19/20 + 3

Zachary Harris
Homework #5
Optimal Control

1.

control input

Dynamics $\begin{cases} \dot{\Theta} = \omega - \beta - \eta_v \\ \dot{\beta} = \eta_u \end{cases}$ $\quad \eta_v \sim \mathcal{N}(0, \sigma_v^2) \quad ; \quad \eta_u \sim \mathcal{N}(0, \sigma_u^2)$

Measurement $\{ Z_k = \Theta_k + v_k \quad v_k \sim \mathcal{N}(0, \sigma_n^2)$

SOLN:

4/4

$$x = \begin{bmatrix} \Theta \\ \beta \end{bmatrix} \qquad \begin{aligned} \dot{x}_1 &= -x_2 + u - \eta_v \\ \dot{x}_2 &= \eta_u \end{aligned}$$

$$F = \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix} \quad G = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad L = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \text{ with } w = \begin{bmatrix} \eta_v \\ \eta_u \end{bmatrix}$$

$$Q_c' = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_u^2 \end{bmatrix}$$

$$\Phi_{k-1} = e^{F\Delta t} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix}$$ 1

$$\Gamma_{k-1} = \int_{t_{k-1}}^{t_k} \Phi(t_k, \tau) G(\tau)\, d\tau = \begin{bmatrix} \Delta t & -\Delta t^2/2 \\ 1 & \Delta t \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \Delta t \\ 1 \end{bmatrix}$$ 1

$$Q_{k-1} = \int_{t_{k-1}}^{t_k} \Phi(t_k, \tau) L(\tau) Q_c' L(\tau)^T \Phi^T(t_k, \tau)\, d\tau$$

$$\Phi(t_k, \tau) L(\tau) = \begin{bmatrix} 1 & -(t_k - \tau) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & -(t_k - \tau) \\ 0 & 1 \end{bmatrix}$$

$$L^T(\tau) \Phi^T(t_k, \tau) = \begin{bmatrix} -1 & 0 \\ -(t_k - \tau) & 1 \end{bmatrix}$$

$$Q_{k-1} = \int_{t_{k-1}}^{t_k} \begin{bmatrix} -1 & -(t_k - \tau) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_u^2 \end{bmatrix} \begin{bmatrix} -1 & 0 \\ -(t_k - \tau) & 1 \end{bmatrix} d\tau$$

$$= \int_{t_{k-1}}^{t_k} \begin{bmatrix} -\sigma_v^2 & -(t_k - \tau)\sigma_u^2 \\ 0 & \sigma_u^2 \end{bmatrix} \begin{bmatrix} -1 & 0 \\ -(t_k - \tau) & 1 \end{bmatrix} d\tau = \int_{t_{k-1}}^{t_k} \begin{bmatrix} \sigma_v^2 + (t_k - \tau)^2 \sigma_u^2 & -(t_k - \tau)\sigma_u^2 \\ -(t_k - \tau)\sigma_u^2 & \sigma_u^2 \end{bmatrix} d\tau$$

$$Q_{k-1} = \begin{bmatrix} \sigma_v^2 \Delta t + \sigma_u^2 \frac{\Delta t^3}{3} & -\frac{\Delta t^2}{2} \sigma_u^2 \\ -\frac{\Delta t^2}{2} \sigma_u^2 & \Delta t\, \sigma_u^2 \end{bmatrix}$$ 2

```
function f = dblint_kf_zjh

% EN530.603 Kalman filtering of the double integrator with position measurements
% Zachary Harris, auvgeek at gmail.com
% adapted from M. Kobilarov , marin(at)jhu.edu
```

