19/20 + 4 Extra Credit

Applied Optimal Control (530.603) - Fall 2014

Karla Hernández

Homework 5

1. **a)** If we define $\boldsymbol{x}(t) \equiv [\theta(t), \beta(t)]^\top$ then we can write the dynamics as:

4/4

$$\dot{\boldsymbol{x}}(t) = \underbrace{\begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}}_{\equiv \boldsymbol{F}} \boldsymbol{x}(t) + \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\equiv \boldsymbol{G}} \omega(t) + \underbrace{\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}}_{\equiv \boldsymbol{L}} \underbrace{\begin{bmatrix} \eta_v \\ \eta_u \end{bmatrix}}_{\equiv \eta(t)},$$

then, using the notation $\boldsymbol{x}_k \equiv \boldsymbol{x}(t_k)$ we have:

$$\boldsymbol{x}_k = e^{(t_k - t_{k-1})\boldsymbol{F}} \boldsymbol{x}(t_{k-1}) + \int_{t_{k-1}}^{t_k} e^{(t_k - \tau)\boldsymbol{F}} [\boldsymbol{G}\omega(\tau) + \boldsymbol{L}\eta(\tau)] \, d\tau$$

Taking expectations (with respect to $\eta(t)$) and assuming the Gaussians are mean-zero we have:

$$\hat{\boldsymbol{x}}_k \equiv E[\boldsymbol{x}_k] = e^{(t_k - t_{k-1})\boldsymbol{F}} \boldsymbol{x}_{k-1} + \int_{t_{k-1}}^{t_k} e^{(t_k - \tau)\boldsymbol{F}} \boldsymbol{G}\omega(\tau) \, d\tau.$$

If we define Φ as the first order approximation of $e^{(t_k - t_{k-1})\boldsymbol{F}} = e^{\Delta t \boldsymbol{F}}$ we have:

$$\Phi \equiv \boldsymbol{I} + \Delta t \boldsymbol{F} = \begin{bmatrix} 1 & -\Delta t \\ 0 & 1 \end{bmatrix}$$ 1

we can obtain that for Δt small enough:

$$\hat{\boldsymbol{x}}_k = \Phi \boldsymbol{x}_{k-1} + \int_{t_{k-1}}^{t_k} \underbrace{\begin{bmatrix} 1 & (\tau - t_k) \\ 0 & 1 \end{bmatrix}}_{\Phi(t_k, \tau)} \boldsymbol{G}\omega(\tau) \, d\tau.$$

Because we are assuming $\omega(\tau)$ is constant in the sampling intervals we can define:

$$\Gamma \equiv \begin{bmatrix} \Delta t \\ 0 \end{bmatrix}, \quad \omega_k \equiv \omega(t_k),$$ 1

and obtain:

$$\hat{\boldsymbol{x}}_k = \boldsymbol{\Phi}\boldsymbol{x}_{k-1} + \boldsymbol{\Gamma}\omega_{k-1}. \tag{1}$$

Next we will determine how the variance evolves. First, we begin recalling the definition:

$$\boldsymbol{P}_0 \equiv E[(\hat{\boldsymbol{x}}_0 - \boldsymbol{x}_0)(\hat{\boldsymbol{x}}_0 - \boldsymbol{x}_0)^\top].$$

We know that:

$$\begin{aligned}
&\boldsymbol{P}_k \equiv E[(\hat{\boldsymbol{x}}_k - \boldsymbol{x}_k)(\hat{\boldsymbol{x}}_k - \boldsymbol{x}_k)^\top] \\
&= \boldsymbol{\Phi}\boldsymbol{P}_{k-1}\boldsymbol{\Phi}^\top + \int_{t_{k-1}}^{t_k} \begin{bmatrix} 1 & (\tau - t_k) \\ 0 & 1 \end{bmatrix} \boldsymbol{L} \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_u^2 \end{bmatrix} \boldsymbol{L}^\top \begin{bmatrix} 1 & (\tau - t_k) \\ 0 & 1 \end{bmatrix}^\top d\tau \\
&= \boldsymbol{\Phi}\boldsymbol{P}_{k-1}\boldsymbol{\Phi}^\top + \int_{t_{k-1}}^{t_k} \begin{bmatrix} 1 & (\tau - t_k) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_u^2 \end{bmatrix} \begin{bmatrix} 1 & (\tau - t_k) \\ 0 & 1 \end{bmatrix}^\top d\tau \\
&= \boldsymbol{\Phi}\boldsymbol{P}_{k-1}\boldsymbol{\Phi}^\top + \underbrace{\int_{t_{k-1}}^{t_k} \begin{bmatrix} \sigma_v^2 + (\tau - t_k)^2\sigma_u^2 & (\tau - t_k)\sigma_u^2 \\ (\tau - t_k)\sigma_u^2 & \sigma_u^2 \end{bmatrix} d\tau.}_{\boldsymbol{Q}_{k-1}}
\end{aligned}$$

