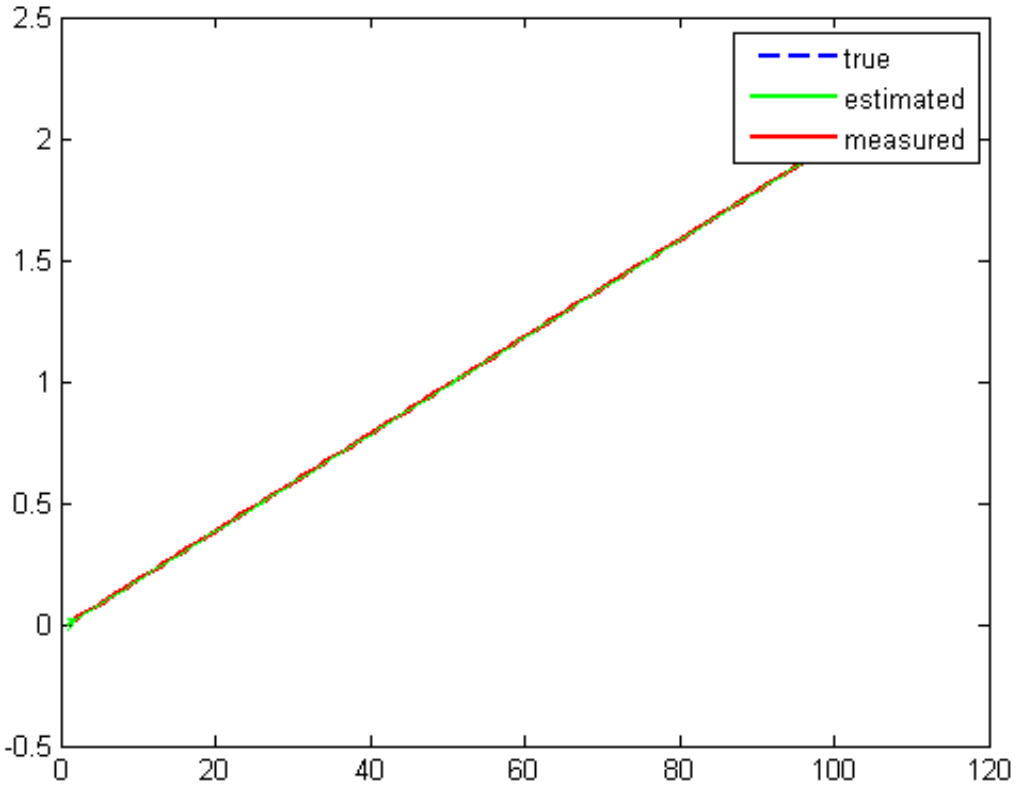
```
function f = int_kf_test
% Kalman filtering of the double integrator with position measurements

% timing
dt = 1;   % time-step
N = 100;   % total time-steps
T = N*dt;  % final time

% noise terms
S.q = (3*10^-6)^2;    % external disturbance variance
S.r = (1.5*10^-5)^2;    % measurement noise variance

% F matrix
S.F = [1 -dt;
       0 1];

% G matrix
S.G = [dt;
        0];

% Q matrix
S.Q = [(3*10^-6)*dt+dt^3/3*(3*10^-9), -dt^2/2*(3*10^-9);
       -dt^2/2*(3*10^-9),   (3*10^-9)*dt];

% R matrix
S.R = S.r;

% H matrix
S.H = [1, 0];

% initial estimate of mean and covariance
x = [0; 1.7*10^-7];
P = [1*10^-4, 0;
        0, 1*10^-12];

xts = zeros(2, N+1); % true states
xs = zeros(2, N+1);  % estimated states
Ps = zeros(2, 2, N+1); % estimated covariances

zs = zeros(1, N);  % estimated state

pms = zeros(1, N); % measured position

xts(:,1) = x;
xs(:,1) = x;
Ps(:,:,1) = P;

Bs = zeros(1, N+1);
Bs(1) = x(2);
for s=2:N
   Bs(s)=Bs(s-1)+ 3*10^-6*randn(1);
end

for k=1:N
  u = 0.02+Bs(k)+3*10^-6*randn(1); % pick some known control

  xts(:,k+1) = S.F*xts(:,k) + S.G*(0.02 + S.q^0.5*randn);  % true state

  [x,P] = kf_predict(x,P,u,S);  % prediction
```

```
  z = xts(1,k+1) + sqrt(S.r)*randn;   % generate random measurement

  [x,P] = kf_correct(x,P,z,S);  % correction

  % record result
  xs(:,k+1) = x;
  Ps(:,:,k+1) = P;
  zs(:,k) = z;
end

plot(xts(1,:), '--', 'LineWidth',2)
hold on
plot(xs(1,:), 'g', 'LineWidth',2)
plot(2:N+1,zs(1,:), 'r', 'LineWidth',2)

legend('true', 'estimated','measured')

% 95% confidence intervals of the estimated position
plot(xs(1,:) + 1.96*reshape(sqrt(Ps(1,1,:)),N+1,1)', '-g')
plot(xs(1,:) - 1.96*reshape(sqrt(Ps(1,1,:)),N+1,1)', '-g')


function [x,P] = kf_predict(x, P, u, S)

x = S.F*x + S.G*u;
P = S.F*P*S.F' + S.Q;

function [x,P] = kf_correct(x, P, z, S)

K = P*S.H'*inv(S.H*P*S.H' + S.R);
P = (eye(length(x)) - K*S.H)*P;
x = x + K*(z - S.H*x);
```



Published with MATLAB® R2014a