#1.b

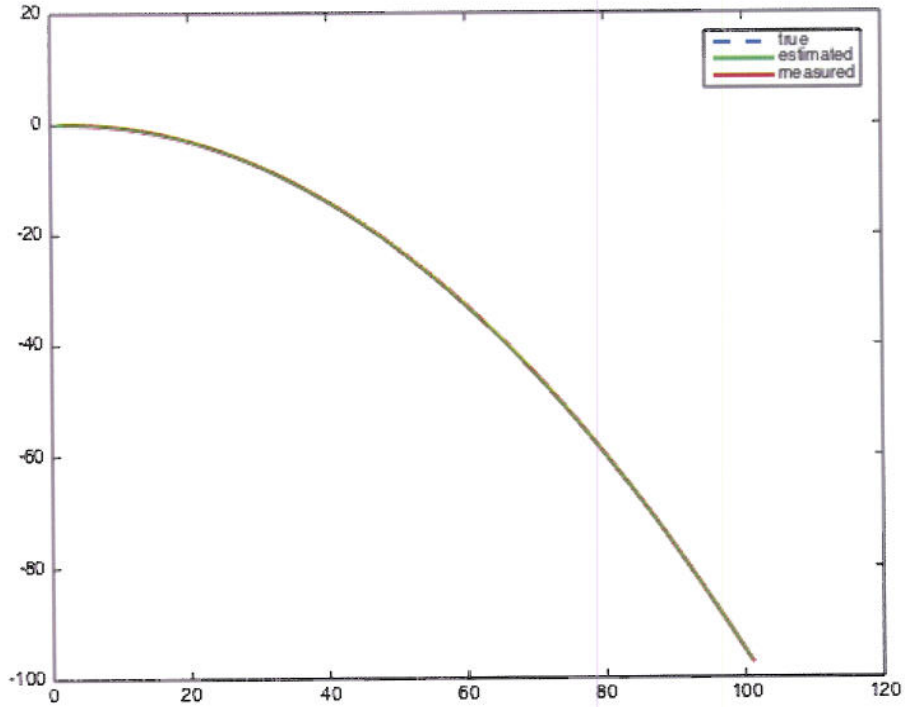
3/4

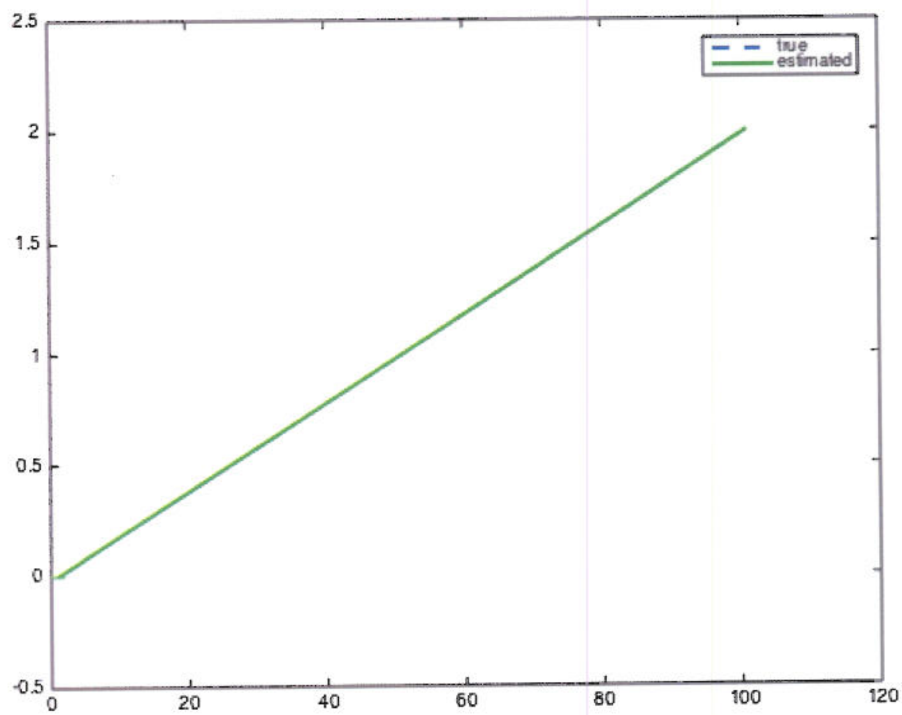
```
% timing
dt = 1;    % time-step
N = 100;    % total time-steps
T = N*dt;  % final time

% noise terms
S.v = (3E-6)^2; % external disturbance variance
S.u = (3E-9)^2; % external disturbance variance
S.n = (1.5E-5)^2;  % measurement noise variance

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

% Gamma matrix
S.Gamma = [dt; 1];

% Q matrix
S.Q = [dt^3/3*S.u+S.v*dt -dt^2/2*S.u;
         -dt^2/2*S.u dt*S.u];

% R matrix
S.R = S.n;

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

% initial estimate of mean and covariance
x = [0; 1.7E-7];
P = diag([1E-4,1E-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

bias = x(2);

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

```
for k=1:N
  %u = cos(k/N); % pick some known control
  bias = bias + dt*sqrt(S.u)*randn;
  u = 0.02+bias+sqrt(S.v)*randn;

  xts(:,k+1) = S.Phi*xts(:,k) + S.Gamma*u;  % true state
```

True state propagation not correct. Check Solution

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

  z = xts(1,k+1) + sqrt(S.n)*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

figure(1)
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')

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

legend('true', 'estimated')

% 95% confidence intervals of the estimated position
plot(xs(2,:) + 1.96*reshape(sqrt(Ps(2,1,:)),N+1,1)', '-g')
plot(xs(2,:) - 1.96*reshape(sqrt(Ps(2,1,:)),N+1,1)', '-g')
```


