Problem 49.

$$\mathbb{E}\{(X-g(Y)\}'R(X-g(Y))\}$$
$$= \mathbb{E}\{(X-\mathbb{E}\{X|Y\}+\mathbb{E}\{X|Y\}-g(Y)\}'R(X-\mathbb{E}\{X|Y\}+\mathbb{E}\{X|Y\}-g(Y))\}$$
$$= \mathbb{E}\{\big(X-\mathbb{E}\{X|Y\}\big)'R(X-\mathbb{E}\{X|Y\}) + (X-\mathbb{E}\{X|Y\})'R\big(\mathbb{E}\{X|Y\}-g(Y)\big) + \big(\mathbb{E}\{X|Y\}-g(Y)\big)'R(X-\mathbb{E}\{X|Y\}) + (\mathbb{E}\{X|Y\}-g(Y)\}'R(\mathbb{E}\{X|Y\}-g(Y))\}$$

Consider

$$\mathbb{E}_{X,Y}\{(X-\mathbb{E}\{X|Y\})'R\big(\mathbb{E}\{X|Y\}-g(Y)\big)+\big(\mathbb{E}\{X|Y\}-g(Y)\big)'R(X-\mathbb{E}\{X|Y\})\}$$
$$= \mathbb{E}_Y\{\mathbb{E}_{X|Y}\{(X-\mathbb{E}\{X|Y\})'R\big(\mathbb{E}\{X|Y\}-g(Y)\big)+\big(\mathbb{E}\{X|Y\}-g(Y)\big)'R(X-\mathbb{E}\{X|Y\})\}\}$$
$$= \mathbb{E}_Y\{(\mathbb{E}\{X|Y\}-\mathbb{E}\{X|Y\})'R\big(\mathbb{E}\{X|Y\}-g(Y)\big)+\big(\mathbb{E}\{X|Y\}-g(Y)\big)'R(\mathbb{E}\{X|Y\})-\mathbb{E}\{X|Y\})\}\}$$
$$= 0$$

Therefore,

$$\mathbb{E}\{(X-g(Y)\}'R(X-g(Y))\}$$
$$= \mathbb{E}\{(X-\mathbb{E}\{X|Y\})'R(X-\mathbb{E}\{X|Y\}) + (\mathbb{E}\{X|Y\}-g(Y)\}'R(\mathbb{E}\{X|Y\}-g(Y))\}$$
$$= \mathbb{E}\{(X-\mathbb{E}\{X|Y\})'R(X-\mathbb{E}\{X|Y\})\} + \mathbb{E}\{(\mathbb{E}\{X|Y\}-g(Y)\}'R(\mathbb{E}\{X|Y\}-g(Y))\}$$

Consider R is positive definite, then we have $\mathbb{E}\big\{(\mathbb{E}\{X|Y\}-g(Y)\}'R\big(\mathbb{E}\{X|Y\}-g(Y)\big)\big\} \geq 0$

So, for any function $g(Y)s$

$$\mathbb{E}\{\big(X-g(Y)\}'R\big(X-g(Y)\big)\} \geq \mathbb{E}\{(X-\mathbb{E}\{X|Y\})'R(X-\mathbb{E}\{X|Y\})\}$$

Problem 51

$$\pi_1 = P\pi_0$$

$$p(x_1=1|y_1) = \frac{p(y_1|x_1=1)\pi_{11}}{p(y_1|x_1=1)\pi_{11}+p(y_1|x_1=2)\pi_{12}}$$
$$= \frac{\exp\,(-(y_1-1))\pi_{11}}{\exp\big(-(y_1-1)\big)\,\pi_{11}+\exp\,(-(y_1-2))\pi_{12}}$$

$$p(x_1=2|y_1) = \frac{p(y_1|x_1=2)\pi_{11}}{p(y_1|x_1=1)\pi_{11}+p(y_1|x_1=2)\pi_{12}}$$
$$= \frac{\exp\,(-(y_1-2))\pi_{11}}{\exp\big(-(y_1-1)\big)\,\pi_{11}+\exp\,(-(y_1-2))\pi_{12}}$$

