1. Find the stationary point of the curve \( f(x, y) = xy \) subject to the constraint \( x + y = 1 \) using the method of Lagrange multipliers.

**Solution:** Define \( g(x, y, \lambda) = f(x, y) + \lambda(x + y - 1) \). Let its partial derivatives respect to \( x, y, \lambda \) to be zeros:

\[
\begin{align*}
\frac{\partial g}{\partial x} &= y + \lambda = 0 \\
\frac{\partial g}{\partial y} &= x + \lambda = 0 \\
\frac{\partial g}{\partial \lambda} &= x + y - 1 = 0
\end{align*}
\]

\[
\begin{cases}
x = \frac{1}{2} \\
y = \frac{1}{2} \\
\lambda = -\frac{1}{2}
\end{cases}
\]

The stationary point of the curve subject to the constraint is at \( \left(\frac{1}{2}, \frac{1}{2}\right) \). The above plot shows relationships between the curve and the constraint. See Matlab script `hw12_1.m` in appendix for detail.

2. The goal of this problem is to find position estimates for a missile moving in 3D. …

   a. Plot the 3 coordinates of the missile position in space …

**Solution:** It is plotted below. Matlab script `hw12_2.m` is listed in appendix.
b. Write the state equation, ...

**Solution**: The state equation is written as:

\[
\begin{bmatrix}
  a_1^{(i)} \\
  a_2 \\
  a_3 \\
  a_4 \\
  a_5 \\
  a_6
\end{bmatrix} =
\begin{bmatrix}
  1 & 0 & 0 & \Delta t & 0 & 0 \\
  0 & 1 & 0 & 0 & \Delta t & 0 \\
  0 & 0 & 1 & 0 & 0 & \Delta t \\
  1 & 0 & 0 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 & 0 & 0 \\
  0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\cdot
\begin{bmatrix}
  a_1^{(i-1)} \\
  a_2 \\
  a_3 \\
  a_4 \\
  a_5 \\
  a_6
\end{bmatrix} +
\begin{bmatrix}
  w_1^{(i)} \\
  w_2 \\
  w_3 \\
  w_4 \\
  w_5 \\
  w_6
\end{bmatrix}
\]

where \( w_i \sim N(0,0.1) \), \( \Delta t = 0.01 \) second is the time step. Rewrite the above equation in a simpler form: \( a^{(i)} = F a^{(i-1)} + w^{(i-1)} \).

c. Write the measurement equation, ...

**Solution**: The measurement equation is written as:

\[
\begin{bmatrix}
  x_1^{(i)} \\
  x_2 \\
  x_3
\end{bmatrix} =
\begin{bmatrix}
  1 & 0 & 0 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 & 0 & 0 \\
  0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
\cdot
\begin{bmatrix}
  x_1^{(i)} \\
  x_2 \\
  x_3
\end{bmatrix} +
\begin{bmatrix}
  v_1^{(i)} \\
  v_2 \\
  v_3
\end{bmatrix}
\]
where \( v_k \sim N(0,3) \). Rewrite the above equation in a simpler form: \( x^{(i)} = H_{a^{(i)}} + v^{(i)} \).

d. Write a Matlab Kalman filter function, …

**Solution:** See Matlab script `hw12_2.m` in appendix in detail.

e. Run the Kalman filter …

**Solution:** See Matlab script `hw12_2.m` in appendix in detail. The estimations are plotted below:

3. Read Chapter 19.4 of the book by …

**Solution:** Done.

**Appendix:**

- `hw12_1.m`:

```matlab
function hw12_1
% Syntax: hw12_1
%
% Description: CMSC828D HW12_1
%
% Author: Haiying Liu
% Date: Dec. 2, 2000
%
```
dbstop if error

msg = nargchk(0, 0, nargin);
if (~isempty(msg))
    error(strcat('ERROR:', msg));
end

clear msg;

figure;

meshX  = -2:0.1:2;
meshY  = -2:0.1:2;
[x, y] = meshgrid(meshX, meshY);

f_xy   = x .* y;

mesh(x, y, f_xy);
grid on;

lx = -2:0.1:2;
ly = 1 - lx;
lz = ones(length(lx), 1);

hold on;
plot3(lx, ly, lz);

plot3(.5, .5, 1, '*');
text(.5, .5, 1.5, 'stationary point');
text(2, -1, 1.5, 'x + y = 1');

xlabel('x');
ylabel('y');
zlabel('f(x,y)');
view(-157, 26);

print -djpeg hw12_1;
• hw12_2.m:

function hw12_2
% Syntax: hw12_2
%
% Description: CMSC828D HW12_2
%
% Author: Haiying Liu
% Date: Dec. 2, 2000
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dbstop if error

msg = nargchk(0, 0, nargin);
if (~isempty(msg))
    error(strcat('ERROR:', msg));
end

clear msg;

%======================================================================

%= a. Plot the 3 coordinates of the missile position in space as a function of time on a single figure.

