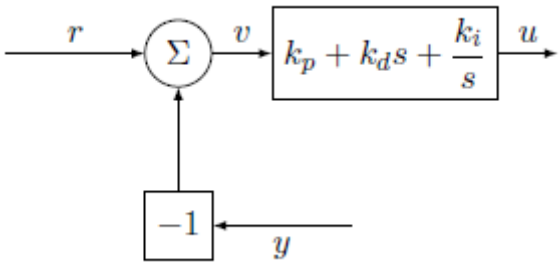
AE 353 Homework #8:

PID and Time Delay

1. a) PID Controller:



The transfer function from y to u can be written as:

$$T_{yu}(s) = -\left(k_p + k_d s + \frac{k_i}{s}\right)$$

$$T_{yu}(j\omega) = -\left(k_p + k_d j\omega + \frac{k_i}{j\omega}\right)$$

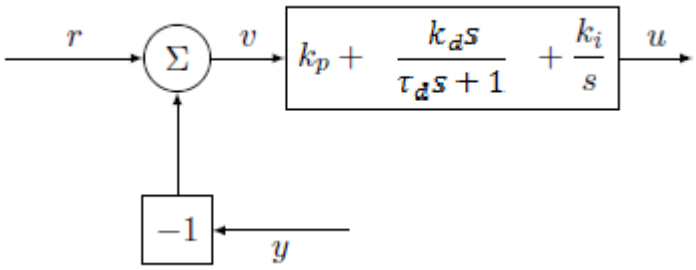
$$= -k_p + j\left(\frac{k_i}{\omega} - k_d\omega\right)$$

Therefore,

$$\left|T_{yu}(j\omega)\right| = \sqrt{k_p^2 + \left(\frac{k_i}{\omega} - k_d\omega\right)^2}$$

As $\omega \to \infty$, we can see that $\left|T_{yu}(j\omega)\right|$ will also tend to infinity. Thus theoretically, a high frequency measurement noise might get amplified to a great extent.

b) Another way to implement the PID controller:



Here,

$$T_{yu}(s) = -\left(k_p + \frac{k_d s}{\tau_d s + 1} + \frac{k_i}{s}\right)$$

After re-arranging the terms, $\left|T_{yu}(j\omega)\right|$ can be written as:

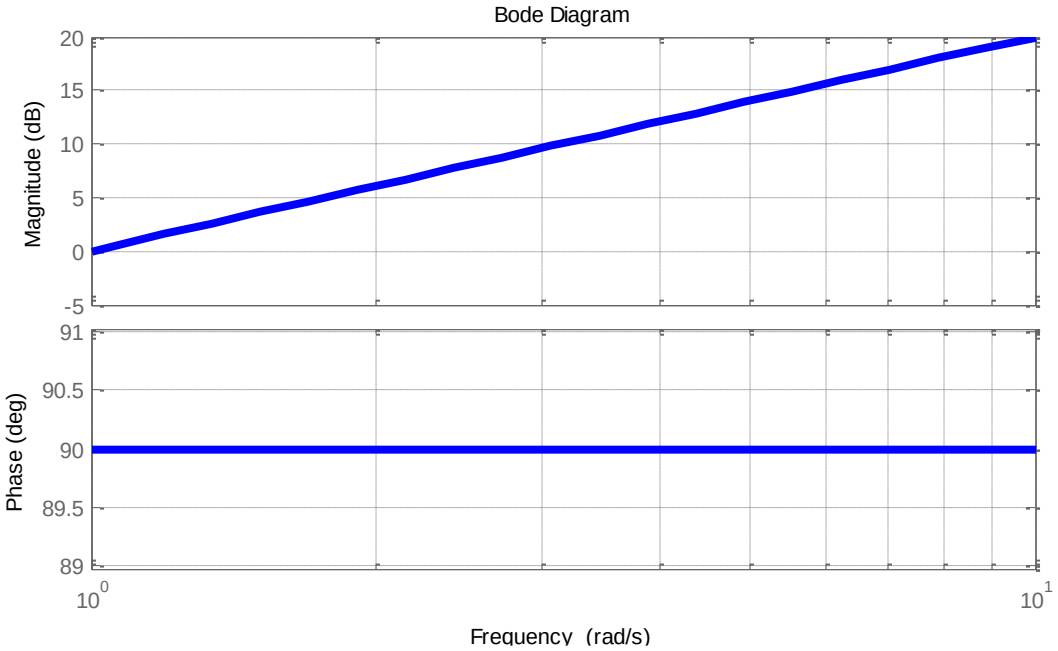
$$\left|T_{yu}(j\omega)\right| = \sqrt{\left(k_p + \frac{k_d \tau_d}{\tau_d^2 + \frac{1}{\omega^2}}\right)^2 + \left(\frac{k_i}{\omega} - \frac{\frac{k_d}{\omega}}{\tau_d^2 + \frac{1}{\omega^2}}\right)^2}$$

