3:

lem set 4, number 1)

The code I used in this problem is shown below:

```
xg(:,1) = [4 -4]';
xg(:,2) = [6 0]';
xg(:,3) = [-5 5]';

for k = 1:3
    record_xg(:,1,k) = xg(:,k);
    for i = 1:6
        x1 = xg(1,k);
        x2 = xg(2,k);
        f = [x1+x2+x1*x2+5; x1^2+2*x2-x2^2-2];
        df_dx = [1+x2 1+x1;
                 2*x1 2-2*x2];
        xg(:,k) = xg(:,k) - inv(df_dx)*f;
        record_xg(:,i+1,k) = xg(:,k);
        record_normf(k,i) = norm(f);
    end
    x1 = xg(1,k);
    x2 = xg(2,k);
    f = [x1+x2+x1*x2+5; x1^2+2*x2-x2^2-2];
    record_normf(k,i+1) = norm(f);
end
```

The output for ***record_xg*** and ***record_normf*** are shown below:

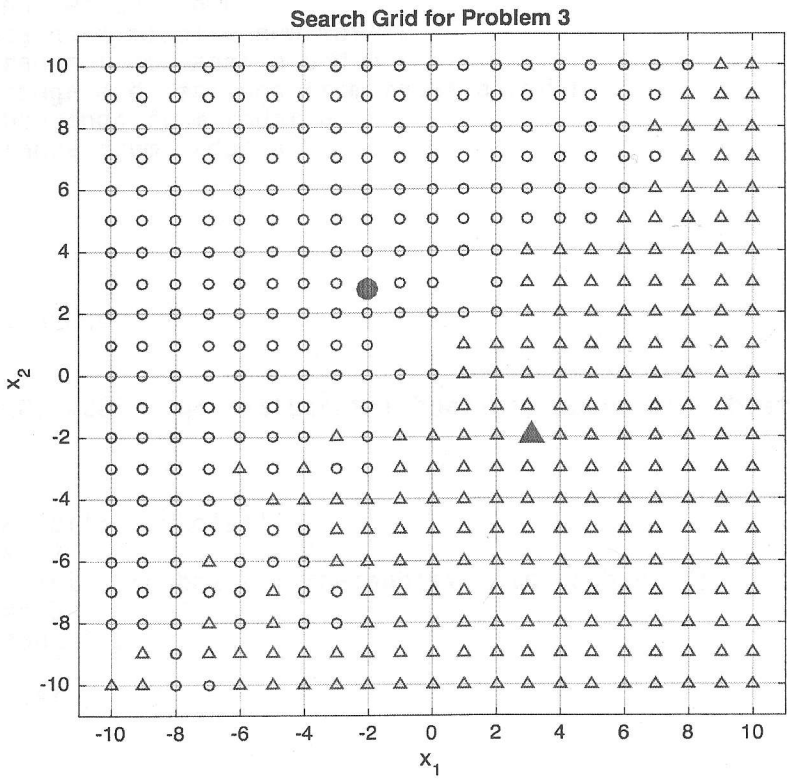
	Iteration	0	1	2	3	4	5	6
first case	xg(1)	4	3.1429	3.1184	3.132	3.132	3.132	3.132
	xg(2)	-4	-2.3143	-1.9733	-1.968	-1.9681	-1.9681	-1.9681
	norm f(xg)	14.866068	2.5547839	0.1159971	0.0001719	3.32E-10	1.83E-15	9.93E-16
second case	xg(1)	6	3.3659	3.0274	3.134	3.132	3.132	3.132
	xg(2)	0	-1.1951	-1.9313	-1.9685	-1.9681	-1.9681	-1.9681
	norm f(xg)	35.735136	6.3462840	0.4947724	0.0107439	4.08E-06	4.91E-13	4.44E-16
third case	xg(1)	-5	-2.8182	-2.1319	-2.0548	-2.0542	-2.0542	-2.0542
	xg(2)	5	3.2727	2.8127	2.7935	2.7944	2.7944	2.7944
	norm f(xg)	21.540659	4.1664779	0.4085405	0.0057713	6.13E-07	1.19E-14	2.66E-15

For the second part (grid going from -10 to 10 for initial conditions of both variables), the following matlab orithm was used:

```
%Create grid
n = 0;
for i = -10:1:10
    for j = -10:1:10
        n = n+1;
        xg(:,n) = [i j]';
    end
end

%Run the problem for all initial conditions defined above
for k = 1:n
    record_xg(:,1,k) = xg(:,k);
    for i = 1:6
        x1 = xg(1,k);
        x2 = xg(2,k);
        f = [x1+x2+x1*x2+5; x1^2+2*x2-x2^2-2];
        df_dx = [1+x2 1+x1;
                 2*x1 2-2*x2];
        xg(:,k) = xg(:,k) - inv(df_dx)*f;
        record_xg(:,i+1,k) = xg(:,k);
        record_normf(k,i) = norm(f);
    end
    x1 = xg(1,k);
    x2 = xg(2,k);
    f = [x1+x2+x1*x2+5; x1^2+2*x2-x2^2-2];
    record_normf(k,i+1) = norm(f);
end

%Plot all results that converge (norm_f < 1e-5)
figure('units','normalized','outerposition',[0 0 1 1]);
for k = 1:n
    if(record_normf(k,end) < 1e-5)
        x1_init = record_xg(1,1,k);
        x1f = record_xg(1,end,k);
        x2_init = record_xg(2,1,k);
        x2f = record_xg(2,end,k);
        if ((x1f < 0) & (x2f > 0))
            plot(x1_init,x2_init,'k*','MarkerSize',5); hold on;
            plot(x1f,x2f,'b*','MarkerSize',10); hold on;
        else
            plot(x1_init,x2_init,'ko','MarkerSize',5); hold on;
            plot(x1f,x2f,'ro','MarkerSize',10); hold on;
        end
    end
end
grid on;
```

