Question 1.

(a). The signals sent by two channels can be modeled as a 2-state Markov chain. The measurement of the signals are corrupted by additive a Gaussian noise and a sinusoidal bias.

(b). Suppose the posterior of estimated state at time step k is

$$\pi_k = p(s_k|y_1, y_2, \dots, y_k) = [p_k, 1 - p_k]$$

The transition probability $P_{ij} = p(s_{k+1} = j|s_k = i)$.

The emission probability $p(y_k|s_k = j) = N(j + Asin(\omega k), 1)$

Given the posterior of estimated state at time k, the transition probability and emission probability $p(y_k|s_k)$. The posterior of estimated state at time $k + 1$ is computed as follows:

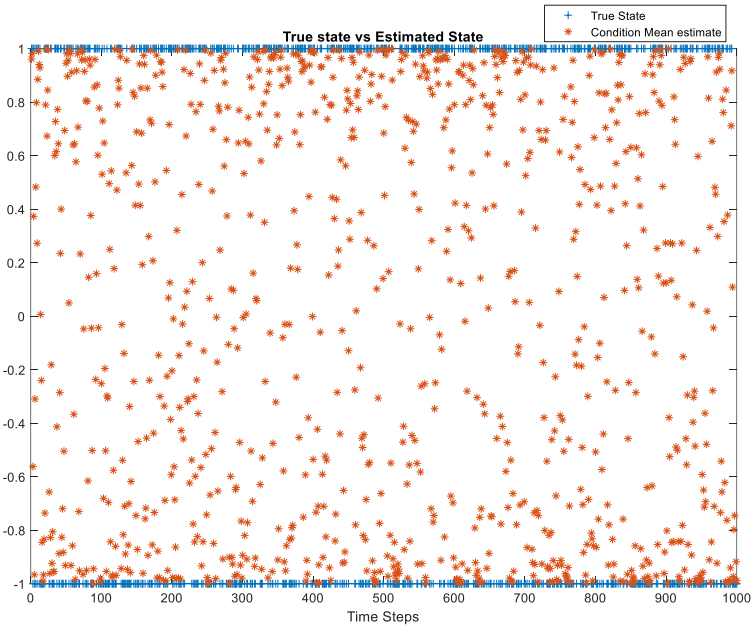
$$p_{k+1} = \pi_{k+1}(-1) = p(s_{k+1} = -1|y_1, y_2, \dots, y_{k+1})$$

$$= \frac{p(y_{k+1}|s_{k+1} = -1)\sum_i p(s_k = i|y_1, y_2, \dots, y_k)P_{i1}}{\sum_l p(y_{k+1}|s_{k+1} = l)\sum_i p(s_k = i|y_1, y_2, \dots, y_k)P_{il}}$$

$$= \frac{p(y_{k+1}|s_{k+1} = -1)\{p_k P_{11} + (1 - p_k)P_{21}\}}{\sum_l p(y_{k+1}|s_{k+1} = l)\{p_k P_{1l} + (1 - p_k)P_{2l}\}}$$

$\pi_{k+1} = [p_{k+1}, 1 - p_{k+1}]$.

(c). The plot of conditional mean estimate is shown as the following plot:

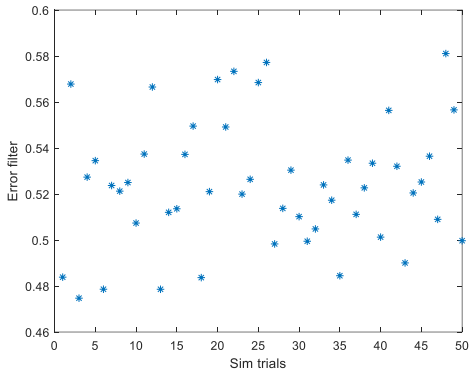


MATLAB code:

```
clc,clear,close
P = [0.8,0.2;0.2,0.8];
pi0 = [0.1,0.9];
tspan = 1000;
y = zeros(1,tspan);
s_true = zeros(1,tspan);
pi_d = pi0;
A = 0.5;
omega = 5;
%genenrate true state sequence and observation vector (y)
for i = 1:tspan
    s = find(mnrnd(1,pi_d)) - 2;
    if s == 0
        s = 1;
    end
    s_true(i) = s;
    y(i) = s + randn + A*sin(omega*i);
    pi_d = pi_d*P;
    pi_d = pi_d / sum(pi_d);
end

p_prv = 0.5;
p_nxt = p_prv;
s_est = zeros(1,tspan);
%implementation of filter
for i = 1:tspan
    p_nxt = normpdf(y(i),-1 + A*sin(omega*i),1)*(p_prv * P(1,1) + (1 -
p_prv)* P(2,1))/(normpdf(y(i),-1 + A*sin(omega*i),1)*(p_prv * P(1,1) + (1 -
p_prv)* P(2,1)) + ...
        normpdf(y(i),1 + A*sin(omega*i),1)*(p_prv * P(1,2) + (1 - p_prv)*
P(2,2)));
    s_est(i) = -p_nxt + (1 - p_nxt);
    p_prv = p_nxt;
end

plot(1:tspan,s_true,'+',1:tspan,s_est,'*');
title('True state vs Estimated State');
legend('True State','Condition Mean estimate');
xlabel('Time Steps')
```

(d). Through 50 independent simulations, the mean squared error of the state estimate is $0.5244s$.

The distribution of the 50 simulations is shown in the following plot:



MATLAB code:

```
clc,clear,close
P = [0.8,0.2;0.2,0.8];
pi0 = [0.1,0.9];
tspan = 1000;
num_sim = 50;
A = 0.5;
omega = 5;
errors = zeros(1,num_sim);
for j = 1:num_sim
    pi_d = pi0;
    y = zeros(1,tspan);
    s_true = zeros(1,tspan);
    %genenrate true state sequence and observation vector (y)
    for i = 1:tspan
        s = find(mnrnd(1,pi_d)) - 2;
        if s == 0
            s = 1;
        end
        s_true(i) = s;
        y(i) = s + randn + A*sin(omega*i);
        pi_d = pi_d*P;
        pi_d = pi_d / sum(pi_d);
    end

    p_prv = 0.5;
    p_nxt = p_prv;
    s_est = zeros(1,tspan);
    %implementation of filter
    for i = 1:tspan
        p_nxt = normpdf(y(i),-1 + A*sin(omega*i),1)*(p_prv * P(1,1) + (1 -
p_prv)* P(2,1))/(normpdf(y(i),-1 + A*sin(omega*i),1)*(p_prv * P(1,1) + (1 -
p_prv)* P(2,1)) + ...
            normpdf(y(i),1 + A*sin(omega*i),1)*(p_prv * P(1,2) + (1 - p_prv)*
P(2,2)));
        s_est(i) = -p_nxt + (1 - p_nxt);
```

```
        p_prv = p_nxt;
    end
    errors(j) = norm(s_true - s_est);
end
plot(1:num_sim,errors,'*')
xlabel('Sim trials')
ylabel('Error')
```

(e). Given the observation sequence $Y = [y_1, y_2, \dots, y_N]$

$$p(s_k|y_1, y_2, \dots, y_N) = \frac{p(s_k|y_1, y_2, \dots y_k)p(y_{k+1}, y_{k+2}, \dots, y_N|s_k)}{p(y_1, y_2, \dots, y_N)}$$
$$\propto p(s_k|y_1, y_2, \dots y_k)p(y_{k+1}, y_{k+2}, \dots, y_N|s_k)$$

where $p(s_k|y_1, y_2, \dots y_k)$ can be computed through optimal filter.

Assumptions: All conditional probability models are time invariant and all parameters are known.

