**(1) Simulate the adaptive control system when and**

% ------------------------------------------------------------------------

**%% setup**

**% true parameter value**

**m = 2;**

**% estimator parameters > 0**

**gamma = 1.25;**

**% controller parameters > 0**

**lambda = 4;**

**% reference model parameters > 0**

**RT = 1/3; % rise time**

**POS = 0.01; % percent overshoot**

**zeta = -log(POS)/sqrt(pi^2 + log(POS)^2); % damping ratio**

**lambda\_2 = round((1.8/RT)^2,1); % "stiffness"**

**lambda\_1 = round(2\*sqrt(lambda\_2)\*zeta,1); % "damping"**

**% simulation time-step**

**dt = 1/60;**

**%% initial conditions**

% time

t(1) = 0;

% state

**x(1) = 0.5;**

x\_dot(1) = 0;

x\_dotdot(1) = 0;

% desired trajectory

r(1) = 0

x\_m\_dotdot(1) = 0

x\_m\_dot(1) = 0;

**x\_m(1) = 0.5;**

% error

x\_tilda(1) = 0;

x\_tilda\_dot(1) = 0;

x\_tilda\_dotdot(1) = 0;

% control input

s(1) = 0;

v(1) = 0;

u(1) = 0;

% parameter estimate

m\_hat\_dot(1) = 1e-6;

m\_hat(1) = m\_hat\_dot(1)\*dt;

% ------------------------------------------------------------------------

**%% time-loop**

for i = 2:(3/dt)

% time

t(i) = t(i-1) + dt;

**% state**

**x\_dotdot(i) = u(i-1)/m;**

x\_dot(i) = x\_dot(i-1) + x\_dotdot(i)\*dt;

x(i) = x(i-1) + x\_dot(i)\*dt;

**% desired trajectory**

**r(i) = 0;**

**x\_m\_dotdot(i) = lambda\_2\*r(i) - lambda\_1\*x\_m\_dot(i-1) - lambda\_2\*x\_m(i-1);**

x\_m\_dot(i) = x\_m\_dot(i-1) + x\_m\_dotdot(i)\*dt;

x\_m(i) = x\_m(i-1) + x\_m\_dot(i)\*dt;

% error

x\_tilda(i) = x(i) - x\_m(i);

x\_tilda\_dot(i) = (x\_tilda(i) - x\_tilda(i-1))/dt;

**% control input**

**s(i) = x\_tilda\_dot(i) + lambda\*x\_tilda(i);**

**v(i) = x\_m\_dotdot(i) - 2\*lambda\*x\_tilda\_dot(i) - lambda^2\*x\_tilda(i);**

**u(i) = m\_hat(i-1)\*v(i);**

**% parameter estimate**

**m\_hat\_dot(i) = -gamma\*v(i)\*s(i);**

**m\_hat(i) = m\_hat(i-1) + m\_hat\_dot(i)\*dt;**

end

% -------------------------------------------------------------------------

**%% Display**

figure(1)

set(gcf,'Units', 'normalized', 'Position',[0.2 0.2 0.7 0.5]); % large

subplot(1,2,2);

hold on

plot(t, m\_hat,'r', 'LineWidth',2);

plot(t, m\*ones(length(t)),'--k', 'LineWidth',2);

xlabel('time (s)')

ylabel('m-hat (kg)')

title('Parameter Estimate')

legend('m-hat','m')

grid on

subplot(1,2,1);

hold on

plot(t, x, 'b', 'LineWidth',2);

plot(t, x\_m,'--k', 'LineWidth',2);

xlabel('time (s)')

