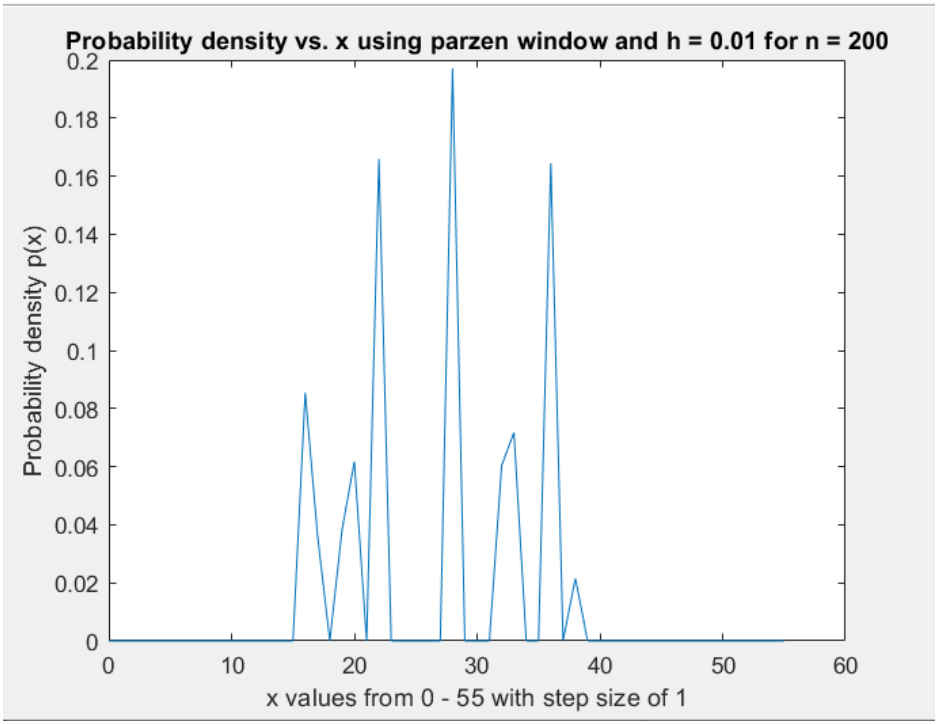
CSE 802 Homework 04

By Abhiram Durgaraju

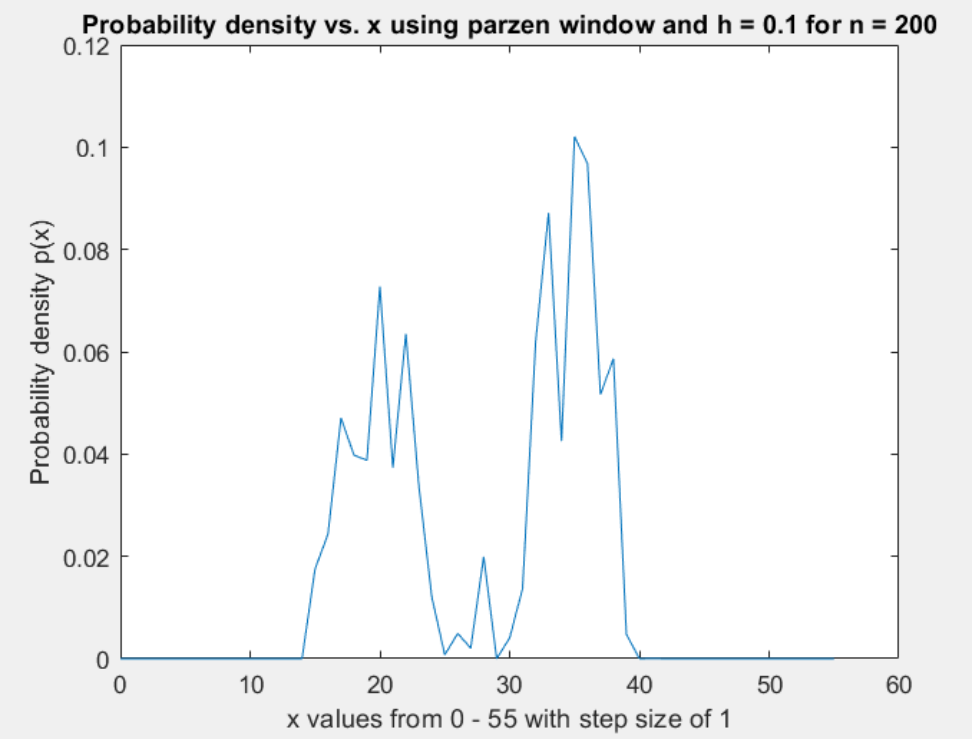
Problem 1)

200 points:

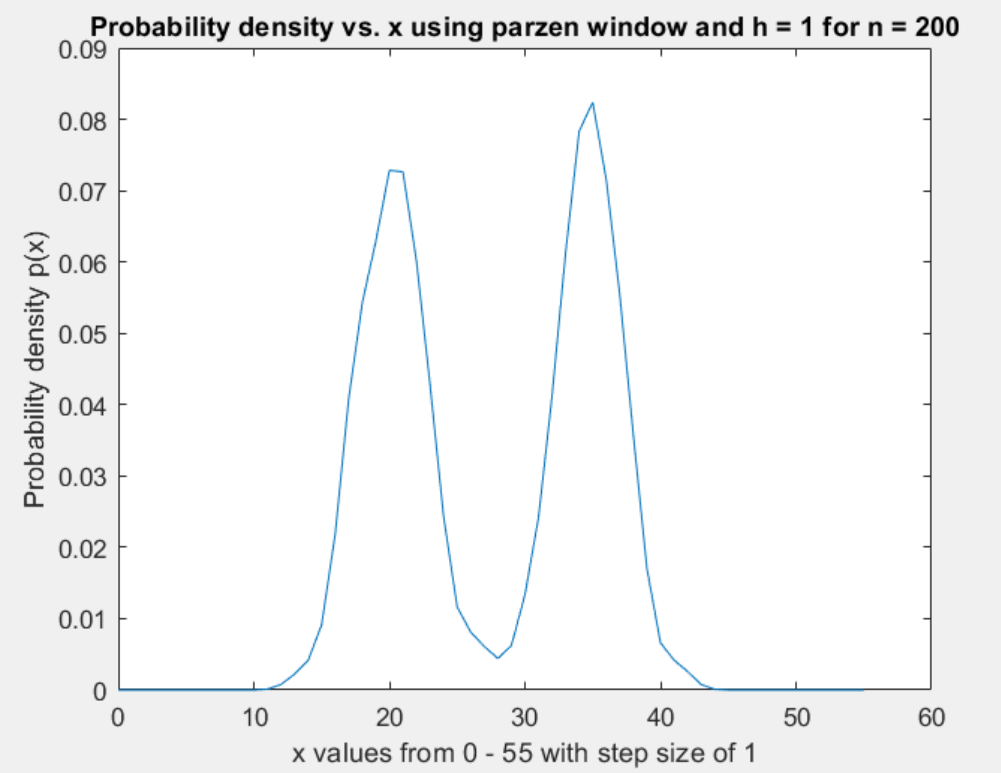
h = 0.01;



