17/20 + 3 Extra Credit

HW 5

ABHINAV KUNAPAREDDY

4/4

① $\dot{x} = F x(t) + G u(t) + L w(t)$

$$\Rightarrow x(t_k) = \Phi(t_k, t_{k-1})\, x(t_{k-1}) + \int_{t_{k-1}}^{t_k} \Phi(t_k, \tau)\left[G u(\tau) + L w(\tau)\right] d\tau$$

$$= \Phi(t_k, t_{k-1})\, x(t_{k-1}) + \int_{t_{k-1}}^{t_k} e^{F(t_k - \tau)}\left[G u(\tau) + L w(\tau)\right] d\tau$$

Now assuming $u, w,$ are are constant in the interval,

$$= \Phi(t_k, t_{k-1})\, x(t_{k-1}) + \int_{t_{k-1}}^{t_k} e^{F(t_k - \tau)}\left(G u_{k-1} + L w_{k-1}\right) d\tau$$

$$= \Phi_{k-1}\, x_{k-1} + \int_{0}^{t_k - t_{k-1}} e^{F(t_k - t_{k-1})}\, e^{F(-\tau)}\left(G u_{k-1} + L w_{k-1}\right) d\tau$$

$$= \Phi_{k-1}\, x_{k-1} + e^{F\Delta t}\left[\int_0^{-\Delta t} e^{-F\tau}\, d\tau\right]\left(G u_{k-1} + L w_{k-1}\right)$$

∴ The discrete dynamics is given by,

$$x_k = \Phi_{k-1} x_{k-1} + \Gamma_{k-1} u_{k-1} + \Lambda_{k-1} w_{k-1}$$

where $\Phi_{k-1} = \Phi(t_k, t_{k-1})$

$$= e^{F\Delta t}$$

$$\Gamma_{k-1} = e^{F\Delta t}\left[\int_0^{\Delta t} e^{-F\tau}\, d\tau\right] G$$

$$\Lambda_{k-1} = e^{F\Delta t}\left[\int_0^{\Delta t} e^{-F\tau}\, d\tau\right] L$$

Now

$$\left.\begin{aligned} \dot{\theta} &= \omega - \beta - \eta_v \\ \dot{\beta} &= \eta_w \end{aligned}\right\} \Rightarrow \begin{bmatrix}\dot{\theta}\\ \dot{\beta}\end{bmatrix} = \begin{bmatrix}0 & -1\\ 0 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \beta\end{bmatrix} + \begin{bmatrix}1\\ 0\end{bmatrix}\omega + \begin{bmatrix}-1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\eta_v\\ \eta_w\end{bmatrix}$$

$$\therefore F = \begin{bmatrix}0 & -1\\ 0 & 0\end{bmatrix} \quad G = \begin{bmatrix}1\\ 0\end{bmatrix} \quad L = \begin{bmatrix}-1 & 0\\ 0 & 1\end{bmatrix}$$

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

$$= \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix}$$ 1

$$\Gamma_{k-1} = e^{F\Delta t} \left[\int_0^{\Delta t} e^{-F\tau} \, d\tau \right] G$$

$$= \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \left[\int_0^{\Delta t} \begin{bmatrix} 1 & +\tau \\ 0 & 1 \end{bmatrix} d\tau \right] G$$

$$= \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta t & \frac{\Delta t^2}{2} \\ 0 & \Delta t \end{bmatrix} G$$

$$= \begin{bmatrix} \Delta t & -\frac{\Delta t^2}{2} \\ 0 & \Delta t \end{bmatrix}_{2\times 2} \begin{bmatrix} 1 \\ 0 \end{bmatrix}_{2\times 1} = \begin{bmatrix} \Delta t \\ 0 \end{bmatrix}$$ 1

$$\Lambda_{k-1} = \begin{bmatrix} \Delta t & -\frac{\Delta t^2}{2} \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -\Delta t & -\frac{\Delta t^2}{2} \\ 0 & \Delta t \end{bmatrix}$$

- Discrete dynamics of system is

$$x_k = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} x_{k-1} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} u_{k-1} + \begin{bmatrix} -\Delta t & -\frac{\Delta t^2}{2} \\ 0 & \Delta t \end{bmatrix} w_{k-1}$$

i.e, $$\begin{bmatrix} \theta_k \\ \beta_k \end{bmatrix} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta_{k-1} \\ \beta_{k-1} \end{bmatrix} + \begin{bmatrix} \Delta t \\ 0 \end{bmatrix} w_{k-1} + \begin{bmatrix} -\Delta t & -\frac{\Delta t^2}{2} \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} \eta_{\theta\, k-1} \\ \eta_{\omega\, k-1} \end{bmatrix}$$

Mean evolves according to

$$\hat{x}_k = \bar{\Phi}_{k-1}\, \hat{x}_{k-1} + \Lambda_{k-1}\, u_{k-1}$$

And covariance as P_a

$$P_k = \Phi_{k-1} P_{k-1} \Phi_{k-1}^T + Q_{k-1}$$

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

$$= \int_{t_{k-1}}^{t_k} e^{F(t_k - \tau)} L Q_c' \left(e^{F(t_k - \tau)} \cdot L \right)^T d\tau$$

$$= \int_0^{\Delta t} e^{F(t_k - \tau' - t_{k-1})} L Q_c' \left(e^{F(t_k - \tau' - t_{k-1})} L \right)^T d\tau'$$

$$= \int_0^{\Delta t} e^{F\Delta t} e^{-F\tau'} L Q_c' L^T \left(e^{F\Delta t} e^{-F\tau'} \right)^T d\tau'$$

$$= e^{F\Delta t} \left[\int_0^{\Delta t} e^{-F\tau'} L Q_c' L^T \left(e^{-F\tau'} \right)^T d\tau' \right] \left(e^{F\Delta t} \right)^T$$

where $Q_c' = E\left(\begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} \begin{bmatrix} \eta_1 & \eta_2 \end{bmatrix} \right)$