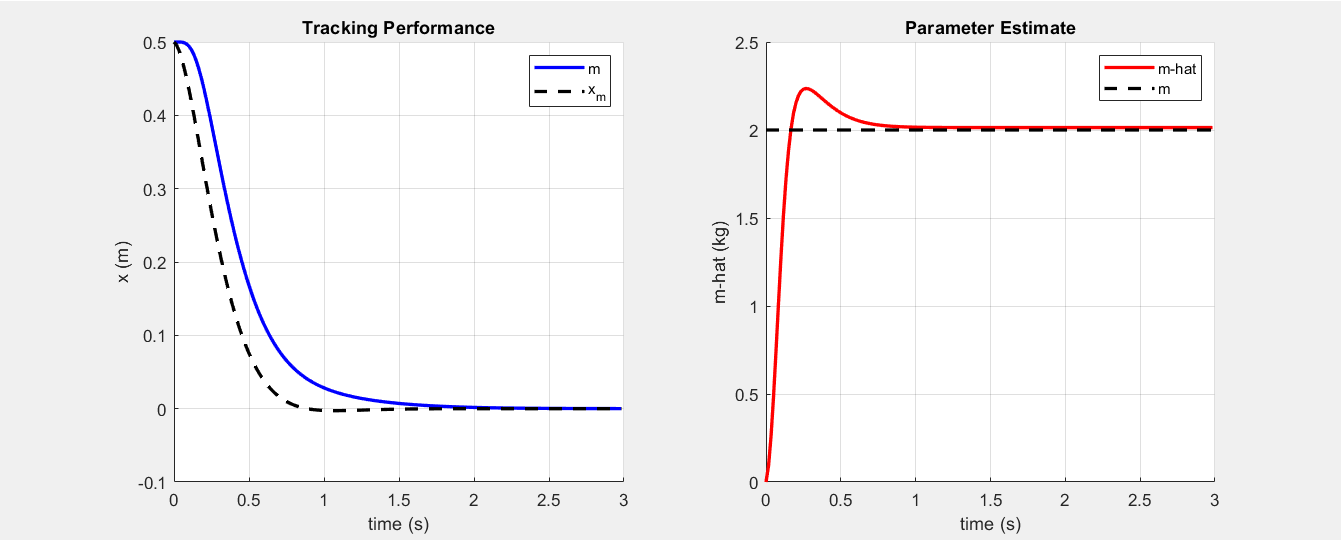
ylabel('x (m)')

title('Tracking Performance')

legend('m','x\_m')