Problem 54

The transition probability is:

$$p(x_{t+1} = j|x_t = i) = Q_{ij}\frac{b_j}{b_i + b_j} \qquad if\ j \neq i$$

$$p(x_{t+1} = i|x_t = i) = Q_{ii} + \sum_j Q_{ij}\frac{b_i}{b_i + b_j}$$

If there exists a stationary distribution π_∞ such that

$$\pi_\infty(i)\, p(x_{t+1} = j|x_t = i) = \pi_\infty(j)\, p(x_{t+1} = i|x_t = j)$$

Then, we have:

$$\pi_\infty(i)Q_{ij}\frac{b_j}{b_i + b_j} = \pi_\infty(j)Q_{ji}\frac{b_i}{b_i + b_j}$$

Consider Q is symmetric, $Q_{ij} = Q_{ji}$.

$$\pi_\infty(i)b_j = \pi_\infty(j)b_i$$

Then we have:

$$\frac{\pi_\infty(i)}{b_i} = \frac{\pi_\infty(j)}{b_j}$$

The equation above applies to any i, j. Therefore, the stationary distribution is proportional to b_i.

Problem 55

I chose a dataset which records the Australian Credit Approval. The dataset has 14 features for each data point along with one classification label. I implemented LDA and logistic regression for the dataset. The correct classification rate for LAD is 92.75% and for logistic regression is 88.41%. The performance of two methods are close but the logistic regression needs more training time.

The MATLAB code is attached below:

```
%problem 55
clear,clc,close all
data = importdata('australian');
data = data.textdata;
[numData, numFeature] = size(data);
dataset = zeros(numData,numFeature);
for i = 1:numData
    for j = 1:numFeature
        cell_data = str2num(data{i,j});
```

```
        dataset(i,j) = cell_data(1);
    end
end
randIndices = randperm(numData);
trainingNum = round(numData*0.9);
trainingData  = dataset(randIndices(1:trainingNum),:);
testData = dataset(randIndices(trainingNum+1:end),:);
MdLinear = fitcdiscr(trainingData(:,2:end),trainingData(:,1));
predictedClass_LDA = predict(MdLinear, testData(:,2:end));
predictedClass_LDA = max(predictedClass_LDA,0);
x0 = rand(1,13)/100;
trainingFeature = trainingData(:,2:end);
trainingLabel = max(trainingData(:,1),0);
options =
optimset('PlotFcns',@optimplotfval,'MaxIter',100000,'MaxFunEvals',100000);
[x,val,etflag] =
fminunc(@(x)costFcn(x,trainingFeature,trainingLabel),x0,options);
predicted_LR = 1./(1+exp(-testData(:,2:end)*x'));
for i = 1:length(predicted_LR)
    if predicted_LR(i) >= 0.5
        predicted_LR(i) = 1;
    else
        predicted_LR(i) = 0;
    end
end
testLabel = max(testData(:,1),0);
LDA_rate= 0;
LR_rate = 0;
for i = 1:length(testLabel)
    if abs(predicted_LR(i)-testLabel(i)) < 0.01
        LR_rate = LR_rate + 1;
    end
    if abs(predictedClass_LDA(i) - testLabel(i)) < 0.01
        LDA_rate = LDA_rate + 1;
    end
end


function cost = costFcn(x,trainingData,trainingLabel)
cost = 0;
numData = size(trainingData,1);
for i = 1:numData
    dataline = trainingData(i,:);
    sumLine = sum(dataline.*x);
    y = trainingLabel(i);
    p = 1/(1+exp(-sumLine));
    cost = cost - (y*log(p) + (1-y)*log(1-p));
end
end
```

Problem 57 & 58

I use real world example that uses both MCMC and Gibbs sampling.

The Ricker model is one classical discrete population model, which gives the expected number of individuals N_{t+1} in generation $t+1$ as a function of the number of individuals in the previous generation t. This model is described by the following equation:

$$N_{t+1} = N_t \exp\{r\left(1-\frac{N_t}{K}\right)\}$$

where r is the maximum per capita growth rate, K is the environmental carrying capacity. The log-transformation of Ricker model is written as

$$x_{t+1} = x_t + a - bexp(x_t) + v_t$$

$$y_t = x_t + n_t$$

where $x_t = \log(N_t)\,, a = r, b = \frac{r}{K}, v_t \sim N(0, \sigma_1^2), n_t \sim N(0, \sigma_2^2)$.