This code was able to identify two convergence points:

$$x_{g1} = \begin{bmatrix} 3.132 \\ -1.9681 \end{bmatrix}, \quad x_{g2} = \begin{bmatrix} -2.0542 \\ 2.7944 \end{bmatrix}$$



There are two convergence points on this range. From the generated plot it can be seen that the initial guess converges to the closest convergence point in most cases. This leads to a division of the search space into two regions separated by a straight line that is perpendicular to the line joining the two convergence points. It must be noted that in some cases when the initial guess is not too close to either of the points, the algorithm may converge to the farther convergence point. Interestingly, if the initial guess is at almost the same distance from the two convergence points, the algorithm may not converge.

yes.

PS 4 #2

$$J(x) = \frac{1}{2}[z - h(x)]^T[z - h(x)]$$

$$\Delta x = (H^TH + \lambda I)^{-1} H^T[z - h(x_g)] \quad \text{where } H = \left.\frac{\partial h}{\partial x}\right|_{x_g}$$

Let $\lambda = \frac{1}{\epsilon}$, $\epsilon = \frac{1}{\lambda}$ then $\epsilon \to 0$ as $\lambda \to \infty$

$$\Delta x(\epsilon) = (H^TH + \tfrac{1}{\epsilon} I)^{-1} H^T[z - h(x_g)] = \epsilon(\epsilon H^TH + I)^{-1} H^T[z - h(x_g)]$$

Let $\tilde{J}(\epsilon) = J(x_g + \Delta x(\epsilon))$. We need to show that $\left.\frac{d\tilde{J}}{d\epsilon}\right|_{\epsilon=0} < 0$ in order to prove that $\exists\, \lambda \geq 0$ s.t. $J(x_g + \Delta x) < J(x_g)$

Note that $\Delta x(\epsilon = 0) = 0$.

$$\left.\frac{d\tilde{J}}{d\epsilon}\right|_{\epsilon=0} = \left(\left.\frac{\partial J}{\partial x}\right|_{x_g}\right)\left(\left.\frac{\partial \Delta x(\epsilon)}{\partial \epsilon}\right)\right|_{\epsilon=0}$$

$$\frac{\partial J}{\partial x} = -[z - h(x)]^T H$$

$$\left.\frac{\partial \Delta x(\epsilon)}{\partial \epsilon}\right|_{\epsilon=0} = \left.\epsilon\left\{\frac{\partial}{\partial \epsilon}\left[(\epsilon H^TH + I)^{-1} H^T[z - h(x_g)]\right]\right\} + (\epsilon H^TH + I)^{-1} H^T[z - h(x_g)]\right|_{\epsilon=0} \quad \text{(product rule)}$$

$$= H^T[z - h(x_g)] \quad \text{(when evaluated at } \epsilon = 0\text{)}$$

Thus, $$\left.\frac{d\tilde{J}}{d\epsilon}\right|_{\epsilon=0} = -[z - h(x_g)]^T HH^T[z - h(x_g)]$$

$$= -\underbrace{\{H^T[z - h(x_g)]\}^T\{H^T[z - h(x_g)]\}}_{\text{Quadratic form guaranteed positive for nonzero } H^T[z - h(x_g)]} < 0$$

Note that if $H^T[z - h(x_g)] = 0$ then LS soln has achieved a min, max, or saddle point and $\Delta x = 0$, so

