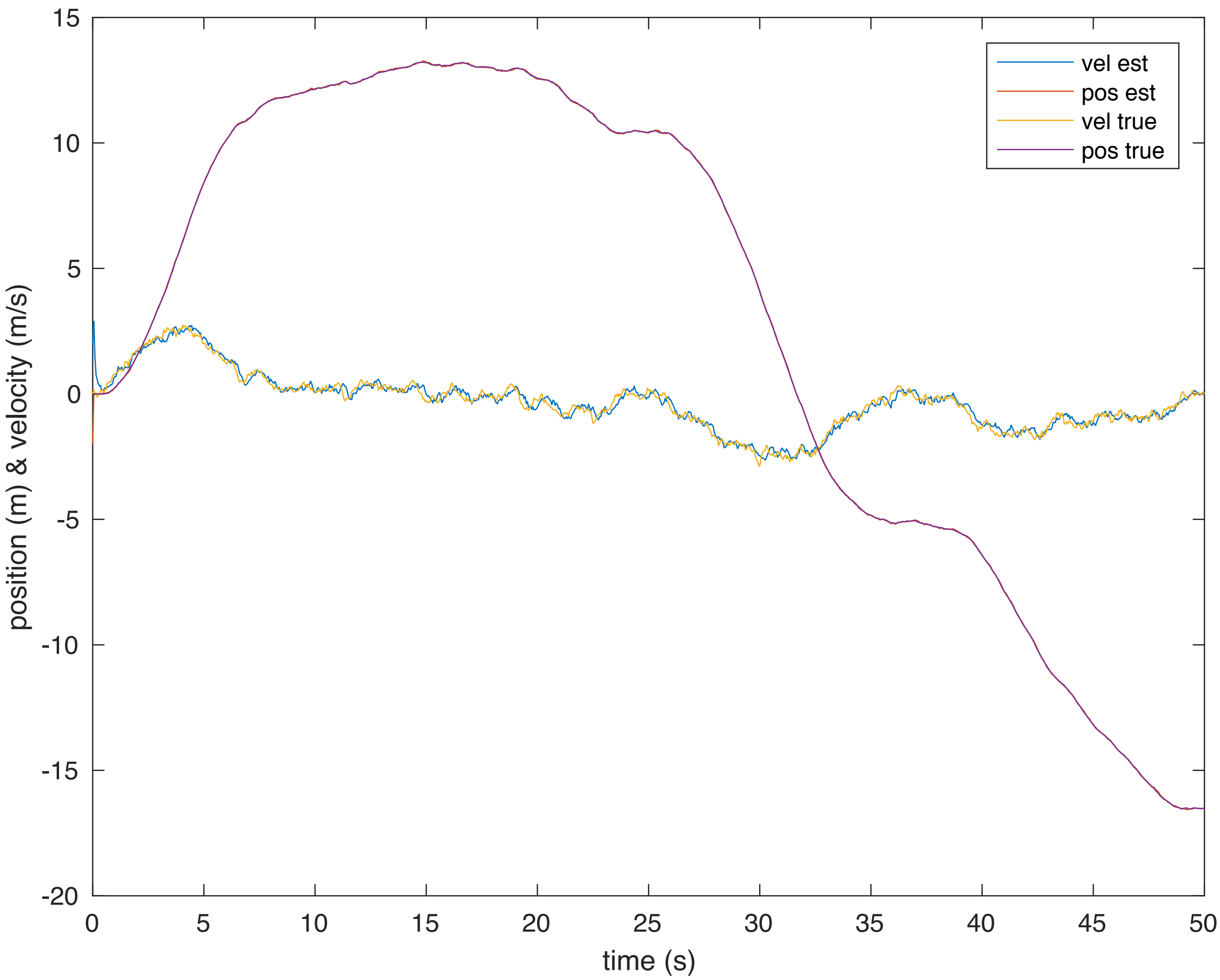
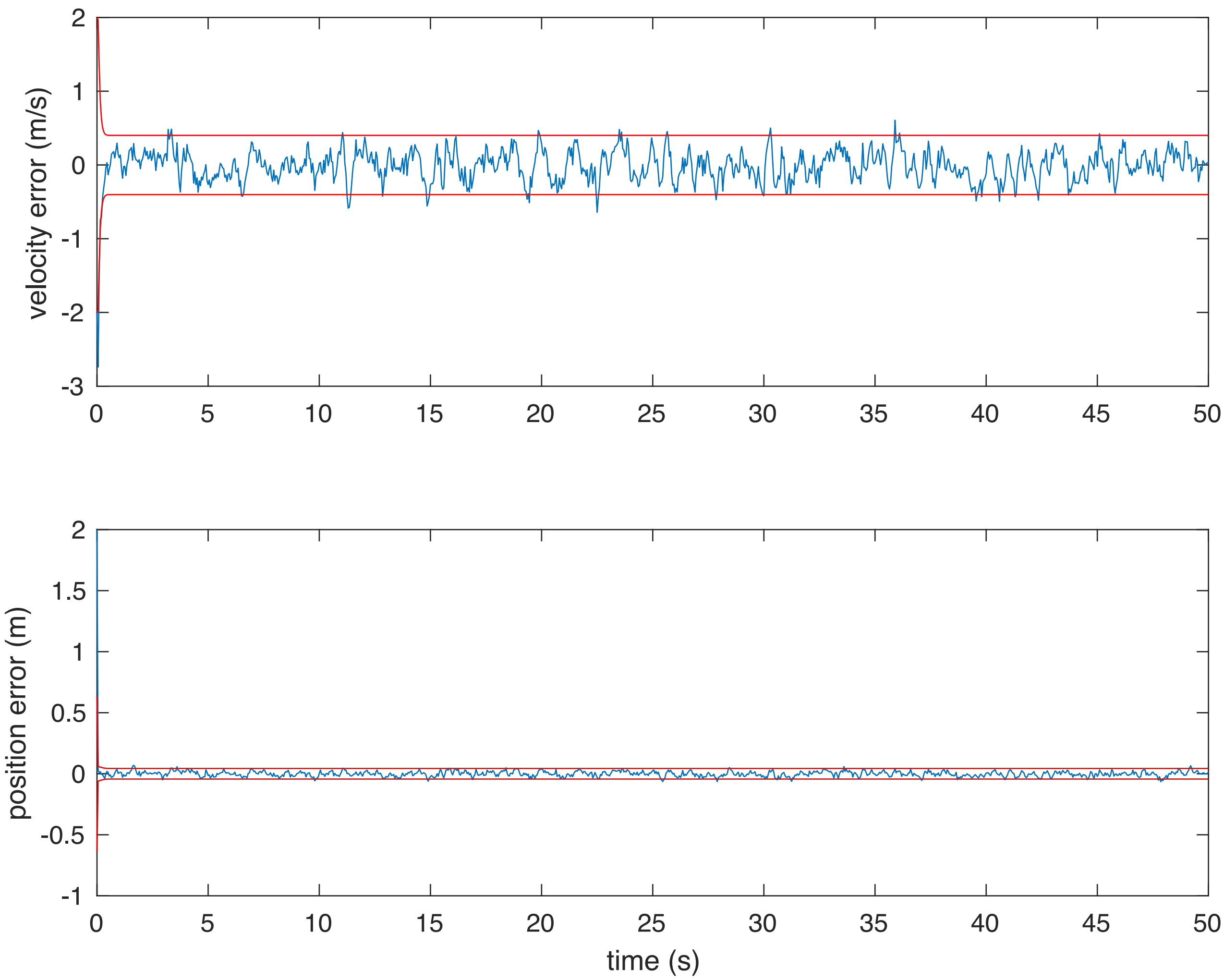
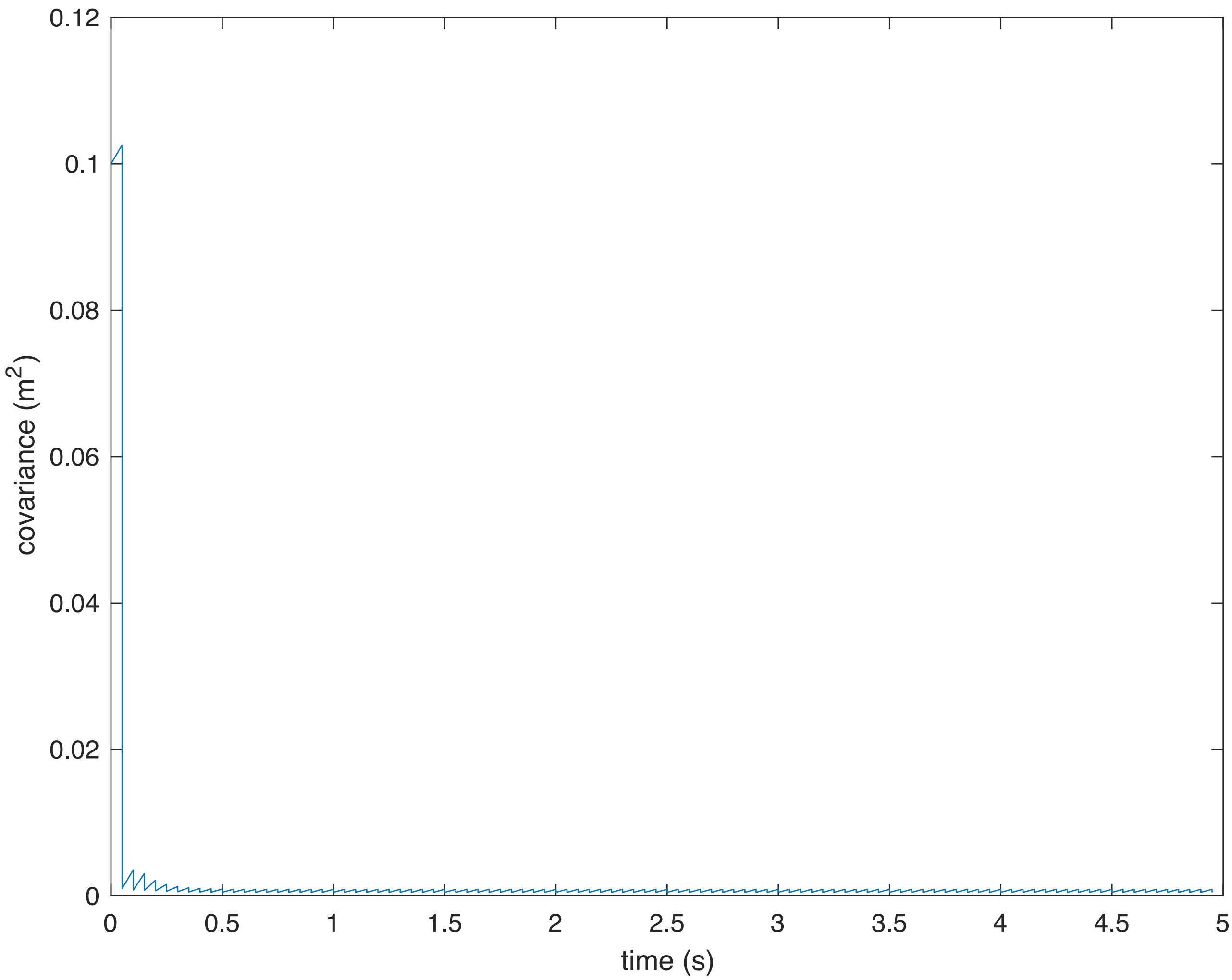
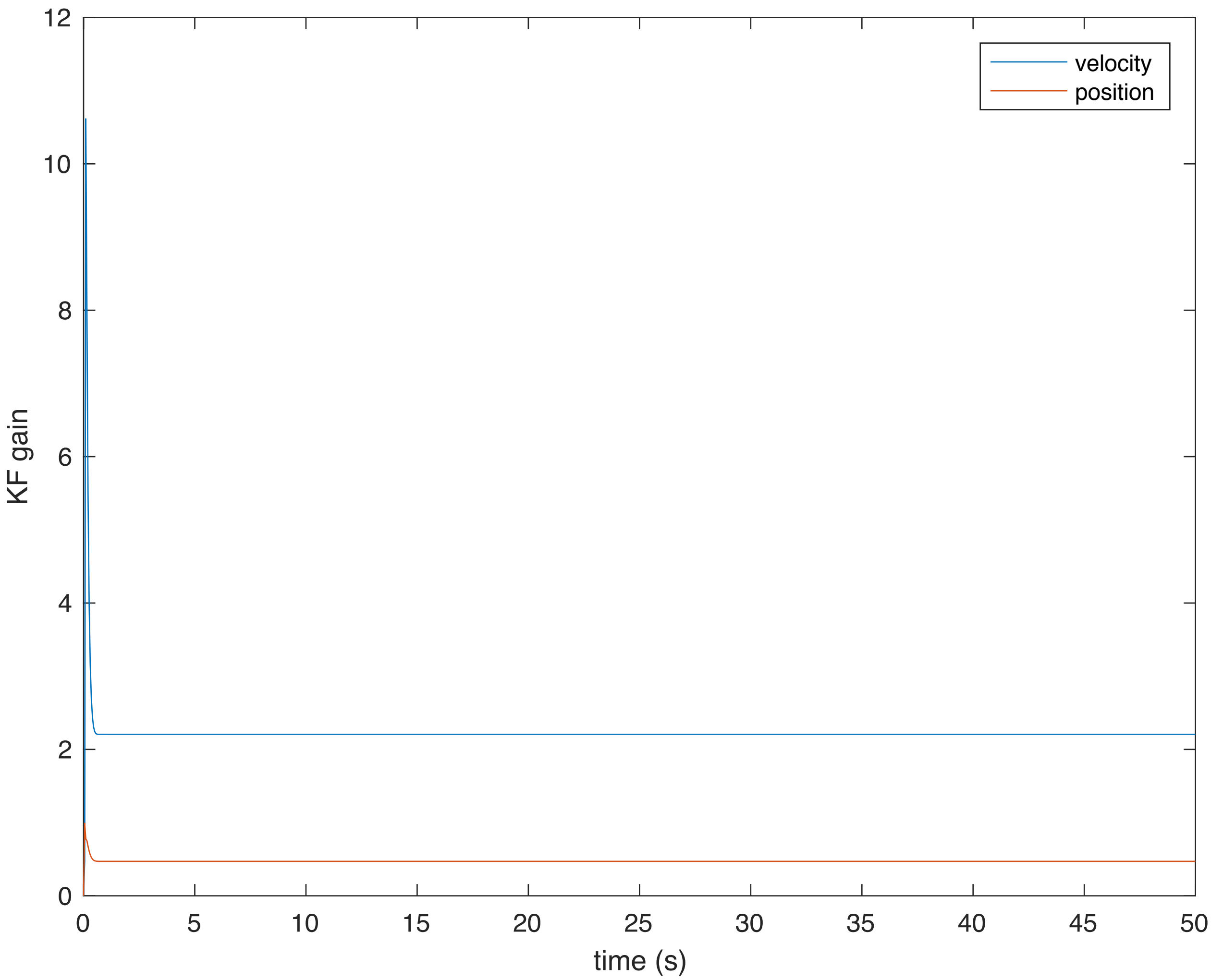


vel est
pos est
vel true
pos true
position (m) & velocity (m/s)
time (s)
15
10
5
0
-5
-10
-15
-20
0
5
10
15
20
25
30
35
40
45
50


velocity error (m/s)
position error (m)
time (s)



0.12
0.1
0.08
0.06
0.04
0.02
0
covariance (m2)
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
time (s)



velocity
position
KF gain
time (s)
0
2
4
6
8
10
12
0
5
10
15
20
25
30
35
40
45
50

```
% UUV simulation
% HW#1 Autonomous Systems
% Sep 2017

% Most of the file below is devoted to storing and plotting variables
% The Kalman filter implementation is a dozen lines or so

clear all;

% Define model parameters
m = 100;
b = 20;