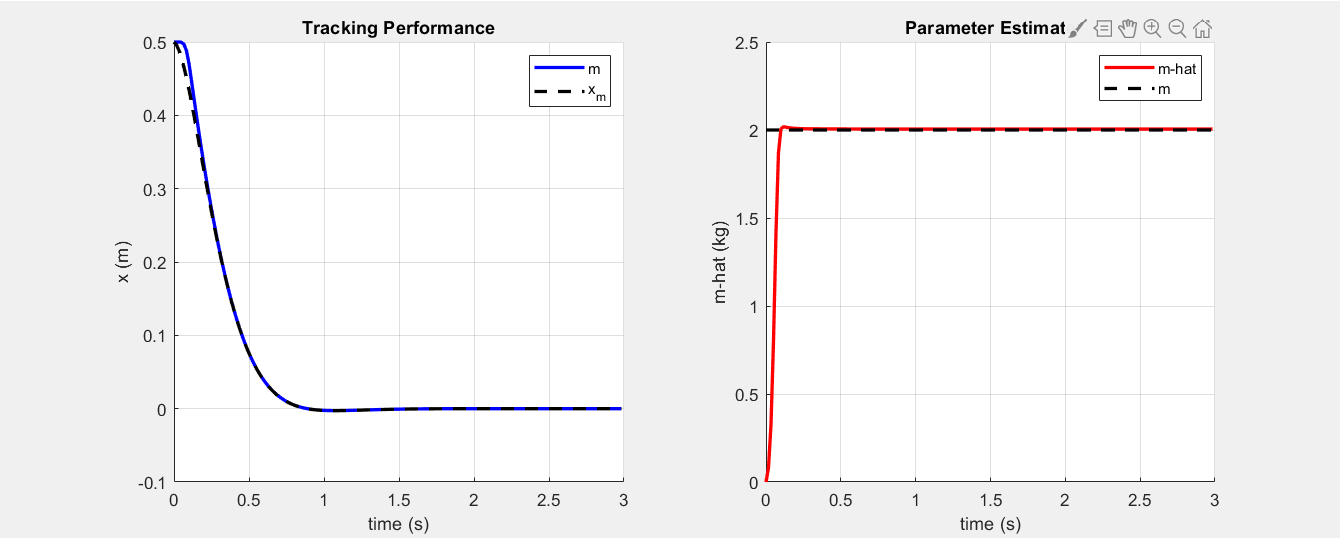
grid on

%% ------------------------------------------------------------------------



The MRAC parameter value used to create this plot were , . These values resulted in convergence of the mass estimate to be approximately the same as truth (2 kg), however, other values result in different settling points. If gamma or lambda is less than 1.25 then the mass estimate converges to some value less than 2 kg, and if gamma or lambda is greater than 1.25 then the mass estimate converges to some value greater than 2 kg.

It is possible that a different combination with a lower gamma and greater lambda or vise-versa could also result in the mass estimate converging to 2 kg. In fact, , results in …



But for the purposes of actual implementation where the true mass value is not known, tuning these controller parameters to get convergence to the true mass value would be impossible, and even if it was possible, it would not help in the case when the mass changes or when a different input signal is used, because the results will vary depending on the input and actual mass.

**(2) Simulate the adaptive control system when and**

% ------------------------------------------------------------------------

**%% setup**

**% true parameter value**

**m = 2;**

**% estimator parameters > 0**

**gamma = 2;**

**% controller parameters > 0**

**lambda = 4;**

**% reference model parameters > 0**

**RT = 1/4; % rise time**

**POS = 0.01; % percent overshoot**

**zeta = -log(POS)/sqrt(pi^2 + log(POS)^2); % damping ratio**

**lambda\_2 = round((1.8/RT)^2,1); % "stiffness"**

**lambda\_1 = round(2\*sqrt(lambda\_2)\*zeta,1); % "damping"**

**% simulation time-step**

**dt = 1/60;**

**%% initial conditions**

% time

t(1) = 0;

% state

**x(1) = 0;**

x\_dot(1) = 0;

x\_dotdot(1) = 0;

% desired trajectory

r(1) = 0

x\_m\_dotdot(1) = 0

x\_m\_dot(1) = 0;

**x\_m(1) = 0;**

% error

x\_tilda(1) = 0;

x\_tilda\_dot(1) = 0;

x\_tilda\_dotdot(1) = 0;

% control input

s(1) = 0;

v(1) = 0;

u(1) = 0;

% parameter estimate

m\_hat\_dot(1) = 1e-6;

m\_hat(1) = m\_hat\_dot(1)\*dt;

% ------------------------------------------------------------------------

**%% time-loop**

for i = 2:(20/dt)

% time

t(i) = t(i-1) + dt;

**% state**

**x\_dotdot(i) = u(i-1)/m(i-1);**

x\_dot(i) = x\_dot(i-1) + x\_dotdot(i)\*dt;

x(i) = x(i-1) + x\_dot(i)\*dt;

**% desired trajectory**

**noise(i) = rand(1)\*0.05\*(-1)^(i);**

**r(i) = sin(4\*t(i)) + noise(i);**

**x\_m\_dotdot(i) = lambda\_2\*r(i) - lambda\_1\*x\_m\_dot(i-1) - lambda\_2\*x\_m(i-1);**

x\_m\_dot(i) = x\_m\_dot(i-1) + x\_m\_dotdot(i)\*dt;

x\_m(i) = x\_m(i-1) + x\_m\_dot(i)\*dt;

% error

x\_tilda(i) = x(i) - x\_m(i);

x\_tilda\_dot(i) = (x\_tilda(i) - x\_tilda(i-1))/dt;