The problem is that assume we know the prior distribution of x_0, the values of b, σ_1, σ_2 and a set of observations $\{y_1, y_2, \dots, y_T\}$, how can we estimate the posterior of a ?

$$p(x_{1:T}, a|y_{1:T}) = \frac{p(a)p(y_{1:T}|x_{1:T})p(x_{1:T})}{p(y_{1:T})} \propto p(a)p(y_{1:T}|x_{1:T})p(x_{1:T})$$

where

$$p(x_{1:T}) = p(x_1)\prod_{t=2}^{T} p(x_t|x_{t-1})$$

$$p(y_{1:T}|x_{1:T}) = \prod_{t=1}^{T} p(y_t|x_t)$$

The objective is to sample from the distribution $p(x_{1:T}, a|y_{1:T})$. However, it is very hard to analytically compute this posterior. Then we can use Gibbs sampling: we first sample $a^{i+1} \sim p(a|x_{1:T}^i, y_{1:T})$, then we sample $x_{1:T}^{i+1} \sim p(x_{1:T}|a^{i+1}, y_{1:T})$.Then the collection of $\{a^i\}_{i=1}^N$ formulates the posterior of a.

The generate steps of the algorithm are as follows:

Step 1: Define a prior of parameter $p(a)$ *and obtain a sample* $a^0 \sim p(a)$*;*

Step 2:

Loop://Gibbs sampling

$x_{1:T}^{i+1} \sim p(x_{1:T}|a^i, y_{1:T})$

$a^{i+1} \sim p(a|x_{1:T}^{i+1}, y_{1:T})$ *//MH sampling*

$i \leftarrow i + 1$

until meet some terminating conditions

Then we look at the details of two sampling step inside the Gibbs sampling loop:

1. We can use Sequential Monte Carlo (SMC) to sampling from $p(x_{1:T}|a^i, y_{1:T})$.
   Step 1: $\{x_0^i\}_{i=1}^N \sim p(x_0)$
   Step 2: for each i
   $X_{1:T}^i \leftarrow \{\}, w^i \leftarrow 1$
   for $t = 1{:}T$
   $x_t^i \sim p(x_t|x_{t-1}^i, a^i)$
   $X_{1:T}^i = \{X_{1:T}^i, x_t^i\}$
   $w^i \leftarrow w^i p(y_t|x_t^i)$
   $\{X_{1:T}^i, w^i\}$
   Sample $X_{1:T}^i$ according to the corresponding w^i

2. Use Metropolis-Hastings (MH) algorithm to sample from $a^{i+1} \sim p(a|x_{1:T}^{i+1}, y_{1:T})$

   Sample from prior $a_0^{i+1} \sim p(a)$

   Sample $a_1^{i+1} \sim q(a|a_0^{i+1})$

   $A^{i+1} = \{\}$

   Loop $k = 1:K$:

   $$p\left(a_{k-1}^{i+1}\middle|x_{1:T}^{i+1}, y_{1:T}\right)$$
   $$\propto p\left(a_{k-1}^{i+1}\right)p\left(x_{1:T}^{i+1}, y_{1:T}\middle|a_{k-1}^{i+1}\right)$$
   $$= p\left(a_{k-1}^{i+1}\right)p(x_0)\prod_{t=1}^{T} p\left(y_t\middle|x_t^{i+1}\right)\prod_{t=1}^{T} p\left(x_t\middle|x_{t-1}, a_{k-1}^{i+1}\right) = \tilde{p}(a_{k-1}^{i+1})$$
   $$p\left(a_k^{i+1}\middle|x_{1:T}^{i+1}, y_{1:T}\right)$$
   $$\propto p\left(a_k^{i+1}\right)p\left(x_{1:T}^{i+1}, y_{1:T}\middle|a_k^{i+1}\right)$$
   $$= p\left(a_k^{i+1}\right)p(x_0)\prod_{t=1}^{T} p\left(y_t\middle|x_t^{i+1}\right)\prod_{t=1}^{T} p\left(x_t\middle|x_{t-1}, a_k^{i+1}\right) = \tilde{p}(a_k^{i+1})$$