true
estimated
measured


true
estimated

```
stop = [];

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

x = S.Phi*x + S.Gamma*u;
P = S.Phi*P*S.Phi' + 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® R2014b

Zachary Harris

2. 4/4 Model:

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1} \qquad w_k \sim \mathcal{N}(0, Q_k)$$

$$z_k = h_k(x_k) + v_k \qquad v_k \sim \mathcal{N}(0, R_k)$$

$$x_{k|k-1} \sim \mathcal{N}(\hat{x}_{k|k-1}, P_{k|k-1})$$

Following the least-squares approach, show that the minimizer of the cost function

$$J(x) = \tfrac{1}{2}(x - \hat{x}_{k|k-1})^T P_{k|k-1}^{-1}(x - \hat{x}_{k|k-1}) + \tfrac{1}{2}[z_k - h(x)]^T R_k^{-1}[z_k - h(x)]$$

after linearizing the function $h(x)$ around $\hat{x}_{k|k-1}$ corresponds to the EKF correction

$$x = \hat{x}_{k|k-1} + K_k[z_k - h(\hat{x}_{k|k-1})]$$

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1}$$

SOLUTION:

To linearize $h(x)$, we simply use $h(x_k) \cong h(\hat{x}_{k|k-1}) + \underbrace{\partial_x h(\hat{x}_{k|k-1})}_{H_k}(x_k - \hat{x}_{k|k-1})$

Thus, our measurement model becomes:

$$z_k = h(\hat{x}_{k|k-1}) + H_k(x_k - \hat{x}_{k|k-1}) + v_k$$

The cost function becomes (dropping all subscripts):

$$J(x) = \tfrac{1}{2}(x - \hat{x})^T P^{-1}(x - \hat{x}) + \tfrac{1}{2}[z_k - h(\hat{x}) - H_k(x - \hat{x})]^T R^{-1}[z_k - h(\hat{x}) - H(x - \hat{x})]$$

The necessary condition for optimality is $\nabla_x J = 0$. Thus, we obtain:

$$\nabla J = P^{-1}(x - \hat{x}) - H^T R^{-1}[z - h(\hat{x}) - H(x - \hat{x})] = 0$$

Solving for x, we obtain

$$-(P^{-1}x + H^T R^{-1} H x) = -P^{-1}\hat{x} - H^T R^{-1}[z - h(\hat{x}) - H\hat{x}]$$

$$x = -[P^{-1} + H^T R^{-1} H]^{-1}[-P^{-1}\hat{x} - H^T R^{-1}(z - h(\hat{x}) - H\hat{x})]$$

$$= \underbrace{[P^{-1} - H^T R^{-1} H]^{-1}[P^{-1} - H^T R^{-1} H]}_{I}\hat{x} + [P^{-1} - H^T R^{-1} H]^{-1} H^T R^{-1}(z - h(\hat{x}))$$

$$x = \hat{x} + [P^{-1} + H^T R^{-1} H]^{-1} H^T R^{-1}(z - h(\hat{x}))$$

Thus, we can write: $x_k = \hat{x}_{k|k-1} + K_k[z_k - h(\hat{x}_{k|k-1})]$

Zachary Harris

2 (cont.)

Currently, we have $K_k = \left[P^{-1} + H^T R^{-1} H\right]^{-1} H^T R^{-1}$, but we rewrite it using the matrix inversion lemma:

$$K_k = \left[P - PH^T(HPH^T + R)^{-1} HP\right] H^T R^{-1}$$
$$= PH^T\left[I - (HPH^T + R)^{-1} HPH^T\right] R^{-1}$$
$$= PH^T(HPH^T + R)^{-1}\left[(HPH^T + R) - HPH^T\right] R^{-1}$$
$$K_k = PH^T(HPH^T + R)^{-1}$$

Now, we check the sufficient condition for optimality: $\nabla_x^2 J$

$$\nabla_x^2 J = P^{-1} + H^T R^{-1} H,$$

which must be positive definite for $\hat{x}$ to be a minimum.

Zachary Harris

3.

a) $$\begin{bmatrix} x_k \\ y_k \\ \theta_k \\ r_k \end{bmatrix} = \begin{bmatrix} x_{k-1} + \Delta t \cos(\theta_{k-1}) r_{k-1} \Omega_{k-1} \\ y_{k-1} + \Delta t \sin(\theta_{k-1}) r_{k-1} \Omega_{k-1} \\ \theta_{k-1} + \Delta t \omega_{k-1} \\ r_{k-1} \end{bmatrix} + w_{k-1}$$

$$F = \begin{bmatrix} 1 & 0 & -\Delta t \sin\theta \, r \, \Omega & \Delta t \cos\theta \, \Omega \\ 0 & 1 & \Delta t \cos\theta \, r \, \Omega & \Delta t \sin\theta \, \Omega \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

b)

```
function f = uni_ekf_zjh

% EN530.603 Extended Kalman filtering of the unicycle with bearing and range
% measurements
% Zachary Harris, auvgeek at gmail.com
% adapted from M. Kobilarov , marin(at)jhu.edu
```

#3.a

```
rng('default')

S.f = @uni_f;   % mobile-robot dynamics
S.h = @br_h;    % bearing-reange sensing
S.n = 4;        % state dimension

% single beacon at (-2,2) : system is unobservable
S.pbs = [-2;
         2];    % beacon positions

% two beacons at (-2,2) and (2,2) : system is observable (two or more)
S.pbs = [-2, 2;
         2, 2];    % beacon positions


nb = size(S.pbs,2); % number of beacons
S.r = 2*nb;       % measurement dimension

% timing
dt = .1;
%N = 2580;
N = 50;
T = dt*N;
S.dt = dt;

% noise models
%S.Q = .3*dt*diag([.1 .1 .01]);
S.Q = (dt)^2*diag([0.01,0.01,0.01,0.0001]);
S.R = .4*diag(repmat([.1; .01], nb, 1));

