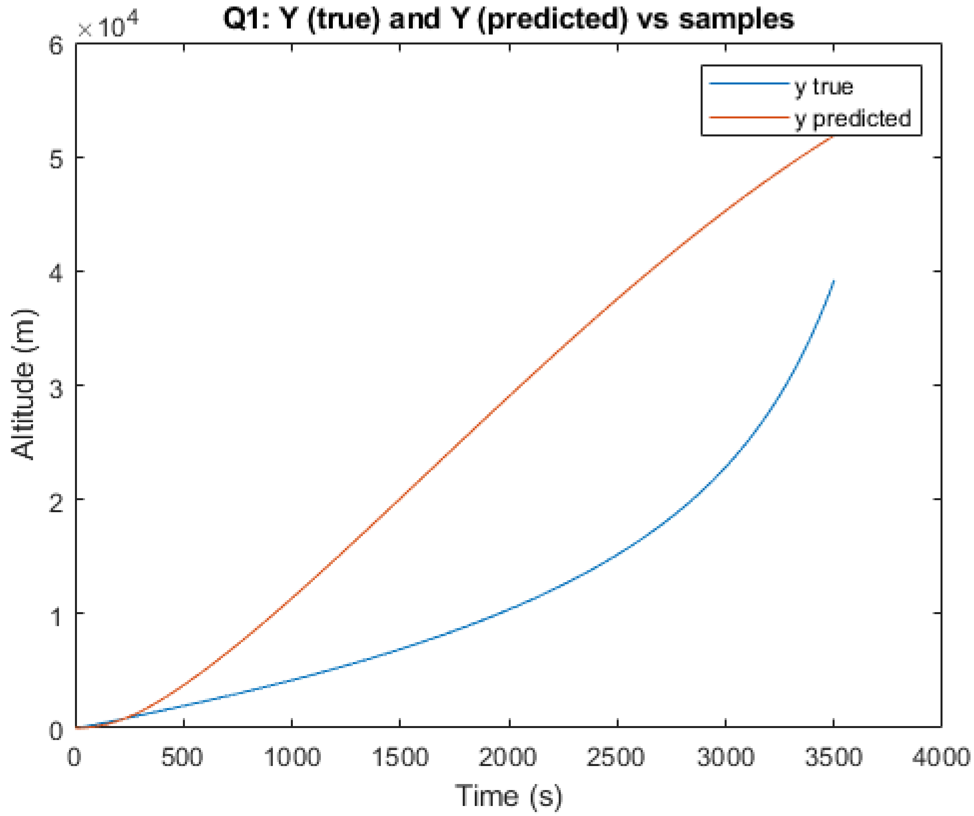
Table of Contents


```
%FILTERING AND IDENTIFICATION
%SC42025
%
%NAME: ANIKET ASHWIN SAMANT
%ID: 4838866
clear all;
clf;

% This assumes that the file "rocket.mat" is present in the same
 directory.

load rocket.mat;
deltaT = 0.1;
m = 100;
g = 9.81;
yinit = 0;
C = [1 0];

A = [1 deltaT; 0 1];

B = [(deltaT^2)/(2*m) -0.5*(deltaT^2) -(deltaT^2)/(2*m);
    deltaT/m -deltaT -deltaT/m];
prediction_size = size(ytrue);
```