**% control input**

**s(i) = x\_tilda\_dot(i) + lambda\*x\_tilda(i);**

**v(i) = x\_m\_dotdot(i) - 2\*lambda\*x\_tilda\_dot(i) - lambda^2\*x\_tilda(i);**

**u(i) = m\_hat(i-1)\*v(i);**

**% parameter estimate**

**m\_hat\_dot(i) = -gamma\*v(i)\*s(i);**

**m\_hat(i) = m\_hat(i-1) + m\_hat\_dot(i)\*dt;**

**% change m**

**if abs(t(i) - 3) <= dt**

**m(i) = 10;**

**elseif t(i) > 6 && t(i) < 9**

**m(i) = m(i-1) - 1\*dt;**

**elseif t(i) > 12 && t(i) < 20**

**tt = tt + (pi)\*dt/2;**

**m(i) = 6 + cos(tt);**

**else**

**m(i) = m(i-1);**

**end**

end

% -------------------------------------------------------------------------

**%% Display**

. . .

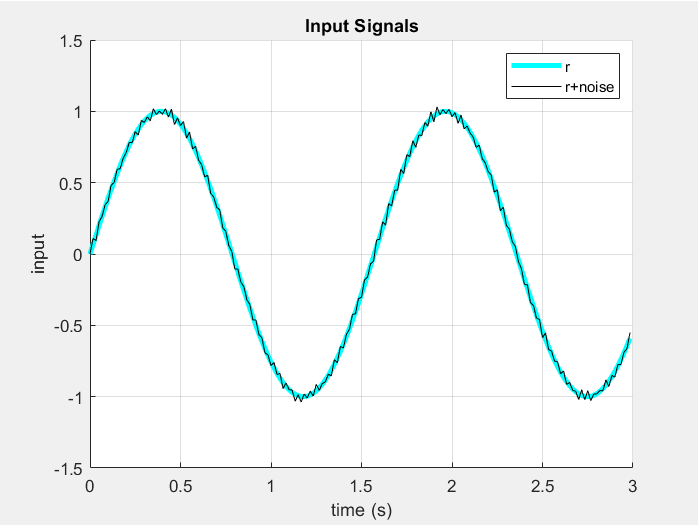
% -------------------------------------------------------------------------

You may notice in the following figures that unlike the figure shown in the class lecture slides, the estimate for the mass () does not converge to the true mass, instead it continues to oscillate up and down around the true mass. These oscillations in the parameter estimate also result in an oscillatory reference tracking error. This occurred regardless of my choice for gamma or lambda, however, with larger gamma and lambda values the oscillations became smaller (up until a point, after which the system broke).

I tested two sets of MRAC controller parameters:

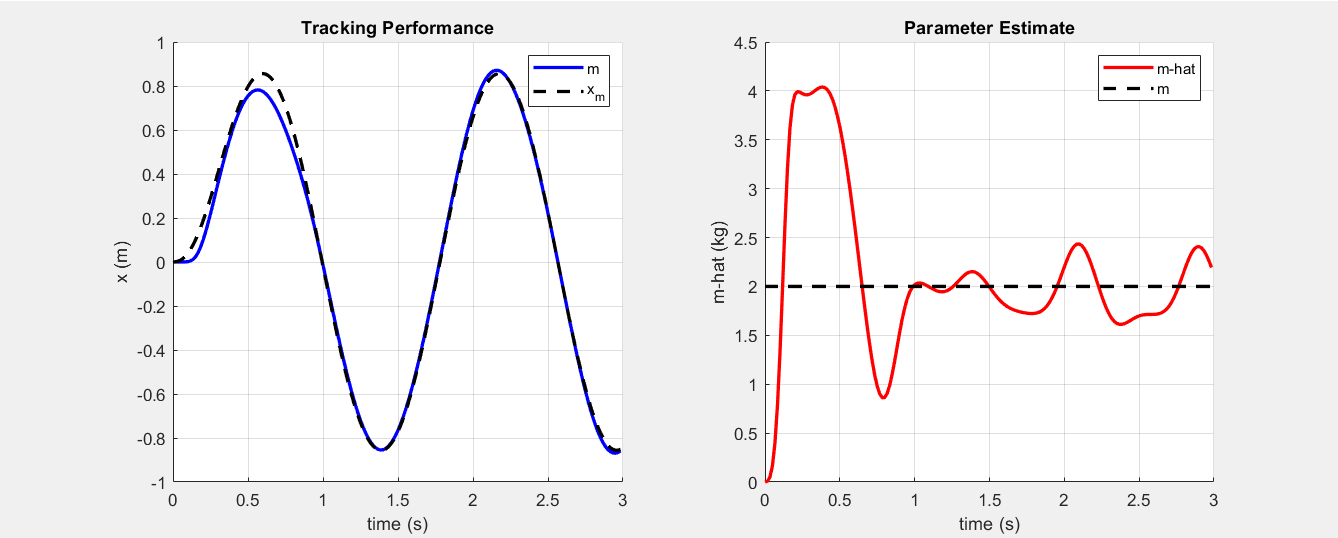
(i) , (ii) ,

The second pair appears to be better for the simple case give for this assignment where always or when it is changing, so long as the input is clean. However, when I tested with a noisy input signal (see figure bellow), the second pair broke completely, whereas the first pair still worked just about as well as with a clean signal.