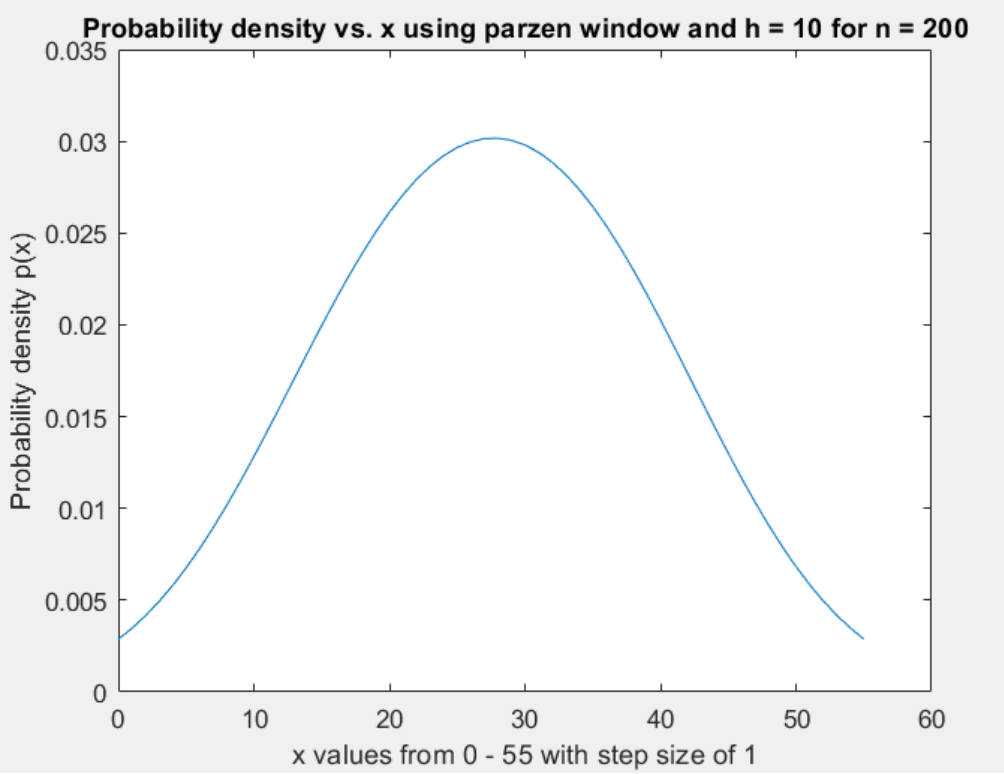
h = 0.1



h = 1

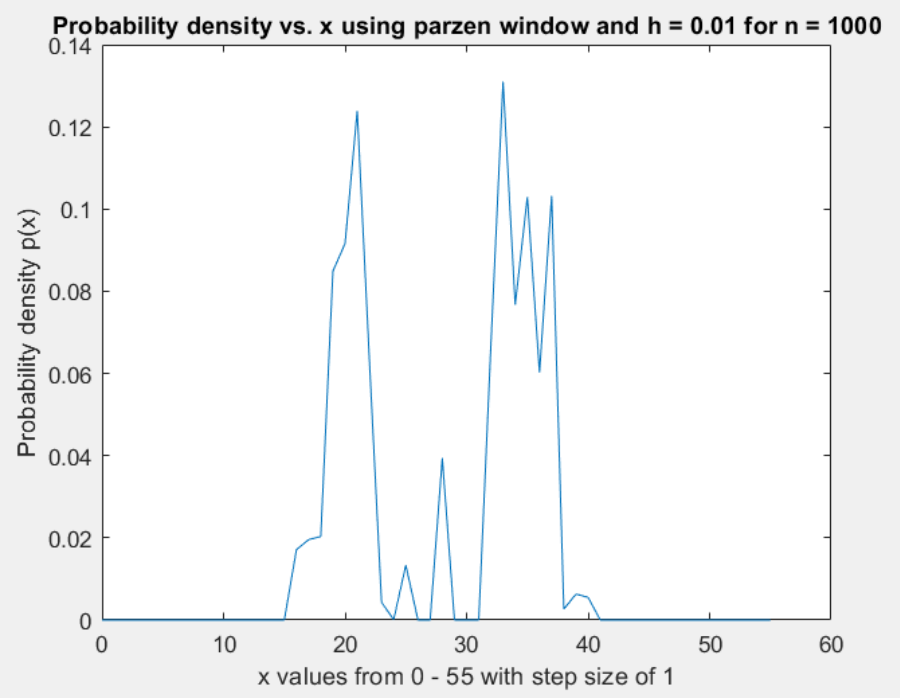

Probability density vs. x using parzen window and h = 1 for n = 200
Probability density p(x)
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
10
20
30
40
50
60
x values from 0 - 55 with step size of 1

h = 10

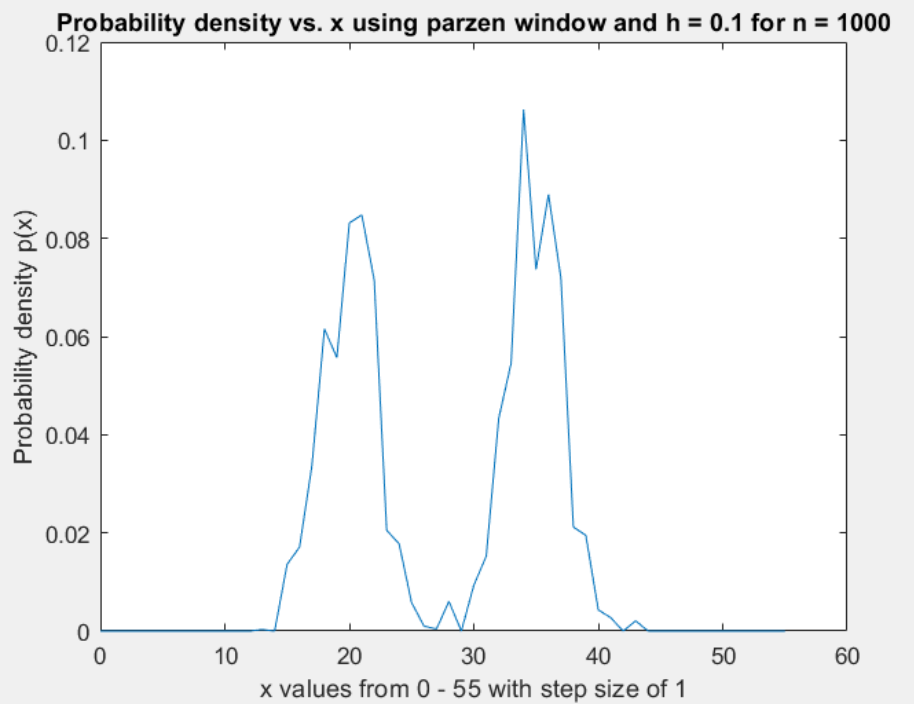

Probability density vs. x using parzen window and h = 10 for n = 200
Probability density p(x)
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
10
20
30
40
50
60
x values from 0 - 55 with step size of 1