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

$$\therefore \theta_{k-1} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \left(\int_0^{\Delta t} \begin{bmatrix} \sigma_v^2 T^2 + \sigma_v^2 & \sigma_v^2 T \\ \sigma_v^2 T & \sigma_v^2 \end{bmatrix} dT \right) \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sigma_v^2 \left(\frac{\Delta t^3}{3}\right) + \sigma_v^2 (\Delta t) & \sigma_v^2 \left(\frac{\Delta t^2}{2}\right) \\ \frac{\sigma_v^2 (\Delta t)^2}{2} & \sigma_v^2 \Delta t \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\Delta t & 1 \end{bmatrix}$$

$$= \frac{\Delta t}{6} \begin{bmatrix} 2\left(\sigma_v^2 \Delta t^2 + 3\sigma_v^2\right) & -3\sigma_v^2 \Delta t \\ -3\sigma_v^2 \Delta t & 6\sigma_v^2 \end{bmatrix}$$

(2)

3/4

$$x_k = f(x_{k-1}, u_{k-1}) + w_{k-1}$$

$$z_k = h_k(x_k) + v_k$$

$$w_k \sim \mathcal{N}(0, Q_k)$$

$$v_k \sim \mathcal{N}(0, R_k)$$

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

To minimize $J = \frac{1}{2}(x - \hat{x})^T P^{-1} (x - \hat{x}) + \frac{1}{2}(z_k - h(x))^T R_k^{-1} (z_k - h(x))$

$z = h(x) + v$

$\Rightarrow \Delta z = H \Delta x + v$ where $H = \frac{\partial h}{\partial x}$

linearizing $h(x)$ around $\hat{x}$ gives

$$h(x) \approx h(\hat{x}) + \underbrace{\frac{\partial h}{\partial x}\Big|_{x=\hat{x}}}_{H_k} (x - \hat{x})$$

$$\therefore h(x) \approx h(\hat{x}) + H_k (x - \hat{x})$$

Minimize,

$$\therefore J \approx \frac{1}{2}(x-\hat{x})^T P^{-1}(x-\hat{x}) + \frac{1}{2}\left[z_k - h(\hat{x}) - H_k(x-\hat{x})\right]^T R_k^{-1} \left[z_k - h(\hat{x}) - H_k(x-\hat{x})\right]$$

$$\Delta J = 0$$

$$\Rightarrow P^{-1}(x-\hat{x}) - H_k^T R_k^{-1}\left(z_k - h(\hat{x}) + H_k\hat{x} - H_k x\right) = 0$$

$$\Rightarrow P^{-1}(x-\hat{x}) - H_k^T R_k^{-1}\left(z_k - h(\hat{x}) + H_k\hat{x}\right)$$

$$\Rightarrow \left(P^{-1} + H_k^T R_k^{-1} H_k\right)x = P^{-1}\hat{x} + H_k^T R_k^{-1}\left(z_k - h(\hat{x})\right) + H_k^T R_k^{-1} H_k\,\hat{x}$$

$$\Rightarrow \left(P^{-1} + H_k^T R_k^{-1} H_k\right)(x-\hat{x}) = H_k^T R_k^{-1}\left(z_k - h(\hat{x})\right)$$

$$\Rightarrow x - \hat{x} = \left(P^{-1} + H_k^T R_k^{-1} H_k\right)^{-1} H_k^T R_k^{-1}\left(z_k - h(\hat{x})\right)$$

$(A+B)^{-1} \neq A^{-1} + B^{-1}$

$$\left(P + (H_k)^{-1} R_k H_k^{-1}\right) H_k^T R_k^{-1}\left(z_k - h(\hat{x})\right)$$

$$= P H_k^T \left(H_k P H_k^T + R_k\right)^{-1}\left(z_k - h(\hat{x})\right)$$

$$\therefore x = \hat{x} + PH_k^T \left(H_k P H_k^T + R_k\right)^{-1} \left(z_k - h(\hat{x})\right)$$

i.e.,

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

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

(2) (3) (a)

4/4

$$\begin{pmatrix} x_{k-1} + \Delta t \cos(\theta_{k-1})\, q_{k-1}\, \Omega_{k-1} \\ y_{k-1} + \Delta t \sin(\theta_{k-1})\, q_{k-1}\, \Omega_{k-1} \\ \theta_{k-1} + \Delta t\, \omega_{k-1} \\ q_{k-1} \end{pmatrix}$$

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

From the plots attached, we can see that 'q' oscilates very close to it's true value '1' (between 1.0025 and 0.9875)

(b) The corresponding error plots are attached.

3/4

After ~~modify~~ modifying uni_br_nobs.m for bearing only measurements and implementing it in uni_ekf_test2.m, we get the output as even when only bearing is used, system is observable with ~~if~~ two beacons and unobservable with just one beacon

Can't Understand this !