$$J(x_g + \Delta x) = J(x_g)$$

```
% p4p3.m
% Problem Set 4 Problem 3

clear; clc;
rng(39);
format long

%----- Problem setup
xTrue = [5;5;1.5];   % Make these less than 20 and positive
N = 11;              % Keep fixed at 11 to prevent confusion
Rmat = diag(ones(N,1));
Rs = diag(0.5*ones(N-1,1));
Rmat = Rmat+[zeros(N-1,1),Rs;zeros(1,N)]+[zeros(1,N);Rs,zeros(N-1,1)];
Ra = chol(Rmat);

%----- Simulate the measurements
wVec = Ra'*randn(N,1);
dt = 0.1;
thist = [0:N-1]'*dt;
zhist = xTrue(1)*cos(xTrue(2)*thist + xTrue(3)) + wVec;

%----- Nonlinear Least Squares Estimation
Rainv = inv(Ra);
xbar = [4;6;1.5];
% Uncomment the two lines below to randomize the initial guess
%rng('shuffle');
%xbar = 20*rand(3,1)
dxtilde = 1;
x1 = xbar(1); x2 = xbar(2); x3 = xbar(3);
JcostOld = (norm(Rainv'*(zhist -  x1*cos(x2*thist + x3))))^2;
iiMax = 100;
ii0 = 0;

while((norm(dxtilde) > 1e-9) & (ii0 < iiMax))
  ii0 = ii0 + 1;
  alpha = 1;
  x1 = xbar(1); x2 = xbar(2); x3 = xbar(3);
  H = Rainv'*[cos(x2*thist + x3), -x1*thist.*sin(x2*thist + x3), ...
              -x1*sin(x2*thist + x3)];
  dz = Rainv'*(zhist - x1*cos(x2*thist + x3));
  [Qtilde,Rtilde] = qr(H);
  Rtilde0 = Rtilde(1:3,1:3);
  dztilde = Qtilde'*dz;
  dxtilde = inv(Rtilde0)*dztilde(1:3);
  xbarTemp = xbar + alpha*dxtilde;
  x1 = xbarTemp(1); x2 = xbarTemp(2); x3 = xbarTemp(3);
  JcostNew = (norm(Rainv'*(zhist -  x1*cos(x2*thist + x3))))^2;
  ii = 0;
  while((JcostNew >= JcostOld) && (ii < iiMax))
    ii = ii + 1;
    alpha = alpha/2;
    xbarTemp = xbar + alpha*dxtilde;
```

```
    x1 = xbarTemp(1); x2 = xbarTemp(2); x3 = xbarTemp(3);
    JcostNew = (norm(Rainv'*(zhist -  x1*cos(x2*thist + x3))))^2;
  end
  xbar = xbarTemp;
  JcostOld = JcostNew;
end

xbar
Pxx = inv(Rtilde0)*inv(Rtilde0')
JcostNew


xbar =
   5.756634618776078
   5.097288218285141
   1.380791590497606
Pxx =
   0.313104806716449  -0.008094595013304   0.001678190096156
  -0.008094595013304   0.088954264913419  -0.035286805857673
   0.001678190096156  -0.035286805857673   0.023426389060185
JcostNew =
   9.232038549815501
```

Modified Missile Tracking Problem

Part (a). This part instructs you to use the new measurement data file. Let $\boldsymbol{x} = [x_1, x_2, x_3, x_4]^T = [y_{10}, v_{10}, y_{20}, v_{20}]^T$ be the parameter vector. The true value of $\boldsymbol{x}$ for this problem is

$$\boldsymbol{x} = \begin{bmatrix} 1500 \\ 900 \\ 1700 \\ 1500 \end{bmatrix}$$

where distance units are in meters and velocity units are in meters per second.

Part (b). It's very important to obtain a good initial initial guess for $\boldsymbol{x}$. Otherwise, the Gauss-Newton method will settle into an incorrect local minimum. Start by modeling the noise-free range measurements as

$$\rho_a^2 = (l_a - y_1)^2 + y_2^2$$
$$\rho_b^2 = (l_b - y_1)^2 + y_2^2$$

Difference the squared range measurements to yield a linear equation in y_1:

$$\begin{aligned} \rho_a^2 - \rho_b^2 &= (l_a - y_1)^2 - (l_b - y_1)^2 \\ &= l_a^2 - 2l_a y_1 - l_b^2 + 2l_b y_1 \end{aligned}$$

from which

$$y_1 = \frac{\rho_a^2 - \rho_b^2 + l_b^2 - l_a^2}{2(l_b - l_a)}$$

Similarly, sum the squared range measurements to solve for y_2 once y_1 is known. Assume $y_2 >= 0$ and take the positive square root from the quadratic equation:

$$\rho_a^2 + \rho_b^2 = (l_a - y_1)^2 + (l_b - y_1)^2 + 2y_2^2$$
$$y_2 = \sqrt{\tfrac{1}{2}\left[\rho_a^2 + \rho_b^2 - (l_a - y_1)^2 - (l_b - y_1)^2\right]}$$

Then, using two position estimates at different times (preferably far apart), solve a linear system of four equations and four unknowns obtain an initial guess of $\boldsymbol{x}$. Form the four equations from

$$y_{1j} = x_1 + x_2 t_j$$
$$y_{2j} = x_3 + x_4 t_j - \frac{1}{2} g t_j^2$$

for two distinct values of j. Let $\boldsymbol{y} = [y_{1p}, y_{2p} + gt_p^2/2, y_{1q}, y_{2q} + gt_q^2/2]^T$ for indices $p \neq q$. Then form the equation

$$\boldsymbol{y} = A\boldsymbol{x}$$

where

$$A = \begin{bmatrix} 1 & t_p & 0 & 0 \\ 0 & 0 & 1 & t_p \\ 1 & t_q & 0 & 0 \\ 0 & 0 & 1 & t_q \end{bmatrix}$$

and find an initial guess of $\boldsymbol{x}$ by

$$\boldsymbol{x}_g = A^{-1}\boldsymbol{y}$$

Repeat this process for different choices of indices p and q to ensure that you have a reliable initial guess. Then set up and solve the nonlinear estimation problem using the framework and range measurement linearization given in the `missileProblemRevisited.pdf` lecture notes.

Part (c). For this part, you can use the same initial guess $\boldsymbol{x}_g$ that you used for Part (b). Set up and solve the nonlinear estimation problem using the framework and range and bearing measurement linearizations given in the `missileProblemRevisited.pdf` lecture notes. The final solution should not be much different from that of Part (b). The approximate estimation error covariance matrix P_{xx} should be smaller along its diagonal elements, but only slightly: the additional bearing measurements do not reduced the covariance significantly.