   If $\tilde{p}\left(a_k^{i+1}\right)q\left(a_{k-1}^{i+1}\middle|a_k^{i+1}\right) > \tilde{p}(a_{k-1}^{i+1})q(a_k^{i+1}|a_{k-1}^{i+1})$

   $A^{i+1} = \{A^{i+1}, a_k^{i+1}\}$

   else

   $u \sim U(0,1)$

   If $u \leq \frac{\tilde{p}\left(a_k^{i+1}\right)q\left(a_{k-1}^{i+1}\middle|a_k^{i+1}\right)}{\tilde{p}(a_{k-1}^{i+1})q(a_k^{i+1}|a_{k-1}^{i+1})}$

   $A^{i+1} = \{A^{i+1}, a_k^{i+1}\}$

   else

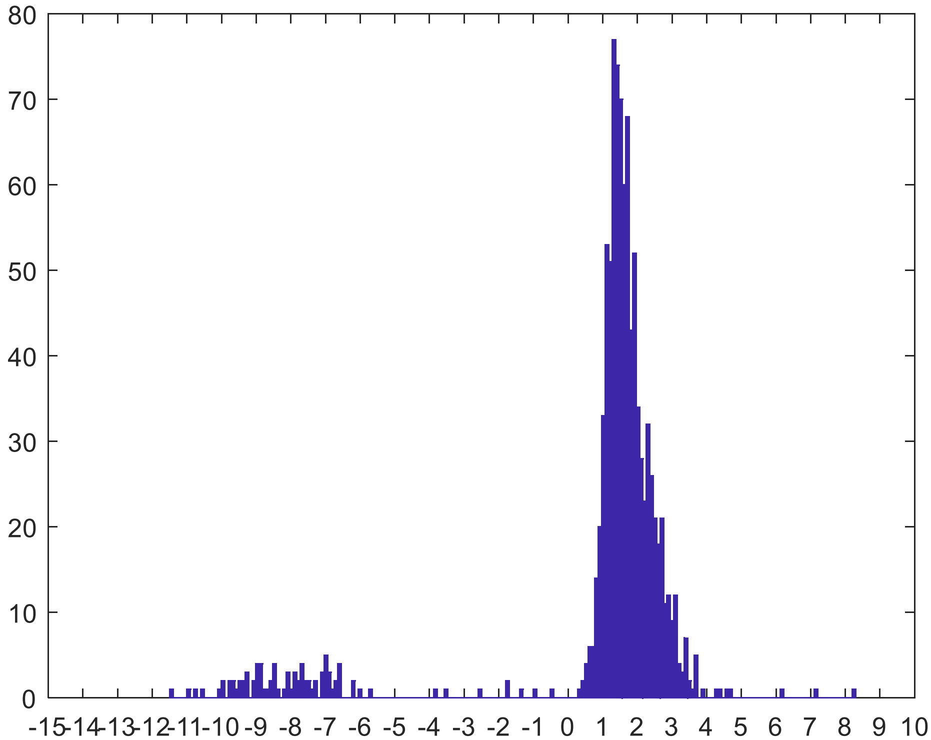
   $A^{i+1} = \{A^{i+1}, a_{k-1}^{i+1}\}$

   Uniformly sample from A^{i+1}, we obtain a^{i+1}

In the implementation, I set the known parameters to be $b = 0.5, \sigma_1^2 = 0.5, \sigma_2^2 = 1, p(a) = N(0,100), p(x_0) = N(0,1)$.

Generate a set of observations $\{y_1, y_2, \ldots, y_T\}$ with $T = 40$ and $a = 1$.

Then, I run the algorithm and the posterior of the parameter a is shown as the following plot:



As we can observe, the peak of the distribution is very closed to the ground truth value $a = 1$.The mean of all samples is 0.9418, which is closed to 1 and the variance of the posterior is 8.0818, which is significantly reduced compared to the prior.