(4) The corresponding plots are attached

3/4

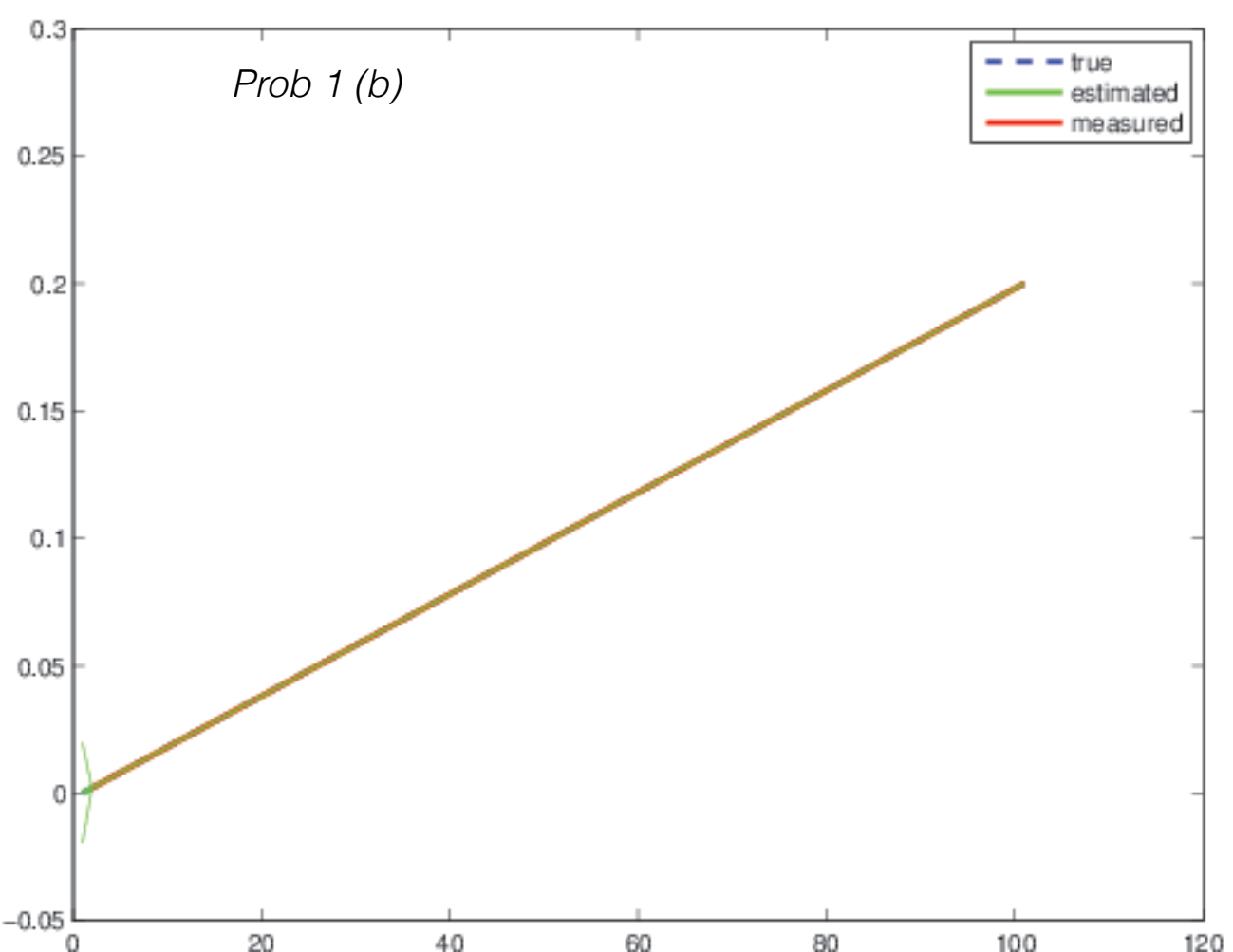
Compared to batch estimation, recursive estimation yields a significantly robust result when the prior values are reasonably accurate to the true value of x. If they are too noisy, the estimate is ~~at~~ not good, but with reasonable good prior values, the recursive estimator is significantly better.