% Linear state-space continuous-time model
F = [-b/m 0; 1 0];
G = [1/m; 0];
H = [0 1];
J = [0];

Ts = 0.05;      % sample period

sys = ss(F,G,H,J);      % continuous LTI object
sysd = c2d(sys,Ts);     % discrete LTI object
[A,B,C,D] = ssdata(sysd);

Sig0 = diag([1 0.1]);
Sig = Sig0;
R = 1*diag([0.01, 0.0001]); % HW values
Q = 0.001;
% R = 1*diag([0.05, 0.0001]);   % Good for simulating evolution of probabilities
% Q = 0.1;
% R = 0.0000001*diag([0.05, 0.0001]);   % Low noise conditions
% Q = 0.0000000000001;

% initialize variables
tfinal = 50;
t = 0:Ts:tfinal;
N = length(t);

u = zeros(1,N);
u(1:100) = 50;
u(501:600) = -50;

X = zeros(2,N);
X0 = [0; 0];
X(:,1) = X0;

z = zeros(1,N);

% Create noisy truth data
for i=2:N
    X(:,i) = A*X(:,i-1) + B*u(i-1) + [sqrt(R(1,1))*randn; sqrt(R(2,2))*randn];
    z(i) = X(2,i) + sqrt(Q)*randn;
end

vtr = X(1,:);
xtr = X(2,:);
```

```
% initialize KF states, variables
mu = zeros(2,N);
mu0 = [2; -2];
mu(:,1) = mu0;
mubar = mu;

K = zeros(2,N);
K(1,1) = 0.0;
K(2,1) = 0.0;

Mv = zeros(1,N);
Mx = zeros(1,N);
Pv = zeros(1,N);
Px = zeros(1,N);
Mv(1) = Sig(1,1);
Mx(1) = Sig(2,2);
Pv(1) = Sig(1,1);
Px(1) = Sig(2,2);

% implement KF
for i = 2:N
  % prediction step
  mubar(:,i) = A*mu(:,i-1) + B*u(i);   % state estimate
  Sigbar = A*Sig*A' + R;               % covariance estimate
  Mv(i) = Sigbar(1,1);                 % storing covariance values for plotting
  Mx(i) = Sigbar(2,2);

  % correction step
  K(:,i) = (Sigbar*C')/(C*Sigbar*C' + Q);    % KF gain
  mu(:,i) = mubar(:,i) + K(:,i)*(z(i)-C*mubar(:,i));   % state estimate