% initial mean and covariance
%xt = [0; 0; pi/4]; % true state
%P = .2*diag([1 1 .1]) % covariance
xt = [0 0 0 1]'; %true state
P = diag([0.01,0.01,0.01,0.04]); %initial covariance
x = xt + sqrt(P)*randn(S.n, 1); % initial estimate with added noise

xts = zeros(S.n, N+1); % true states
xs = zeros(S.n, N+1);  % estimated statesß
Ps = zeros(S.n, S.n, N+1); % estimated covariances
ts = zeros(N+1,1); % times

zs = zeros(S.r, N);  % measurements
```

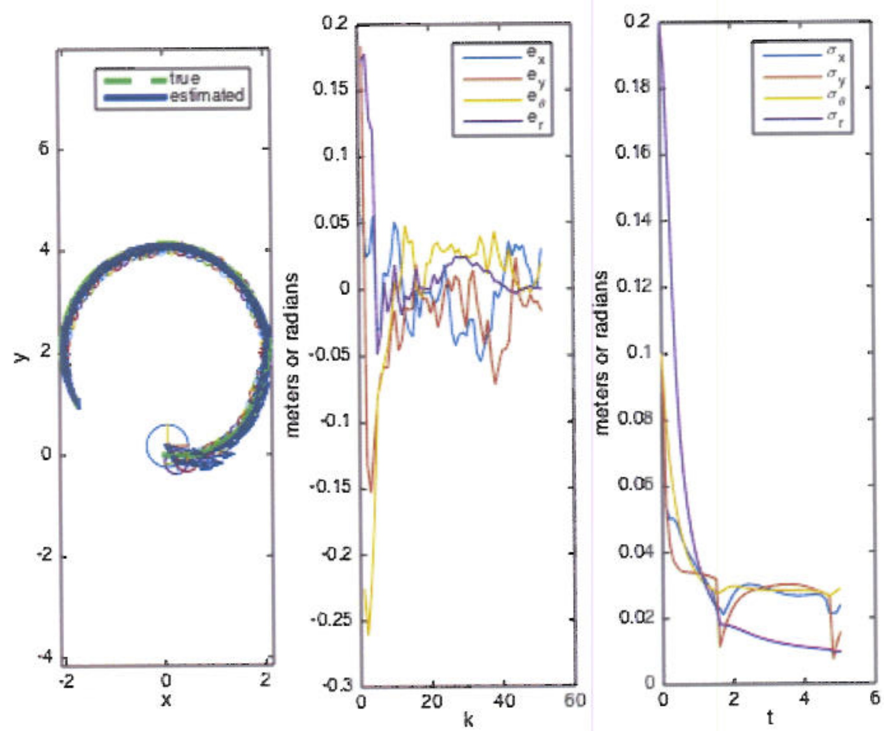
```
xts(:, 1) = xt;
xs(:, 1) = x;
Ps(:, :, 1) = P;
ts(1) = 0;

ds = zeros(S.n, N+1);  % errors
ds(:,1) = x - xt;

for k=1:N,
  u = dt*[2; 1];  % known controls

  xts(:,k+1) = S.f(xts(:,k), u, S) + sqrt(S.Q)*randn(S.n,1);  % true state

  [x,P] = ekf_predict(x, P, u, S);  % predict
  ts(k+1) = k*dt;

  z = S.h(xts(:,k+1), S) + sqrt(S.R)*randn(S.r,1);  % generate measurement
  [x,P] = ekf_correct(x, P, z, S);  % correct

  xs(:,k+1) = x;
  Ps(:,:,k+1) = P;

  zs(:,k) = z;
  ds(:,k+1) = x - xts(:,k+1);  % actual estimate error
  ds(:,k+1) = fix_state(ds(:,k+1));
end

subplot(1, 3, 1)

plot(xts(1,:), xts(2,:), '--g','LineWidth',3)
hold on
plot(xs(1,:), xs(2,:), '-b','LineWidth',3)
legend('true', 'estimated')

xlabel('x')
ylabel('y')
axis equal
axis xy

% beacon
plot(S.pbs(1,:), S.pbs(2,:), '*r');

for k=1:1:N
  plotcov2(xs(1:2,k), 1.96^2*Ps(1:2,1:2,k));
end
quiver(xts(1,:), xts(2,:), .5*cos(xts(3,:)), .5*sin(xts(3,:)),'g');
quiver(xs(1,:), xs(2,:), .5*cos(xs(3,:)), .5*sin(xs(3,:)),'b');

subplot(1,3,2)

plot(ds')

mean(sqrt(sum(ds.*ds, 1)))
```

```
xlabel('k')
ylabel('meters or radians')
legend('e_x','e_y','e_\theta','e_r')

subplot(1,3,3)

plot(ts, reshape(sqrt(Ps(1,1,:)),N+1,1), ...
     ts, reshape(sqrt(Ps(2,2,:)),N+1,1), ...
     ts, reshape(sqrt(Ps(3,3,:)),N+1,1), ...
     ts, reshape(sqrt(Ps(4,4,:)),N+1,1));
legend('\sigma_x','\sigma_y','\sigma_\theta','\sigma_r')

xlabel('t')
ylabel('meters or radians')
```

ans =

0.0683

4/4




```
stop = [];

function [x, varargout] = uni_f(x, u, S)
% dynamical model of the unicycle
c = cos(x(3));
s = sin(x(3));
```