MATLAB code:

```
%p57,p58
clc,clear,close all
%generate observations Y
a = 1;
b = 0.5;
omega_v = .5;
omega_n = 1;
x0 = normrnd(0,1,1);
T = 40;
Y = zeros(1,T);
x_cur = x0;
for i = 1:T
    x_cur = x_cur + a - b*exp(x_cur) + normrnd(0,omega_v);
    y = x_cur + normrnd(0,omega_n);
```

```
    Y(i) = y;
end

%estimate the parameters a, b
ab = mvnrnd([0,0],[100,0;0,2]);
[X,x0] = sampleX(ab(1),b,T,omega_v);
N = 1000; %gibbs sampling iterations
%Gibbs sampling loop
As = zeros(1,N);
Bs = zeros(1,N);
for i = 1:N
    paramSamples = MCMC_params(X,Y,omega_v,omega_n,x0);
    param = paramSamples(:,unidrnd(size(paramSamples,2)))
    [X,x0] = SMCTraj(param(1),b,omega_v,omega_n,Y,T);
    As(i) = param(1);
    Bs(i) = param(2);
end


function [X,x0] = SMCTraj(a,b,omega_v,omega_n,Y,T)
N = 8000;
x0 = normrnd(0,1,[N,1]);
sampleTrajs = zeros(N,T+2);
sampleTrajs(:,2) = x0;
for j = 1:N
    x_cur = x0(j);
    for i = 1:T
        x_cur = x_cur + a - b*exp(x_cur) + normrnd(0,omega_v);
        sampleTrajs(j,i+2) = x_cur;
        y = Y(i);
        sampleTrajs(j,1) = sampleTrajs(j,1) +
log(max(normpdf(y,x_cur,omega_n),exp(-100)));
    end

end
    [~,sampleTrajIndex] = max(sampleTrajs(:,1));
    x0 = sampleTrajs(sampleTrajIndex,2);
    X = sampleTrajs(sampleTrajIndex,3:end);
end
function [X,x0] = sampleX(a,b,T,omega_v)
X = zeros(1,T);
x0 = normrnd(0,1,1);
x_cur = x0;
for i = 1:T
    x_cur = x_cur + a - b*exp(x_cur) + normrnd(0,omega_v);
    X(i) = x_cur;
end
end

function params = MCMC_params(X,Y,omega_v,omega_n,x0)
ab0 = mvnrnd([0,0],[2,0;0,2]);
N = 8000;
%a = a0;b = b0;
a_prev = ab0(1);b_prev = 0.5;%b_prev = ab0(2);
params = zeros(2,N);
%params(:,1) = [a0;b0];
```

```
for i = 1:N
    ab = mvnrnd([a_prev,b_prev],[0.005,0;0,0.005]);%proposal distribution
    a = ab(1);
    %b = ab(2);
    %%%%
    b = 0.5;
    %%%
    logProb1 = evalTrajectoryLogProb(a_prev,b_prev,omega_v,omega_n,X,Y,x0);
    logProb2 = evalTrajectoryLogProb(a,b,omega_v,omega_n,X,Y,x0);
    logProb1 = logProb1 + normpdf(a,a_prev,0.005);
    logProb2 = logProb2 + normpdf(a_prev,a,0.005);
    if logProb2 > logProb1
        params(:,i) = [a,b];
        a_prev = a;
        b_prev = b;
    else
        u = rand();
        if u <= exp(logProb2-logProb1)
            params(:,i) = [a;b];
            a_prev = a;
            b_prev = b;
        else
            params(:,i) = [a_prev;b_prev];
        end
    end
end
end


function logProb = evalTrajectoryLogProb(a,b,omega_v,omega_n,X,Y,x0)
logProb = log(normpdf(x0,3,1));
x_prev = x0;
for i = 1:length(X)
    x = X(i);
    y = Y(i);
    logProb = logProb + log(max(normpdf(x,x_prev + a -
b*exp(x_prev),omega_v),exp(-100))) + log(max(normpdf(y,x,omega_n),exp(-
100)));
end
end
```

Reference:

Gao, M., Chang, X., & Wang, X. (2012). Bayesian parameter estimation in dynamic population model via particle Markov chain Monte Carlo.