500 points:

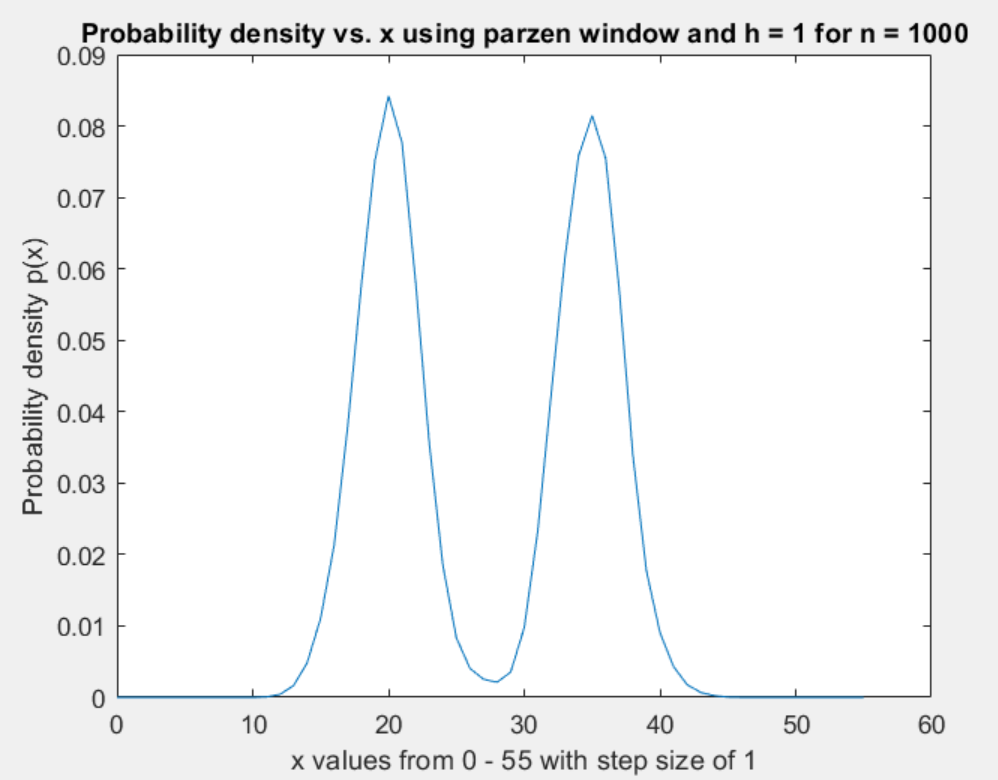
h=0.01



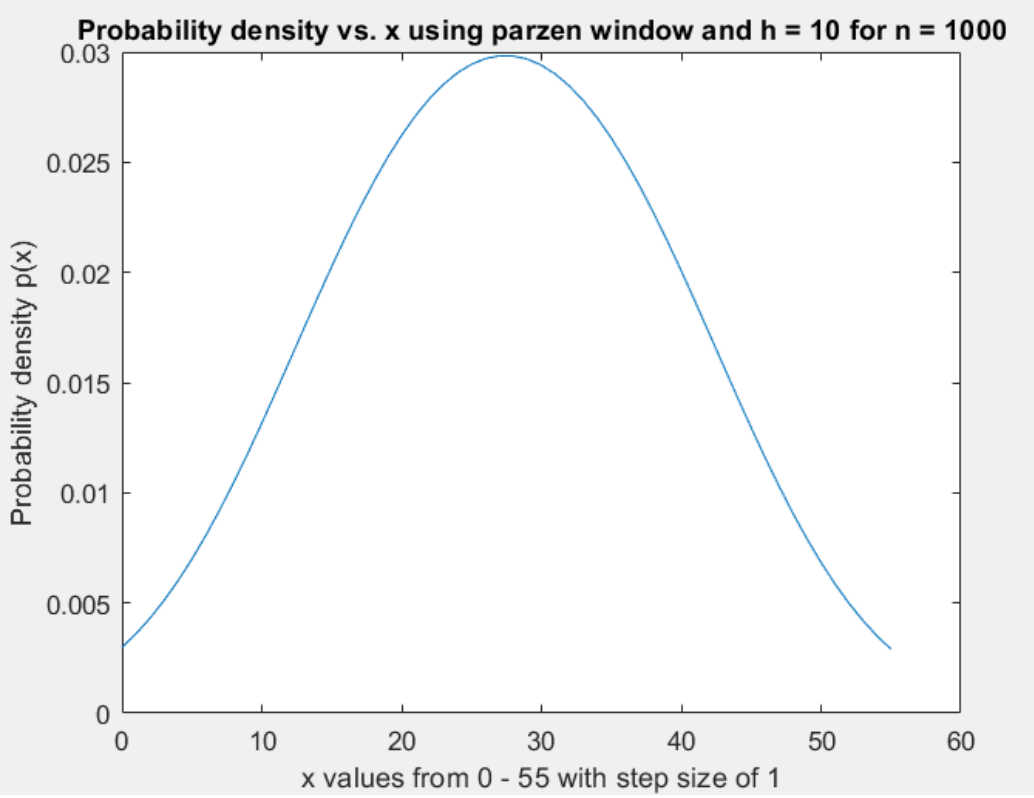
h=0.1



h=1

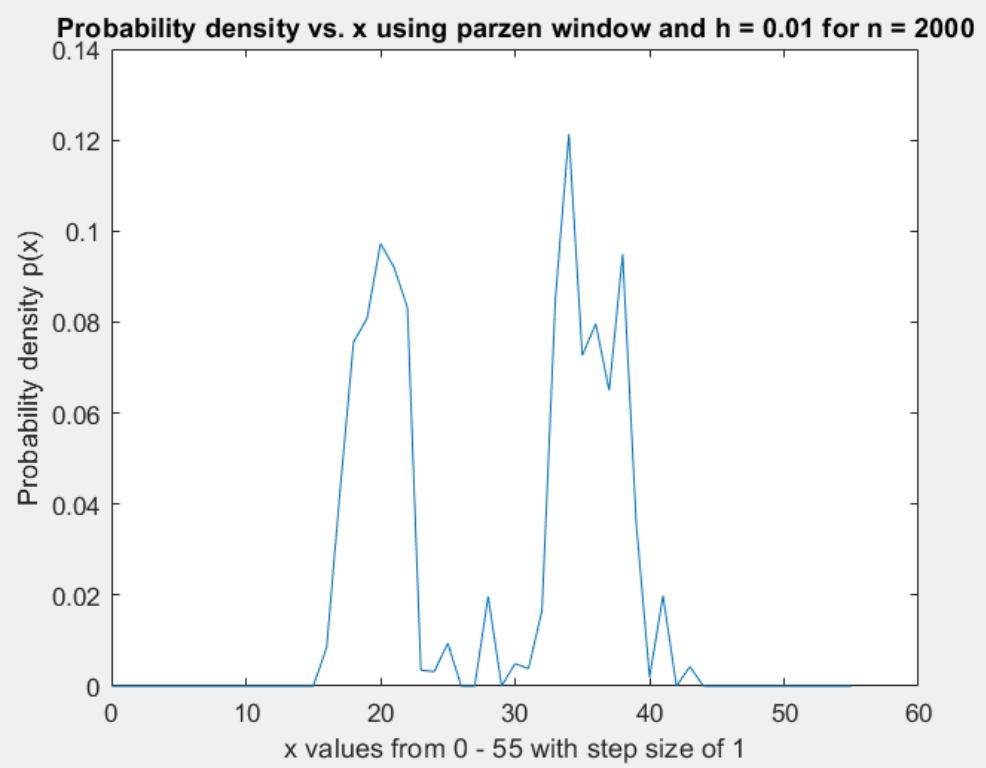


h=10

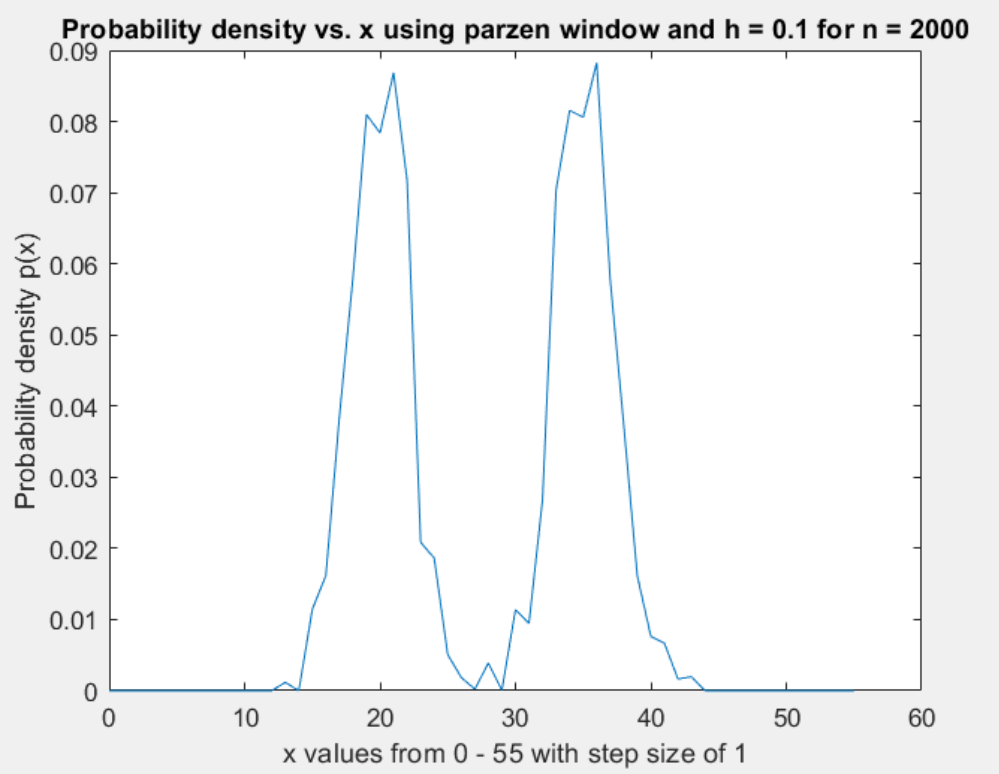


1000 points:

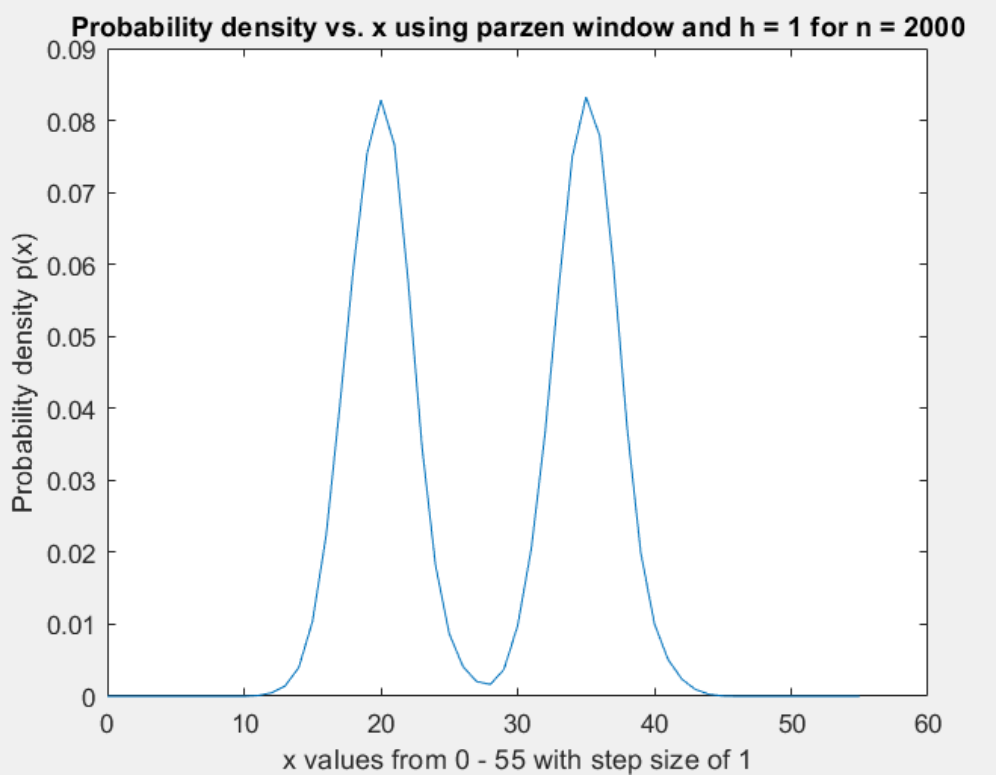
h=0.01


Probability density vs. x using parzen window and h = 0.01 for n = 2000
Probability density p(x)
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
10
20
30
40
50
60
x values from 0 - 55 with step size of 1