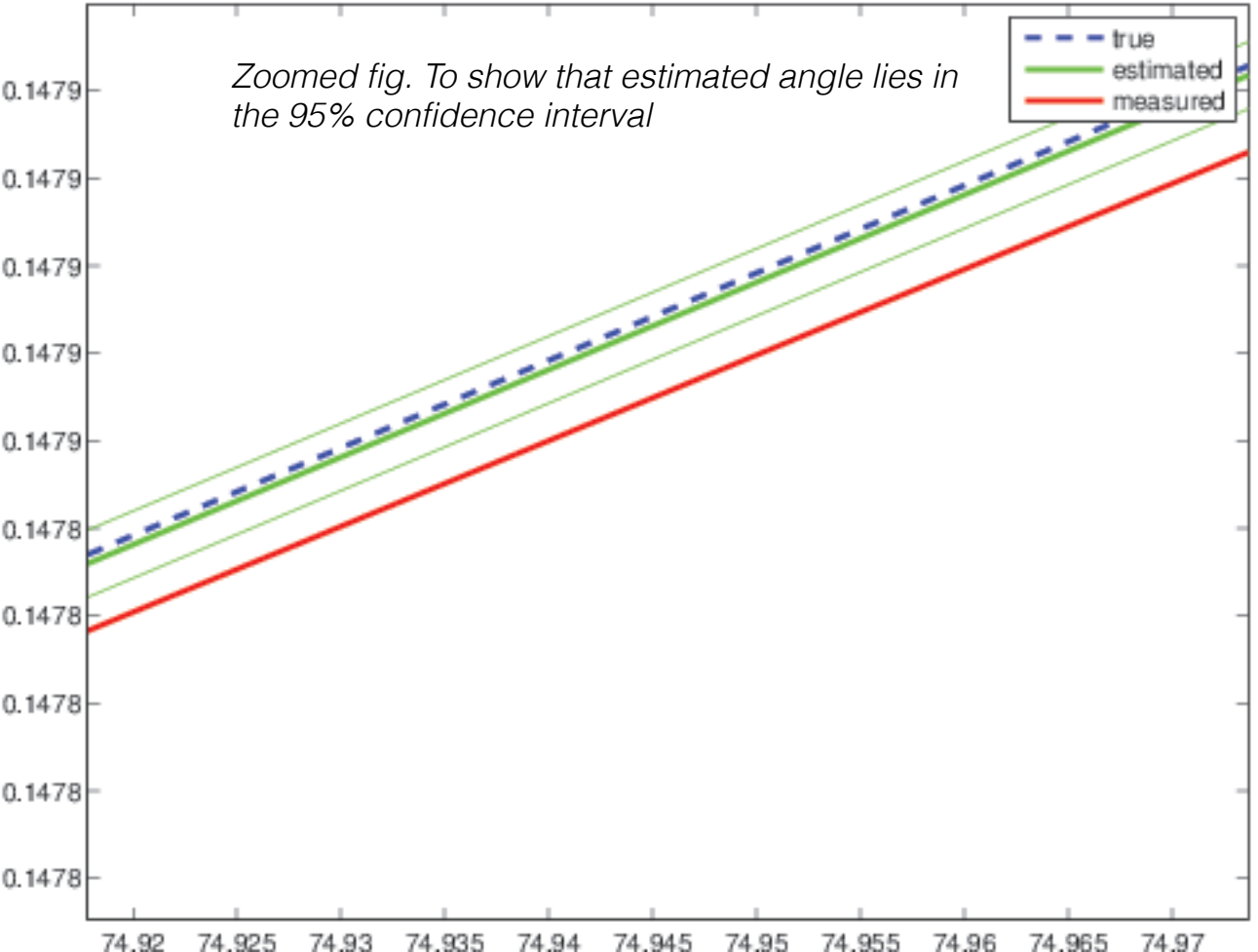
```
function f = att_kf_test
% Kalman filtering for single-axis attitude estimation
% timing
dt = .1;   % time-step
N = 100;   % total time-steps
T = N*dt;  % final time

% noise terms
S.q = [3e-06^2 0;0 3e-09^2];    % external disturbance variance
S.r = 1.5e-05^2;    % measurement noise variance

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

% G matrix
S.G = [dt;
       0];
S.L=[-dt -dt^2/2;0 dt];

b=S.q(4);a=S.q(1);
% Q matrix
S.Q =(dt/6)*[2*(b*dt^2+3*a), -3*b*dt;
          -3*b*dt, 6*b];

% R matrix
S.R = S.r;

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

% initial estimate of mean and covariance
x = [0; 1.7e-07];
P = diag([1e-04 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

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

for k=1:N
  u = 0.02+xs(2,k)+sqrt(a)*randn(1); % pick some known control

  xts(:,k+1) = S.F*xts(:,k) + S.G*u + S.L*[sqrt(a)*randn(1);sqrt(b)*randn(1)];  % 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
```

1b) 3/4

True estimate should be propagated differently
Check Soln -1

```
  % 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);
```


Prob 1 (b)
true
estimated
measured
0.3
0.25
0.2
0.15
0.1
0.05
0
−0.05
0
20
40
60
80
100
120


Zoomed fig. To show that estimated angle lies in the 95% confidence interval
true
estimated
measured
0.1479
0.1479
0.1479
0.1479
0.1479
0.1478
0.1478
0.1478
0.1478
0.1478
74.92
74.925
74.93
74.935
74.94
74.945
74.95
74.955
74.96
74.965
74.97

```
function f = uni_ekf_test1
% EN530.603 Extended Kalman filtering of the unicycle with bearing and range↙
measurements
%
% M. Kobilarov , marin(at)jhu.edu

clear

%rng('default')
rng(10212);

S.bearing_only = 0;

% 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

if S.bearing_only
  S.h = @b_h;    % bearing sensing
  S.r = nb;      % measurement dimension
  S.R = .4*diag(repmat([.1], nb, 1));
else
  S.h = @br_h;   % bearing-reange sensing
  S.r = 2*nb;      % measurement dimension
  S.R = .4*diag(repmat([.1; .01], nb, 1));
end

S.n = 4;        % state dimension
S.f = @uni_f;  % mobile-robot dynamics

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

% noise models
S.Q = .3*dt*diag([.01 .01 .01 .0001]);


% initial mean and covariance
xt = [0; 0; 0;1]; % true state

P = .01*diag([1 1 1 4]) ;% 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
```

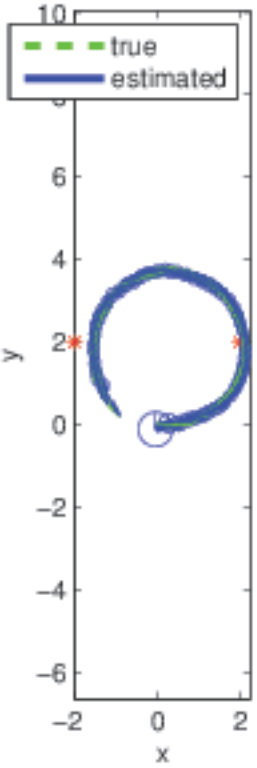
```
zs = zeros(S.r, N);  % measurements

xts(:, 1) = xt;
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, 4, 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(1,4,2)

plot(ds')

mean(sqrt(sum(ds.*ds, 1)))
xlabel('k')
ylabel('meters or radians')
legend('e_x','e_y','e_\theta','e_r')
```