Denote $\beta_k = p(y_{k+1}, y_{k+2}, \dots, y_N|s_k)$. Consider we have a 2-state Markov chain, $\beta_k = [p'_k, 1 - p'_k]$

Then the recursion is:

$$\beta_k = \sum_{s_{k+1}} \beta_{k+1} p(y_{k+1}|s_{k+1}) p(s_{k+1}|s_k)$$

Thus, given $\beta_{k+1} = [p'_{k+1}, 1 - p'_{k+1}]$,

$$p'_k = p'_{k+1} p(y_{k+1}|s_{k+1} = -1) P_{11} + (1 - p'_{k+1}) p(y_{k+1}|s_{k+1} = 1) P_{12}$$

$\beta_k = [p'_k, 1 - p'_k]$.

Denote un-normalized distribution at time step k as $\tilde{\boldsymbol{\gamma}}_k = p(s_k|y_1, y_2, \dots y_k)p(y_{k+1}, y_{k+2}, \dots, y_N|s_k) = [\gamma_k^1, \gamma_k^2]$, and normalized distribution as $\boldsymbol{\gamma}_k$.

The smooth algorithm is as follows:

for state s_k at each time step:

$$p_k = \frac{p(y_k|s_k = -1)\{p_{k-1}P_{11} + (1 - p_{k-1})P_{21}\}}{\sum_l p(y_k|s_k = l)\{p_{k-1}P_{1l} + (1 - p_{k-1})P_{2l}\}}$$

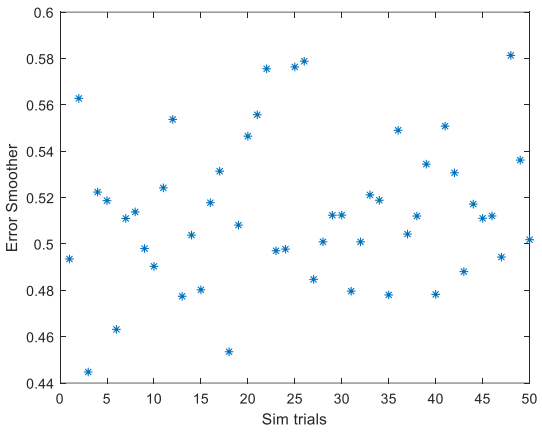
$$p'_k = p'_{k+1} p(y_{k+1}|s_{k+1} = -1) P_{11} + (1 - p'_{k+1}) p(y_{k+1}|s_{k+1} = 1) P_{12}$$

$$\gamma_k^1 = p_k p'_k$$

$$\gamma_k^2 = (1 - p_k)(1 - p'_k)$$

$$\boldsymbol{\gamma}_k = [\frac{\gamma_k^1}{\gamma_k^1 + \gamma_k^2}, \frac{\gamma_k^2}{\gamma_k^1 + \gamma_k^2}]$$

(f). The errors of 50 simulations are shown in the figure below and the mean squared error is 0.5141, which is less than the mean squared error of the filter-based state estimation:



MATLAB code:

```
clc,clear,close
P = [0.8,0.2;0.2,0.8];
pi0 = [0.1,0.9];
tspan = 1000;
num_sim = 50;
A = 0.5;
omega = 5;
errors = zeros(1,num_sim);
errors_filter = zeros(1,num_sim);
for j = 1:num_sim
    pi_d = pi0;
    y = zeros(1,tspan);
    s_true = zeros(1,tspan);
    %genenrate true state sequence and observation vector (y)
    for i = 1:tspan
        s = find(mnrnd(1,pi_d)) - 2;
        if s == 0
            s = 1;
        end
        s_true(i) = s;
        y(i) = s + randn + A*sin(omega*i);
        pi_d = pi_d*P;
        pi_d = pi_d / sum(pi_d);
    end

    p_nxt = 0.5;
    s_est = zeros(1,tspan);
    s_est_filter = zeros(1,tspan);
    %implementation of filter
    for i = 1:tspan
        %forward filter
```

```
        p_nxt = normpdf(y(i),-1 + A*sin(omega*i),1)*(p_nxt * P(1,1) + (1 -
p_nxt)* P(2,1))/(normpdf(y(i),-1 + A*sin(omega*i),1)*(p_nxt * P(1,1) + (1 -
p_nxt)* P(2,1)) + ...
            normpdf(y(i),1 + A*sin(omega*i),1)*(p_nxt * P(1,2) + (1 - p_nxt)*
P(2,2)));
        p_back = [1;1];
        %backward algorithm
        for k = tspan:-1:i
            B = diag([normpdf(y(k),-1 + A*sin(omega*k),1),normpdf(y(k),1 +
A*sin(omega*k),1)]);
            p_back = P*B*p_back;
            p_back = p_back/sum(p_back);
        end
        gamma1 = p_nxt*p_back(1);
        gamma2 = (1 - p_nxt) * p_back(2);
        gammas = [gamma1, gamma2]/(gamma1 + gamma2);
        s_est(i) = -gammas(1) + gammas(2);
        s_est_filter(i) = -p_nxt + (1 - p_nxt);
    end
    errors(j) = norm(s_true - s_est);
    errors_filter(j) = norm(s_true - s_est_filter);
end
plot(1:num_sim,errors,'*')
xlabel('Sim trials')
ylabel('Error Smoother')
figure
plot(1:num_sim,errors_filter,'*')
xlabel('Sim trials')
ylabel('Error filter')
```