```
x = [x(1) + c*x(4)*u(1);
     x(2) + s*x(4)*u(1);
     x(3) + u(2);
      x(4)];

x = fix_state(x, S);

if nargout > 1
  % F-matrix
  varargout{1} = [1, 0, -s*x(4)*u(1) c*x(4)*u(1);
                  0, 1, c*x(4)*u(1) s*x(4)*u(1);
                  0 0 1 0
                  0 0 0 1];
end


function [y, varargout] = br_h(x, S)

p = x(1:2);

y = [];
H = [];
for i=1:size(S.pbs, 2)
  pb = S.pbs(:, i); %i-th beacon
  d = pb - p;
  r = norm(d);

  th = fangle(atan2(d(2), d(1)) - x(3));
  y = [y; th; r];

  if nargout > 1
    % H-matrix
    H = [H;
         d(2)/r^2, -d(1)/r^2, -1 0;
         -d'/r, 0 0];
  end
end

if nargout > 1
  varargout{1} = H;
end


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

[x, F] = S.f(x, u, S);
x = fix_state(x, S);  % fix any [-pi,pi] issues
P = F*P*F' + S.Q;


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

[y, H] = S.h(x, S);
```

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

e = z - y;
e = fix_meas(e, S);  % fix any [-pi,pi] issues
x = x + K*e;


function x = fix_state(x, S)
x(3) = fangle(x(3));

function z = fix_meas(z, S)
for i=1:size(S.pbs,2)
  z(2*i-1) = fangle(z(2*i-1));
end

function a = fangle(a)
% make sure angle is between -pi and pi
a = mod(a,2*pi);
if a < -pi
  a = a + 2*pi;
else
  if a > pi
    a = a - 2*pi;
  end
end
```

Published with MATLAB® R2014b

```
function f = uni_ekf_test

% EN530.603 Extended Kalman filtering of the unicycle with bearing measurement
% Zachary Harris, auvgeek at gmail.com
% adapted from M. Kobilarov , marin(at)jhu.edu
```