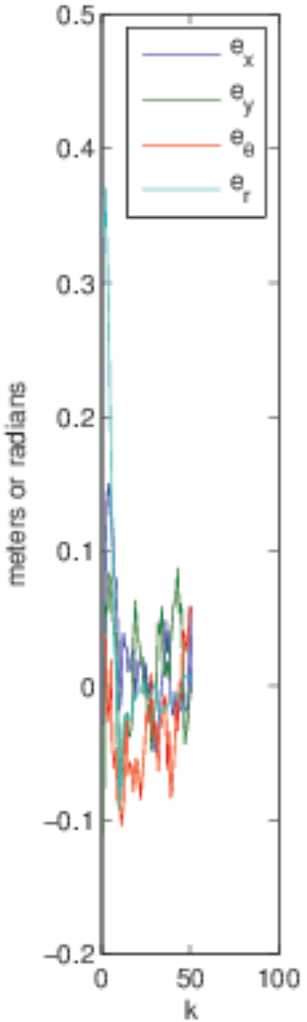
```
subplot(1,4,3)

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')

subplot(1,4,4)
plot(ts, xts(4,:), '--g','LineWidth',3);
xlabel('t')
ylabel('meters')




function [x, varargout] = uni_f(x, u, S)
% dynamical model of the unicycle
c = cos(x(3));
s = sin(x(3));
r=x(4);
x = [x(1) + c*r*u(1);
     x(2) + s*r*u(1);
     x(3) + u(2);
     x(4)];

x = fix_state(x, S);

if nargout > 1
  % F-matrix
  varargout{1} = [1, 0, -s*r*u(1), c*u(1);
                  0, 1, c*r*u(1), s*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
```

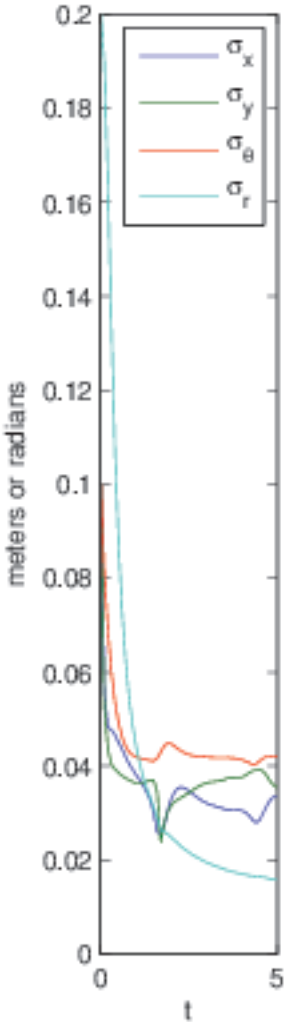
```
    H = [H;
         d(2)/r^2, -d(1)/r^2, -1,0;
         -d'/r, 0,0];
  end
end

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


function [y, varargout] = b_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];
  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));
```

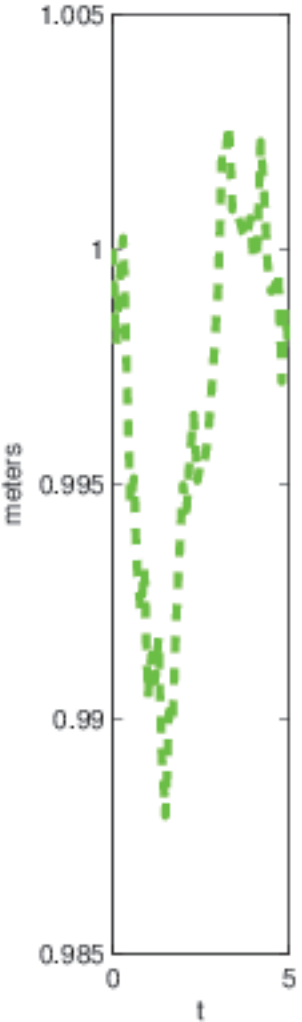
```
function z = fix_meas(z, S)
z;
S.r;
for i=1:size(S.pbs,2)
  if S.bearing_only
    z(i) = fangle(z(i));
  else
    z(2*i-1) = fangle(z(2*i-1));
  end
end

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
```

Prob 3 (a)


true
estimated
y
x


e_x
e_y
e_θ
e_r
meters or radians
k


σ_x
σ_y
σ_θ
σ_r
meters or radians
t


meters
t

```
function f = uni_ekf_test2
% Extended Kalman filtering of the unicycle with bearing only measurements
%
% M. Kobilarov , marin(at)jhu.edu

clear

%rng('default')
rng(10212)

S.bearing_only = 1;

% 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

if S.bearing_only
  S.h = @b_h;    % bearing sensing
  S.r = nb;      % measurement dimension
  S.R = .4*diag(repmat([.1], nb, 1));
else
  S.h = @br_h;   % bearing-reange sensing
  S.r = 2*nb;      % measurement dimension
  S.R = .4*diag(repmat([.1; .01], nb, 1));
end

S.n = 3;       % state dimension
S.f = @uni_f;  % mobile-robot dynamics

% timing
dt = .1;
%N = 2500;
N = 50;
T = dt*N;
S.dt = dt;

% noise models
S.Q = .3*dt*diag([.1 .1 .01]);


% initial mean and covariance
xt = [0; 0; pi/4]; % true state

P = .2*diag([1 1 .1]); % 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;
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')
```

```
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));
legend('\sigma_x','\sigma_y','\sigma_\theta')

xlabel('t')
ylabel('meters or radians')
disp(' When 1 beacon is used system is');
uni_br_nobs2(1);
disp(' When 2 beacons are used system is');

uni_br_nobs2(2);


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*u(1);
     x(2) + s*u(1);
     x(3) + u(2)];

x = fix_state(x, S);

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

if nargout > 1
```

```
  varargout{1} = H;
end


function [y, varargout] = b_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];
  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)
z;
S.r;
for i=1:size(S.pbs,2)
  if S.bearing_only
    z(i) = fangle(z(i));
```