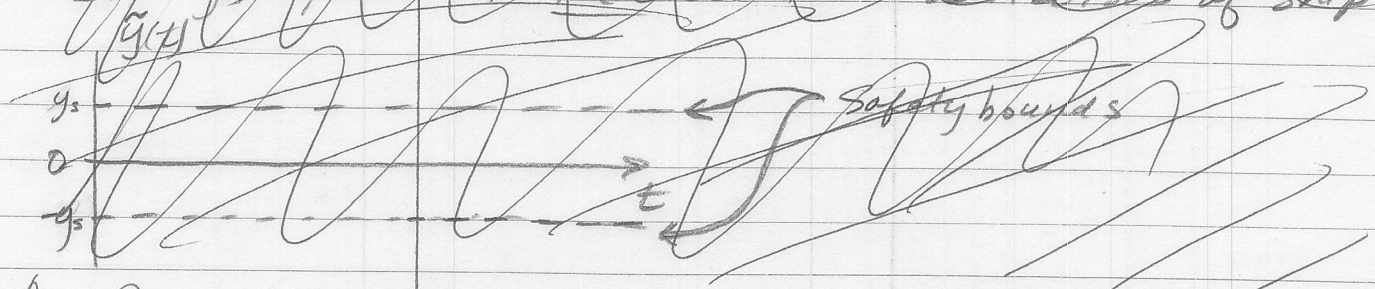
Part (d). You will find that the rank of the measurement sensitivity matrix H is 4, so the initial state is, in theory, observable from the bearing measurements of a single radar. But the singular values of H will reveal that the problem is poorly conditioned (H is nearly less than rank 4). Thus, the initial state is only *weakly* observable from the single-radar, bearings-only data.

Part (e). Upon examining the approximate estimation error covariance of the single-radar, bearings-only solution, you will find that position estimates are useless: the square root of the diagonal elements of P_{xx} corresponding to position estimates are a km or more in size. However, the velocity measurements may be useful: their square-root variance is less than 10 cm/second.

Analysis of 1st-order Markov process with finite Pyyss

(ALSO known as OU process)

Let $\tilde{y}(t)$ represent the cross-track deviation of ship:



$\tilde{y} \neq 0$ for travel along the rhumbline. Under H_0, $y(t)$ is measured by estimator (based on GPS + gyrocompass complementary filter) and regulated toward 0 by a controller. Under H_a, $y(t)$ can be modeled as an OU process. ~~$y(k) = \tilde{y}(k\Delta t)$~~

The OU process is just a first-order Markov process:

$$y(k+1) = f\,y(k) + v(k), \quad k = 0, 1, \ldots \qquad (1)$$

where the sequence $\{v(k)\}$ is zid with $v(k) \sim N(0, (1-f^2)\sigma_{d_0}^2)$ and where $f = e^{-T/\tau_{d_0}}$. Assume $y(0) = 0$.

Note that $\bar{y}(k) = E[y(k)] = 0$ and that $P_{yy}(k) \triangleq E[(y(k) - \bar{y}(k))^2]$

is given by $$P_{yy}(k+1) = f^2 P_{yy}(k) + (1-f^2)\sigma_{d_0}^2 \qquad (2)$$

The steady-state value of P_{yy} as $k \to \infty$ is found by

$$P_{yyss} = f^2 P_{yyss} + (1-f^2)\sigma_{d_0}^2$$

$$\Rightarrow P_{yyss} = \sigma_{d_0}^2$$

Also, the soln of (1) is given by $$y(k) = \sum_{i=0}^{k-1} f^{(k-1)-i}\, v(i)$$

The autocorrelation of $y(k)$ is the same as the autocovariance since $\bar{y}(k) = 0$.

$$R(k,j) \triangleq E[y(k)\,y(j)]$$

$$= E\left[\left(f\,y(k-1) + v(k-1)\right)\left(f\,y(j-1) + v(j-1)\right)\right]$$

$$= E[\, f^2 y(k-1)y(j-1) + f y(k-1)v(j-1) + f y(j-1)v(k-1) + v(k-1)v(j-1)\,]$$

$$= f^2 R(k-1, j-1) + f E[y(k-1)v(j-1)] + f E[y(j-1)v(k-1)] + E[v(k-1)v(j-1)]$$

It's convenient here to invoke the solution of $y(k)$:

$$\boxed{y(k) = \sum_{i=0}^{k-1} f^{(k-1)-i} v(i), \quad k = 1, 2, \ldots}$$

Now it's easier to calculate $R(k, j)$:

$$R(k+1, j+1) = E\Big[\left(f^k v(0) + f^{k-1} v(1) + \cdots + f v(k-1) + v(k)\right) \cdot \left(f^j v(0) + f^{j-1} v(1) + \cdots + f v(j-1) + v(j)\right) \Big]$$

Note that cross-terms vanish, leaving only

$$R(k+1, j+1) = f^{j+k}(1-f^2)\sigma_{d_0}^2 + f^{(k-1)(j-1)}(1-f^2)\sigma_{d_0}^2 + \cdots$$