(g). Derivation of ML estimator using EM algorithm for amplitude A.

$$\ln p(Y,S|A) = \ln \prod_{i=1}^{N} p(y_i|s_i)p(s_i|s_{i-1}) = \sum_{i=1}^{N} \ln p(y_i|s_i) + \sum_{i=1}^{N} \ln p(s_i|s_{i-1})$$
$$= \sum_{i=1}^{N} \sum_{j\in\{-1,1\}} I(s_i = j) \ln p(y_i|s_i = j)$$
$$+ \sum_{i=1}^{N} \sum_{j,k\in\{-1,1\}} I(s_i = j, s_{i-1} = k) \ln p(s_i = j|s_{i-1} = k)$$
$$= \sum_{i=1}^{N} \sum_{j\in\{-1,1\}} I(s_i = j)\{\ln \frac{1}{\sqrt{2\pi}} - \frac{1}{2}(y_i - j - Asin(\omega i)^2\}$$
$$+ \sum_{i=1}^{N} \sum_{j,k\in\{-1,1\}} I(s_i = j, s_{i-1} = k) \ln P_{i-1,i}$$

Then the auxiliary likelihood is as follows:

$$Q(A^I, A) = \sum_{i=1}^{N} \sum_{j\in\{-1,1\}} \gamma_i(j)\{\ln \frac{1}{\sqrt{2\pi}} - \frac{1}{2}(y_i - j - Asin(\omega i)^2\} + \sum_{i=1}^{N} \sum_{j,k\in\{-1,1\}} \gamma_i(j,k) \ln P_{kj}$$

$$= const - \sum_{i=1}^{N} \sum_{j\in\{-1,1\}} \frac{\gamma_i(j)}{2}(y_i - j - Asin(\omega i)^2 + \sum_{i=1}^{N} \sum_{j,k\in\{-1,1\}} \gamma_i(j,k) \ln P_{kj}$$

where:

$\gamma_i(j) = p(s_i = j|y_1, y_2, \dots, y_N, A^I)$,

$\gamma_i(j,k) = p(s_i = j, s_{i-1} = k|y_1, y_2, \dots, y_N, A^I)$

$$= \frac{p(s_{i-1} = k|y_1, y_2, \dots, y_{i-1}, A^I)P_{kj}p(y_i|s_i = j, A^I)p(y_{i+1}, y_{i+2}, \dots, y_N|s_i = j, A^I)}{\sum_{s_i, s_{i-1}\in\{-1,1\}} p(s_{i-1}|y_1, y_2, \dots, y_{i-1}, A^I)P_{jk}p(y_i|s_i, A^I)p(y_{i+1}, y_{i+2}, \dots, y_N|s_i, A^I)}$$

Then the EM algorithm goes as follows:

E step:

$$Q(A^I, A) = const - \sum_{i=1}^{N} \sum_{j\in\{-1,1\}} \frac{\gamma_i(j)}{2}(y_i - j - Asin(\omega i)^2 + \sum_{i=1}^{N} \sum_{j,k\in\{-1,1\}} \gamma_i(j,k) \ln P_{kj}$$

M step:

$$A^{I+1} = \underset{A}{\operatorname{argmax}}\, Q(A^I, A)$$

Compute the derivative of $Q(A^I, A)$ with respect to A.

$$0 = \sum_{i=1}^{N} \sum_{j\in\{-1,1\}} \gamma_i(j)\{y_i - j - A\sin(\omega i)\}\sin(\omega i) = \sum_{i=1}^{N} \{y_i - (\gamma_i(1) - \gamma_i(-1) - A\sin(\omega i))\sin(\omega i)$$

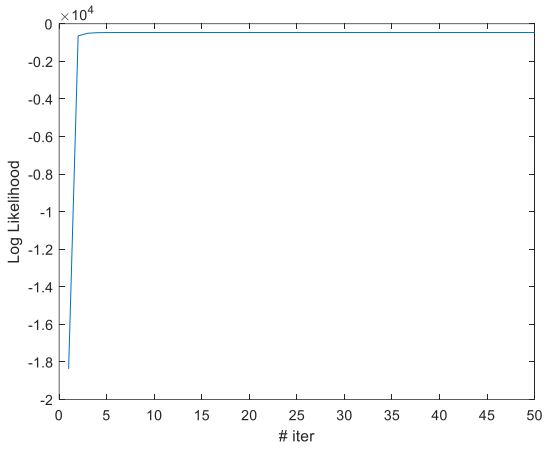
$$A^{I+1} = \frac{\sum_{i=1}^{N}\{(y_i - (\gamma_i(1) - \gamma_i(-1))\sin(\omega i)\}}{\sum_{i=1}^{N}\sin^2(\omega i)}$$

(h). We evaluate the log likelihood of the dataset given a parameter A^I:

$$logP(Y|A^I) = Q(A^I, A^I)$$

Run the EM algorithm and use $A = 0.5$ to generate data and the final estimate of the parameter is $\hat{A} = 0.5142$.

The time history of log likelihood is shown as follows:



MATLAB code:

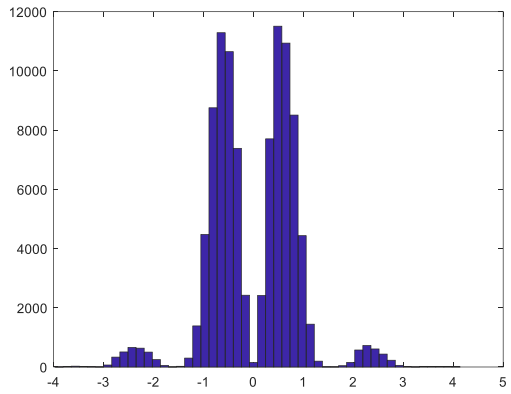
```
%question 1 EM
clc,clear,close
P = [0.8,0.2;0.2,0.8];
pi0 = [0.1,0.9];
tspan = 1000;
num_sim = 50;
A = 0.5;
omega = 5;
pi_d = pi0;
y = zeros(1,tspan);
s_true = zeros(1,tspan);
%genenrate true state sequence and observation vector (y)
for i = 1:tspan
    s = find(mnrnd(1,pi_d)) - 2;
    if s == 0
        s = 1;
    end
    s_true(i) = s;
    y(i) = s + randn + A*sin(omega*i);
    pi_d = pi_d*P;
```

```
    pi_d = pi_d / sum(pi_d);
end
%EM algorithm
A = 10;
num_iter = 50;
Logliks = zeros(1,num_iter);
for i = 1:num_iter%iterate 100 times
    % E step
    [gamma_marginal, gamma_joint] = computePosterior(y, A, tspan);
    Logliks(i) = evalLoglik(gamma_marginal, gamma_joint, A, omega, tspan, y);
    % M step
    A = sum((y - (-gamma_marginal + (1 -
gamma_marginal))).*sin(omega*(1:tspan)))/sum(sin(omega*(1:tspan)).*sin(omega*
(1:tspan)));
end
plot(1:num_iter, Logliks)
function [gamma_marginal, gamma_joint] = computePosterior(y, A, tspan)
forward_prob = zeros(1,tspan);
backward_prob = zeros(1,tspan);
gamma_marginal = zeros(1,tspan);%prob of being -1
gamma_joint = zeros(4,tspan-1);%(-1,-1), (-1,1), (1,-1), (1,1)
P = [0.8,0.2;0.2,0.8];
omega = 5;
p_back = [1;1];
%backward algorithm
for k = tspan:-1:1
    B = diag([normpdf(y(k),-1 + A*sin(omega*k),1),normpdf(y(k),1 +
A*sin(omega*k),1)]);
    p_back = P*B*p_back;
    p_back = p_back/sum(p_back);
    backward_prob(k) = p_back(1);
end
p_nxt = 0.5;
for i = 1:tspan
    %forward filter
    p_nxt = normpdf(y(i),-1 + A*sin(omega*i),1)*(p_nxt * P(1,1) + (1 -
p_nxt)* P(2,1))/(normpdf(y(i),-1 + A*sin(omega*i),1)*(p_nxt * P(1,1) + (1 -
p_nxt)* P(2,1)) + ...
        normpdf(y(i),1 + A*sin(omega*i),1)*(p_nxt * P(1,2) + (1 - p_nxt)*
P(2,2)));
    gamma1 = p_nxt*backward_prob(i);
    gamma2 = (1 - p_nxt) * (1 - backward_prob(i));
    gammas = [gamma1, gamma2]/(gamma1 + gamma2);
    gamma_marginal(i) = gammas(1);
    forward_prob(i) = p_nxt;
end

for i = 1:tspan-1
    %s_{i-1} = -1, s_i = -1
    gamma_joint(1,i) = forward_prob(i)*P(1,1)*normpdf(y(i+1),-1 +
A*sin(omega*(i+1)),1)*backward_prob(i+1);
    %s_{i-1} = -1, s_i = 1
    gamma_joint(2,i) = forward_prob(i)*P(1,2)*normpdf(y(i+1),1 +
A*sin(omega*(i+1)),1)*(1 - backward_prob(i+1));
    %s_{i-1} = 1, s_i = -1
    gamma_joint(3,i) = (1 - forward_prob(i))*P(2,1)*normpdf(y(i+1),-1 +
A*sin(omega*(i+1)),1)* backward_prob(i+1);
```

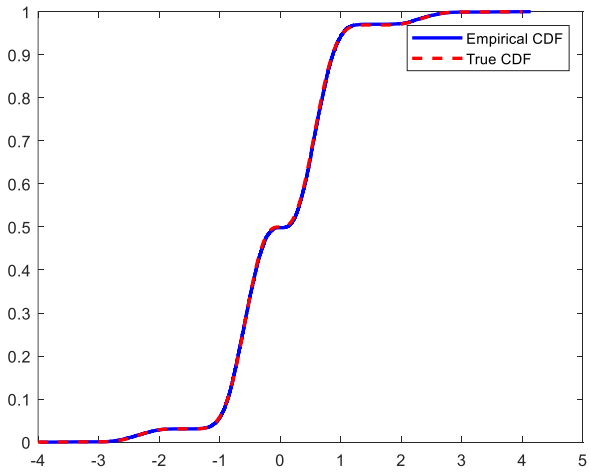
```
    %s_{i-1} = 1, s_i = 1
    gamma_joint(4,i) = (1 - forward_prob(i))*P(2,2)*normpdf(y(i+1),1 +
A*sin(omega*(i+1)),1)* (1 - backward_prob(i+1));
    gamma_joint(:,i) = gamma_joint(:,i)/sum(gamma_joint(:,i));
end

end

function Loglik = evalLoglik(gamma_marginal, gamma_joint, A, omega, tspan, y)
P = [0.8,0.2;0.2,0.8];
Loglik = -sum(gamma_marginal/2.*(y + 1 - A*sin(omega*(1:tspan))).^2) - sum((1
- gamma_marginal)/2.*(y - 1 - A*sin(omega*(1:tspan))).^2);

for i = 1:size(gamma_joint,1)
    gamma = gamma_joint(:,i);
    Loglik = Loglik + gamma(1)*log(P(1,1)) + gamma(2)*log(P(1,2)) +
gamma(3)*log(P(2,1)) + gamma(4)*log(P(2,2));
end
end
```

Question 2.