As $\omega \to \infty$, $\left|T_{yu}(j\omega)\right| = k_p + k_d/\tau_d$

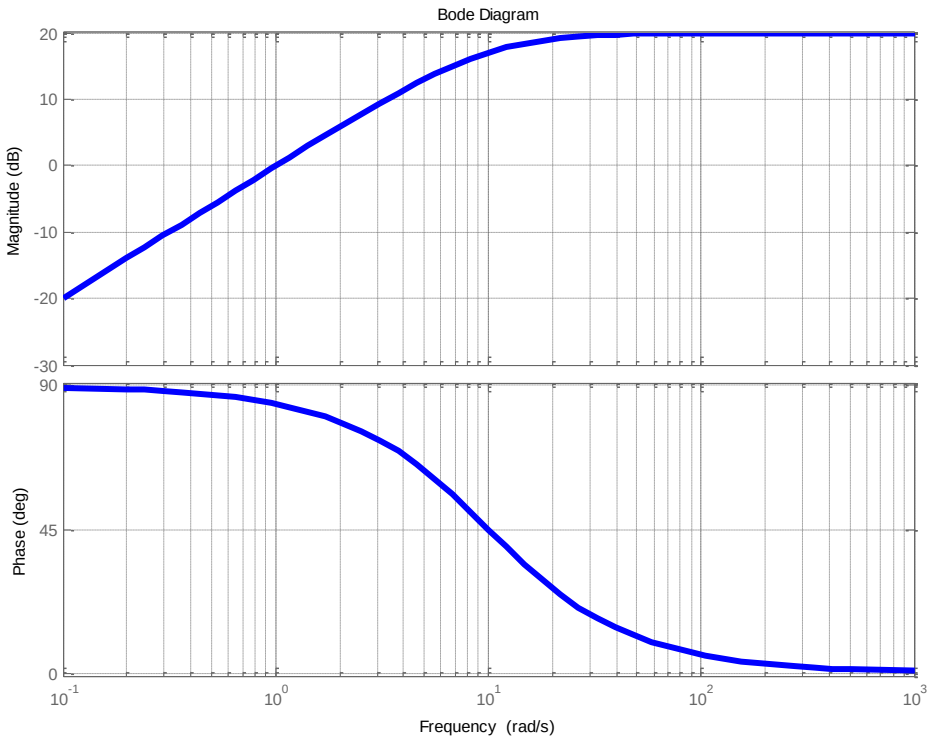
In this case the high-frequency measurement noise will not be indefinitely amplified as in the previous implementation.

To look at the Bode plots of the transfer functions, we can use the following commands on MATLAB:

```
T = tf([1 0],1)
bode(T)
```




```
T = tf([1 0], [0.1 1])
bode(T)
```



As we can see from the plots, after a certain frequency the gain in the approximate derivative case stops increasing. In the first plot we can see the gain increase indefinitely with frequency.

Let us now look at the time response of the system:

The transfer function from v to u looks like:

$$T_{vu}(s) = k_p + \frac{k_d s}{\tau_d s + 1} + \frac{k_i}{s} = \frac{\left(k_p + \frac{k_d}{\tau_d}\right)s^2 + \left(\frac{k_p}{\tau_d} + k_i\right)s + \frac{k_i}{\tau_d}}{s^2 + \frac{1}{\tau_d}s}$$