Let's look at some easy ones. First, let $k = j$. Then

$$R(k+1, k+1) = P(k+1) = f^{2k}(1-f^2)\sigma_{d_0}^2 + f^{2(k-1)}(1-f^2)\sigma_{d_0}^2 + f^{2(k-2)}(1-f)^2\sigma_{d_0}^2 + \cdots + f^2(1-f^2)\sigma_{d_0}^2 + (1-f^2)\sigma_{d_0}^2$$

$$\Rightarrow \boxed{R(k+1, k+1) = (1-f^2)\sigma_{d_0}^2 \sum_{i=0}^{k} f^{2i} = P(k+1) \qquad (3)}$$

This should be the same as $P_{yy}(k+1)$. Let's see if it works out for low k:

$$P_{yy}(1) = f^2 \cancel{P_{yy}(0)}^{\,\to 0} + (1-f^2)\sigma_{d_0}^2 \qquad (\text{from } (2))$$

Similarly, $R(1,1) = (1-f^2)\sigma_{d_0}^2$

Note that this is much like Bar Shalom 4-4 (from problem set 4)

$$P_{yy}(2) = f^2 P_{yy}(1) + (1-f^2)\sigma_{d_0}^2 \qquad \text{(from (2))}$$

$$= f^2(1-f^2)\sigma_{d_0}^2 + (1-f^2)\sigma_{d_0}^2$$

$$= (1-f^2)\sigma_{d_0}^2\left[f^2 + f^0\right]$$

Similarly, $R(2,2) = (1-f^2)\sigma_{d_0}^2\left[f^2 + f^0\right]$

So it looks like (3) is a correct formula for the OU process. This should mean that for large k (3) should converge to $P_{yyss} = \sigma_{d_0}^2$, which would imply that

$$\lim_{k\to\infty} \sum_{i=0}^{k} f^{2i} = \frac{1}{1-f^2} \qquad \text{for} \quad 0 \le f < 1$$

Is this true? Yes, I've confirmed it to be true.

Ok, but is there a formula for $\sum_{i=0}^{k} f^{2i}$ with $0 \le f < 1$?

Here is a formula: $\sum_{i=0}^{k} r^i = \frac{r^{k+1}-1}{r-1}$

Here is the needed formula: $\sum_{i=0}^{k} r^{2i} = \frac{(1-r^{2k+2})}{1-r^2}$

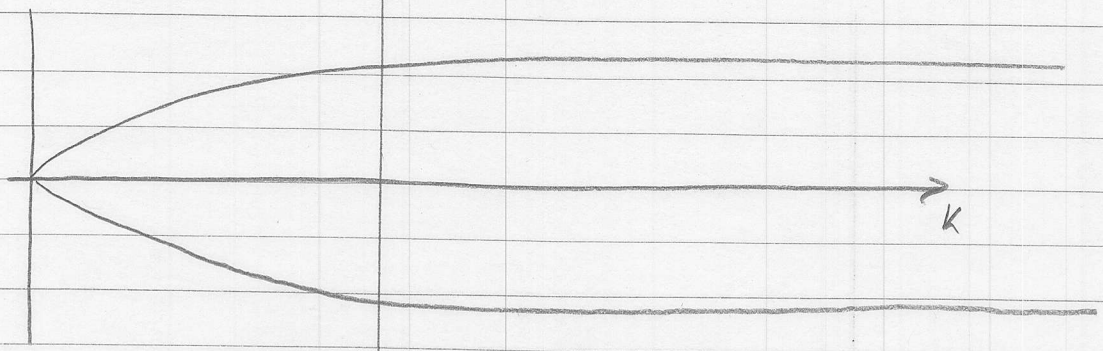
(From Wikipedia article on Geometric progression)

Now we can reduce (3) to

$$R(k+1,k+1) = P_{yy}(k+1) = \sigma_{d_0}^2(1-f^2)\frac{(1-f^{2k+2})}{(1-f^2)}$$

$$= \sigma_{d_0}^2(1-f^{2k+2}) \qquad (4)$$

What does this look like?



In only a few short time constants we get to steady state.

Let's look at $R(k+2, k+1)$:

$$R(k+2,k+1) = E\Big[\Big(f^{k+1}v(0) + f^{k}v(1) + \cdots + f^{2}v(k-1) + f\,v(k) + v(k+1)\Big) \cdot \Big(f^{k}v(0) + f^{k-1}v(1) + \cdots + f\,v(k-1) + v(k)\Big)\Big]$$

Note that all terms with $v(k+1)$ will vanish, since $v(k+1)$ isn't correlated with any $v(0), v(1), \ldots, v(k)$.

Then we have

$$R(k+2,k+1) = f^{k+1}f^{k}(1-f^2)\sigma_{d_0}^2 + f^{k}f^{k-1}(1-f^2)\sigma_{d_0}^2 + \cdots + f^{2}f(1-f^2)\sigma_{d_0}^2 + f(1-f^2)\sigma_{d_0}^2$$

