CH E 152B Homework 2

Connor Hughes

April 11, 2021

Exercise 4: Level control simulation

We shall simulate the performance of the controllers from Exercise 1.

Part (a)

Note that for piecewise constant input over the sample time, the solution to the differential equation model in Exercise 1 is

$$y(k+1) = y(k) + \Delta(d(k) - u(k))$$

where $t = k\Delta$, $y(0) = 0$, and Δ is the sample time.

Part (b)

Now let's introduce measurement error; if we measure $d(k)$, say that the measured value is

$$d_m(k) = d(k) + v(k)$$

where $v(k)$ is a normally distributed random variable with zero mean and variance $(0.05)^2$. Note that in `MATLAB`, the command `randn` gives a normally distributed (pseudo) random variable with zero mean and unit variance, so `0.05*randn` gives a random variable with zero mean and variance $(0.05)^2$. So your measurement model is in `MATLAB`

```
d_m(k) = d(k) + 0.05*randn
```

Simulate your *feedforward* controller of Exercise 1 for $t = 600$ min with a sample time of $\Delta = 1$ min with a step change of magnitude 0.5 in $d(k)$ at time $t = 300$ min. Let's say the tank is scaled so that $y(k) = 1$ is full, $y(k) = -1$ is empty, and the setpoint $y(k) = 0$ is half full.

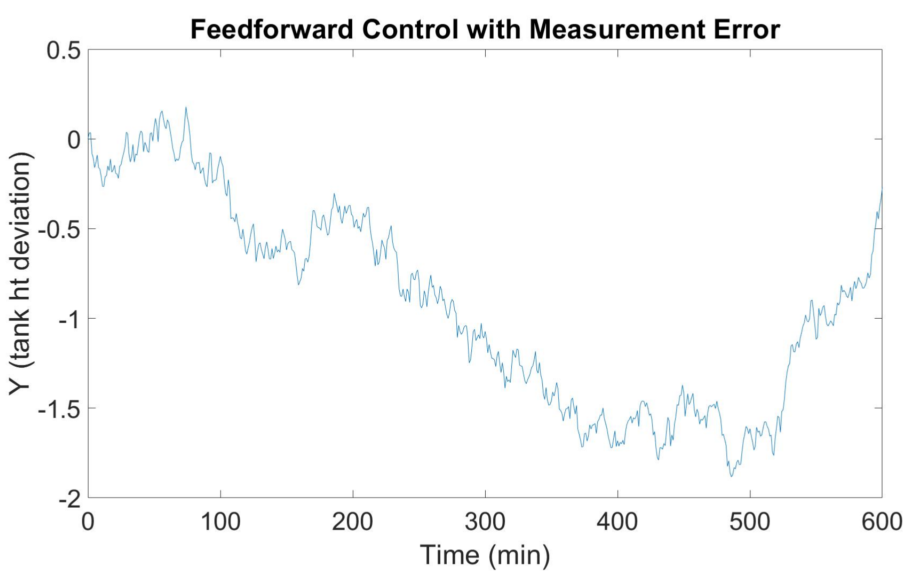
Run several simulations of the feedforward controller for different realizations of the measurement noise. Does your tank ever overflow using feedforward control? Does the tank ever empty and cavitate the pump using feedforward control?

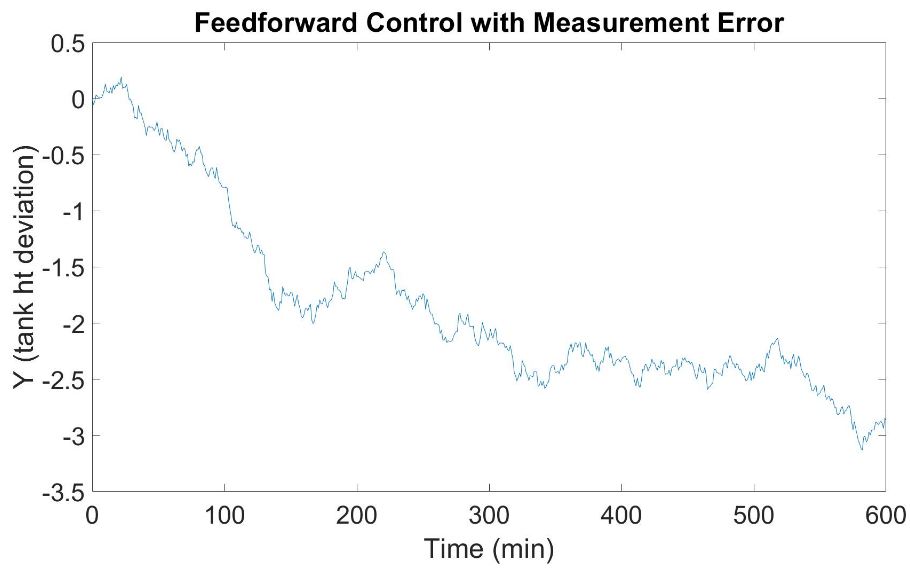
Solution:

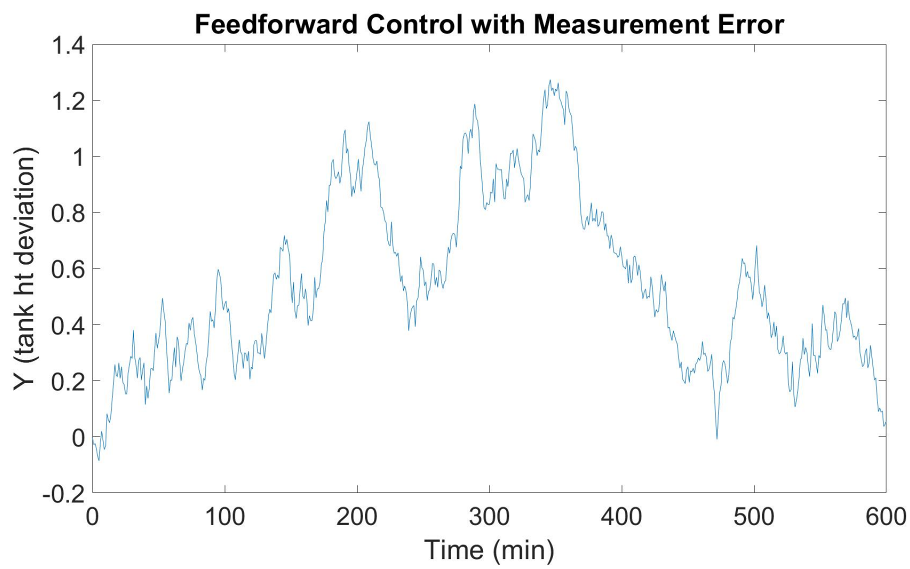
The `MATLAB` code on the following page was used to simulate the system with feedforward control.

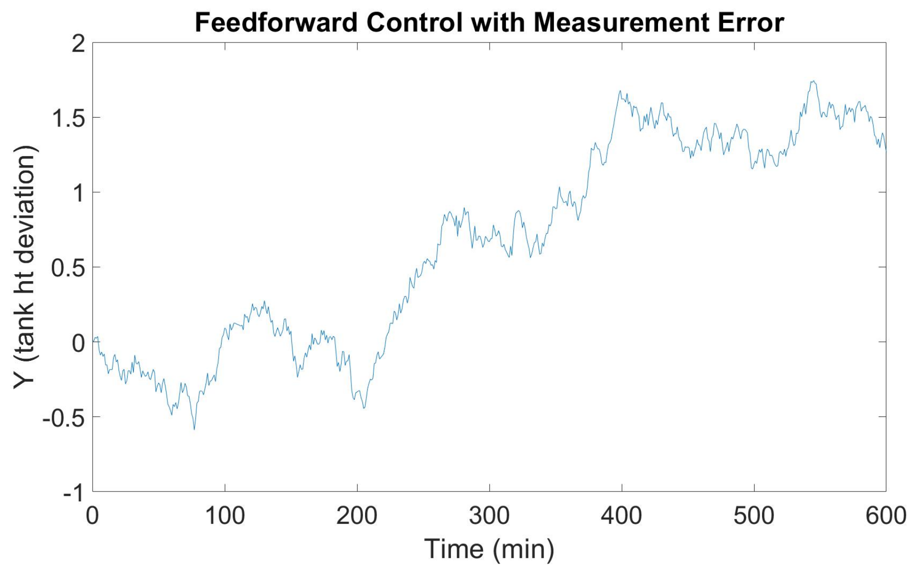
```matlab
%initialize parameters:
del = 1;   %sample time = 1 minute
tspan = [0:del:600];   %time span covers 600 min

%set initial condition & set point:
y_0 = 0;
y_t = 0;

%initialize input vector:
d_1 = zeros(1, 300) + 0.5;
d_2 = zeros(1, 301) + 1;
d = cat(2, d_1, d_2);

%add measurement error:
dm = d + 0.05*randn(1, length(tspan));

%generate feedforward control input using control law  u(t) = dm(t):
u = dm;

%simulate system:
y = zeros(1, length(tspan));
y(1) = y_0;
for i = 1:(length(tspan) - 1)
    y(i + 1) = fdfwd_tank_lvl(y(i), d(i), u(i), del);
end

%plot tank level deviation vs time:
plot(tspan, y)
xlabel("Time (min)", 'FontSize', 32)
ylabel("Y (tank ht deviation)", 'FontSize', 32)
title("Feedforward Control with Measurement Error", 'FontSize', 36)
ax = gca;
ax.FontSize = 28;

function y_kp1 = fdfwd_tank_lvl(y_k, d_k, u_k, del)
    %compute tank level deviation at next time step:
    y_kp1 = y_k + del*(d_k - u_k);
end
```