---------- Question 1 ----------------

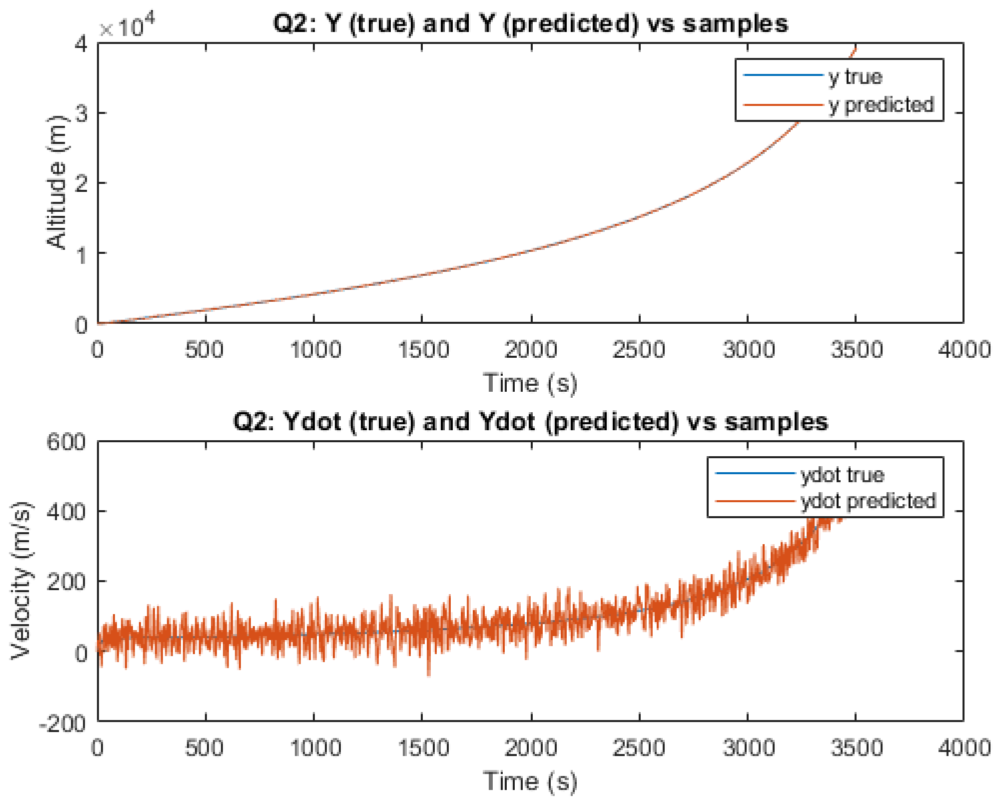
```
y_predicted = zeros(prediction_size);
x_current = [y(1); ydottrue(1)];
for i = 1:prediction_size
    x_next = A*x_current + B*u(i, 1:3)';
    y_predicted(i) = C*x_current;
    x_current = x_next;
end

% Plotting the ytrue values and the predicted y values against the
 sample
% indices
figure(1);
plot(1:prediction_size, ytrue, 1:prediction_size, y_predicted);
legend('y true', 'y predicted');
title('Q1: Y (true) and Y (predicted) vs samples');
xlabel('Time (s)');
ylabel('Altitude (m)');

%As we can clearly see the predicted y values are very different from
 the
%ytrue values provided, and the predicted trajectory coincides only
 for the
%first few samples, beyond which it diverges greatly.
```



---------------- Question 2 ------------------

```
p = [0.8 0.7];
K = place(A', C',p)';

% Initializing the predicted value arrays
y_predicted_asymp = zeros(prediction_size);
ydot_predicted_asymp = zeros(prediction_size);

x_current_asymp = [ytrue(1); ydottrue(1)];

for i = 1:prediction_size
    %predicting the next state using the asymptotic observer
    x_next_asymp = (A - K*C)*x_current_asymp + B*u(i, 1:3)' + K*y(i);
    y_predicted_asymp(i) = C*x_current_asymp;
    ydot_predicted_asymp(i) = [0 1]*x_current_asymp;
    x_current_asymp = x_next_asymp;
end

%Plotting the values of ytrue and the predicted y values against the
 sample
%indices
figure(2);
subplot(2,1,1);
plot(1:prediction_size, ytrue, 1:prediction_size, y_predicted_asymp);
```