Writing this in the controllable canonical form, we have:

$$\dot{z} = \begin{bmatrix} -\frac{1}{\tau_d} & 0 \\ 1 & 0 \end{bmatrix} z + \begin{bmatrix} 1 \\ 0 \end{bmatrix} v$$

$$u = \begin{bmatrix} k_i - \frac{k_d}{\tau_d^2} & \frac{k_i}{\tau_d} \end{bmatrix} z + \left[\left(k_p + \frac{k_d}{\tau_d}\right)\right] v$$

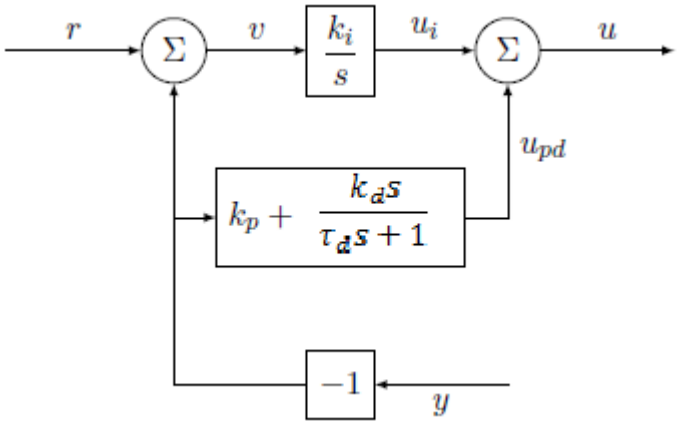
From the above equation we can see that if $z(t)\ and\ v(t) = 0$ for all $t < 0, u(t) = 0$.

At $t = 0$, $v(0) = 1$ and $z(0) = 0$

Therefore, $u(0) = k_p + k_d/\tau_d$

Therefore when there is a step change in the reference signal, the controller output changes suddenly from 0 to $k_p + k_d/\tau_d$. This is the input that will be provided to your dynamical system. It would be wise to protect the system from such sudden jumps in input.

c) Next method of implementing a PID controller:



The transfer function from y to u is:

$$T_{yu}(s) = -\left(k_p + \frac{k_d s}{\tau_d s + 1} + \frac{k_i}{s}\right)$$

After re-arranging the terms, $\left|T_{yu}(j\omega)\right|$ can be written as:

$$\left|T_{yu}(j\omega)\right| = \sqrt{\left(k_p + \frac{k_d \tau_d}{\tau_d^2 + \frac{1}{\omega^2}}\right)^2 + \left(\frac{k_i}{\omega} - \frac{\frac{k_d}{\omega}}{\tau_d^2 + \frac{1}{\omega^2}}\right)^2}$$

As $\omega \to \infty$, $\left|T_{yu}(j\omega)\right| = k_p + k_d/\tau_d$. The behavior is same as in part (b). The high frequency noise will not be indefinitely amplified. This could be classified as good behavior.

Following the recommended steps, let us note down the transfer function from y to u_{pd}:

$$T_{yu_{pd}}(s) = -\left(k_p + \frac{k_d s}{\tau_d s + 1}\right) = \frac{-\left(k_p + \frac{k_d}{\tau_d}\right)s - \frac{k_p}{\tau_d}}{s + 1/\tau_d}$$

Writing this in the controllable canonical form, we get:

$$\dot{z_1} = \left[-\frac{1}{\tau_d}\right] z_1 + y$$

$$u_{pd} = \left[\frac{k_d}{\tau_d^2}\right] z_1 + \left[-\left(k_p + \frac{k_d}{\tau_d}\right)\right] y$$

The transfer function from v to u_i, can be given as:

$$T_{vu_i}(s) = \frac{k_i}{s}$$

In the controllable canonical form we get:

$$\dot{z_2} = [0] z_2 + v$$

$$u_i = [k_i] z_2$$

We can now combine the above two state-space systems. We know that $= u_{pd} + u_i$, $v = r - y$ and that we have two inputs now (r and y).

Therefore:

$$\dot{z} = \begin{bmatrix} -\frac{1}{\tau_d} & 0 \\ 0 & 0 \end{bmatrix} z + \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} y \\ r \end{bmatrix}$$

$$u = \begin{bmatrix} \frac{k_d}{\tau_d^2} & k_i \end{bmatrix} z + \begin{bmatrix} -\left(k_p + \frac{k_d}{\tau_d}\right) & 0 \end{bmatrix} \begin{bmatrix} y \\ r \end{bmatrix}$$

If z, r and y are < 0 for $t < 0$, then $u(t) = 0$ for $t < 0$.