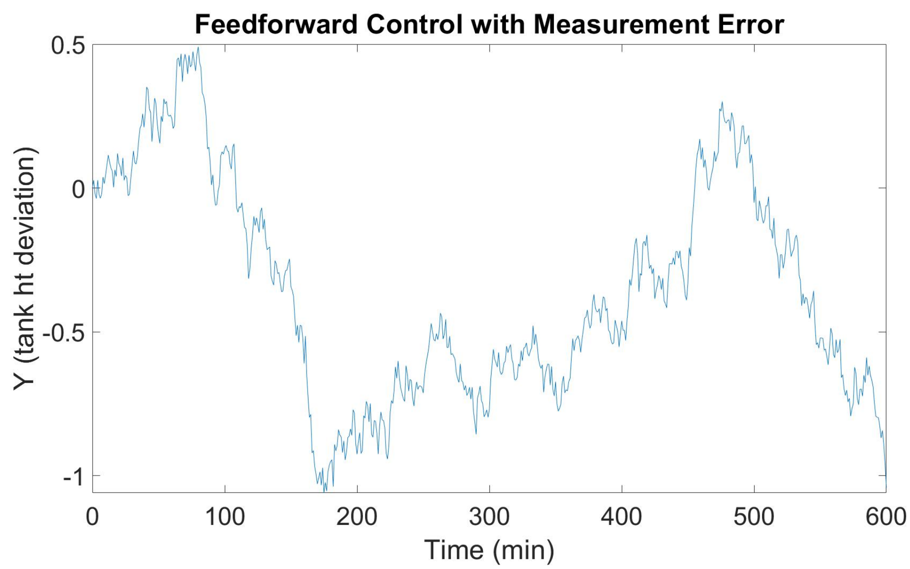
The following 5 plots show the tank height deviation for different realizations of the measurement noise:




Feedforward Control with Measurement Error
Y (tank ht deviation)
Time (min)
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
0
100
200
300
400
500
600


Feedforward Control with Measurement Error
Y (tank ht deviation)
Time (min)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
100
200
300
400
500
600





As we can see from the plots above, the feedforward controller is ineffective at maintaining the tank level at the set point in the presence of zero-mean normally distributed measurement noise with a standard deviation of 0.05. In the first two plots and the fifth plot, Y reaches a value below -1, meaning the tank empties and the pump would cavitate. In the third and fourth plots, Y reaches a value greater than 1, indicating that the tank overflows.