For small Δt we can use:

$$\boldsymbol{Q}_{k-1} \approx \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_u^2 \end{bmatrix} \Delta t$$

Check soln for higher order terms

and in general the variance evolves according to:

$$\boldsymbol{P}_k = \boldsymbol{\Phi}\boldsymbol{P}_{k-1}\boldsymbol{\Phi}^\top + \boldsymbol{Q}_{k-1}. \tag{2}$$

4/4

b) The output can be seen in figure (1). A printout of the code is included at the end of the homework.

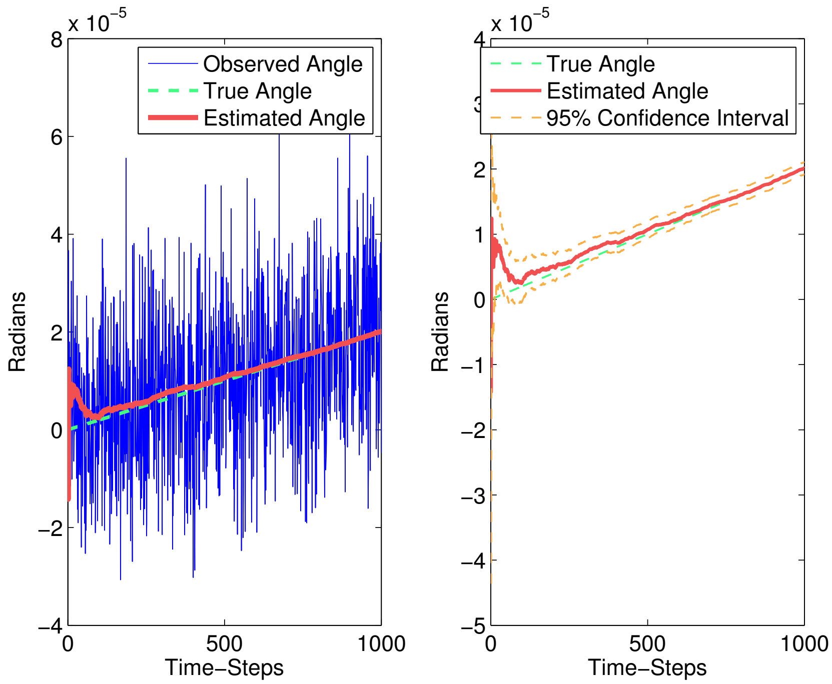

Figure 1: Output for problem 1b.

2. If we linearize $h(\boldsymbol{x})$ around $\hat{\boldsymbol{x}}_{k|k-1}$ we have:

3/4

$$h(\boldsymbol{x}) \approx h(\hat{\boldsymbol{x}}_{k|k-1}) + \boldsymbol{H}_k(\boldsymbol{x} - \hat{\boldsymbol{x}}_{k|k-1})$$

where $\boldsymbol{H}_k$ is the Jacobian of $h(\boldsymbol{x})$ evaluated at $\hat{\boldsymbol{x}}_{k|k-1}$. Substituting this approximation into the function we wish to minimize yields:

$$\begin{aligned} J(\boldsymbol{x}) = &\frac{1}{2}(\boldsymbol{x} - \hat{\boldsymbol{x}}_{k|k-1})^\top \boldsymbol{P}_{k|k-1}^{-1}(\boldsymbol{x} - \hat{\boldsymbol{x}}_{k|k-1}) \\ &+ \frac{1}{2}[\boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{k|k-1})]^\top \boldsymbol{R}_k^{-1}[\boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{k|k-1})] \\ &+ \frac{1}{2}[\boldsymbol{H}_k(\boldsymbol{x} - \hat{\boldsymbol{x}}_{k|k-1})]^\top \boldsymbol{R}_k^{-1}[\boldsymbol{H}_k(\boldsymbol{x} - \hat{\boldsymbol{x}}_{k|k-1})] \\ &- [\boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{k|k-1})]^\top \boldsymbol{R}_k^{-1}[\boldsymbol{H}_k(\boldsymbol{x} - \hat{\boldsymbol{x}}_{k|k-1})]. \end{aligned}$$