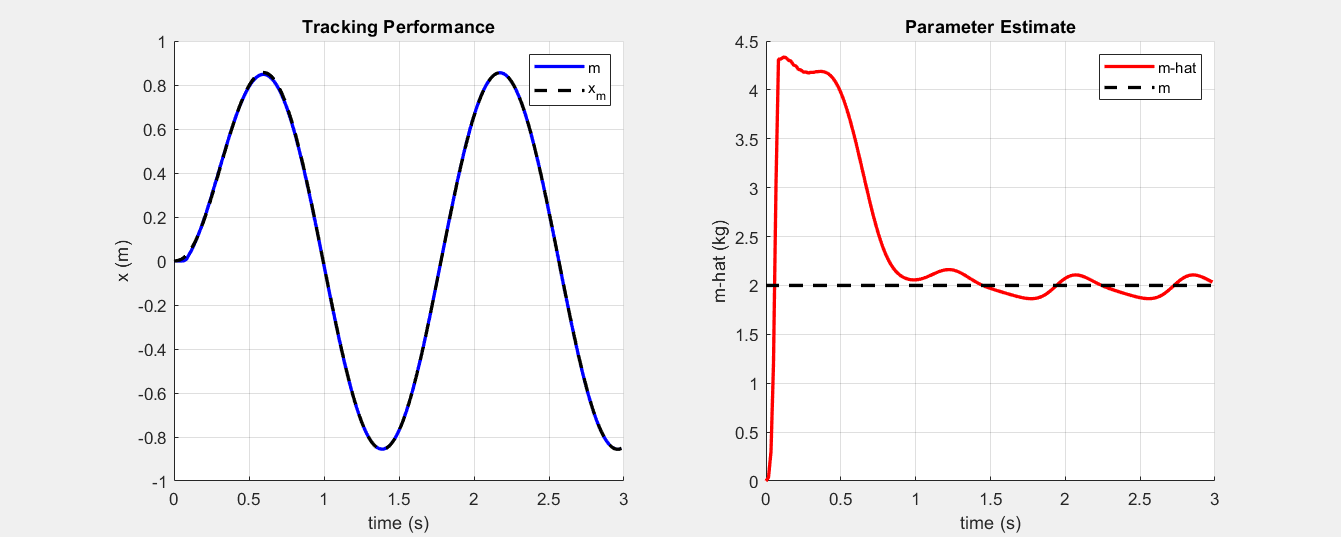


**Constant , no noise:**

,

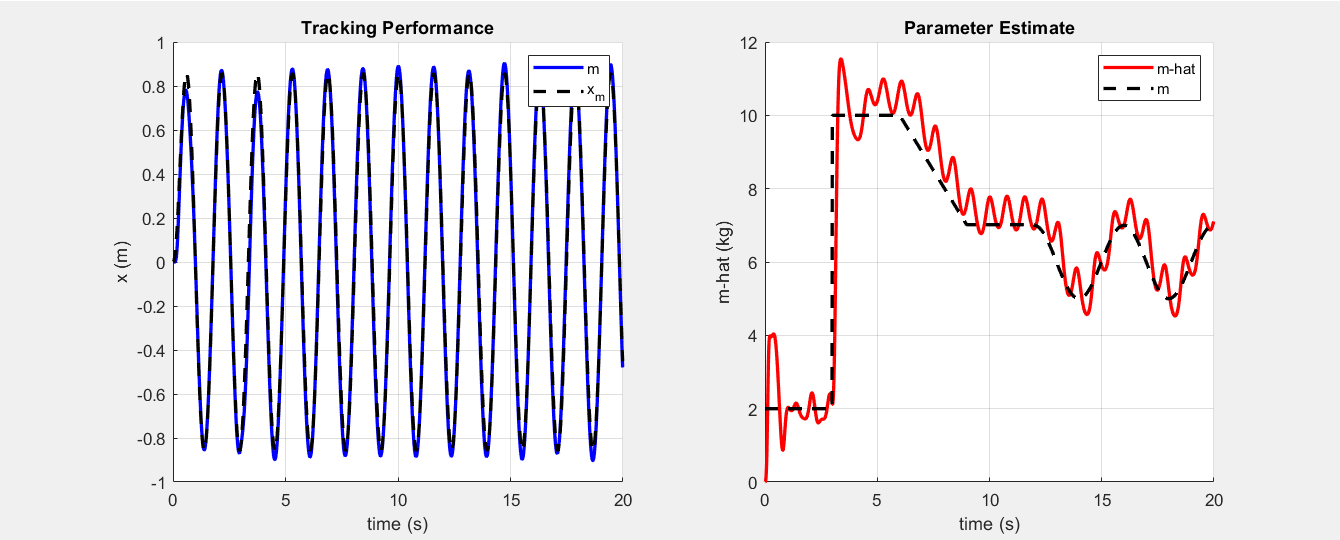


,