At $t = 0$, when $r(0) = 1$, u remains 0. Thus, there is no sudden jump in the output of the controller. This new PID structure is in some sense 'protecting' your system.

2. Rotational motion of a control moment gyro on a spacecraft in state-space form:

$$\dot{x} = -x + u$$

$$y = x$$

a) The transfer function can be easily computed by either taking the Laplace transform of the above equations or by directly using:

$$H(s) = C(sI - A)^{-1}B + D$$
$$= \frac{1}{s+1}$$

b) $T_{delay}(s) = e^{-s\tau_{delay}}$

$$T_{delay}(j\omega) = e^{-j\omega\tau_{delay}} = \cos(\omega\tau_{delay}) - jsin(\omega\tau_{delay})$$

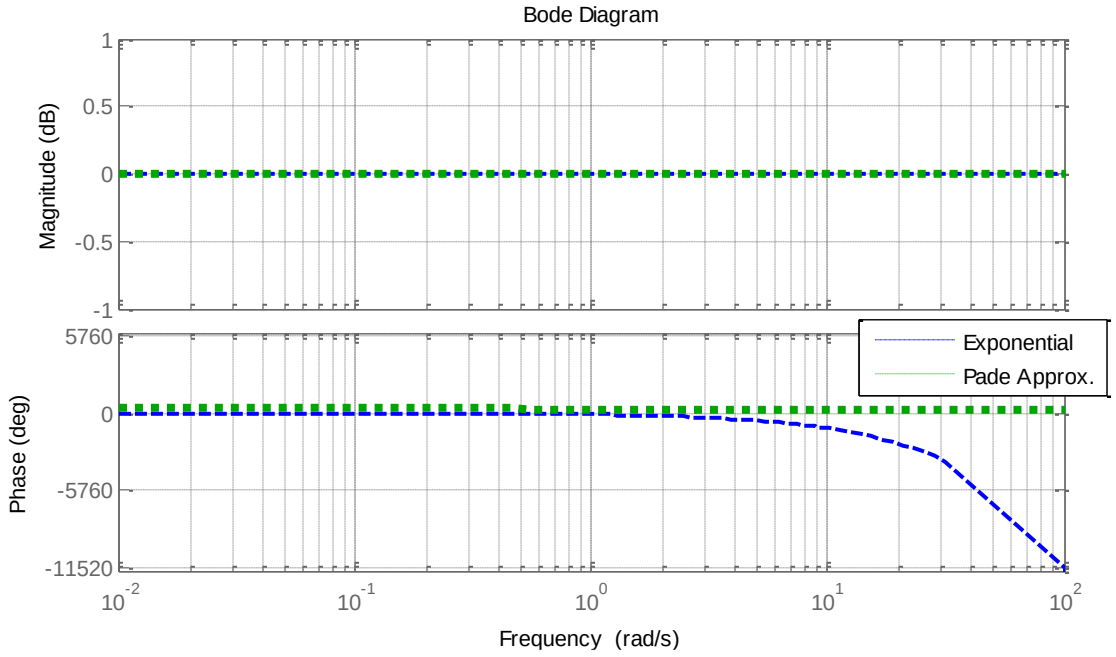
Therefore:

$$\left|T_{delay}(j\omega)\right| = 1$$

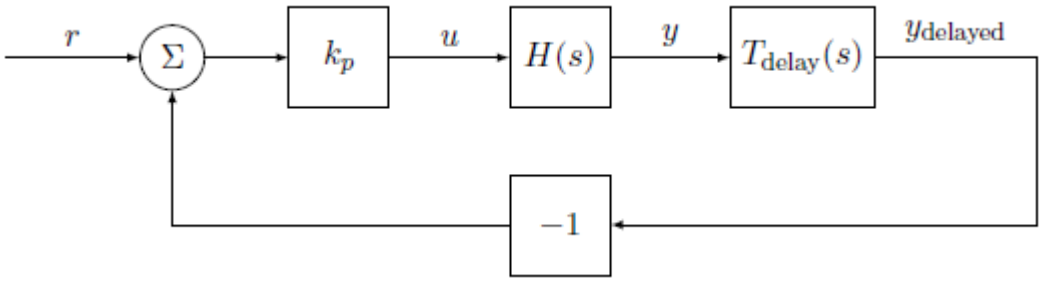
$$\angle T_{delay}(j\omega) = -\omega\tau_{delay}$$