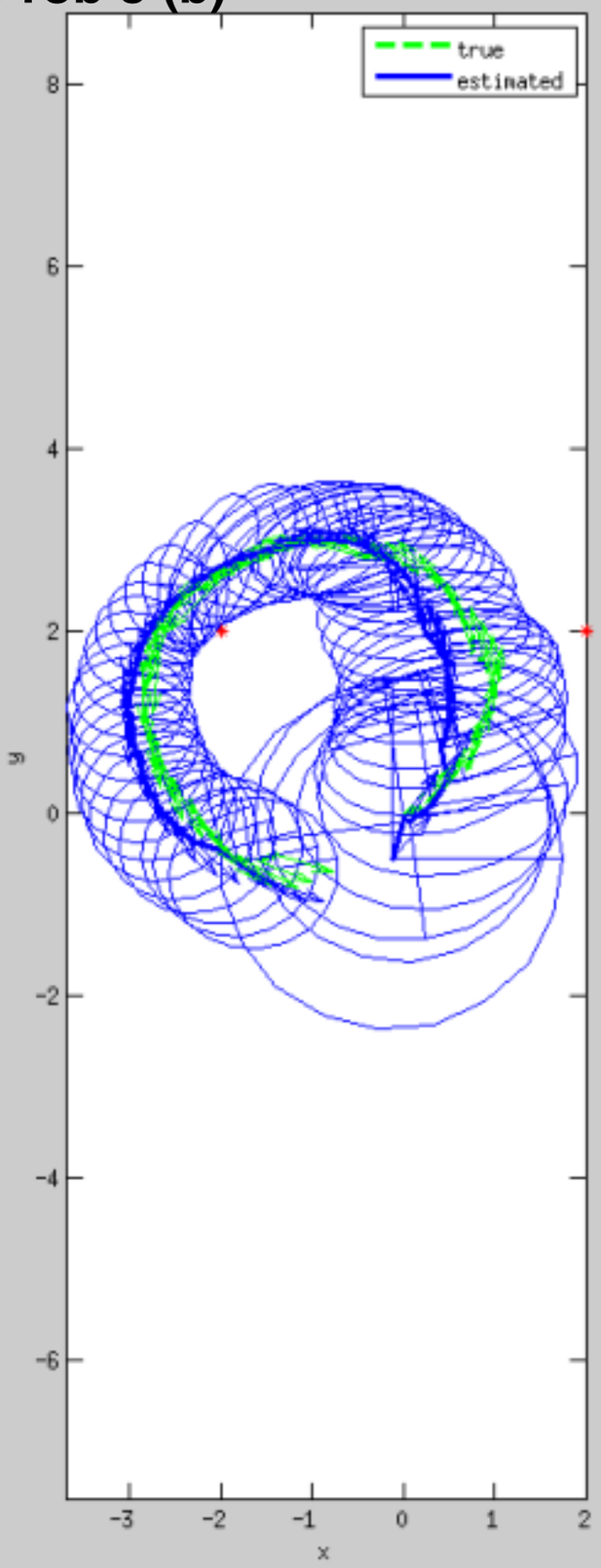
```
  else
    z(2*i-1) = fangle(z(2*i-1));
  end
end

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
```

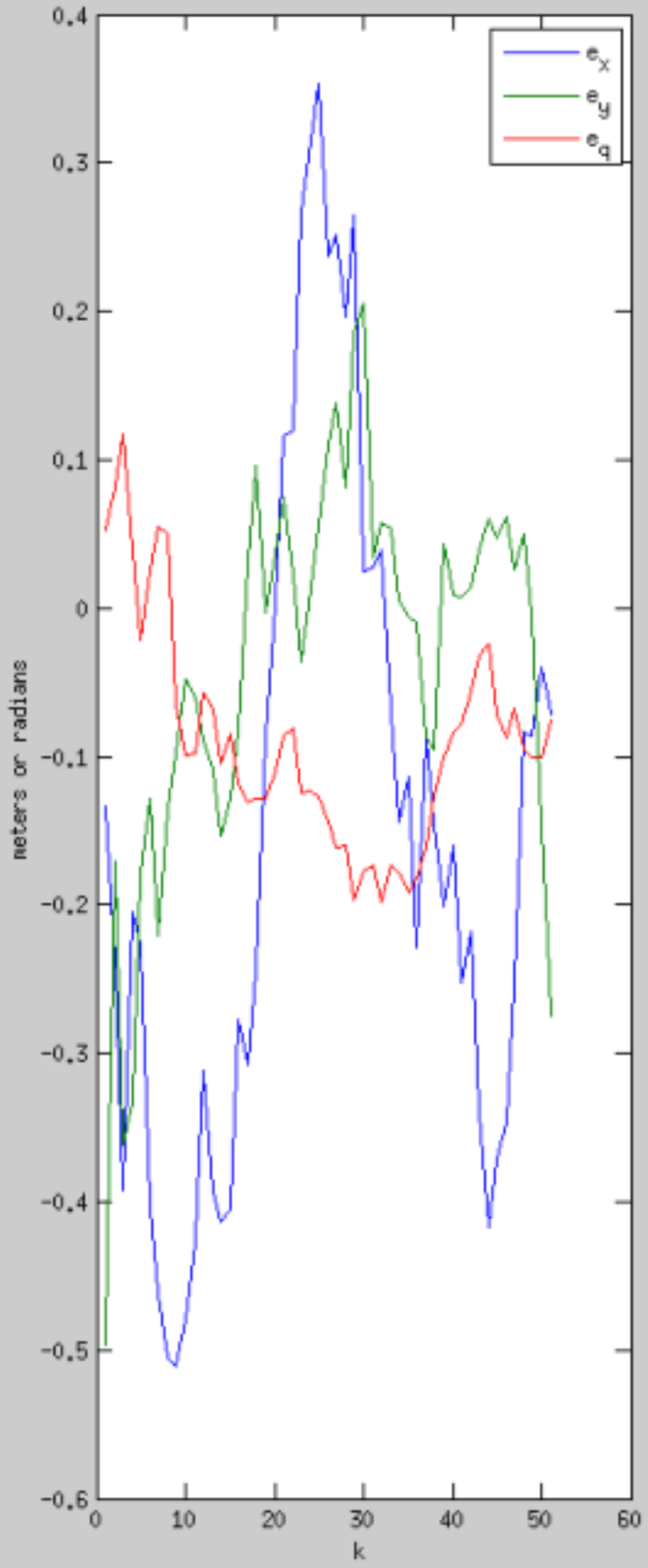
```
function rs = uni_br_nobs2(i)
% EN530.603 computes nonlinear observability of unicycle with range-bearing
% from beacons
% M. Kobilarov , marin(at)jhu.edu

%clear

% one beacon : ubosverbale
if(i==1)S.pbs = [5;
         5];    % beacon positions

%two beacons: observable
else if(i==2)S.pbs = [5, 3;
         5, 6];    % beacon positions
    end
end
syms px py th u1 u2 real

x = [px; py; th];
u = [u1; u2];

% unicycle with bearing-range
f = uni_f(x,u,S);
[z, H] = b_h(x, S);

nz = length(z);
nx = length(x);

dz = H*f;
l = [z; dz];

for i=1:3
  Dl = jacobian(l,x);

  A=simplify(subs(Dl, {px,py,th}, {1,2,pi/4}));

  % try symbolic rank and also with different values for controls
  rs = [  rank(A);
          rank(subs(A, {u1,u2}, {1,1}));
          rank(subs(A, {u1,u2}, {1,0}));
          rank(subs(A, {u1,u2}, {0,1}));
          rank(subs(A, {u1,u2}, {0,0}))]  ;

  if length(find(rs==nx))
    % rank is 3
    Dl;
    break
  end

  % keep adding time-derivatives of z
  dz = jacobian(dz, x)*f;
  l = [l; dz];
end

if max(int32(rs))==3
  disp('OBSERAVABLE')
else
  disp('UNOBSERAVABLE')
```