Part (c)

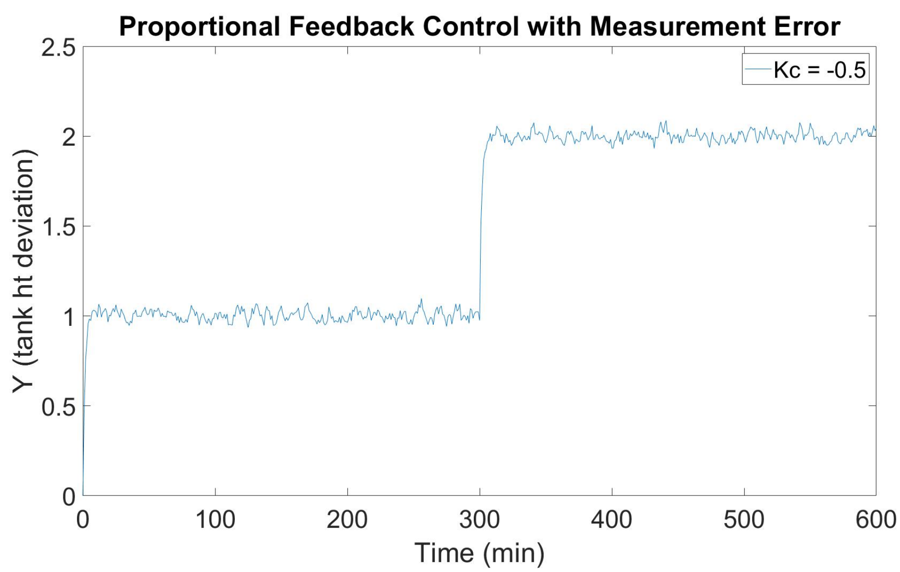
Now simulate the response of a *feedback* controller under proportional control for the same disturbance using gain $K_c = -0.5$. Assume that when you measure $y(k)$, you have zero mean normally distributed measurement noise of the same variance as $d(k)$ from the last part.

Solution:

The following `MATLAB` code was used to simulate the system with proportional feedback control.

```matlab
%% Problem 4c,d, and e: Feedback Control (proportional only)
%initialize proportional gain:
k_c = -0.5;

%simulate system:
y = zeros(1, length(tspan));
y(1) = y_0;
for i = 1:(length(tspan) - 1)
    y(i + 1) = fdbck_P_tank_lvl(y(i), y_t, d(i), k_c, del);
end

%plot tank level deviation vs time:
plot(tspan, y)
xlabel("Time (min)")
ylabel("Y (tank ht deviation)")
title("Proportional Feedback Control with Measurement Error")
legend("Kc = -0.5")
%legend("Kc = -1")
%legend("Kc = -2")

function y_kp1 = fdbck_P_tank_lvl(y_k, y_t, d_k, k_c, del)
    %generate measured tank level deviation (with meas. error):
    y_m = y_k + 0.05*randn(1);
    %generate feedback control based on measured tank level deviation:
    u_k = k_c*(y_t - y_m);
    %compute tank level deviation at next time step:
    y_kp1 = y_k + del*(d_k - u_k);
end
```

The corresponding tank height deviation is shown in the following plot:



As we can see in the plot above, the proportional feedback control with $k_c = -0.5$ is also ineffective at maintaining the tank level at the set point. However, in this case it is not the measurement noise which prevents this. Rather, the steady state error is much too large with the specified gain. For an input deviation of 0.5 (seen from 1-300 min), the steady state error of the controller is equal to 1. With the measurement noise included, we can see that this causes the tank height deviation to exceed 1 at numerous points, indicating an overflow. For an input deviation of 1 (seen from 300-600 min), the steady state error of the controller is equal to 2. By examining the closed loop system solution,