$$= f(1-f^2)\sigma_{d_0}^2 \sum_{i=0}^{k} f^{2i}$$

Thus, $R(k+2,k+1) = f\,R(k+1,k+1) = f\,P_{yy}(k+1)$

It would be easy to show that

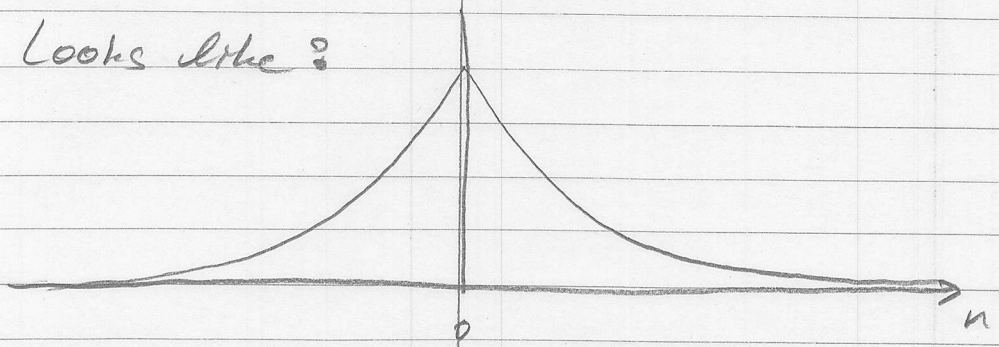
$$\boxed{R(k,k+n) = R(k+n,k) = f^{n}P_{yy}(k) \quad (5)}$$

Note that this depends on k since the process is not stationary.

As $P_{yy}(k) \to P_{yyss}$, then the process does become stationary with

$$R(n) \triangleq R(k+n, k) = f^n P_{yyss}$$

Looks like:



Summary:

$$P_{yyss} = \sigma_{d_0}^2$$

$$P_{yy}(k+1) = \sigma_{d_0}^2 (1 - f^{2k+2})$$

$$R(k, k+n) = R(k+n, k) = f^n P_{yy}(k)$$

$$y(k+1) = \sum_{i=0}^{k} f^{k-i} v(i), \quad k = 0, 1, \ldots$$

For $y(k+1) = f y(k) + v(k), \quad k = 0, 1, \ldots$

$$v(k) \sim N(0, (1-f^2)\sigma_{d_0}^2), \text{ indep.}$$

$$f = e^{-T/\tau_{d_0}}, \quad \tau_{d_0} \text{ is correlation time}$$

BS 4-4

Consider $y(k) = \alpha y(k-1) + (1-\alpha) v(k)$
with $y(0) = 0$ and $0 < \alpha < 1$

---

1) Solution for $y(k)$:

$y(1) = (1-\alpha) v(1)$

$y(2) = \alpha(1-\alpha) v(1) + (1-\alpha) v(2)$

$\Rightarrow y(k) = \sum_{i=0}^{k-1} \alpha^i (1-\alpha) v(k-i)$

Let $j = k-i$, then $\boxed{y(k) = (1-\alpha) \sum_{j=1}^{k} \alpha^{k-j} v(j)}$

2) $\bar{y}(k) \triangleq E[y(k)] = (1-\alpha) \sum_{j=1}^{k} \alpha^{k-j} \bar{v} = \bar{v}(1-\alpha^k)$

$\text{Var}(y(k)) \triangleq E[(y(k) - \bar{y}(k))^2]$

$= E\left\{ \left[\sum_{j=1}^{k} \alpha^{k-j} (1-\alpha) [v(j) - \bar{v}] \right]^2 \right\}$

$= \sum_{j=1}^{k} \alpha^{2(k-j)} (1-\alpha)^2 \sigma^2 = \frac{(1-\alpha)(1-\alpha^{2k}) \sigma^2}{(1+\alpha)}$

3) How do $y(k)$ and $z(k)$ differ?

$z(k) = \left(\frac{1}{\sum_{i=1}^{k} \alpha^{k-i}} \right) \sum_{i=1}^{k} \alpha^{k-i} v(i)$

$y(k) = (1-\alpha) \sum_{i=1}^{k} \alpha^{k-i} v(i)$

They differ in the coefficient of the sum, which tends to create an average from the sum. Look at expectations:

$\bar{z}(k) \triangleq E[z(k)] = \bar{v}$, thus $\bar{y}(k) = (1-\alpha^k) \bar{z}(k)$,
so they differ by the factor $(1-\alpha^k)$, which, since $0 < \alpha < 1$, goes to 1 as $k \to \infty$.

4) Estimate $\bar{v}$ by a sample mean:

$$\hat{\bar{v}}_{sm} = \frac{1}{N}\sum_{k=1}^{N} v(k)$$

which has variance $E\left[(\bar{v} - \hat{\bar{v}}_{sm})^2\right] = \sigma^2/N$

Look for α that produces same variance for large k:

$$\lim_{k\to\infty} \text{Var}[y(k)] = \frac{(1-\alpha)\sigma^2}{1+\alpha} = \frac{\sigma^2}{N}$$