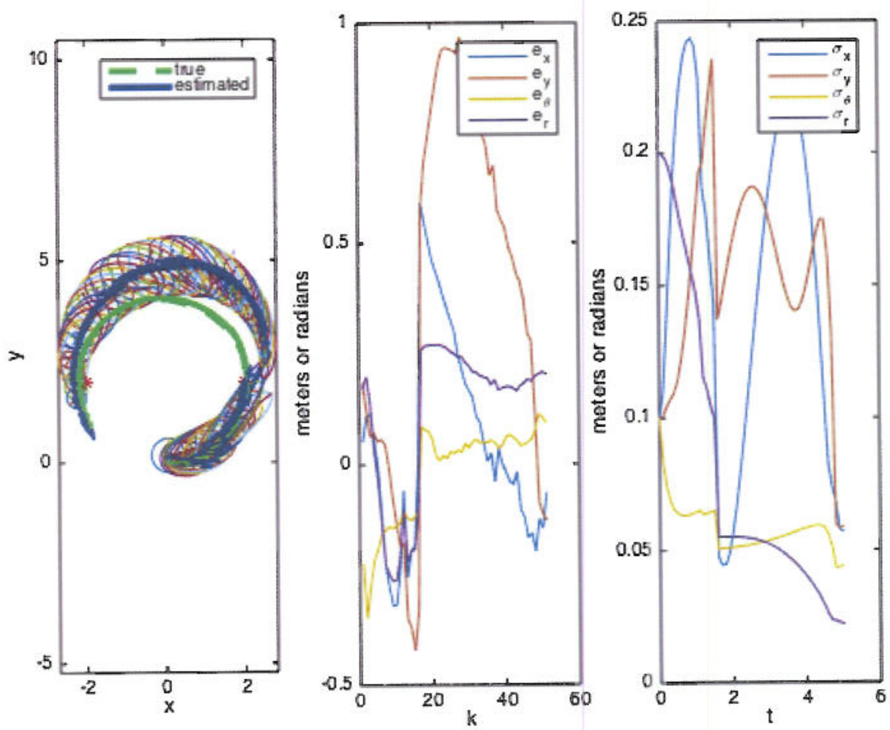
#3.b

```
rng('default')

S.f = @uni_f;  % mobile-robot dynamics
S.h = @br_h;   % bearing-reange sensing
S.n = 4;       % state dimension

% single beacon at (-2,2) : system is unobservable
S.pbs = [-2;
         2];    % beacon positions

% two beacons at (-2,2) and (2,2) : system is observable (two or more)
S.pbs = [-2, 2;
             2, 2];    % beacon positions

nb = size(S.pbs,2); % number of beacons
S.r = 1*nb;       % measurement dimension

% timing
dt = .1;
%N = 2580;
N = 50;
T = dt*N;
S.dt = dt;

% noise models
%S.Q = .3*dt*diag([.1 .1 .01]);
S.Q = (dt)^2*diag([0.01,0.01,0.01,0.0001]);
S.R = .4*diag(repmat([.1], nb, 1));

% initial mean and covariance
%xt = [0; 0; pi/4]; % true state
%P = .2*diag([1 1 .1]) % covariance
xt = [0 0 0 1]'; %true state
P = diag([0.01,0.01,0.01,0.04]); %initial covariance
x = xt + sqrt(P)*randn(S.n, 1); % initial estimate with added noise

xts = zeros(S.n, N+1); % true states
xs = zeros(S.n, N+1);  % estimated states
Ps = zeros(S.n, S.n, N+1); % estimated covariances
ts = zeros(N+1,1); % times

zs = zeros(S.r, N);  % measurements

xts(:, 1) = xt;
```

```matlab
xs(:, 1) = x;
Ps(:, :, 1) = P;
ts(1) = 0;

ds = zeros(S.n, N+1);  % errors
ds(:,1) = x - xt;

for k=1:N,
  u = dt*[2; 1];  % known controls

  xts(:,k+1) = S.f(xts(:,k), u, S) + sqrt(S.Q)*randn(S.n,1);  % true state

  [x,P] = ekf_predict(x, P, u, S);  % predict
  ts(k+1) = k*dt;

  z = S.h(xts(:,k+1), S) + sqrt(S.R)*randn(S.r,1);  % generate measurement
  [x,P] = ekf_correct(x, P, z, S);  % correct

  xs(:,k+1) = x;
  Ps(:,:,k+1) = P;

  zs(:,k) = z;
  ds(:,k+1) = x - xts(:,k+1);  % actual estimate error
  ds(:,k+1) = fix_state(ds(:,k+1));
end

subplot(1, 3, 1)

plot(xts(1,:), xts(2,:), '--g','LineWidth',3)
hold on
plot(xs(1,:), xs(2,:), '-b','LineWidth',3)
legend('true', 'estimated')

xlabel('x')
ylabel('y')
axis equal
axis xy

% beacon
plot(S.pbs(1,:), S.pbs(2,:), '*r');

for k=1:1:N
  plotcov2(xs(1:2,k), 1.96^2*Ps(1:2,1:2,k));
end
quiver(xts(1,:), xts(2,:), .5*cos(xts(3,:)), .5*sin(xts(3,:)),'g');
quiver(xs(1,:), xs(2,:), .5*cos(xs(3,:)), .5*sin(xs(3,:)),'b');

subplot(1,3,2)

plot(ds');

mean(sqrt(sum(ds.*ds, 1)))
xlabel('k')
ylabel('meters or radians')
```

```
legend('e_x','e_y','e_\theta','e_r')

subplot(1,3,3)

plot(ts, reshape(sqrt(Ps(1,1,:)),N+1,1), ...
     ts, reshape(sqrt(Ps(2,2,:)),N+1,1), ...
     ts, reshape(sqrt(Ps(3,3,:)),N+1,1), ...
     ts, reshape(sqrt(Ps(4,4,:)),N+1,1));
legend('\sigma_x','\sigma_y','\sigma_\theta','\sigma_r')

xlabel('t')
ylabel('meters or radians')
```

```
ans =

    0.6115
```



4/4

It is not possible to reconstruct the full state from bearing only measurements. The estimation error in the x and y DOF does not appear to converge towards zero.

```
stop = [];

function [x, varargout] = uni_f(x, u, S)
% dynamical model of the unicycle
c = cos(x(3));
```

```
s = sin(x(3));

x = [x(1) + c*x(4)*u(1);
     x(2) + s*x(4)*u(1);
     x(3) + u(2);
      x(4)];

x = fix_state(x, S);

if nargout > 1
  % F-matrix
  varargout{1} = [1, 0, -s*x(4)*u(1) c*x(4)*u(1);
                  0, 1, c*x(4)*u(1) s*x(4)*u(1);
                  0 0 1 0
                  0 0 0 1];
end


function [y, varargout] = br_h(x, S)

p = x(1:2);

y = [];
H = [];
for i=1:size(S.pbs, 2)
  pb = S.pbs(:, i); %i-th beacon
  d = pb - p;
  r = norm(d);

  th = fangle(atan2(d(2), d(1)) - x(3));
  y = [y; th];

  if nargout > 1
    % H-matrix
    H = [H;
         d(2)/r^2, -d(1)/r^2, -1 0;];
         %-d'/r, 0 0];
  end
end

if nargout > 1
  varargout{1} = H;
end


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

[x, F] = S.f(x, u, S);
x = fix_state(x, S);  % fix any [-pi,pi] issues
P = F*P*F' + S.Q;

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

[y, H] = S.h(x, S);
```