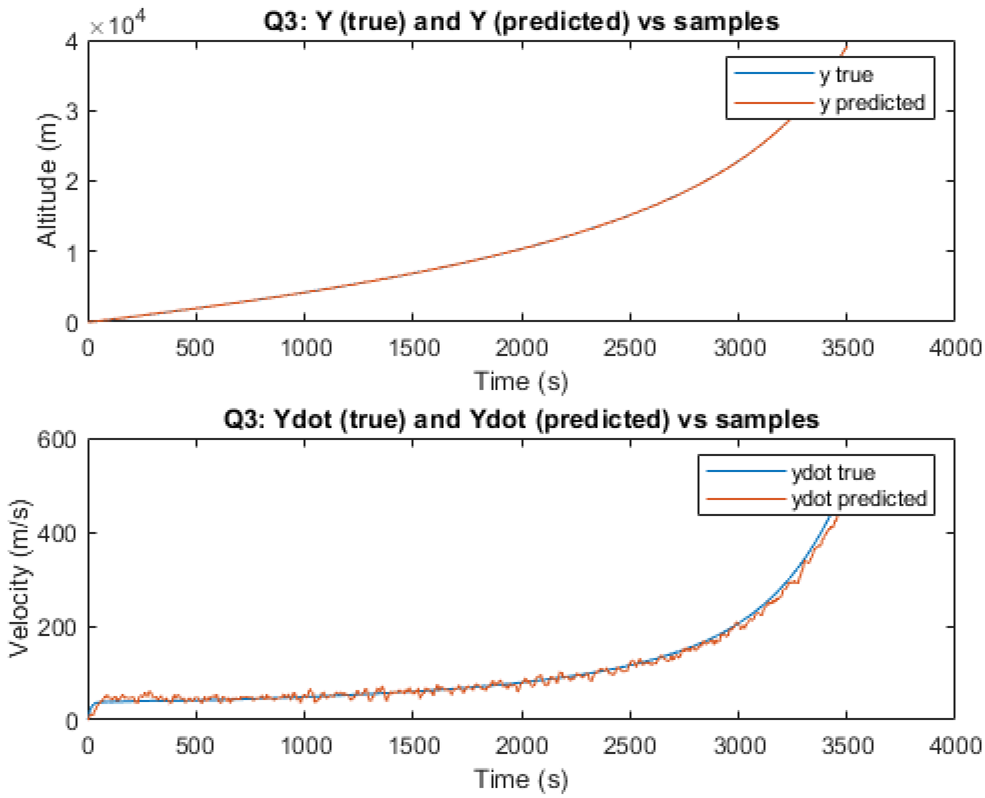
```
legend('y true', 'y predicted');
title('Q2: Y (true) and Y (predicted) vs samples');
xlabel('Time (s)');
ylabel('Altitude (m)');

%Plotting the values of ydottrue and the predicted y dot values
 against the sample
%indices
subplot(2,1,2);
plot(1:prediction_size, ydottrue, 1:prediction_size,
 ydot_predicted_asymp);
legend('ydot true', 'ydot predicted');
title('Q2: Ydot (true) and Ydot (predicted) vs samples');
xlabel('Time (s)');
ylabel('Velocity (m/s)');

% Compared to the previous simulation, we can see that the accuracy of
 the
% simulation has increased greatly due to the asymptotic observer
 feedback.
```



------------ Question 3 -------------

```
S = 0; %given
Q = 1.2*eye(2); %Varying Q values gives different RMSE values.
R = 1000; %given
```

```
% Since S = 0 and Q > 0

P = dare(A',C',Q,R);
%P = 0.5 * (Q + (Q^2 + 4*Q*R)^0.5);
K = (A*P*C')/(C*P*C' + R);

%Initializing the predicted value arrays
y_predicted_kalman = zeros(prediction_size);
ydot_predicted_kalman = zeros(prediction_size);

x_current_kalman = [ytrue(1); ydottrue(1)];

for i = 1:prediction_size
    % Taking the Kalman gain matrix to be stationary
    x_next_kalman = (A - K*C)*x_current_kalman + B*u(i, 1:3)' +
 K*y(i);
    y_predicted_kalman(i) = C*x_current_kalman;
    ydot_predicted_kalman(i) = [0 1]*x_current_kalman;
    x_current_kalman = x_next_kalman;
end

%Plotting the values of ytrue and the predicted y values against the
 sample
%indices
figure(3);
subplot(2,1,1);
plot(1:prediction_size, ytrue, 1:prediction_size, y_predicted_kalman);
legend('y true', 'y predicted');
title('Q3: Y (true) and Y (predicted) vs samples');
xlabel('Time (s)');
ylabel('Altitude (m)');

%Plotting the values of ydottrue and the predicted y dot values
 against the sample
%indices
subplot(2,1,2);
plot(1:prediction_size, ydottrue, 1:prediction_size,
 ydot_predicted_kalman);
legend('ydot true', 'ydot predicted');
title('Q3: Ydot (true) and Ydot (predicted) vs samples');
xlabel('Time (s)');
ylabel('Velocity (m/s)');

% Using different Q values yields different RMSE values, and 1.2
 appears to
% be the value at which the RMSE is at its minimum. However,
 increasing or
% decreasing from 1.2 yields higher RMSE values based on the
 calculated Kalman gain at those Q values.
```