$$\Rightarrow \alpha = \frac{N-1}{N+1}$$

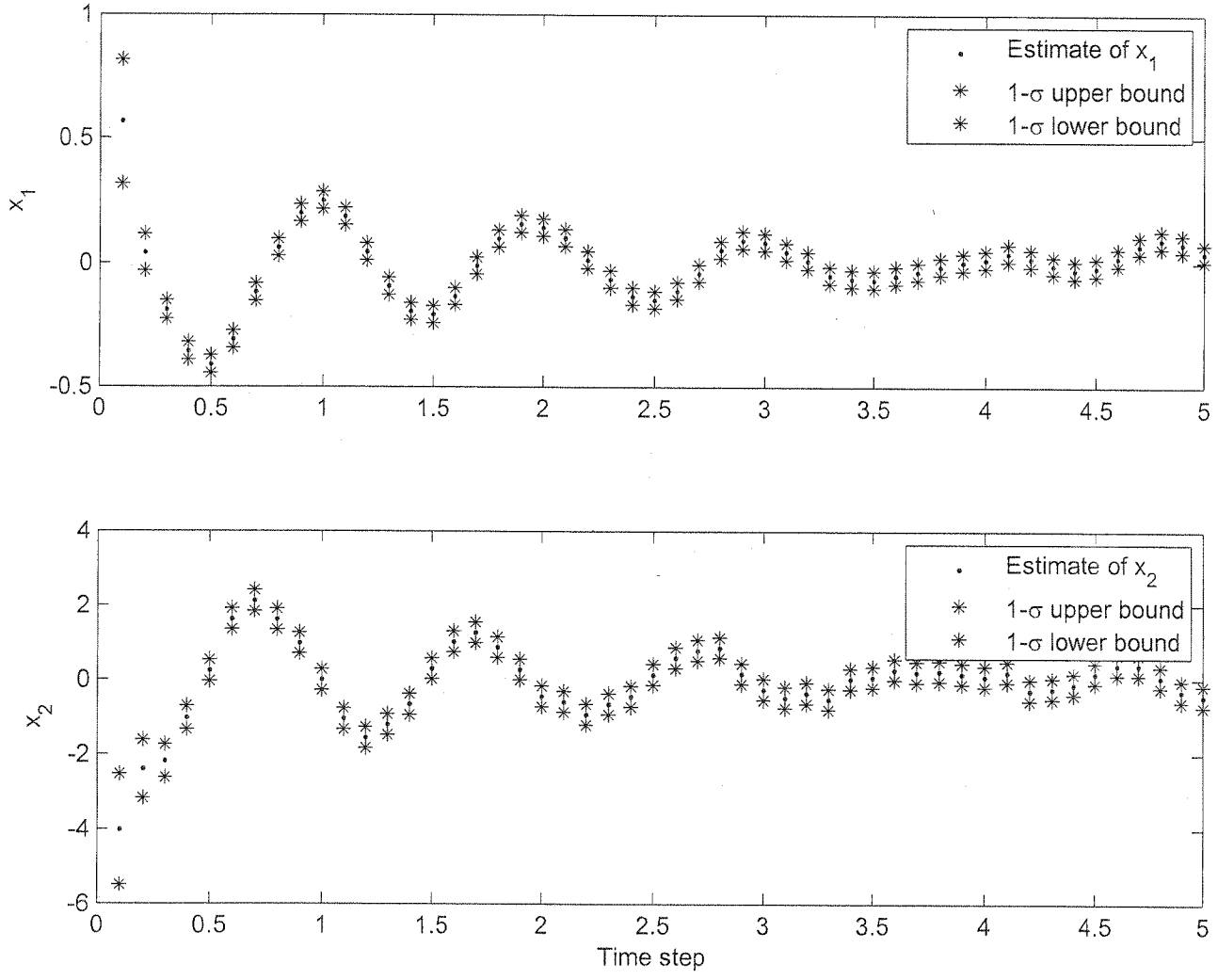
Thus, for $N = 10$, $\alpha = 9/11$.

```
% PS5n3.m
% Implementation of problem 3 on homework set 5
% A basic linear Kalman Filter

clear;clc;
% Run the script with the data
kf_example02a

nd = length(zhist);
nx = length(xhat0);

% The first measurement is at time 1.  Initial estimates are for time 0
Phatk = P0;
xhatk = xhat0;
Fk = Fk;
Gammak = Gammak;
Gk = 0;
uk = 0;
Qk = Qk;
Rkp1 = Rk;
Hkp1 = Hk;

% Allocate space for histories
xhathist = zeros(nd,nx);
errhist = zeros(nd,1);
stdhist = zeros(nd,nx);  % History of expected standard dev's

% Filter
for ii=1:nd
  zkp1 = zhist(ii,:);
  [xhatkp1,Phatkp1,err] = ...
    kf(xhatk,uk,zkp1,Phatk,Fk,Gammak,Gk,Qk,Rkp1,Hkp1);

  xhathist(ii,:) = xhatkp1';
  errhist(ii) = err;
  stdhist(ii,:) = [sqrt(diag(Phatkp1))'];

  xhatk = xhatkp1;
  Phatk = Phatkp1;
end


figure(1);clf;
subplot(211)
plot(thist,xhathist(:,1),'.', thist, xhathist(:,1)+stdhist(:,1),'*',
thist, ...
     xhathist(:,1)-stdhist(:,1),'*');
ylabel('x_1')
legend('Estimate of x_1', '1-\sigma upper bound', '1-\sigma lower
bound');
subplot(212)
plot(thist,xhathist(:,2),'.', thist, xhathist(:,2)+stdhist(:,2),'*',
thist, ...
     xhathist(:,2)-stdhist(:,2),'*');
ylabel('x_2')
legend('Estimate of x_2', '1-\sigma upper bound', ['1-\sigma lower '
...
```