If we take the gradient with respect to $\boldsymbol{x}$ in the previous equation and set it equal to zero we have:

$$\boldsymbol{P}_{k|k-1}^{-1}(\boldsymbol{x} - \hat{\boldsymbol{x}}_{k|k-1}) + \boldsymbol{H}_k^\top \boldsymbol{R}_k^{-1}[\boldsymbol{H}_k(\boldsymbol{x} - \hat{\boldsymbol{x}}_{k|k-1})] - \boldsymbol{H}_k^\top \boldsymbol{R}_k^{-1}[\boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{k|k-1})] = \boldsymbol{0}. \quad (3)$$

Solving for $\boldsymbol{x}$ in (3):

$$\boldsymbol{x} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k[\boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{k|k-1})],$$
$$\boldsymbol{K}_k \equiv [\boldsymbol{P}_{k|k-1}^{-1} + \boldsymbol{H}_k^\top \boldsymbol{R}_k^{-1}\boldsymbol{H}_k]^{-1}\boldsymbol{H}_k^\top \boldsymbol{R}_k^{-1}.$$

Rewriting using the matrix inversion lemma we have:

reproduce steps or show reference to pg number etc -1

$$\boldsymbol{x} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k[\boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{k|k-1})],$$
$$\boldsymbol{K}_k = \underbrace{[\boldsymbol{P}_{k|k-1} - \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^\top(\boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^\top + \boldsymbol{R}_{k|k-1})^{-1}\boldsymbol{H}_k\boldsymbol{P}_{k|k-1}]}_{\equiv \boldsymbol{P}_k}\boldsymbol{H}_k^\top\boldsymbol{R}_k^{-1}$$

which is the update we have in the notes for lecture 10. Furthermore, we know (by the equivalence derived in the notes) that this is the same as:

$$\boldsymbol{x} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{K}_k[\boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_{k|k-1})],$$
$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^\top(\boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^\top + \boldsymbol{R}_k)^{-1}$$

as desired.

3. **a)** The code is modified from that provided online and is included at the end of the homework. Figure (2) is the figure output for this problem; we can see that the errors are close to zero.

4/4

b) The code is modified from that provided online and is included at the end of the homework. Figure (3) is the figure output for this problem; we can see that the errors are not going to zero. Therefore, bearing alone does not seem to be sufficient.

4/4

4. First, I must mention that it is impossible to find a unique weighted least squares estimate using only 2 measurements as we are estimating more parameters (six) than we have equations (two). For this reason, I propose to modify the code to have $6 * 4 = 24$ total observations (i.e., rows in $\boldsymbol{H}$) and generated them so that each 6×6 matrix was nonsingular (e.g., rows 1–6 of $\boldsymbol{H}$ form a nonsingular matrix). The idea would then be to process this data set in four iterations of six data points each and in one whole batch mode and compare the two. Details below.

4/4

Let our noisy observations be denoted by $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{24}$. In addition, define $\boldsymbol{Z}_1 \equiv \{z_1, \ldots, z_6\}, \boldsymbol{Z}_2 \equiv \{z_7, \ldots, z_{12}\}$ and so on until $\boldsymbol{Z}_4$. We then have (I will use the notation $f(\cdot)$ denote the distribution of the quantity in parenthesis):

$$f(\boldsymbol{x}|\hat{\boldsymbol{x}}_1) = \frac{f(\hat{\boldsymbol{x}}_1|\boldsymbol{x})f(\boldsymbol{x})}{\int(\text{numerator})\, d\boldsymbol{x}} \tag{4}$$