------------ Question 4 ---------------

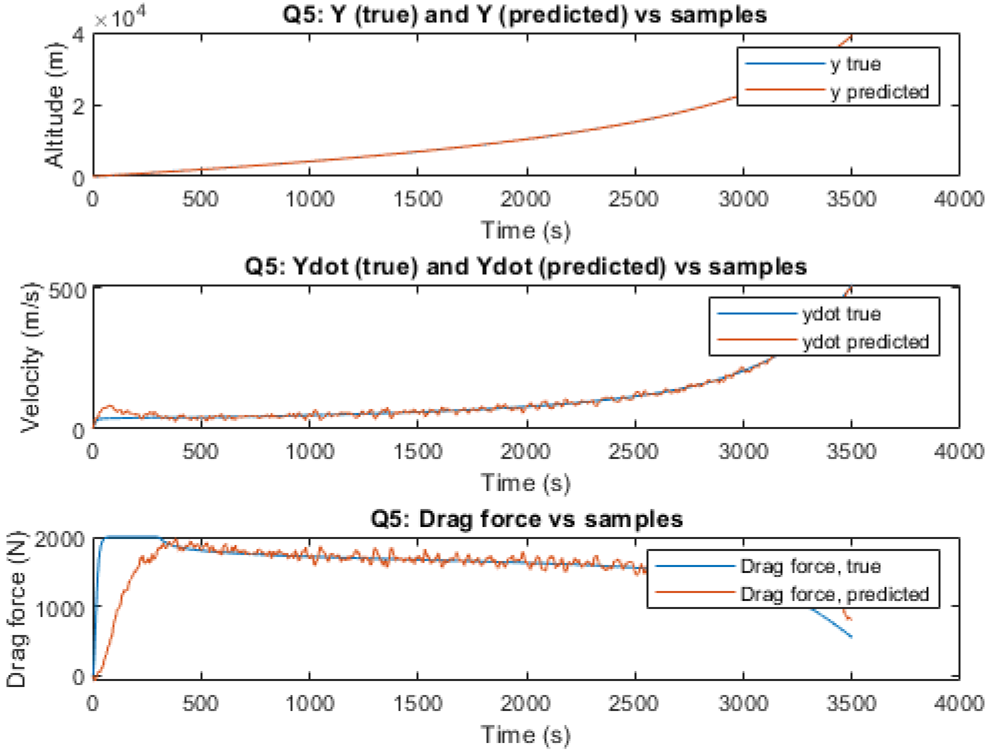
```
%The given data for y can be differentiated in order to calculate the
%difference, and moreover we know the sampling period, so we can also
%calculate the ydot values accordingly. But since we have measurement
 noise
%as well, we can't be sure of the measurement's accuracy. The Kalman
 filter
%provides accurate velocity value predictions, as can be seen from the
%plots.
%Another way to predict the values can be through the implementation
 of a
%non-stationary Kalman filter.
```