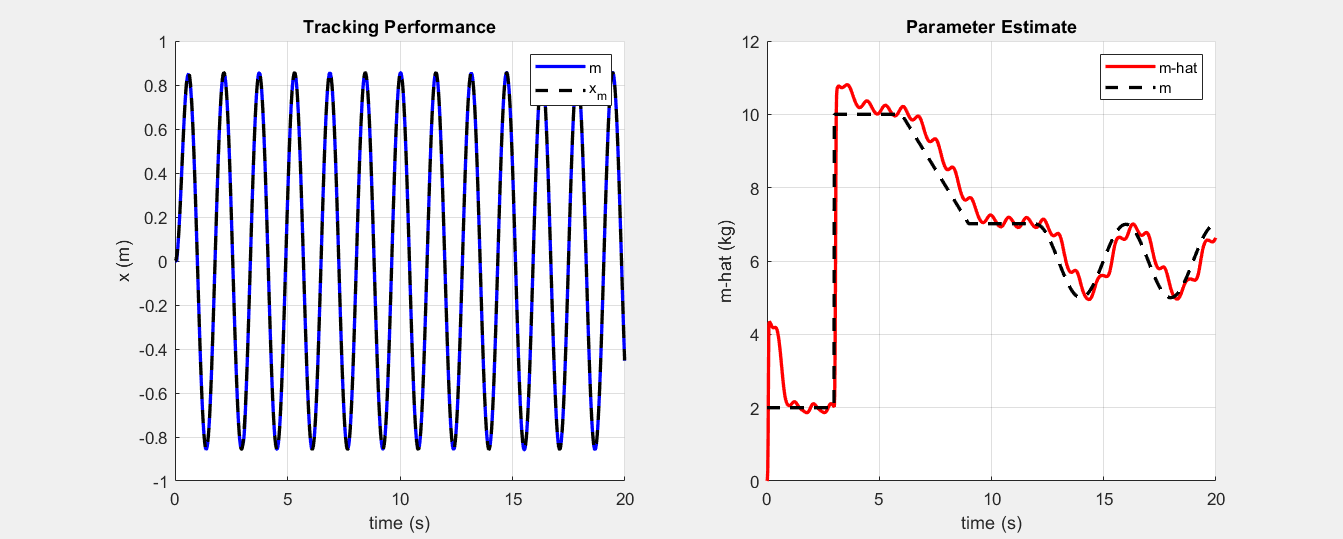


**Changing m, no noise:**

,

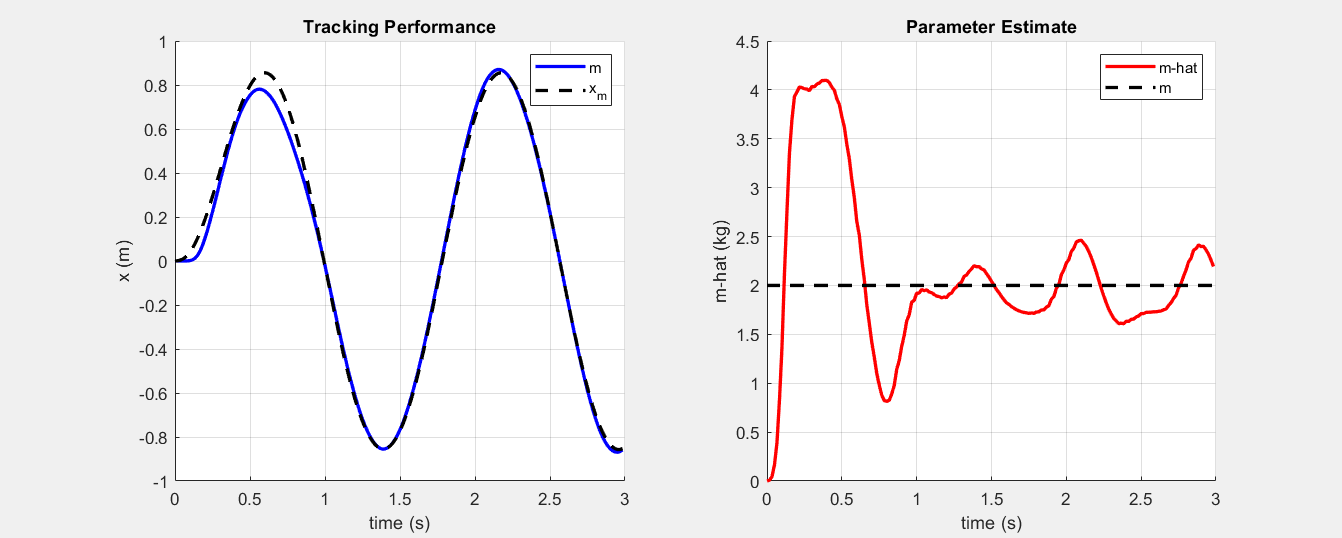


,

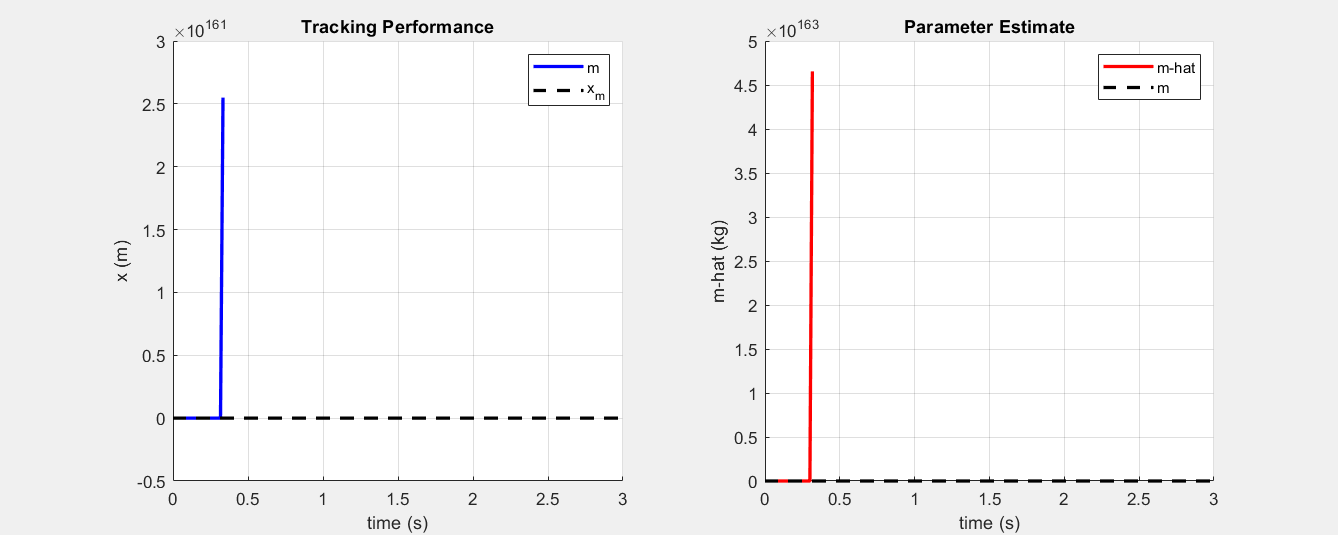