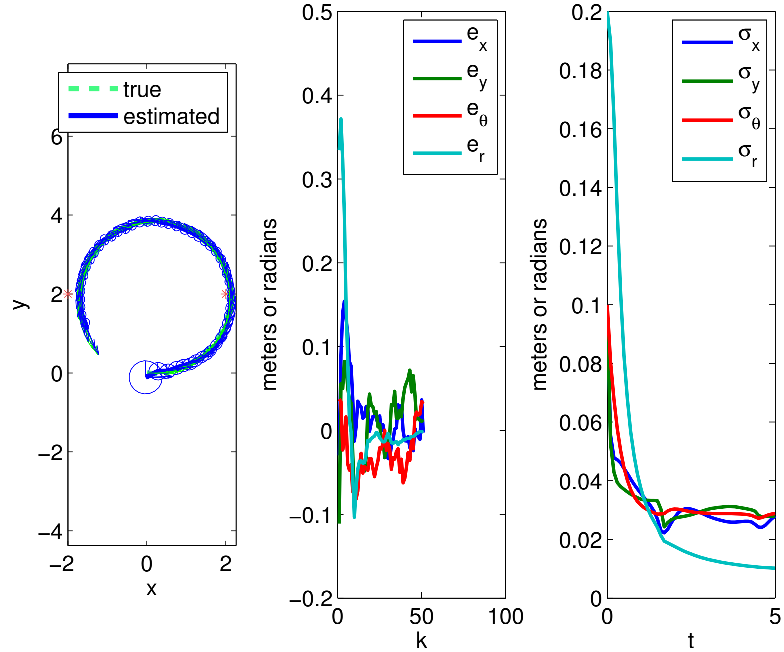

Figure 2: Oputput for problem 3a. The errors go to zero.

where $\hat{x}_1$ is the estimate resulting from using $\boldsymbol{r}$ to do weighted least squares on $\boldsymbol{Z}_1$:

$$\hat{\boldsymbol{x}}_1 = (\boldsymbol{H}_{1:6}^\top \boldsymbol{r}^{-1} \boldsymbol{H}_{1:6})^{-1} \boldsymbol{H}_{1:6}^\top \boldsymbol{r}^{-1} [z_1, \ldots, z_6]^\top \tag{5}$$

where $\boldsymbol{H}_{1:6}$ denotes the matrix with the first six rows of $\boldsymbol{H}$. Then:

$$\hat{\boldsymbol{x}}_1 \sim N\Big(\boldsymbol{B}_1 \boldsymbol{H} \boldsymbol{x}, \boldsymbol{B}_1 \boldsymbol{r} \boldsymbol{B}_1^\top\Big),$$
$$\boldsymbol{B}_1 \equiv (\boldsymbol{H}_{1:6}^\top \boldsymbol{r}^{-1} \boldsymbol{H}_{1:6})^{-1} \boldsymbol{H}_{1:6}^\top \boldsymbol{r}^{-1}.$$

By the way $\boldsymbol{H}$ is generated we know $\boldsymbol{H}_{1:6}$ is nonsingular so that $\hat{\boldsymbol{x}}_1$ does not have a degenerate normal distribution. Now, (4) becomes $f(\boldsymbol{x}|\hat{\boldsymbol{x}}_1) =$

$$\left(e^{-\frac{1}{2}(\hat{x}_1 - \boldsymbol{B}_1 \boldsymbol{H} \boldsymbol{x})^\top (\boldsymbol{B}_1 \boldsymbol{r} \boldsymbol{B}_1^\top)^{-1} (\hat{x}_1 - \boldsymbol{B}_1 \boldsymbol{H} \boldsymbol{x})}\right) \left(e^{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{m}_0)^\top \boldsymbol{P}_0^{-1} (\boldsymbol{x} - \boldsymbol{m}_0)}\right) / \int \text{numerator } dx. \tag{6}$$

The outline of the implementation is as follows:

- Compute $\hat{\boldsymbol{x}}_1$ as in (5) using the first 6 measurements $\boldsymbol{Z}_1$.