------------- Question 5 ----------------

```
% x is now [y ydot d] and A is modified to be a 3x3 matrix.
% w(k) has w3(k) included with it, and x(k+1) has d(k+1) included.

A_new = [1 deltaT -(deltaT^2)/(2*m);
          0 1 -deltaT/m;
          0 0 1];

B_new = [B(:,1:2);0 0];
```

```
C_new = [1 0 0];

S = 0;
Q = 1.2*eye(3);
Q(3,3) = 100;
R = 1000;
P_new = dare(A_new',C_new',Q,R);

K_new = (A_new*P_new*C_new')/(C_new*P_new*C_new' + R);

y_new_predicted_kalman = zeros(prediction_size);
ydot_new_predicted_kalman = zeros(prediction_size);
drag_force_predicted = zeros(prediction_size);

xd_current = [ytrue(1); ydottrue(1); dtrue(1)];

for i = 1:prediction_size
    xd_next = (A_new - K_new*C_new)*xd_current + B_new*u(i, 1:2)' +
 K_new*y(i);
    y_new_predicted_kalman(i) = C_new*xd_current;
    ydot_new_predicted_kalman(i) = [0 1 0]*xd_current;
    drag_force_predicted(i) = [0 0 1]*xd_current;
    xd_current = xd_next;
end

figure(4);
subplot(3,1,1);
plot(1:prediction_size, ytrue, 1:prediction_size,
 y_new_predicted_kalman);
legend('y true', 'y predicted');
title('Q5: Y (true) and Y (predicted) vs samples');
xlabel('Time (s)');
ylabel('Altitude (m)');

subplot(3,1,2);
plot(1:prediction_size, ydottrue, 1:prediction_size,
 ydot_new_predicted_kalman);
legend('ydot true', 'ydot predicted');
title('Q5: Ydot (true) and Ydot (predicted) vs samples');
xlabel('Time (s)');
ylabel('Velocity (m/s)');

subplot(3,1,3);
plot(1:prediction_size, dtrue, 1:prediction_size,
 drag_force_predicted);
legend('Drag force, true', 'Drag force, predicted');
title('Q5: Drag force vs samples');
xlabel('Time (s)');
ylabel('Drag force (N)');

%As we can see, the velocity values can still be calculated, and also
 the
%drag force values. Increasing the Q value increases the variance of
 the
```

```
%velocity estimates though. Since the drag values have a large jump
 after
%the first few samples, we accordingly set Q(3,3) to a high value as
%compared with the Q(1,1) and Q(2,2) values for the altitude and
 velocity
%respectively.
```



------------------- Question 6 ------------------

```
% The RMSE value for ytrue against the predicted values from Q1. As we
 can
% see, the value is quite high since the prediction is hardly
 accurate.
% (Q1)
rms_alt_predicted_Q1 = rms(ytrue - y_predicted)

% The asymptotic observer reduces the RMSE value since we make use of
 the
% observer feedback to estimate our state. (Q2)
rms_alt_asymp_Q2 = rms(ytrue - y_predicted_asymp)

% The stationary Kalman filter further reduces the RMSE value since we
 make
% use of statistical data of the error values to estimate our state.
 (Q3)
rms_alt_kalman_Q3 = rms(ytrue - y_predicted_kalman)
```

```
% RMSE value for Q5, in which we incorporate the drag force into the
 state
% of the system. (Q5)
rms_alt_new_kalman_Q5 = rms(ytrue - y_new_predicted_kalman)

% Velocity RMSE for the asymptotic observer (Q2)
rms_vel_asymp_Q2 = rms(ydot_predicted_asymp - ydottrue)

% Velocity RMSE for the stationary Kalman filter from Q3
rms_vel_kalman_Q3 = rms(ydot_predicted_kalman - ydottrue)

% Velocity RMSE for the stationary Kalman filter from Q5
rms_vel_new_kalman_Q5 = rms(ydot_new_predicted_kalman - ydottrue)

%RMSE of the drag force from Q5
rms_drag_force_Q5 = rms(drag_force_predicted - dtrue);
```

```
rms_alt_predicted_Q1 =

   1.5612e+04


rms_alt_asymp_Q2 =

   31.6591


rms_alt_kalman_Q3 =

   19.3842


rms_alt_new_kalman_Q5 =

   20.0838


rms_vel_asymp_Q2 =

   33.6383


rms_vel_kalman_Q3 =

    9.6603


rms_vel_new_kalman_Q5 =

    9.7348
```

Published with MATLAB® R2018a