(a). Using MH algorithm to simulate the distribution, the histogram of the data points is shown as follows:



The empirical and the true CDFs are shown as follows:



MATLAB code:

```
clc,clear,close all
%question 2a
numSamples = 100000;
propVar = 5;
x = 1;
x_arr = zeros(1,numSamples);
for i = 1:numSamples
    x_cand = normrnd(x,propVar);
    x_prob = cos(x)^2*sin(2*x)^2*normpdf(x);
    x_cand_prob = cos(x_cand)^2*sin(2*x_cand)^2*normpdf(x_cand);
    if x_cand_prob > x_prob
        x = x_cand;
    else
        u = rand;
```

```
        if u < x_cand_prob/x_prob
            x = x_cand;
        end
    end
    x_arr(i) = x;
end

step_cdf = 0.01;
lower_bound = -4;
upper_bound = 4;
cdf = [];
cum_den = 0;
for i = lower_bound:step_cdf:upper_bound
    cum_den = cum_den + cos(i)^2*sin(2*i)^2*normpdf(i)*step_cdf;
    cdf = [cdf,cum_den];
end
cdf = cdf/max(cdf);
figure
[f,x_h] = ecdf(x_arr);
plot(x_h,f,'b','LineWidth',2);
hold on
plot(lower_bound:step_cdf:upper_bound,cdf,'r--','LineWidth',2)
legend('Empirical CDF','True CDF');
```

(b). Although the measurement noise $v(k)$ is non-Gaussian, we assume it is Gaussian with the same variance when we develop Kalman filter.

Given $\omega(k) \sim N(0,1), v(k) \sim N(0,R)$, where R is the variance of the true measurement noise.

The equations of the Kalman filter is as follows:

Prediction:

$$\hat{x}_{k+1|k} = \hat{x}_k$$

$$\Sigma_{k+1|k} = \Sigma_k + 1$$

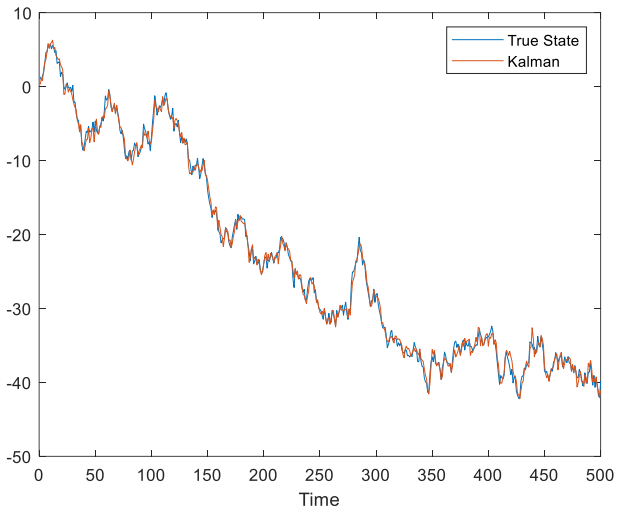
Update:

$$S_{k+1} = \Sigma_{k+1|k} + R$$

$$\hat{x}_{k+1} = \hat{x}_{k+1|k} + \frac{\Sigma_{k+1|k}(y_{k+1} - \hat{x}_{k+1|k})}{S_{k+1}}$$

$$\Sigma_{k+1} = \Sigma_{k+1|k} - \frac{{\Sigma_{k+1|k}}^2}{S_{k+1}}$$

(c). Simulate the Kalman filter on the state space model. The comparison of the true state and the state estimate is as follows:



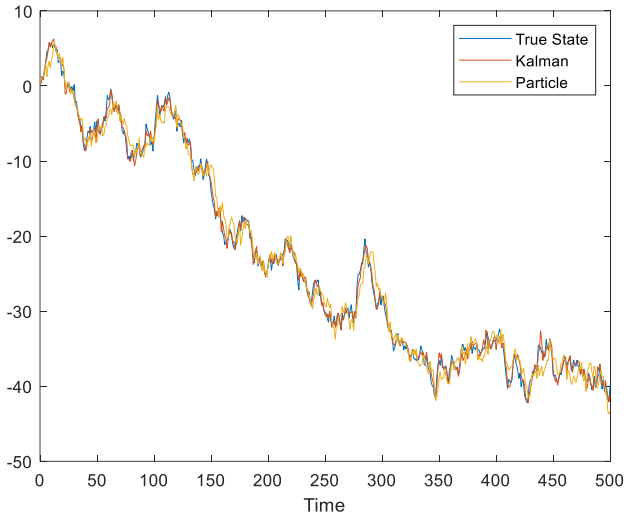
The mean squared error is 0.4971.

MATLAB code:

```
clc,clear,close all
%question 2c
numSamples = 200000;
propVar = 5;
x = 1;
x_arr = zeros(1,numSamples);
for i = 1:numSamples
    x_cand = normrnd(x,propVar);
    x_prob = cos(x)^2*sin(2*x)^2*normpdf(x);
    x_cand_prob = cos(x_cand)^2*sin(2*x_cand)^2*normpdf(x_cand);
    if x_cand_prob > x_prob
        x = x_cand;
    else
        u = rand;
        if u < x_cand_prob/x_prob
            x = x_cand;
        end
    end
    x_arr(i) = x;
end

%state is z,
%generate state and observation data
z = 1;
tspan = 500;
z_arr = zeros(1,tspan);
y_arr = zeros(1,tspan);
for i = 1:tspan
    z = z + normrnd(0,1);
    y = z + x_arr(randi(numSamples,1));
```

```
    z_arr(i) = z;
    y_arr(i) = y;
end

R = var(x_arr);
%Kalman filter
xhat = 0;
Sigma = 1;
x_est_arr = zeros(1,tspan);
for i = 1:tspan
    xhat_pred = xhat;
    Sigma_pred = Sigma + 1;
    S = Sigma_pred + R;
    xhat = xhat_pred + Sigma_pred*(y_arr(i) - xhat_pred)/S;
    Sigma = Sigma_pred - Sigma_pred^2/S;
    x_est_arr(i) = xhat;
end

plot(1:tspan,z_arr,1:tspan,x_est_arr)
legend('True State','State Estimate')
xlabel('Time')
```

(d). The Kalman filter is NOT optimal in terms of mean squared error because the observation noise is not Gaussian.
(e). The comparison of true state, Kalman filter estimate and particle filter estimate is shown as the following plot:



The mean squared error of Kalman filter is 0.4971 and the mean squared error of particle filter is 2.3417.

MATLAB code:

```
%particle filter
particalSize = 2000;
zhat = 10*rand(particalSize,1);
x_est_particle = zeros(1,tspan);
for i = 1:tspan
    zhat_pred = zhat + normrnd(0,1,particalSize,1);
    w = cos(y_arr(i)-zhat_pred).^2.*sin(2*(y_arr(i)-
zhat_pred)).^2.*normpdf(y_arr(i)-zhat_pred);
    partition = mnrnd(particalSize,w/sum(w));
    zhat = [];
    for j = 1:length(partition)
        num = partition(j);
        for k = 1:num
            zhat = [zhat,zhat_pred(j)];
        end
    end
    x_est_particle (i) = mean(zhat);
end
figure
plot(1:tspan,z_arr,1:tspan,x_est_arr)
hold on
plot(1:tspan,x_est_particle)
xlabel('Time')
legend('True State','Kalman', 'Particle');
```