$$y(k+1) = y(k) + \Delta(d(k) - k_c * (y_t - y_m(k)))$$

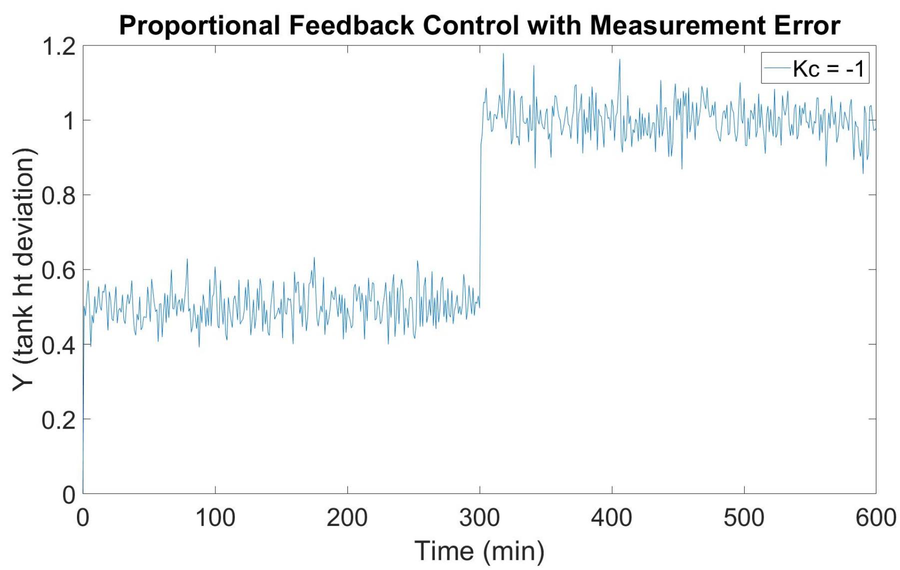
we can confirm that the steady state error is the issue. For $d(k) = 0.5$, $k_c = -0.5$, $y_t = 0$ and $y_m(k) > 1$, we know $(d(k) - k_c(y_t - y_m(k))$ is negative, so y is decreasing. However, for $y_m(k) < 1$, we see that $(d(k) - k_c(y_t - y_m(k))$ is positive, so y is increasing. Thus, with the specified gain, the lowest value at which the proportional feedback controller can maintain the tank height deviation is 1. Similarly, for $d(k) = 0.5$, the best our controller can do is to maintain the tank height deviation at 2.

Part (d)

In order to reduce the offset, let's turn up the gain. Use $k_c = -1$. What is the steady state offset?

Solution:

The same `MATLAB` code shown for part (c) was used again to simulate the system for this problem, after the gain was changed to -1. The following plot shows the tank height deviation with the proportional feedback controller with the new gain:



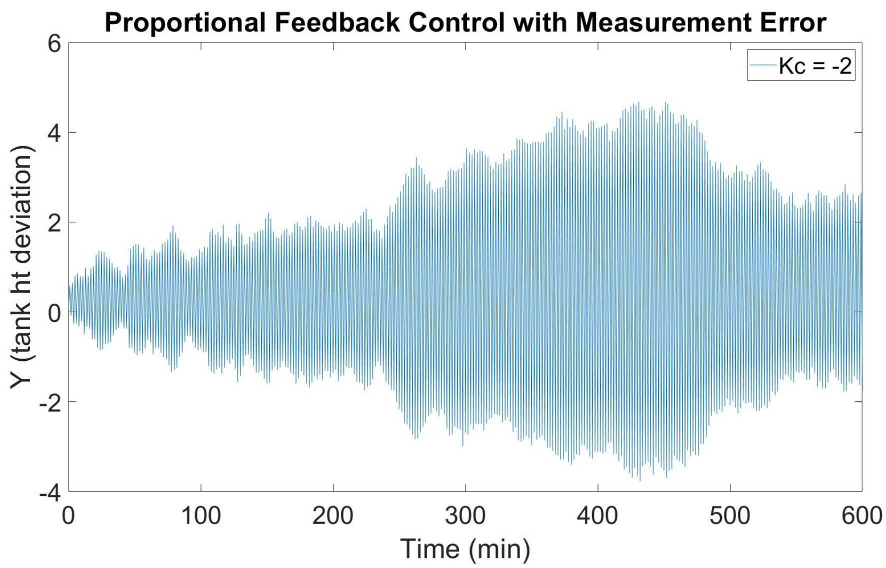
Notice in the above plot that the increased gain has increased the amplitude of the oscillations induced by measurement noise, which is to be expected as the tank height deviation measurement is used as an input to the feedback controller. However, notice also that this increased gain has reduced the steady state error by a factor of 2.

Part (e)

Use $k_c = -2$. What happens to the closed-loop system for this value of gain?

Solution:

Once again, the same `MATLAB` code shown for part (c) was used again to simulate the system for this problem, after the gain was changed to -2. The following plot shows the tank height deviation with the proportional feedback controller with the new gain:



From the above plot, we can see that this gain is sufficiently high that the closed-loop system has become far less stable, and is being driven into very large, high-frequency oscillations as a result of the measurement noise itself.