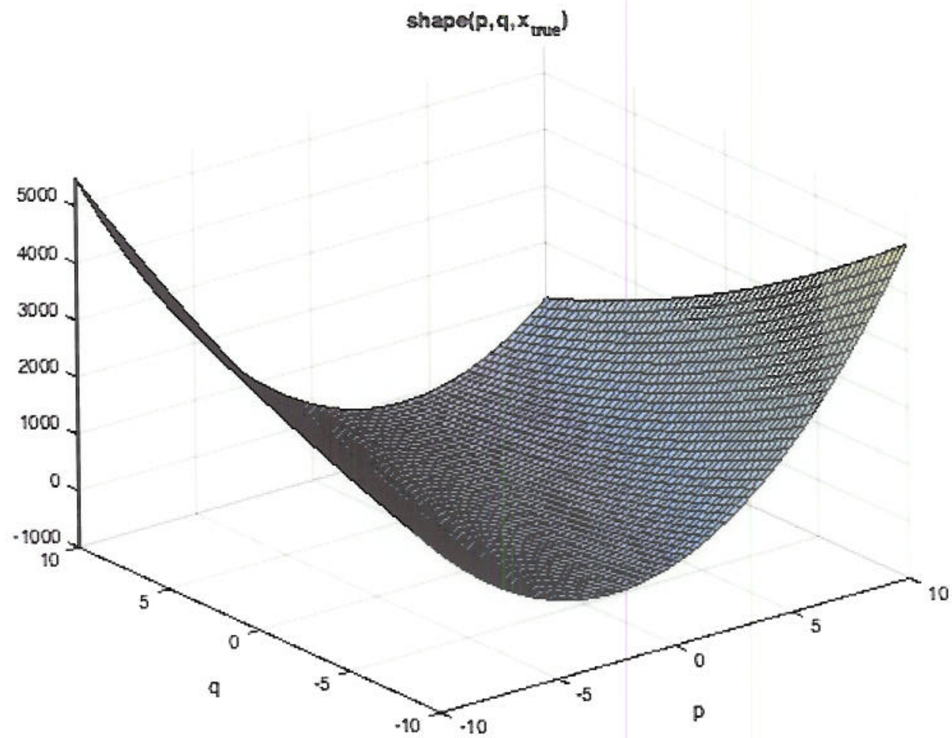
```
P = P - P*H'*inv(H*P*H' + S.R)*H*P;
K = P*H'*inv(S.R);

e = z - y;
e = fix_meas(e, S);  % fix any [-pi,pi] issues
x = x + K*e;

function x = fix_state(x, S)
x(3) = fangle(x(3));

function z = fix_meas(z, S)
for i=1:size(S.r,2)
  z(2*i-1) = fangle(z(2*i-1));
end

function a = fangle(a)
% make sure angle is between -pi and pi
a = mod(a,2*pi);
if a < -pi
  a = a + 2*pi;
else
  if a > pi
    a = a - 2*pi;
  end
end
```

