```
% Tim Coon 01/17/2014
% Q3 Qualifying Exam EENG765
% Problem #4
clear; close all; clc;
set(0,'defaulttextinterpreter','latex')

%% Given Parameters
dt_x = 1;                % (sec) propagation time = time step duration
dt_y = 1;
beta_x = 1/dt_x;         % inverse of the time constant
beta_y = 1/dt_y;
sigma_x = 1;             % (m)
sigma_yn = 3;            % (m)
t_span = 7;              % (sec) simulation time
t1 = 0:dt_x:t_span;      % dynamics time vector
n_states = length(t1);   % number of time steps in process
b_var = 2;                   % (m^2) constant measurement bias variation

%% Calculated Values
% shaping filters
sFilter_x = sqrt(2*sigma_x^2*beta_x);
sFilter_y = sqrt(2*sigma_yn^2*beta_y);
% State-Space matrices from hand calculations
fx = [0 1; 0 -beta_x];          % x-direction system matrix
fy = [0];                       % y-direction system matrix
fb = [0];                       % bias dynamics system matrix
gx = [0 0; sFilter_x 0];        % x-direction input coeff matrix
gy = [0 sFilter_y];             % y-direction input coeff matrix
gb = [0 0];                     % bias input coeff matrix
F = matrix_concat(fx, fy, fb);  % system matrix
n_statevar = size(F,1);         % number of state variables
G = [gx; gy; gb];  % input coefficients, ref Maybeck (5-123)
W = eye(size(G,2));    % PSD matrix for Van Loan Method (identity for UWN)
% [state transition matrix, discrete process noise covariance matrix]
[phi,Qd] = get_phi_Qd(F,G,W,dt_x);

%% Measurement Realizations
H = [0.4 0 2 1];         % Measurement Matrix
R = [];                   % Measurement noise covariance matrix

%% Estimation
xm0 = zeros(n_statevar,1);
Pm0 = zeros(n_statevar);
Pm0(end,end) = b_var;
[x_hat, x_std] = TC_KF_P4(xm0,Pm0,phi,H,Qd,R,n_states,t1);

%% Estimation Plots
figure()
```

```
suptitle({'Estimate of Ant Location';' '})
set(0,'Units','pixels')
sz = get(0,'ScreenSize');
set(gcf,'Position',[0 0 sz(3)/2 sz(4)])
% px ----------------------------------------------------------------------
subplot(411)
hold on
plot(t1,x_hat(1,:),'r--','linewidth',2)
stairs(t1,x_hat(1,:)+x_std(1,:),'k','linewidth',1)
stairs(t1,x_hat(1,:)-x_std(1,:),'k','linewidth',1)
hold off
ylabel('x-position (m)'); xlabel('time (s)');
legend('Estimate','St Dev','location','eastoutside');
% vx ----------------------------------------------------------------------
subplot(412)
hold on
plot(t1,x_hat(2,:),'r--','linewidth',2)
stairs(t1,x_hat(2,:)+x_std(2,:),'k','linewidth',1)
stairs(t1,x_hat(2,:)-x_std(2,:),'k','linewidth',1)
hold off
ylabel('x-velocity (m)'); xlabel('time (s)');
legend('Estimate','St Dev','location','eastoutside');
% py ----------------------------------------------------------------------
subplot(413)
hold on
plot(t1,x_hat(3,:),'r--','linewidth',2)
stairs(t1,x_hat(3,:)+x_std(3,:),'k','linewidth',1)
stairs(t1,x_hat(3,:)-x_std(3,:),'k','linewidth',1)
hold off
ylabel('y-position (m)'); xlabel('time (s)');
legend('Estimate','St Dev','location','eastoutside');
% b  ----------------------------------------------------------------------
subplot(414)
hold on
plot(t1,x_hat(4,:),'r--','linewidth',2)
stairs(t1,x_hat(4,:)+x_std(4,:),'k','linewidth',1)
stairs(t1,x_hat(4,:)-x_std(4,:),'k','linewidth',1)
hold off
ylabel('Sensor Bias'); xlabel('time (s)');
legend('Estimate','St Dev','location','eastoutside');

% path --------------------------------------------------------------------
figure()
subplot(3,1,1:2)
plot(x_hat(1,:),x_hat(3,:),'b-*','linewidth',2)
ylabel('y-position (m)'); xlabel('x-position (m)');
title({'Estimated Path'; ''});
% axis equal
```

```
% xlim([-0.1 0.1]);
ylim([0 2.5]);
ax = subplot(3,1,3);
set(ax,'visible','off')
x_h = num2str(x_hat,'%10.4f');
text(0.2,0.3,x_h)
text(0.1,0.3,'$\hat{x} = $')
```