As we can see the time-delay has unit gain and thus just causes a pure phase shift.

c) The bode plots for the exponential function and the Padé approximation look like:



d) Using the following figure:



We get:

$$y = k_p H(s)\left(r - yT_{delay}(s)\right)$$

$$T_{ry} = \frac{k_p H}{1 + k_p H T_{delay}}$$

$$T_{ry}(s) = \frac{s\left(\frac{k_p \tau_{delay}}{2}\right) + k_p}{s^2\left(\frac{\tau_{delay}}{2}\right) + s\left(1 + \frac{\tau_{delay}}{2} - \frac{k_p \tau_{delay}}{2}\right) + (1 + k_p)}$$

e) Given $k_p = 4$, the denominator looks like:

$$s^2\left(\frac{\tau_{delay}}{2}\right) + s\left(1 - \frac{3\tau_{delay}}{2}\right) + 5$$

The above equation has negative roots for $\tau_{delay} < 0.667$, purely imaginary roots for $\tau_{delay} = 0.667$ and positive roots for $\tau_{delay} > 0.667$.

Thus, the maximum τ_{delay} for which the system is (marginally) stable is $\tau_{delay} = 0.667$ seconds.

f) Loop transfer function:

$$L(s) = k_p H(s) T_{delay}(s)$$

In the absence of time delay:

$$L(j\omega) = k_p H(j\omega) = \frac{k_p}{1 + j\omega}$$

$$|L(j\omega)| = \frac{k_p\sqrt{1 + \omega^2}}{1 + \omega^2}$$

On solving for the crossover frequency, we get:

$$\omega_c = \sqrt{k_p^2 - 1} = \sqrt{15} = 3.873\ rad/s$$

$$L(j\omega_c) = \frac{1}{k_p}(1 - j\omega_c) = 0.25 - 0.96825j$$

$$\angle L(j\omega_c) = \tan^{-1}(-\omega_c) = -1.318\, rad = -75.52^o$$

Therefore phase difference at this crossover frequency = $-75.52^o + 180^o = 104.48^o$

The phase shift that would be applied to the loop transfer function by a delay of $\tau_{delay} = 0.667$:

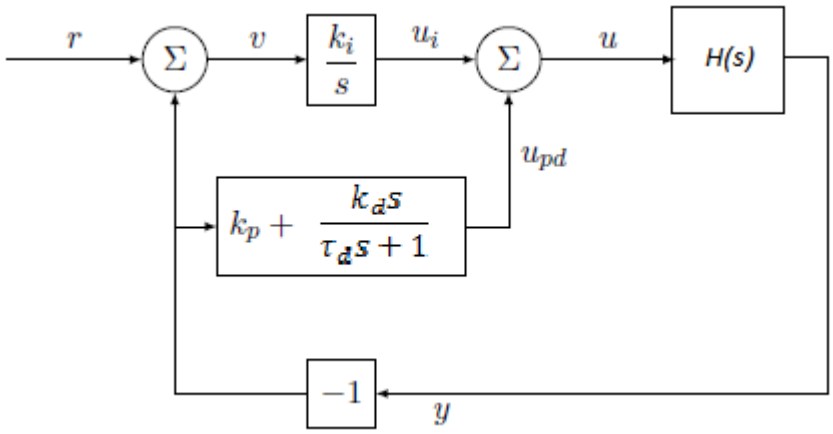
$$\angle T_{delay}(j\omega_c) = -\omega_c \tau_{delay} = -2.5833\, rad = -148.01^o$$

3. State space form for robot arm:

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -1/5 \end{bmatrix} x \begin{bmatrix} 0 \\ 1/5 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} x$$