Problem 59

Suppose $\{x_k\}$ is an one-dimensional stochastic process. In the simulation, I set $x_0 \sim N(0{,}1000)$ and the process noise $\omega_k \sim N(0{,}2)$.

Given the sample set at time step k: $\{x_k^i\}_{i=1}^N$

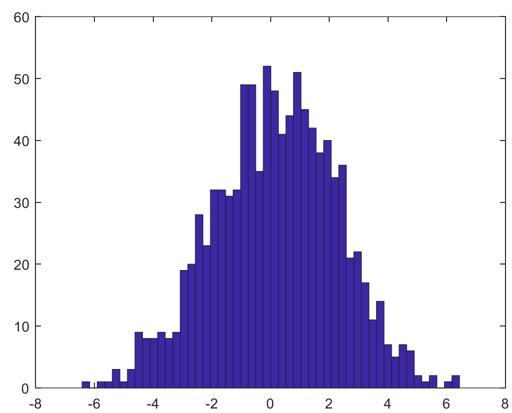
For $i = 1{:}\,N$

$$x_{k+1}^i \sim N(\sin(x_k^i), 2)$$

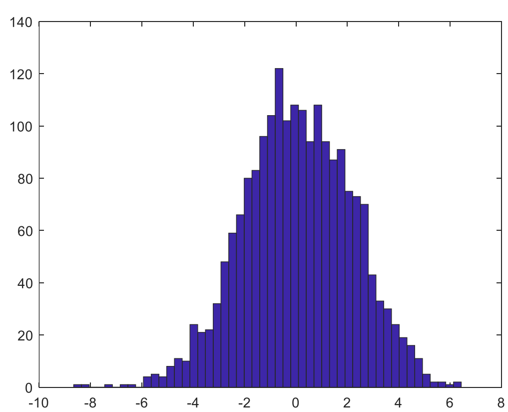
endFor

The Distribution of x_k at $k = 1{,}2{,}3$

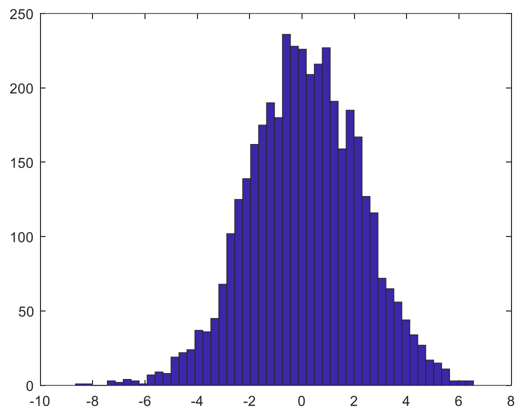
$k = 1$



$k = 2$



$k = 3$

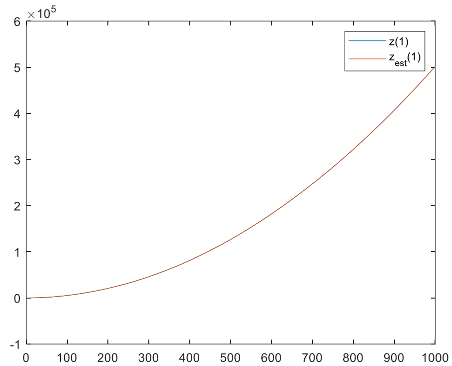


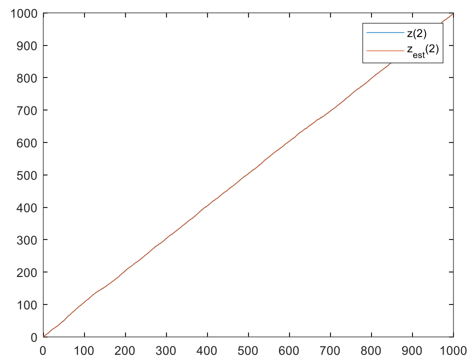
MATLAB code:

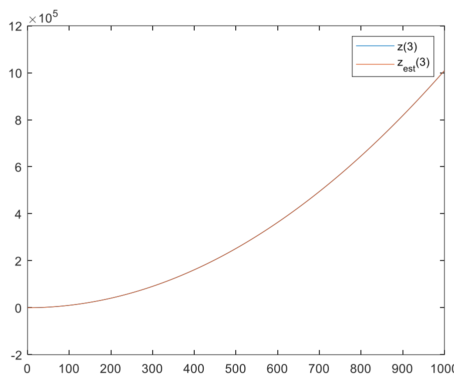
```
clc,clear,close all
x0 = normrnd(0,1000,[1,1000]);
T = 1000;
omega_var = 2;
x = x0;
x_next = [];
for i = 1:T
    for j = 1:length(x)
        x_next = [x_next,normrnd(sin(x(j)),omega_var)];
    end
    x = x_next;
    hist(x,50)
end
```

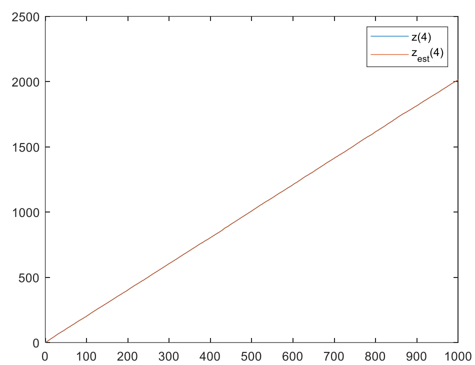
Problem 61.

There are 4 state variables. After applying Kalman filter to the problem , we can obtain the time history of real state variables and estimated state variables as follows:









The mean absolute errors of the four state estimations are $e(1) = 0.3243, e(2) = 0.3380. e(3) = 0.3345, e(4) = 0.3453$. Also, given that the acceleration is constant, the trajectories of velocities are lines and the trajectories of positions are parabola.

The MATLAB code is shown as follows:

```
clc,clear,close all
T = 0.1;
A = [1 T 0 0;0 1 0 0;0 0 1 T;0 0 0 1];
C = [1 0 0 0;0 0 1 0];
f = [T^2/2 0;T 0;0 T^2/2;0 T];
r = [1;2];
z0 = [0;0.1;0;-0.1];
z = z0;
sigma_ob = 1;
sigma_process = 0.1;
Q = diag(sigma_process^2*[1,1,1,1]);
R = diag(sigma_ob^2*[1,1]);
P = eye(4);
z_est0 = zeros(4,1);
z_est = z_est0;
```

```
z_array = [];
z_est_array = [];
tspan = 10000;
for i = 1:tspan
    omega = normrnd(0,sigma_process,[4,1]);
    niu = normrnd(0,sigma_ob,[2,1]);
    z = A*z + f*r + omega;
    y = C*z + niu;
    z_est = A*z_est + f*r;
    P = A*P*A'+Q;
    K = P*C'*inv(C*P*C' + R);
    z_est = z_est + K*(y-C*z_est);
    P = (eye(4) - K*C)*P;
    z_array = [z_array,z];
    z_est_array = [z_est_array,z_est];
end
figure(1)
plot(T*(1:tspan),z_array(1,:),T*(1:tspan),z_est_array(1,:))
legend('z(1)','z_{est}(1)')
figure(2)
plot(T*(1:tspan),z_array(2,:),T*(1:tspan),z_est_array(2,:))
legend('z(2)','z_{est}(2)')
figure(3)
plot(T*(1:tspan),z_array(3,:),T*(1:tspan),z_est_array(3,:))
legend('z(3)','z_{est}(3)')
figure(4)
plot(T*(1:tspan),z_array(4,:),T*(1:tspan),z_est_array(4,:))
legend('z(4)','z_{est}(4)')
mean(abs(z_array(1,:)-z_est_array(1,:)))
mean(abs(z_array(2,:)-z_est_array(2,:)))
mean(abs(z_array(3,:)-z_est_array(3,:)))
mean(abs(z_array(4,:)-z_est_array(4,:)))
```

Problem 62.