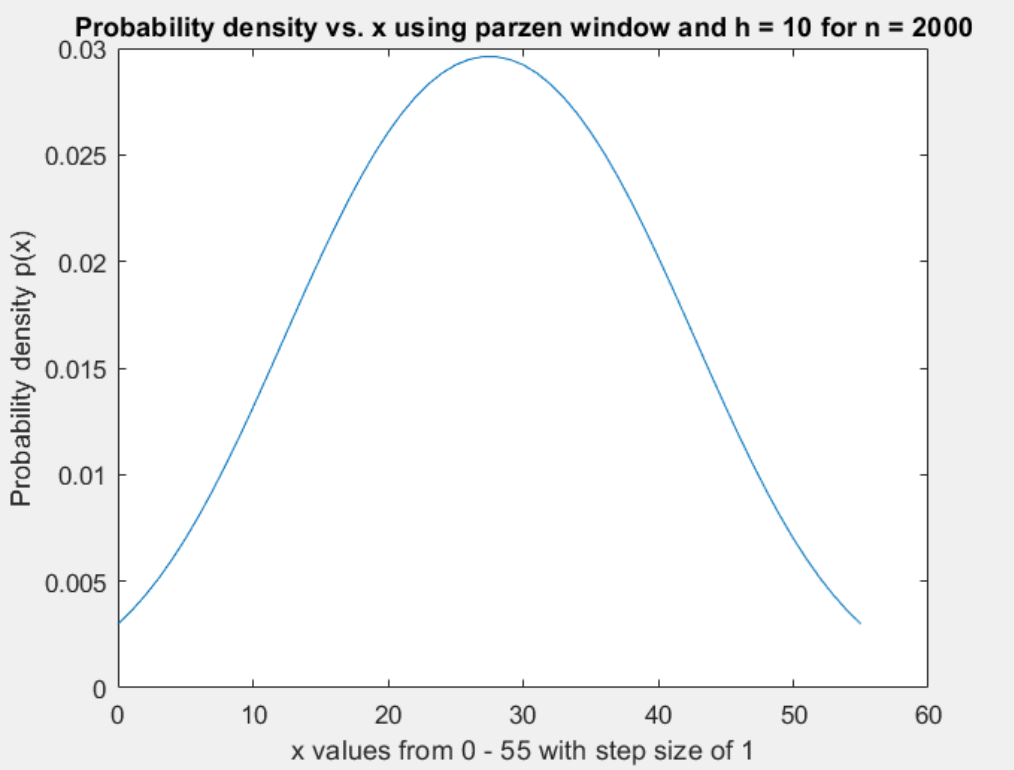
h=0.1


Probability density vs. x using parzen window and h = 0.1 for n = 2000
Probability density p(x)
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
10
20
30
40
50
60
x values from 0 - 55 with step size of 1

h=1


Probability density vs. x using parzen window and h = 1 for n = 2000
Probability density p(x)
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
10
20
30
40
50
60
x values from 0 - 55 with step size of 1

H=10


Probability density vs. x using parzen window and h = 10 for n = 2000
Probability density p(x)
0.03
0.025
0.02
0.015
0.01
0.005
0
0
10
20
30
40
50
60
x values from 0 - 55 with step size of 1

c)

As the window width increases, in general, the bimodal distribution starts to show from the estimated density. However, increasing the window width too much, to h = 10, it “smooths” out the density resulting in a false unimodal approximation. At very low window width however, we see there is sharp variation in estimated density.

As the number of training samples increase, the effect of variation on window width is suppressed. The density curve is generally smoother for a given window width when compared to densities at lower training sample size. The bimodal distribution is seen, with less jaggedness, for h = 1. However, at h=10, regardless on the sample size, the density estimation converges to a unimodal gaussian distribution.

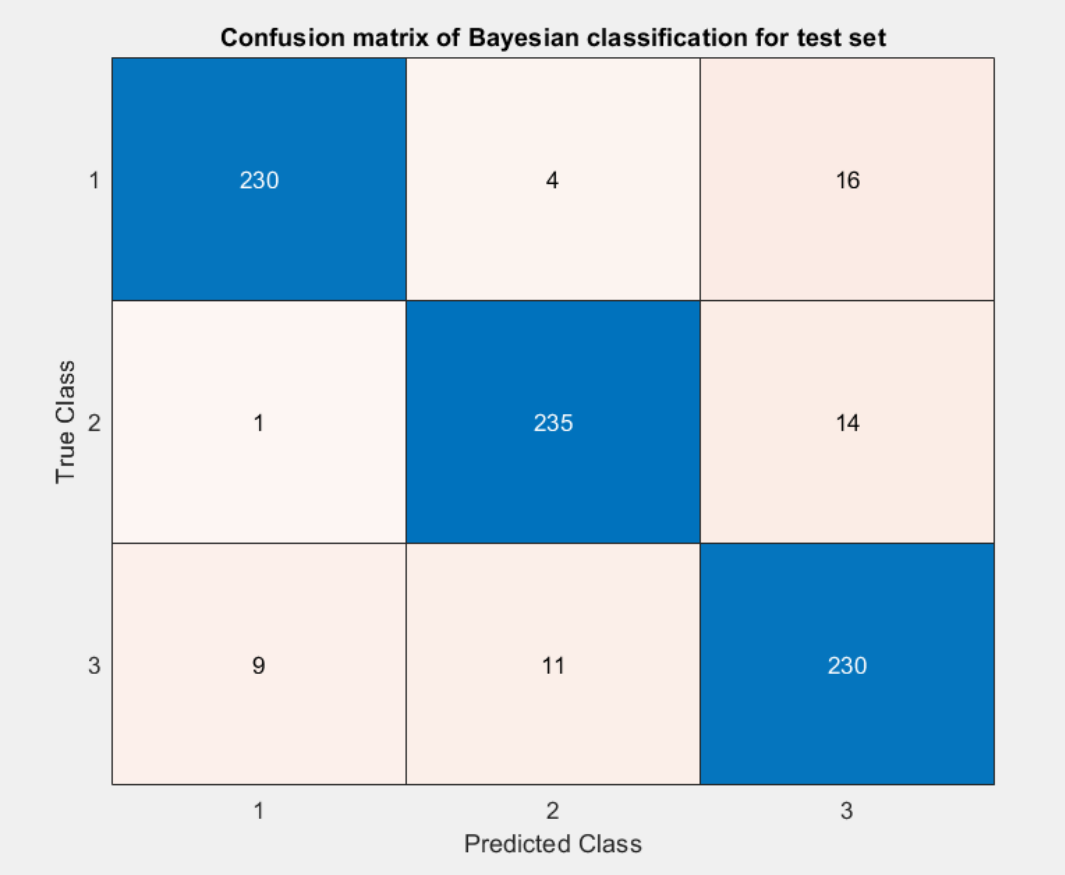
Code used for problem 1:

```
rng('default');
y1 = normrnd(20,sqrt(5),[1,1000]); %try 100, 500, and 1000
y2 = normrnd(35,sqrt(5),[1,1000]);
y = [y1, y2];
n = 2000;
h1 = 10;  %try for 0.01, 0.1, 1, and 10
%h_n = h1/sqrt(n);
h_n = h1;
p_n_list = zeros(1,55);
count = 1;
for x=0:1:55
    p_n = 0;
    for i=1:n
        p_n = p_n + (1/h_n)*normpdf((x-y(i))/h_n);
    end
    p_n = p_n/n;
    p_n_list(1, count) = p_n;
    count = count+1;
end

x = [0:1:55];
plot(x, p_n_list);
xlabel('x values from 0 - 55 with step size of 1');
ylabel('Probability density p(x)');
title('Probability density vs. x using parzen window and h = 10 for n = 2000');
```