**Constant , with noise:**

,

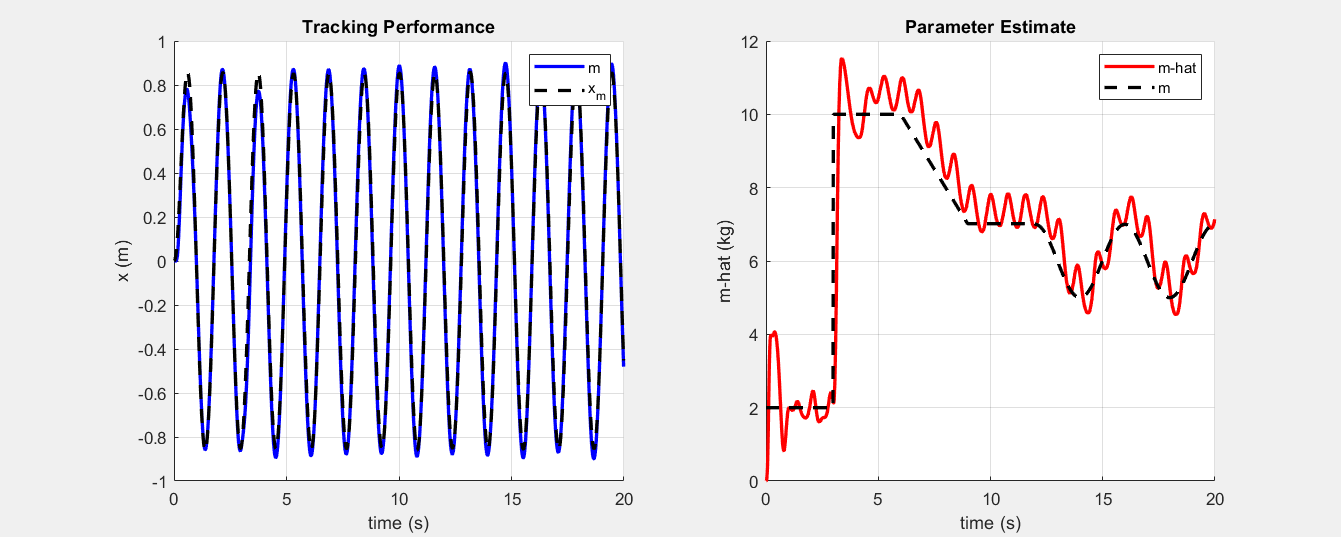


,



**Changing , with noise:**

,



,