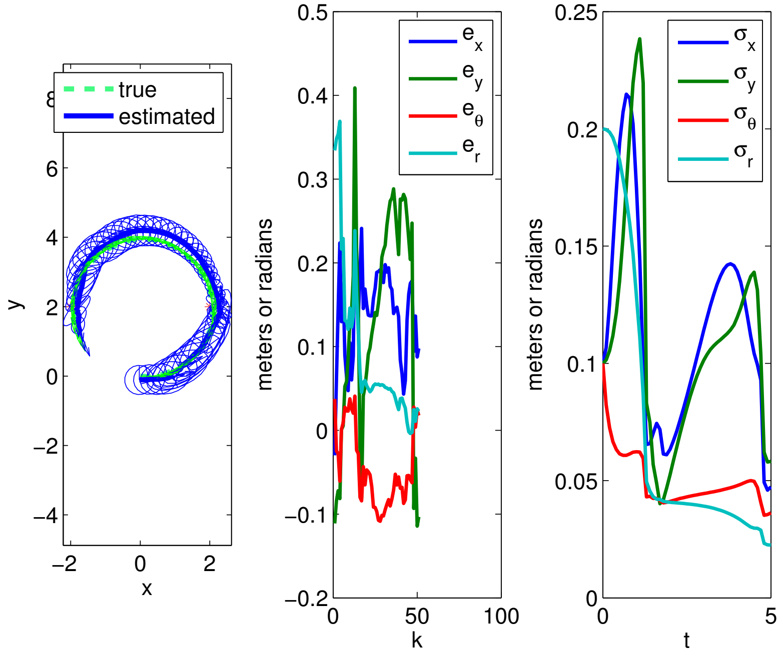

Figure 3: Oputput for problem 3b. The estimates are less stable and not all errors are going to zero.

- Compute $f(\boldsymbol{x}|\hat{\boldsymbol{x}}_1)$ as in (6).
- Let $f(\boldsymbol{x}|\hat{\boldsymbol{x}}_1)$ be the new prior for $\boldsymbol{x}$ for the next iteration.
- Compute $\hat{\boldsymbol{x}}_2$ as in (5) using the second 6 measurements $\boldsymbol{Z}_2$.
- Compute $f(\boldsymbol{x}|\hat{\boldsymbol{x}}_2)$ **using** $f(\boldsymbol{x}|\hat{\boldsymbol{x}}_1)$ as the new prior for $\boldsymbol{x}$. Note that $f(\boldsymbol{x}|\hat{\boldsymbol{x}}_2)$ will depend both on $\hat{\boldsymbol{x}}_1$ and $\hat{\boldsymbol{x}}_2$.
- Repeat until we obtain $f(\boldsymbol{x}|\hat{\boldsymbol{x}}_4)$ and use its MLE as the estimate for the true state.

While the discussion above is formulated in a Bayesian framework, it requires integrating the denominator of (6). I did not implement this.

On the other hand, if what is means is for us to use $\boldsymbol{m}_0$ as an initial estimate and to then use re-weighted least squares in a mini batch update mode I thin k it would be unfair to compare this with the batch mode because that uses the true covariance matrix. If we were to implement a re-weighted least squares in batch mode it would require significantly more computation than the online mode and would, again, seem like an “unfair” comparison?