(a). The MATLAB code is attached as follows:

```
clc,clear,close all
tspan = 1000;
S = [0, 10, 20];
P = [0.1,0.1,0.8;0.3,0.3,0.4;0.2,0.2,0.6];
b = [0.3,0.3,0.4];
sigma_sq = 5;
pi_s = b;
p_error = 0;
for i =1:tspan
    b = b*P;
    x = 10*(find(mnrnd(1,b))-1);
    y = normrnd(x,sqrt(sigma_sq));
    pi_s_unnorm = [normpdf(y,0,sqrt(sigma_sq))*pi_s*P(:,1),...
            normpdf(y,10,sqrt(sigma_sq))*pi_s*P(:,2),...
            normpdf(y,20,sqrt(sigma_sq))*pi_s*P(:,3)];
    pi_s =  pi_s_unnorm/sum(pi_s_unnorm);
    p_error = 1 - max(pi_s) + p_error;
end
ave_p_error = p_error/tspan
```

The average probability error corresponding to difference measurement noise are listed as follows:s

$$\sigma^2 = 1,\ \bar{p}_{error} = 2.4893 \times 10^{-8},$$

$$\sigma^2 = 2,\ \bar{p}_{error} = 2.5257 \times 10^{-4},$$

$$\sigma^2 = 5,\ \bar{p}_{error} = 0.0153,$$

(b). The MATLAB code is shown as follows:

```
clc,clear,close all
tspan = 1000;
S = [0, 10, 20];
P = [0.1,0.1,0.8;0.3,0.3,0.4;0.2,0.2,0.6];
b0 = [0.3,0.3,0.4];
sigma_sq = 1;
pi_s = b0;
b = b0;
p_error = 0;
ys = [];
for i =1:tspan
    b = b*P;
    x = 10*(find(mnrnd(1,b))-1);
    y = normrnd(x,sqrt(sigma_sq));
    ys = [ys,y];
    pi_s_unnorm = [normpdf(y,0,sqrt(sigma_sq))*pi_s*P(:,1),...
            normpdf(y,10,sqrt(sigma_sq))*pi_s*P(:,2),...
            normpdf(y,20,sqrt(sigma_sq))*pi_s*P(:,3)];
    pi_s =  pi_s_unnorm/sum(pi_s_unnorm);
    p_error = 1 - max(pi_s) + p_error;
end
p_error_tot = 0;
for i = 1:tspan
    %forward
```

```
    pi_s = b0;
    for j = 1:i
        pi_s_unnorm = [normpdf(ys(j),0,sqrt(sigma_sq))*pi_s*P(:,1),...
            normpdf(ys(j),10,sqrt(sigma_sq))*pi_s*P(:,2),...
            normpdf(ys(j),20,sqrt(sigma_sq))*pi_s*P(:,3)];
        pi_s =  pi_s_unnorm/sum(pi_s_unnorm);
    end
    %backward
    beta = [1,1,1]';
    for k = tspan:-1:i
        beta = 
P*diag([normpdf(ys(k),0,sqrt(sigma_sq)),normpdf(ys(k),10,sqrt(sigma_sq)),norm
pdf(ys(k),20,sqrt(sigma_sq))])*beta;
        beta = beta/sum(beta);
    end
    pi_tot = [pi_s(1)*beta(1),pi_s(2)*beta(2),pi_s(3)*beta(3)];
    pi_tot = pi_tot/sum(pi_tot);
    p_error_tot = 1 - max(pi_tot) + p_error_tot;
end
ave_p_error = p_error/tspan
ave_p_error_tot = p_error_tot/tspan
```

$$\sigma^2 = 1,\ \bar{p}_{error} = 1.6569 \times 10^{-8},$$

$$\sigma^2 = 2,\ \bar{p}_{error} = 1.8695 \times 10^{-4},$$

$$\sigma^2 = 5,\ \bar{p}_{error} = 0.0141$$

As we can observe that, as the measurement noise increases, the average probability error increases. Also, optimal smoother has less average probability error than optimal filter given a measurement noises.