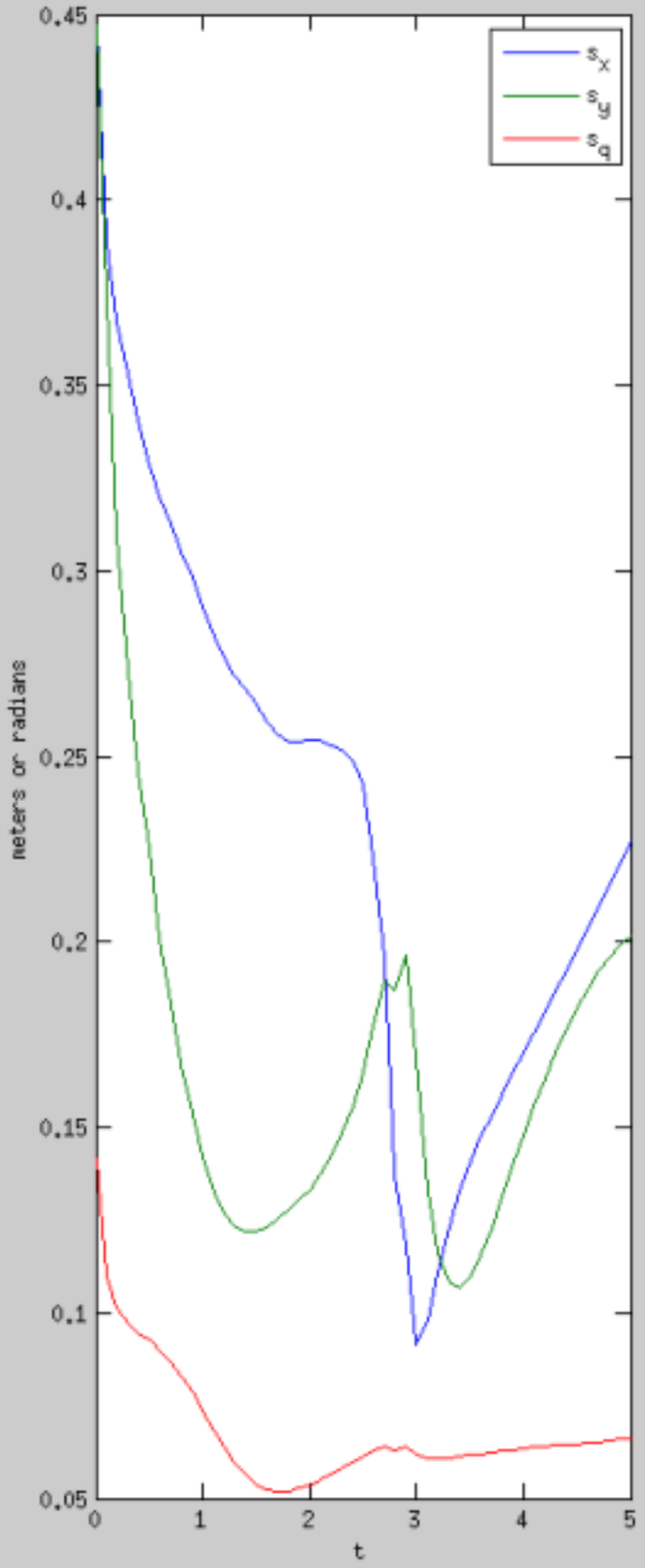
```
end



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

f = [c*u(1);
     s*u(1);
     u(2)];

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



function [y, varargout] = br_h(x, S)
% bearing-range from multiple beacons

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 = 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;
         -d'/r, 0];
  end
end

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



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%% some more models %%%%%%%%%%%%

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

p = x(1:2);
px = p(1);
```

```
py = p(2);

d = p;
r = norm(d);

th = atan2(d(2), d(1));
y = [th; r];

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



function [y, varargout] = b_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 = 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];
end
end
if nargout > 1
  varargout{1} = H;
end



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

p = x(1:2);
px = p(1);
py = p(2);

d = S.p0 - p;
r = norm(d);

y = r;

if nargout > 1
  % H-matrix
  varargout{1} = [-d'/r, 0];
end
```