Matlab Code For Problem 1b

```
% Defining quantities:
dt=10^(-6); %fraction of a sec.
sn=1.5*(10^(-5));
su=3*(10^(-9));
sv=3*(10^(-6));
angrate=0.02; % rad/sec.
theta0=0;
beta0=1.7*(10^(-7));
P0=diag([10^(-4),10^(-12)]);
ci=0.95; % Confidence interval area.
T=10^(3); % Total time-steps.

F=zeros(2,2);
F(1,2)=-1;
Phi=eye(2)+dt*F;
G=[1,0]';
Gamma=dt*G;

Q=[sv^2 0; 0 su^2]*dt;
H=eye(2);
R=zeros(2,2);
R(1,1)=sn^2;

% True angle:
thetak=dt*angrate*ones(1,T+1);
thetak(1)=theta0;
thetak=cumsum(thetak,2);

% Observed angle:
zk=thetak+(sn*randn(1,T+1));

% Real beta values:
betak=randn(1,T+1)*su*dt;
betak(1)=beta0;
betak=cumsum(betak,2);

% Control values:
u=angrate*ones(1,T+1)+betak+sv*randn(1,T+1);

% Real x:
xreal=[thetak;betak];

% Observed x:
xobs=[zk;betak];

% Just for pre-allocation:
xhat=xobs;
xpred=xhat;
cp=zeros(1,T+1);
cm=cp;

Phat=P0;
cp(1)=1.96*sqrt(Phat(1,1));
cm(1)=-1.96*sqrt(Phat(1,1));
```

```
for i=1:T
    %Prediction step;
    xpred(:,i+1)=(Phi*xhat(:,i))+(Gamma*u(i));
    Ppred=(Phi*Phat*(Phi'))+Q;
    %Estimation step:
    K=(Ppred*H')/((H*Ppred*H')+R);
    Phat=(eye(2)-(K*H))*Ppred;
    xhat(:,i+1)=xpred(:,i+1)+(K*(zk(:,i+1)-(H*xpred(:,i+1))));
    % Confidence interval storage:
    cp(i+1)=1.96*sqrt(Phat(1,1));
    cm(i+1)=-1.96*sqrt(Phat(1,1));
end

% Plotting:

red=[242/255 80/255 80/255];
green=[67/255 250/255 131/255];
yellow=[250/255 174/255 67/255];

subplot(1,2,1)
plot(0:T,xobs(1,:),'b','LineWidth',.5)
hold on
plot(0:T,xreal(1,:),'--','Color',green,'LineWidth',2)
plot(0:T,xhat(1,:),'Color',red,'LineWidth',3)
ylabel('Radians'); xlabel('Time-Steps');
legend('Observed Angle','True Angle','Estimated Angle');

subplot(1,2,2)
plot(0:T,xreal(1,:),'--','Color',green,'LineWidth',1.1)
hold on
plot(0:T,xhat(1,:),'Color',red,'LineWidth',2)
plot(1:T,xhat(1,2:end)+cp(1,2:end),'--','Color',yellow,'LineWidth',1)
plot(1:T,xhat(1,2:end)+cm(1,2:end),'--','Color',yellow,'LineWidth',1)
ylabel('Radians'); xlabel('Time-Steps');
legend('True Angle','Estimated Angle','95% Confidence Interval');

fig=gcf;
set(findall(fig,'-property','FontSize'),'FontSize',13)
```

Matlab Code For Problem 3a

```
function f = uni_ekf_test
rng(10212)
S.bearing_only = 0;

% 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)); % measurement noise model
else
```

```
  S.h = @br_h;    % bearing−reange sensing
  S.r = 2∗nb;       % measurement dimension
  S.R = .4∗diag(repmat([.1; .01], nb, 1)); % measurement noise model
end

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

% timing
dt = .1;
N = 50;
T = dt∗N;
S.dt = dt;

% noise models
S.Q = (dt^2)∗diag([.01 .01 .01 .0001]);

% initial mean and covariance
xt = [0; 0; 0; 1]; % true state
P = diag([.01 .01 .01 .04]); % 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];
  xts(:,k+1) = S.f(xts(:,k), u, S) + sqrt(S.Q)∗randn(S.n,1);  % next true state
  [x,P] = ekf_predict(x, P, u, S);  % predict next state
  ts(k+1) = k∗dt;
  z = S.h(xts(:,k+1), S) + sqrt(S.R)∗randn(S.r,1);  % generate noisy 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

% Plotting:
subplot(1, 3, 1)
plot(xts(1,:), xts(2,:), '−−','Color',green,'LineWidth',3)
hold on
plot(xs(1,:), xs(2,:), '−b','LineWidth',3)
legend('true', 'estimated')
xlabel('x')
ylabel('y')
axis equal
axis xy
plot(S.pbs(1,:), S.pbs(2,:), '∗','Color',red); % Beacon
```

```
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','LineWidth',2)
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),'LineWidth',2);
legend('\sigma_x','\sigma_y','\sigma_\theta','\sigma_r')
xlabel('t')
ylabel('meters or radians')

fig=gcf;
set(findall(fig,'-property','FontSize'),'FontSize',13)

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(4);
     x(2) + s*u(1)*x(4);
     x(3) + u(2);
     x(4)];
x = fix_state(x, S);
if nargout > 1
  % F-matrix
  varargout{1} = [1, 0, -s*u(1)*x(4), c*u(1);
                  0, 1, c*u(1)*x(4), 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)
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
```

Matlab Code For Problem 3b

```
function f = uni_ekf_test2
rng(10212)
S.bearing_only = 1;

% The rest of the code is the same as uni_ekf_test.m
```