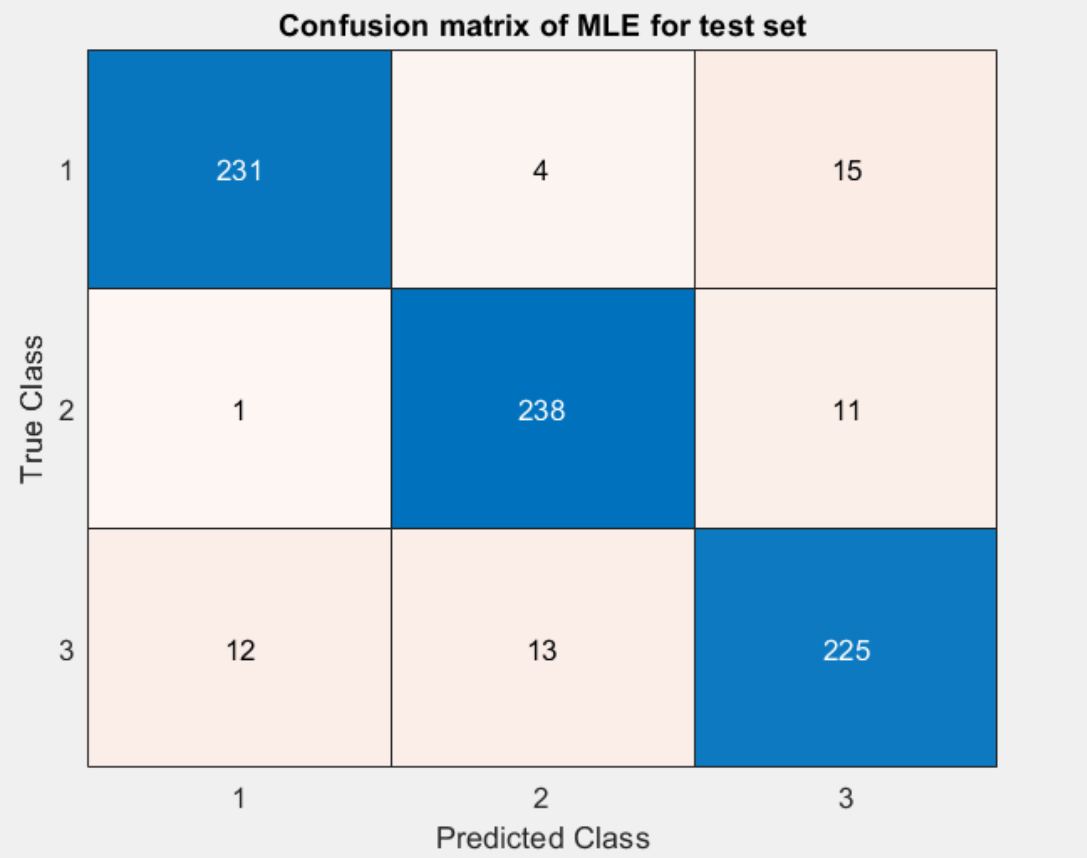
Problem 2)

a)



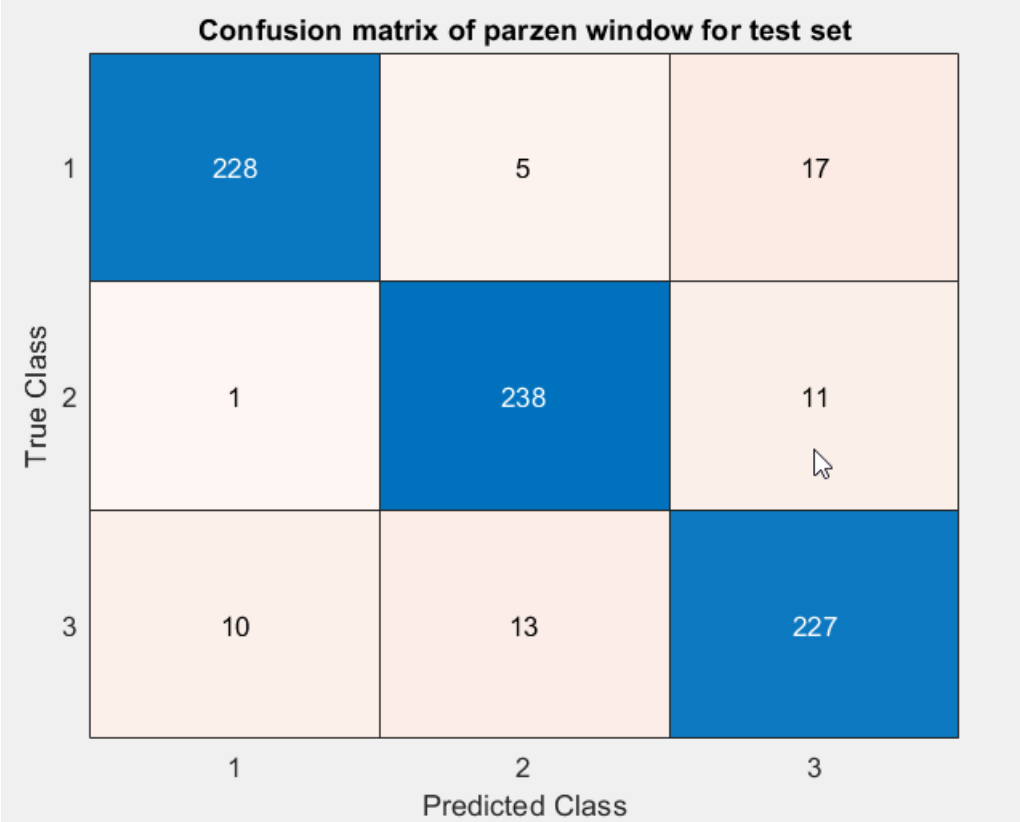
55 samples in the test set were misclassified. Error rate = 55/750 = 0.0733

b)