  Sig = (eye(2)-K(:,i)*C)*Sigbar;      % covariance estimate
  Pv(i) = Sig(1,1);                    % storing covariance values for plotting
  Px(i) = Sig(2,2);
end

figure(1); clf;
mbx = mubar(2,:);
mx = mu(2,:);
for i=1:N/10
    % prediction
    sig = sqrt(Mx(i));
    xx = (mbx(i)-4*sig):6*sig/50:(mbx(i)+4*sig);
    px = 1/(sqrt(2*pi)*sig)*exp(-0.5*(xx-mbx(i)).^2/sig^2);
    plot(xx,px);
%     axis([-2 3 0 1.5]);
    axis([-0.2 2 0 20]);
    xlabel('position (m)');
    ylabel('belief');
    hold on;

    % measurement
    sig = sqrt(Q);
    xx = (z(i)-4*sig):6*sig/50:(z(i)+4*sig);
    pz = 1/(sqrt(2*pi)*sig)*exp(-0.5*(xx-z(i)).^2/sig^2);
    plot(xx,pz);

    % measurement update
    sig = sqrt(Px(i));
```

```
    xx = (mx(i)-4*sig):6*sig/50:(mx(i)+4*sig);
    px = 1/(sqrt(2*pi)*sig)*exp(-0.5*(xx-mx(i)).^2/sig^2);
    plot(xx,px);
    legend('prediction','measurement','measurment update');
    hold off;
    pause(0.001);
end

figure(2); clf;
plot(t,X);
plot(t,mu,t,X);
xlabel('time (s)');
ylabel('position (m) & velocity (m/s)');
legend('vel est','pos est','vel true','pos true');

v_err = vtr - mu(1,:);
x_err = xtr - mu(2,:);
figure(3); clf;
subplot(211); plot(t,v_err,t,2*sqrt(Pv),'r',t,-2*sqrt(Pv),'r');
ylabel('velocity error (m/s)');
subplot(212); plot(t,x_err,t,2*sqrt(Px),'r',t,-2*sqrt(Px),'r');
xlabel('time (s)');
ylabel('position error (m)');

PPx = reshape([Mx; Px],[1,2002]);
tt = reshape([t;t],[1,2002]);
figure(4); clf;
plot(tt(1:200),PPx(1:200));
xlabel('time (s)');
ylabel('covariance (m^2)');

figure(5); clf;
plot(t,K);
xlabel('time (s)');
ylabel('KF gain');
legend('velocity','position');
```