% Read the raw data.
rawData = textread('KF_meas.dat', '%f');
x = reshape(rawData, 3, length(rawData) / 3)';
nX = size(x, 1);

figure;

subplot(1, 2, 1);
plot3(x(:, 1), x(:, 2), x(:, 3), 'g.');
xlabel('x_1');
ylabel('x_2');
zlabel('x_3');
grid on;
view(3);

subplot(1, 2, 2);
dt = 0.01;
t = 1:dt:(nX - 1)* dt + 1;
plot(t, x(:, 1), t, x(:, 2), t, x(:, 3));
xlabel('t');
ylabel('x_1, x_2, x_3');
grid on;

print -djepg hw12_2a;

%======================================================================

%= d. Write a Matlab Kalman filter function ...

% See the function 'KalmanFilter' below.
% e. Run the Kalman filter. Plot the estimates for the missile ...

% Initialization.
phi = [ ...
1 0 0 dt 0 0 ; ...  
0 1 0 0 dt 0 ; ...  
0 0 1 0 0 dt ; ...  
1 0 0 0 0 0 ; ...  
0 1 0 0 0 0 ; ...  
0 0 1 0 0 0 ; ...  ];

H = [ ...
1 0 0 0 0 0; ...  
0 1 0 0 0 0; ...  
0 0 1 0 0 0; ...  ];

Q = 0.01 * eye(6);
R = 4 * eye(3);

state.a = zeros(6, 1);
state.phi = phi;
state.Q = Q;
state.K = [];
state.P = eye(6);
state.PP = eye(6);

measurement.x = [];
measure.H = H;
measure.R = R;

% Estimate for the dynamic system.
est_a = zeros(nX, 6);
for index = 1:nX
    measure.x = x(index, :)';
    state = KalmanFilter(state, measure);
    est_a(index, :) = state.a';
end

subplot(1, 2, 1);
hold on;
plot3(est_a(:, 1), est_a(:, 2), est_a(:, 3), 'k');

subplot(1, 2, 2);
hold on;
t = [1 - dt, t];
est_a = [zeros(6, 1)'; est_a];
plot(t, est_a(:, 1), 'k', t, est_a(:, 2), 'k', t, est_a(:, 3), 'k');
axis tight;

print -djpeg hw12_2f;
function newState = KalmanFilter(state, measure)
    % Syntax: newState = KalmanFilter(state, measure)
    
    % state - structure for state equation including
    %        state.a  : state variable;
    %        state.phi: system matrix;
    %        state.Q  : covariance for state Gaussian noise
    %            at time i-1;
    %        state.K  : gain;
    %        state.P  : covariance matrix for prediction
    %            error, i.e. P'_i;
    %        state.PP : covariance for estimation error,
    %            i.e. P_i-1.
    % measure - structure for measure equation including
    %        measure.x: measure variable;
    %        measure.H: system transfer matrix;
    %        measure.R: covariance for measure
    %            Gaussian noise
    
    % Description: Kalman filter
    %
    % Author: Haiying Liu
    % Date: Dec. 2, 2000
    %

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dbstop if error

msg = nargchk(2, 2, nargin);
if (~isempty(msg))
    error(strcat('ERROR:', msg));
end

clear msg;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initialize the new estimation structure by old one.
newState = state;

% Compute the covariance matrix for prediction error.
newState.P = state.phi * state.PP * (state.phi)' + state.Q;

% Compute Gain.
newState.K = state.P * (measure.H)' * ...
    inv(measure.H * state.P * (measure.H)' + measure.R);

% Predict the new state.
newState.a = state.phi * state.a + newState.K * ...
    (measure.x - measure.H * state.phi * state.a);

% Compute covariance for estimation error used for next round recursion.
newState.PP = (eye(size(newState.K, 1), size(measure.H, 2)) - ...