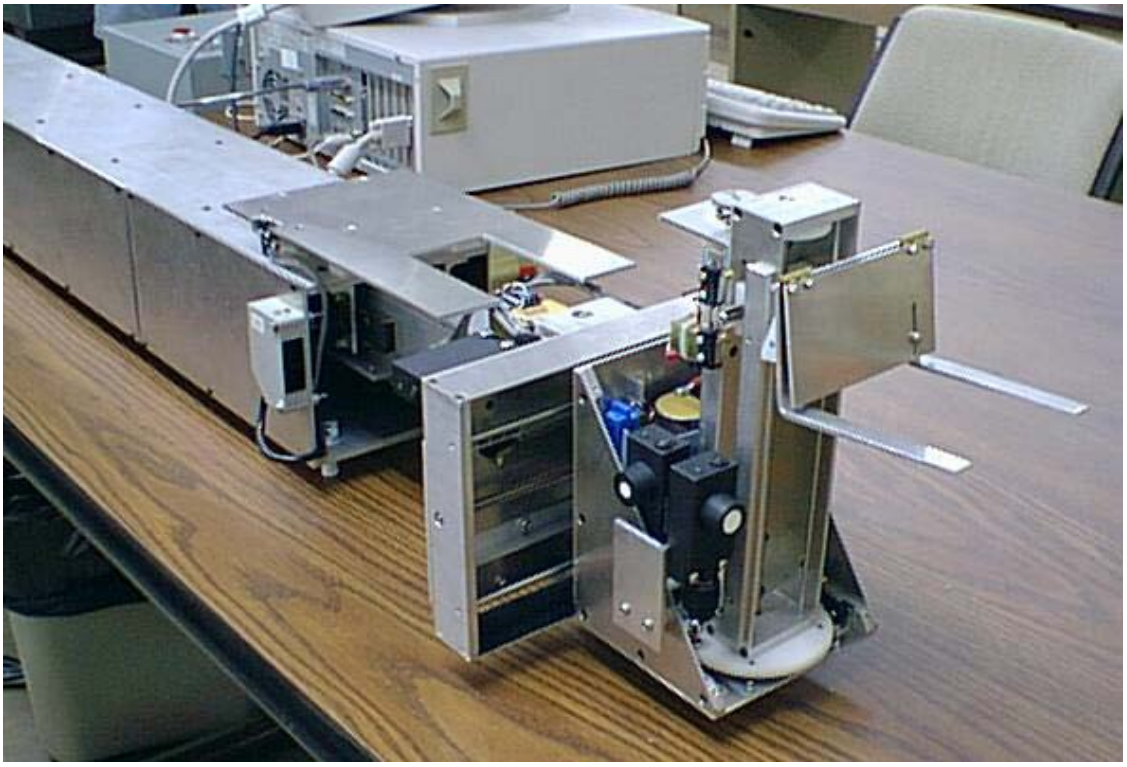


Schematic of Controller Layout and Power Distribution

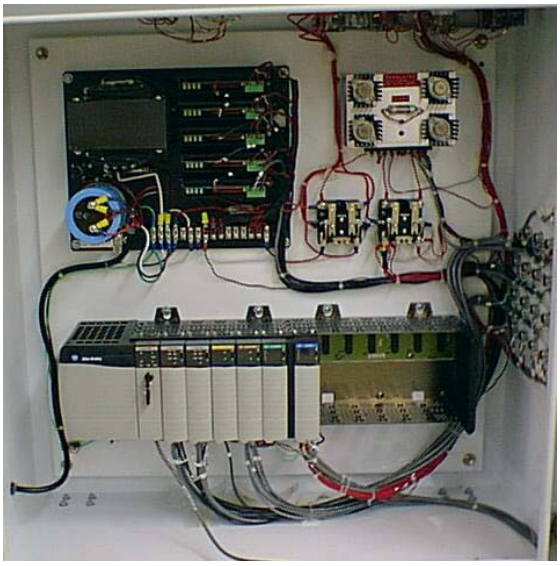


PLC Control System Hardware: Ladder Logic

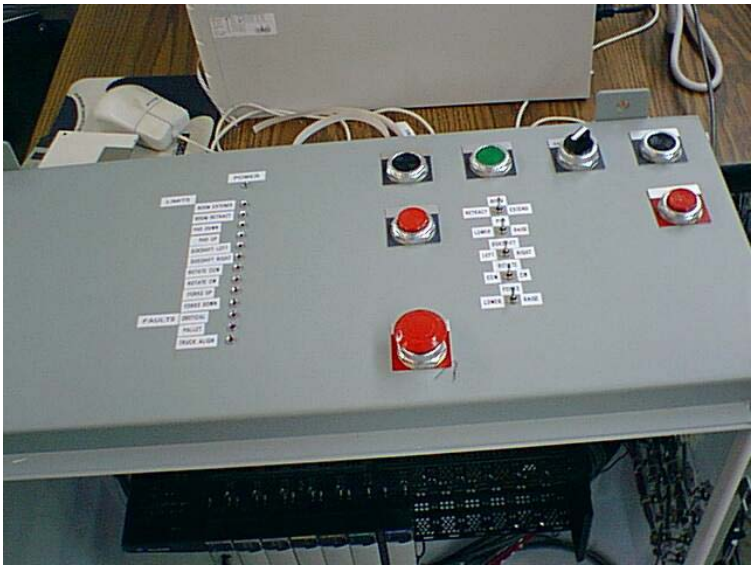
Controlled via PLC



Stewart-Glapat Pallet Handling Device (PHD)



PLC Cabinet



PLC Interface