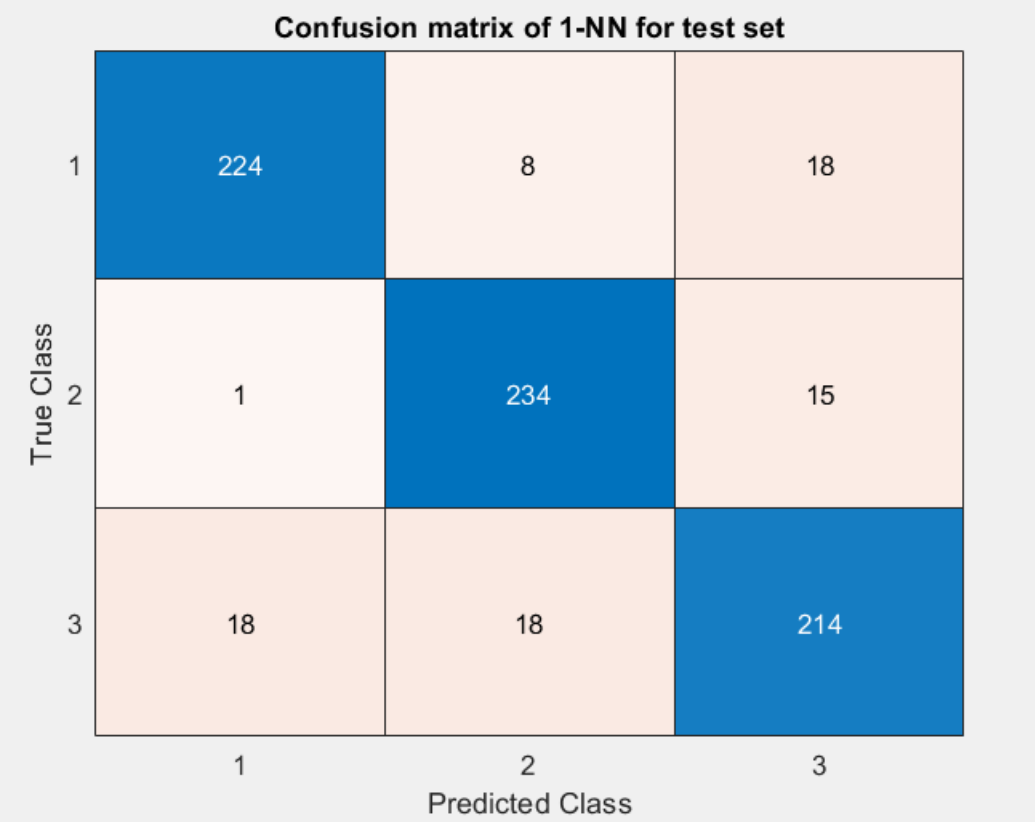
56 samples in the test set were misclassified. Error rate = 56/750 = 0.0746

c)



82 samples were misclassified. Error rate = 57/750 = 0.076

d)



78 samples were misclassified. Error rate = 78/750 = 0.1040

Code used for problem 2:

```
D = importdata('hw04_data.txt');
D_training = [D(1:250, :); D(501:750, :); D(1001:1250, :)];
D_test = [D(251:500, :); D(751:1000, :); D(1251:1500, :)];

%Bayesian classification
Bayes_prediction = zeros(750, 1);
for i=1:750
    p1 = mvnpdf(D_test(i,1:2), [0, 0], 4*[1 0; 0 1]);
    p2 = mvnpdf(D_test(i,1:2), [10, 0], 4*[1 0; 0 1]);
    p3 = mvnpdf(D_test(i,1:2), [5, 5], 5*[1 0; 0 1]);
    [M, I] = max([p1 p2 p3]);
    Bayes_prediction(i) = I;
end

% C_bayes = confusionmat(D_test(:,3), Bayes_prediction);
% confusionchart(C_bayes);
% title('Confusion matrix of Bayesian classification for test set');


%MLE
mean_mle_1 = mean(D_training(1:250, 1:2));
mean_mle_2 = mean(D_training(251:500, 1:2));
mean_mle_3 = mean(D_training(501:750, 1:2));
cov_mle_1 = cov(D_training(1:250, 1:2), 1);
cov_mle_2 = cov(D_training(251:500, 1:2), 1);
cov_mle_3 = cov(D_training(501:750, 1:2), 1);
MLE_prediction = zeros(750, 1);
for i=1:750
    p1 = mvnpdf(D_test(i,1:2), mean_mle_1, cov_mle_1);
    p2 = mvnpdf(D_test(i,1:2), mean_mle_2, cov_mle_2);
    p3 = mvnpdf(D_test(i,1:2), mean_mle_3, cov_mle_3);
    [M, I] = max([p1 p2 p3]);
    MLE_prediction(i) = I;
end
```

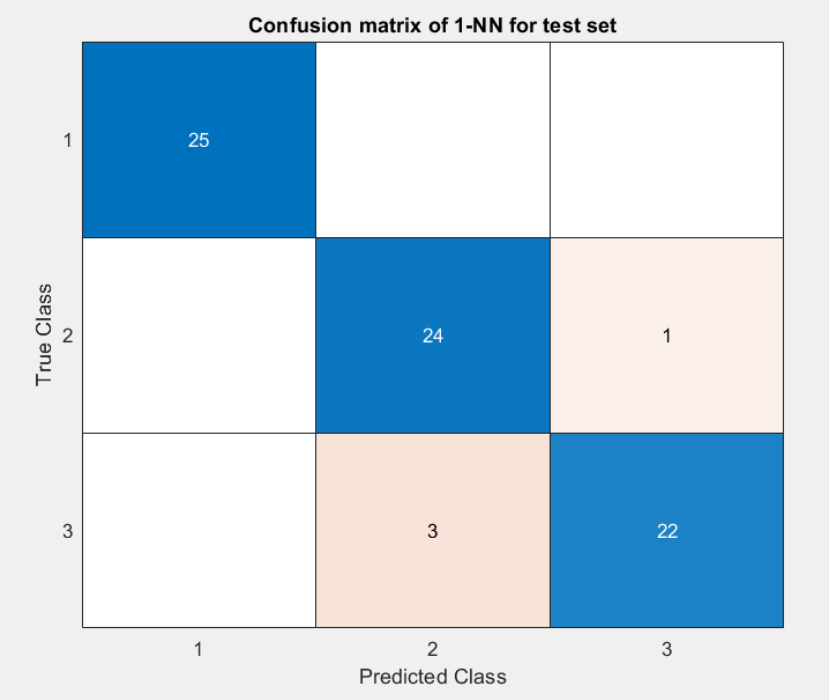
```
% C_MLE = confusionmat(D_test(:,3), MLE_prediction);
% confusionchart(C_MLE);
% title('Confusion matrix of MLE for test set');
%Parzen
n = 750;
h1 = 1;  %try for 0.01, 0.1, 1, and 10
%h_n = h1/sqrt(n);
h_n = 1;  %this will get a better performance 57/750 instead of 82/750
parzen_prediction = zeros(750, 1);
for x=1:n
    p_n_1 = 0;
    p_n_2 = 0;
    p_n_3 = 0;
    for i=1:n/3
        p_n_1 = p_n_1 + (1/h_n)*mvnpdf((D_test(x,1:2)- D_training(i,1:2))/h_n);
        p_n_2 = p_n_2 + (1/h_n)*mvnpdf((D_test(x,1:2)- D_training(i+250,1:2))/h_n);
        p_n_3 = p_n_3 + (1/h_n)*mvnpdf((D_test(x,1:2)- D_training(i+500,1:2))/h_n);
    end
    p_n_1 = p_n_1/n;
    p_n_2 = p_n_2/n;
    p_n_3 = p_n_3/n;
    [M, I] = max([p_n_1 p_n_2 p_n_3]);
    parzen_prediction(x) = I;
end
C_parzen = confusionmat(D_test(:,3), parzen_prediction);
confusionchart(C_parzen);
title('Confusion matrix of parzen window for test set');
%kNN
k = 1;
knn_prediction = zeros(750, 1);
for x=1:750
    euclid_dist = zeros(750, 1);
    for i=1:750
        euclid_dist(i) = norm(D_test(x,1:2) - D_training(i,1:2));
    end
    [M, I] = mink(euclid_dist, k);
    knn_prediction(x) = D_training(I, 3); %this is only for 1-NN
end

% C_knn = confusionmat(D_test(:,3), knn_prediction);
% confusionchart(C_knn);
% title('Confusion matrix of 1-NN for test set');
```