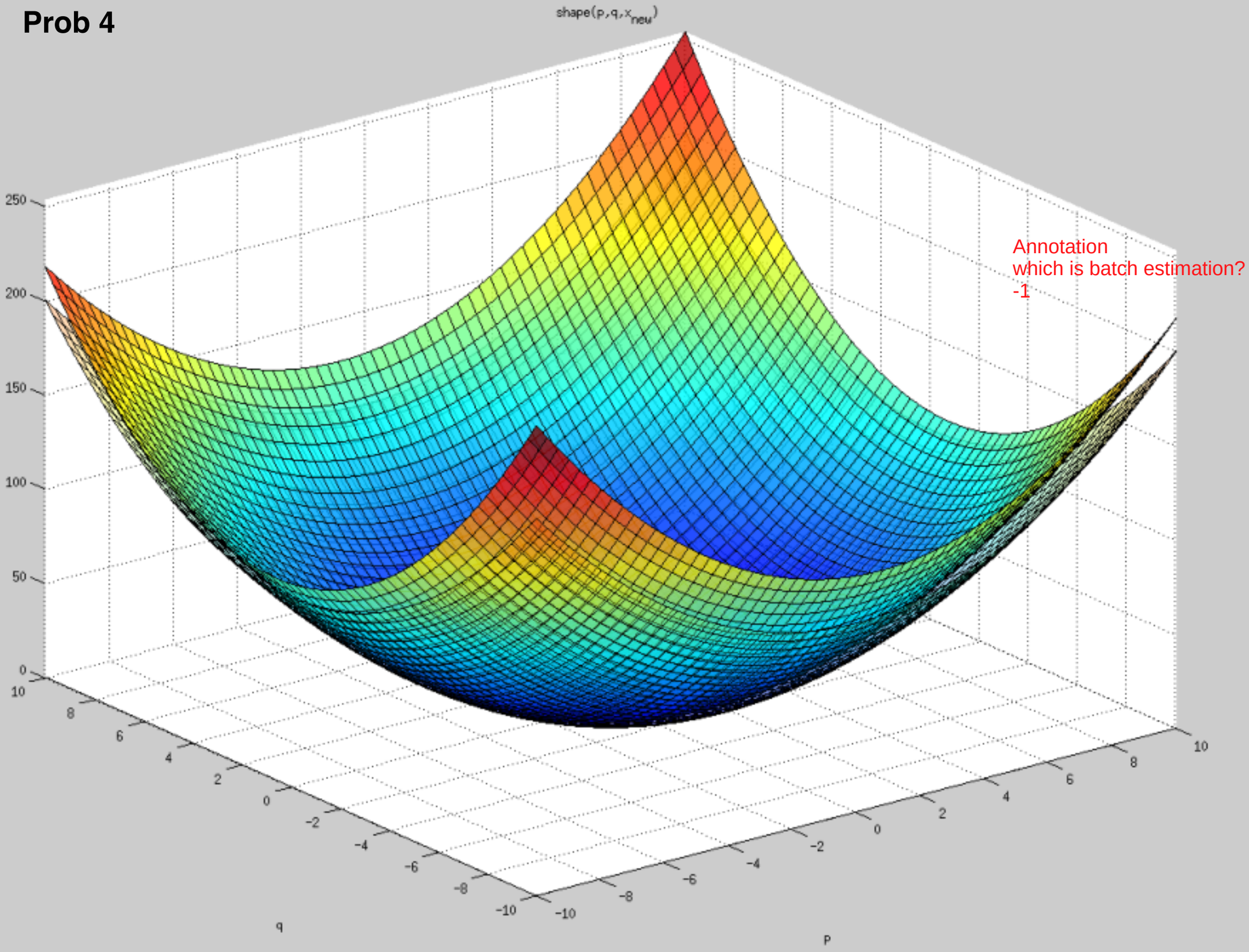
Prob 3 (b)


true
estimated
x
y


e_x
e_y
e_q
meters or radians
k


s_x
s_y
s_q
meters or radians
t

Prob 4



Annotation
which is batch estimation?
-1

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

close all;
% 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];

% 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_tot = shape(p, q, x_true) + randn(k,1)*std
R_tot = diag(repmat(std^2, k, 1));
H_tot = shape_basis(p, q);

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

P_old=diag([16,16,16,16,16,16]);
x_old=[1.2;1.3;.1;.1;.1;.1];

for i=1:4

    H{i}=shape_basis(p(2*i-1:2*i), q(2*i-1:2*i));
    R{i}=diag(repmat(std^2,2, 1));
    z{i}=z_tot(2*i-1:2*i);
    P_new=P_old-P_old*H{i}'*inv(H{i}*P_old*H{i}'+R{i})*H{i}*P_old;
    K{i}=P_new*H{i}'*inv(R{i});
    x_new=x_old+K{i}*(z{i}-H{i}*x_old) ;
end
x_new
% plot estimated
hold on
ge = ezsurf(@(p,q)shape(p,q,x_new),[-s,s]);
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;
```