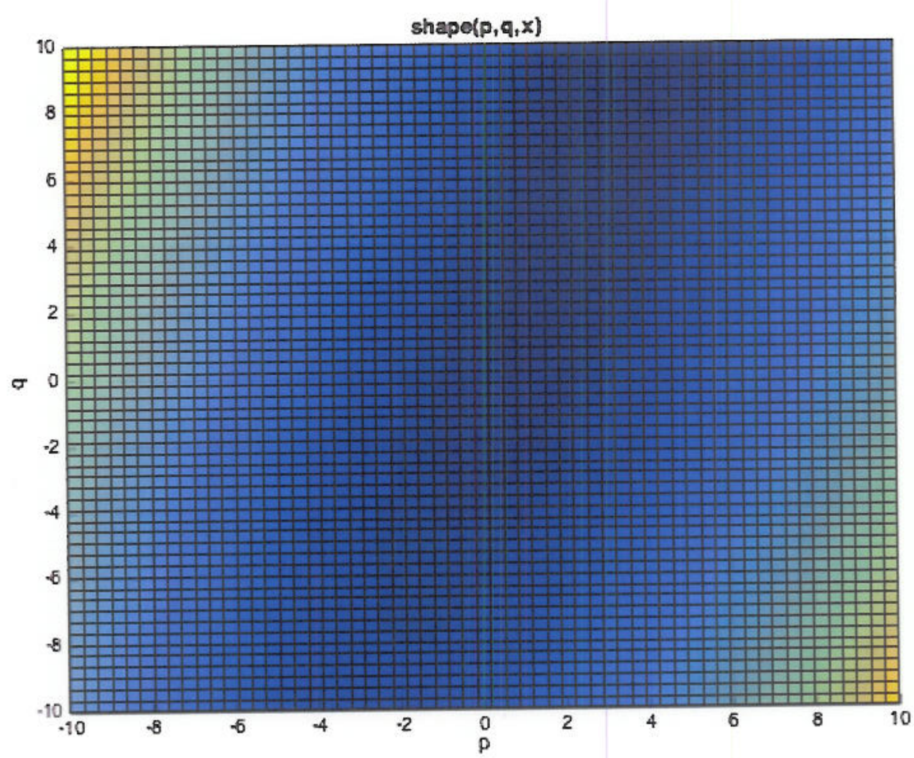
Published with MATLAB® R2014b

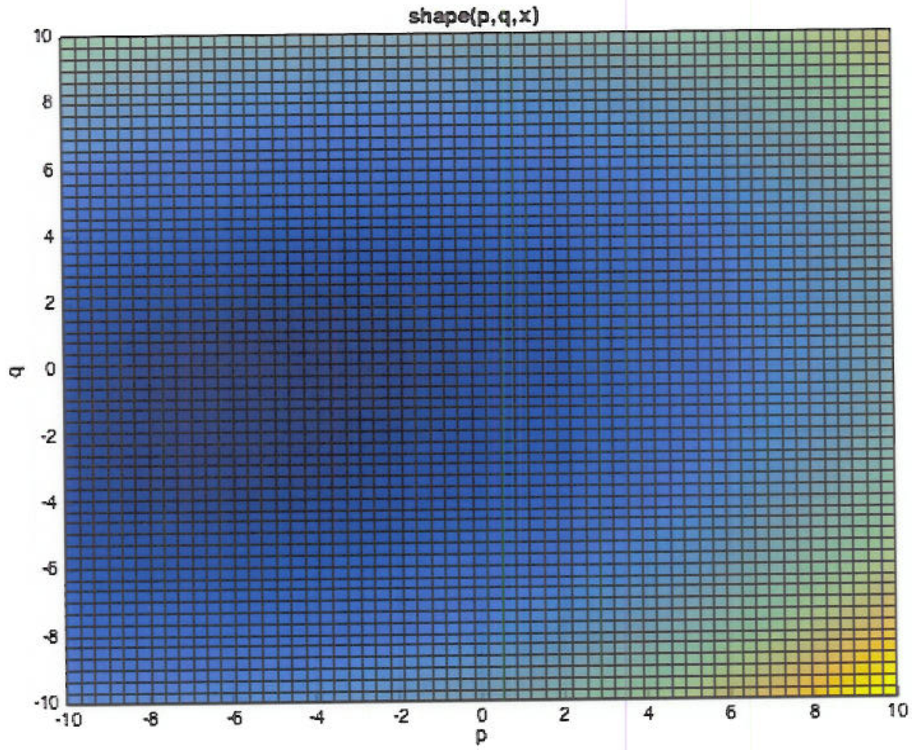
```
function f = shape_fit_zjh
% static batch estimation of a shape defined as a quadratic
% function z = f(p,q) + v, and parametrized using a vector x

% workspace is the square [-s,s]x[-s,s]
s = 10;

% true shape parameter (i.e. a symmetric cup)
%x_true = [1; 1; 0; 0; 0; 0];
m0 = [1.2 1.3 1 1 1 1]';
P0 = diag([16,16,16,16,16,16,]);
x_true = m0 + P0*randn(6,1);

% plot true
gt = ezsurf(@(p,q)shape(p, q, x_true),[-s,s]);
alpha(gt, 0.3)

% measurement standard dev
std = 20;

% #of measurements
k = 8;

% generate random measurements
p = 4*s*(rand(k,1) - .5);
q = 4*s*(rand(k,1) - .5);
z = shape(p, q, x_true) + randn(k,1)*std;

%
R = diag(repmat(std^2, k, 1));
H = shape_basis(p, q);

% estimate optimal parameters x
    x = inv(H'*inv(R)*H)*H'*inv(R)*z;
%

% plot estimated
figure(1)
hold on
ge = ezsurf(@(p,q)shape(p,q,x),[-s,s]);
[AZ,EL] = view;
alpha(ge, .8)


clear R H x

for i = 1:4


R = diag(repmat(std^2, 2, 1));
H = shape_basis(p(i:i+1), q(i:i+1));
```

```
% estimate optimal parameters x
x = inv(H'*inv(R)*H)*H'*inv(R)*z(i:i+1);

end

% plot estimated
figure(2)
hold on
ge = ezsurf(@(p,q)shape(p,q,x),[-s,s]);
view(AZ,EL);
alpha(ge, .8)


function f = shape_basis(p, q)
% quadratic function, although could be any shape
f = [p.^2, q.^2, p.*q, p, q, ones(size(p))];

function z = shape(p, q, x)
z = shape_basis(p, q)*x;


% This method of iteration results in the inversion of a singular matrix,
% so I suspec there is an error in the method. I expected the iterative
% weighting to produce a higher-quality fit compared to the batch
% estimation. From speculation, the noisier to prior, the more iterations
% required to converge to the answer. (Also: I cannot get the graph to
% publish correctly. It works fine when I run the code, but it won't view
% it in 3D in the "publish" feature.)
```

```
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND =  9.302014e-22.
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND =  4.033157e-21.
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND =  5.539355e-21.
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND =  3.307454e-21.
```


shape(p,q,x_{true})
5000
4000
3000
2000
1000
0
-1000
10
5
0
q
-5
-10
-10
-5
0
p
5
10


shape(p,q,x)
10
8
6
4
2
q
0
-2
-4
-6
-8
-10
-10
-8
-6
-4
-2
0
p
2
4
6
8
10



Published with MATLAB® R2014b

Explanation which one is better for Q4?
3/4