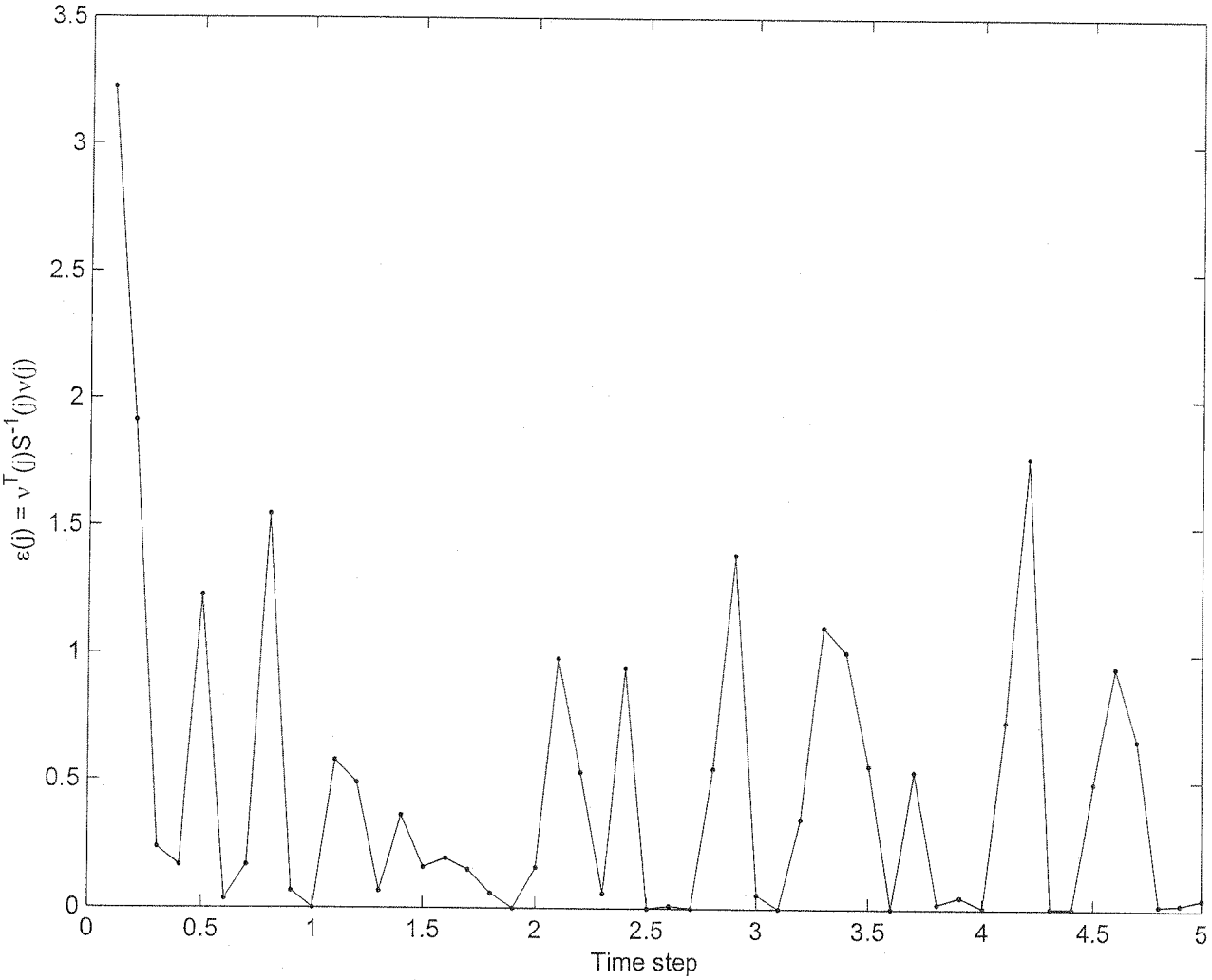
```
                        'bound']);
xlabel('Time step')

figure(2);clf;
plot(thist, errhist);
hold on;
plot(thist,errhist,'.');
ylabel('\epsilon(j) = \nu^T(j)S^{-1}(j)\nu(j)');
xlabel('Time step')

format long
xhatk
Phatk
```

```
-------------- KF function -----------------
function [xhatkp1,Phatkp1,err] = ...
    kf(xhatk,uk,zk,Phatk,Fk,Gammak,Gk,Qk,Rkp1,Hkp1)
  % KF.m  Linear Kalman Filter implementation
  nx = length(xhatk);

  % Propagate
  xbarkp1 = Fk*xhatk + Gk*uk;
  Pbarkp1 = Fk*Phatk*Fk' + Gammak*Qk*Gammak';

  % Measure
  innov = zk - Hkp1*xbarkp1;
  Skp1 = Hkp1*Pbarkp1*Hkp1' + Rkp1;
  Wkp1 = Pbarkp1*Hkp1'*inv(Skp1);
  xhatkp1 = xbarkp1 + Wkp1*innov;
  err = innov'*inv(Skp1)*innov;

  % Basic form
  Phatkp1 = (eye(nx) - Wkp1*Hkp1)*Pbarkp1;
```

```
-------------- Results ---------------------

xhatk =
   0.04035867819641
  -0.45508276484930
Phatk =
   0.00117513862431   0.00069397431279
   0.00069397431279   0.07779334749816
```


Estimate of x_1
1-σ upper bound
1-σ lower bound
x_1
1
0.5
0
-0.5
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

Estimate of x_2
1-σ upper bound
1-σ lower bound
x_2
4
2
0
-2
-4
-6
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Time step


3.5
3
2.5
2
1.5
1
0.5
0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Time step
ε(j) = v^T(j)S^-1(j)v(j)