Part (f)

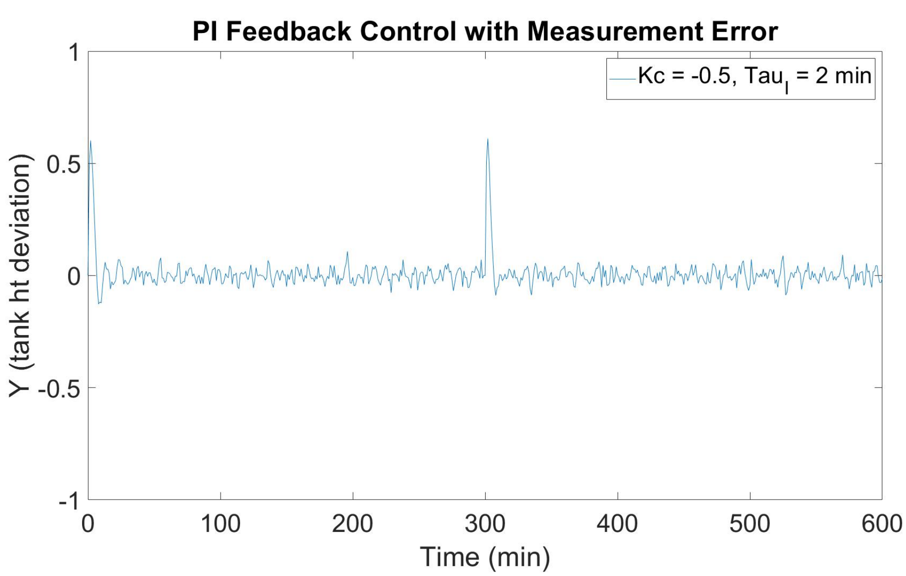
Now use a PI controller with $k_c = -0.5$ and $\tau_I = 2\Delta$. Do you have steady-state offset? Play around a bit with the simulation and decide what you think are the best PI tuning parameter values of k_c and τ_I for this process with this amount of measurement noise.

Solution:

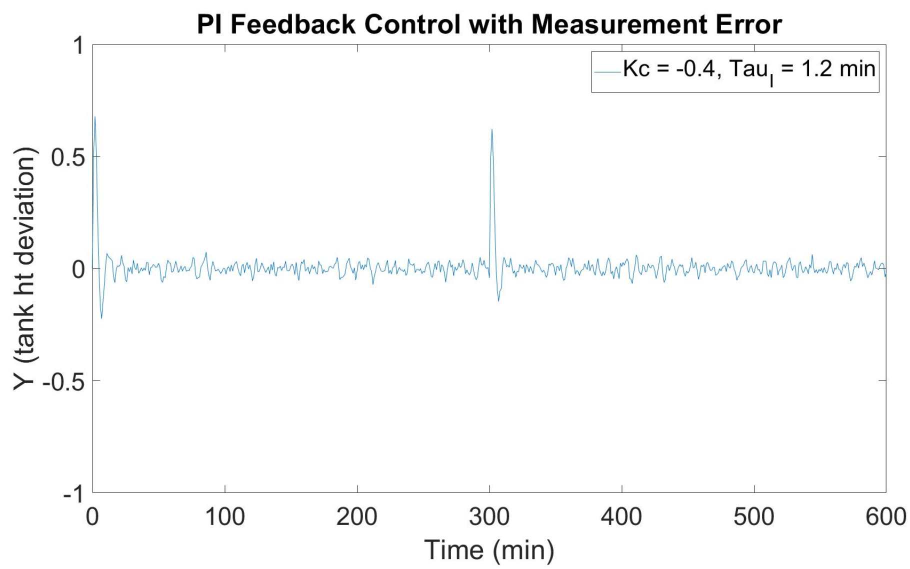
The `MATLAB` code on the following page was used to simulate the system with PI feedback control.

```
%% Problem 4f: Feedback Control (PI)
k_c = -0.4;
tau_I = 1.2*del;
int_term = zeros(1, length(tspan));

%simulate system:
y = zeros(1, length(tspan));
y(1) = y_0;
int_term(1) = y_t - y_0;
for i = 1:(length(tspan) - 1)
    y(i + 1) = fdbck_PI_tank_lvl(y(i), y_t, d(i), k_c, tau_I, int_term(i), del);
    int_term(i + 1) = int_term(i) + del*(y_t - y(i + 1));
end

%plot tank level deviation vs time:
plot(tspan, y)
xlabel("Time (min)", 'FontSize', 32)
ylabel("Y (tank ht deviation)", 'FontSize', 32)
title("PI Feedback Control with Measurement Error", 'FontSize', 36)
legend("Kc = " + k_c + ", Tau_I = " + tau_I/del + " min", 'FontSize', 26)
axis([0 600 -1 1])
ax = gca
ax.FontSize = 28;

function y_kp1 = fdbck_PI_tank_lvl(y_k, y_t, d_k, k_c, tau_I, I, del)
    %generate measured tank level deviation (with meas. error):
    y_m = y_k + 0.05*randn(1);
    %generate feedback control based on measured tank level deviation:
    u_k = k_c*(y_t - y_m) + (k_c/tau_I)*I;
    %compute tank level deviation at next time step:
    y_kp1 = y_k + del*(d_k - u_k);
end
```