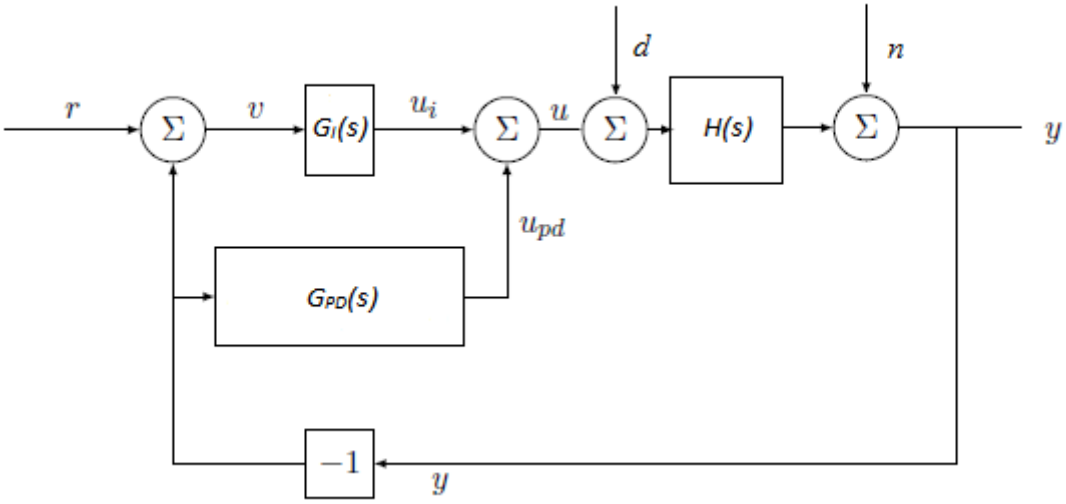
a) The block diagram for this problem would look like:



Where: $H(s) = C(sI - A)^{-1}B + D$

$$H(s) = \frac{1}{5s^2 + s}$$

To obtain the closed loop relation from r to y, let's represent the above figure (including noise and disturbance) as:



Therefore,

$$y = n + Hu$$
$$y = n + H(u_i + u_{pd} + d)$$
$$y = n + H(G_I(r - y) - G_{PD}y + d)$$

On rearranging, we get:

$$y = \frac{HG_I}{1 + HG_I + HG_{PD}}r + \frac{1}{1 + HG_I + HG_{PD}}n + \frac{H}{1 + HG_I + HG_{PD}}d$$

The first 2 design specifications are related to the transfer function T_{ry}. The next one is related to T_{dy} and the last one to T_{ny}.

There are many ways to approach the design process. One of the methods could be to start off with a pole-placement technique:

- Decide a set of poles (number of poles depends on degree of polynomial in denominator) that you think might satisfy the design constraints.
- Compare the coefficients and get values for k_p, k_d, k_i and τ_d.
- Trial and error till you get what you want.

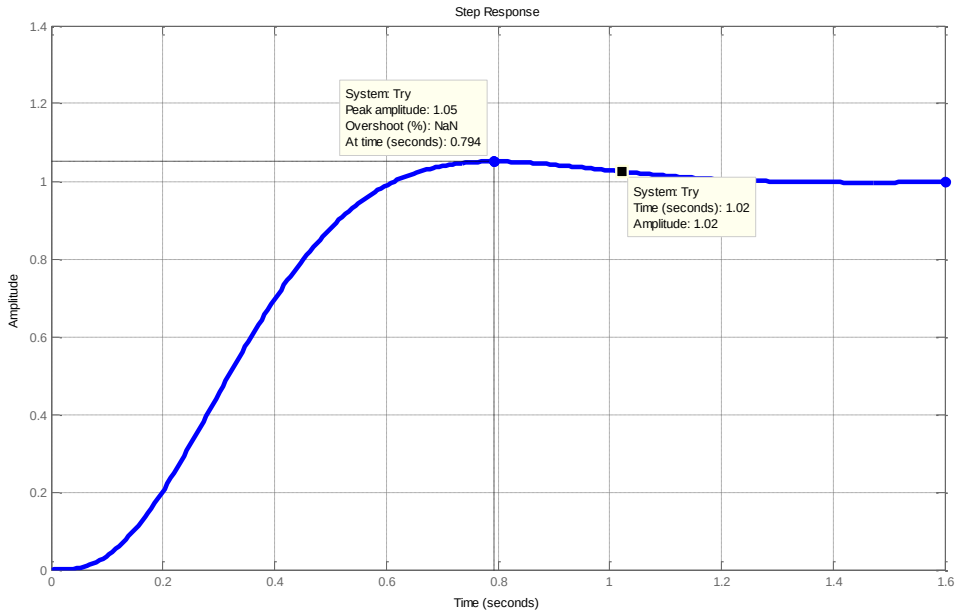
The closed loop function T_{ry} looks like:

$$T_{ry}(s) = \frac{\frac{k_i}{5}s + \frac{k_i}{5\tau_d}}{s^4 + \left(\frac{1}{5} + \frac{1}{\tau_d}\right)s^3 + \left(\frac{k_d + \tau_d k_p + 1}{5\tau_d}\right)s^2 + \left(\frac{k_p + \tau_d k_i}{5\tau_d}\right)s + \frac{k_i}{5\tau_d}}$$

After comparing the coefficients to the desired denominator polynomial and trial error, I end up with the following gains:

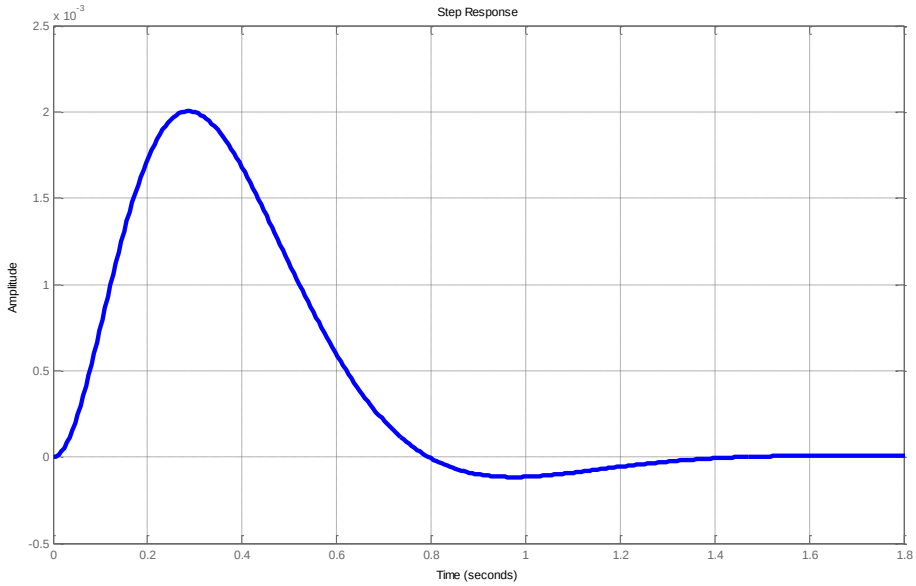
$$k_p = 400, k_d = 60, k_i = 1300, \tau_d = 0.025$$

We get the following closed-loop step response:

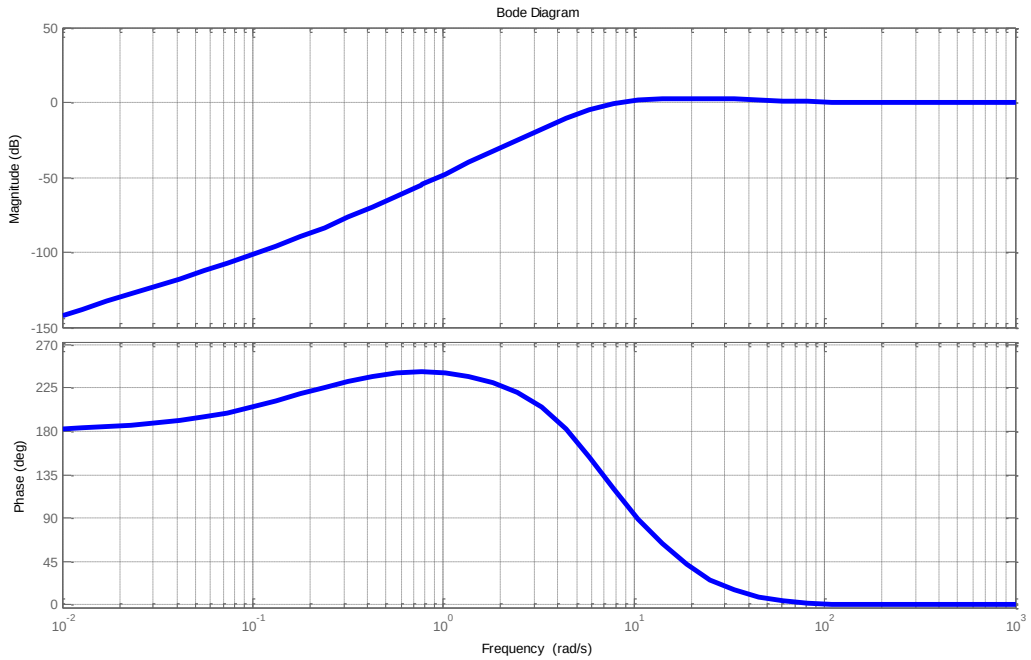


With a 2% Settling time almost equal to 1 second and a peak amplitude of less than 15%.

For a disturbance input, there is no steady state error:



We can look at the Bode plot to check the gain:

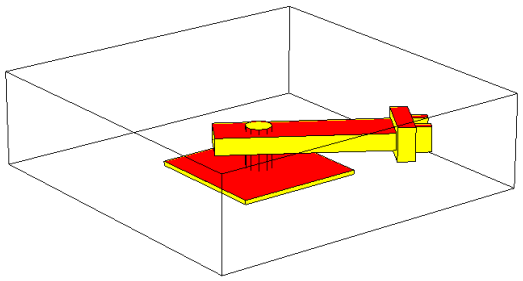


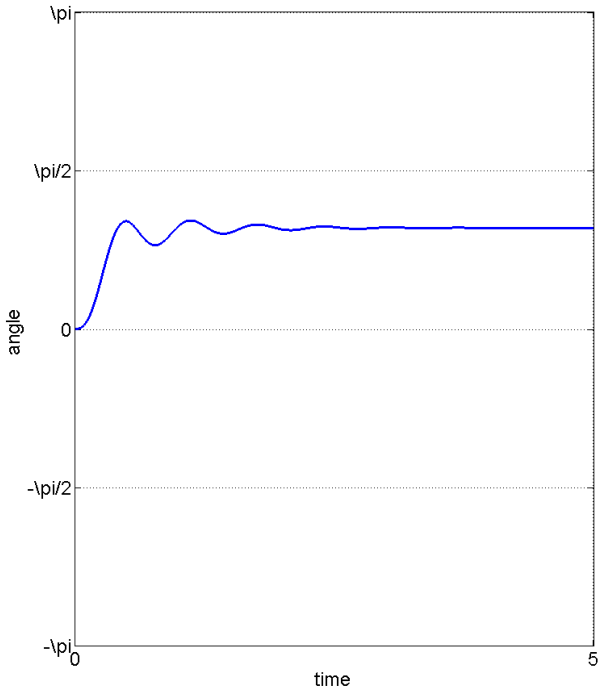
The gain is always lower than 10.

b) To implement the controller, we can edit the ControlLoop function to the following:

```
function [u,userdata] = ControlLoop(t,y,r,userdata,params)
%
% INPUTS
%
%  t is the current time (1x1)
%
%  y is the angle measurement (1x1)
%
%  r is the reference signal (1x1)
%
%  userdata has whatever you put there
%
%  params has a number of important parameters (see above)
%
% OUTPUTS
%
%  u is the applied torque
%
%  userdata has whatever you put there
%
% RULES
%
%  - the code you submit must display nothing in the MATLAB console
%  - the code you submit must not take excessively long to compute
%  - the code you submit must not declare or use global variables
```

```
%
persistent isFirstTime
if isempty(isFirstTime)
    isFirstTime = false;
    fprintf(1,'initialize control loop\n');
    userdata.z = [0;0];
end

% v = r - y;
% zdot = [-1 -1/params.td  0 0;...
%          1   0           0 0;...
%          0   0          -1 0;...
%          0   0           1 0]*userdata.z + [1;0;0;0]*y + [0;0;1;0]*v;

z_dot = [-1/params.td 0; 0 0]*userdata.z + [1 0; -1 1]*[y; r];

userdata.z = userdata.z + params.dt*z_dot;

u = [params.kd/params.td^2 params.ki]*userdata.z - (params.kd/params.td +
params.kp)*y;
```





The gains used were $k_p = 400$ $k_i = 1300$ $k_d = 60$ with a time delay of $\tau = 0.025$ s.