The following plot shows the tank height deviation with the PI feedback controller:



In the above plot, we can see that the PI feedback controller with the specified gains is able to maintain the tank height deviation close to the set point of zero. When the step increases in the input occur at 1 min and at 300 min, the tank height deviation spikes, but the controller is quickly able to drive it back to within a small region around zero. At steady state, the tank height deviation does experience high-frequency, small amplitude oscillations caused by the controller responding to the measurement noise. Ideally, we would reduce or eliminate these oscillations, as the response to these oscillations might require the control valve to constantly make small, rapid changes, which will likely lead to rapid wear. This issue is perhaps best alleviated by using a low pass filter on the input to the controller, but we can also minimize the amplitude of the high-frequency, steady state oscillations by reducing the proportional gain k_c. However, reducing this gain also has the effect of increasing the amplitude of the spikes which occur at the step changes in input flow, which presents a tradeoff. Our central goal with this system is to ensure that the tank does not empty or overflow, while providing a fairly steady output stream even in the presence of significant and rapid disturbances to the input stream. It seems that a small reduction in the proportional gain k_c, combined with a reduction in τ_I (which *increases* the effect of the integral term) allows us to strike a good balance between our two objectives. The plot below was generated using $k_c = -0.4$ and $\tau_I = 1.2$:



In this plot, we see a small reduction in the amplitude of the steady state oscillations, while maintaining approximately the same amplitude of the spikes. This comes at the cost of introducing an additional (negative) spike during the period in which the tank is settling back to its set point after a step change in input, but these additional oscillations are far smaller than the initial spike, and do not cause us any concern from an inventory control perspective as they do not bring the tank close to empty.

Exercise 5: Transfer Function

Give the closed-loop transfer function between setpoint T_{2sp} and T_2 for the cascade/feedforward controller of Excercise 2.

Solution:

The closed-loop transfer function between setpoint T_{2sp} and T_2 for the cascade/feedforward controller can be derived from the block diagram shown for Exercise 2, part (c) on the previous homework. We can simplify the task for ourselves by first deriving the closed-loop transfer function from T_{1sp} to T_1, which contains the inner feedback loop as well as the feedforward control. We will then refer to this closed-loop transfer function as $G(s)$, and use this shorthand when expressing the overall closed-loop transfer function. By inspection of the block diagram for Exercise 2, part (c), we have:

$$\frac{T_1(s)}{T_{1sp}(s)} = G(s) = \frac{g_2}{1 + g_2 g_3 g_{C2}} T_0 + \frac{g_2 g_3 g_{C3}}{1 + g_2 g_3 g_{C2}} T_{0m} + \frac{g_2 g_3 g_{C2}}{1 + g_2 g_3 g_{C2}} T_{1sp}$$

Now, we can consider the rest of the block diagram, and we have:

$$\frac{T_2(s)}{T_{2sp}(s)} = \frac{g_1}{1 + g_1 G g_{C1}} Q_2 + \frac{g_1 G g_{C1}}{1 + g_1 G g_{C1}} T_{2sp}$$