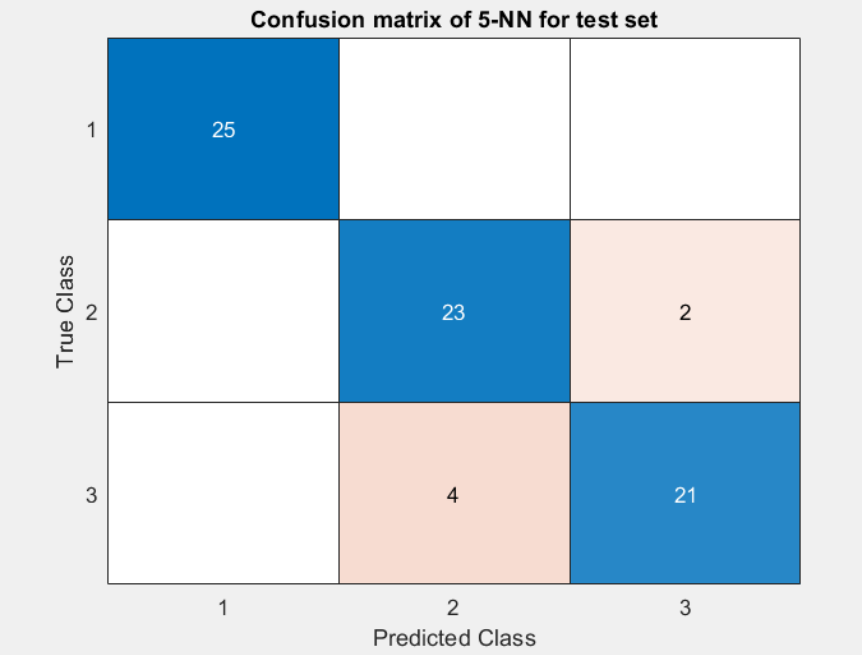
Problem 3)

a)

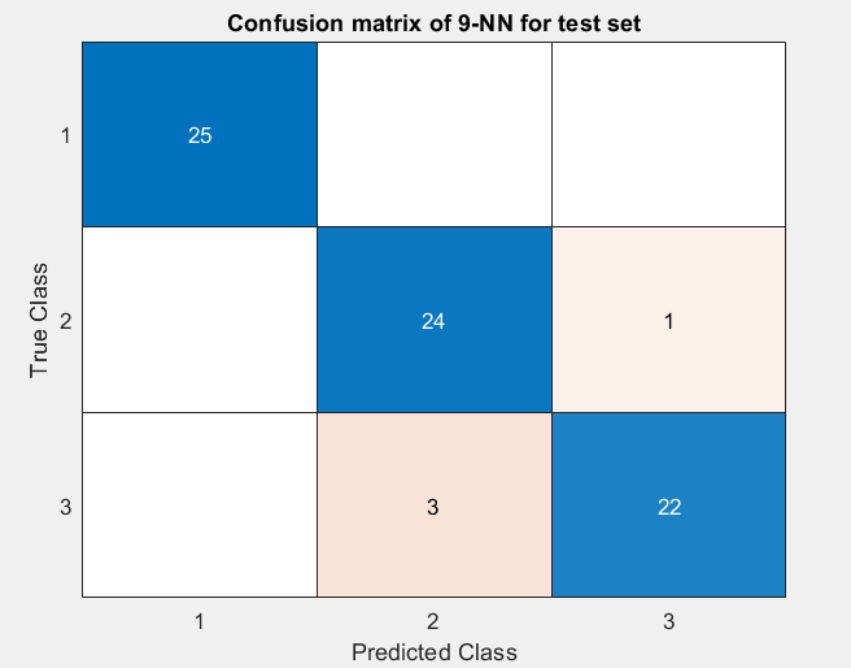
k =1



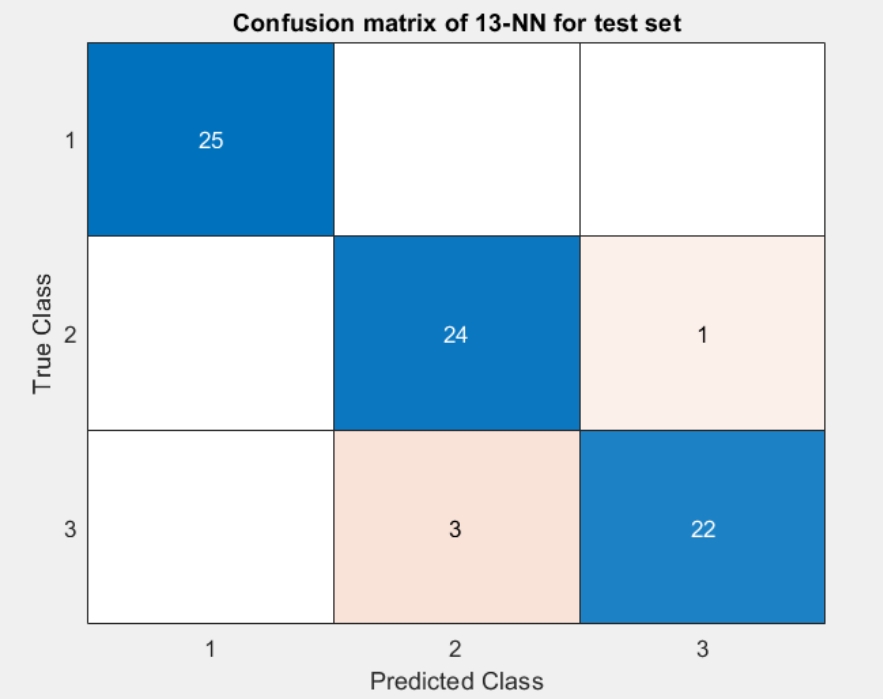
k=5



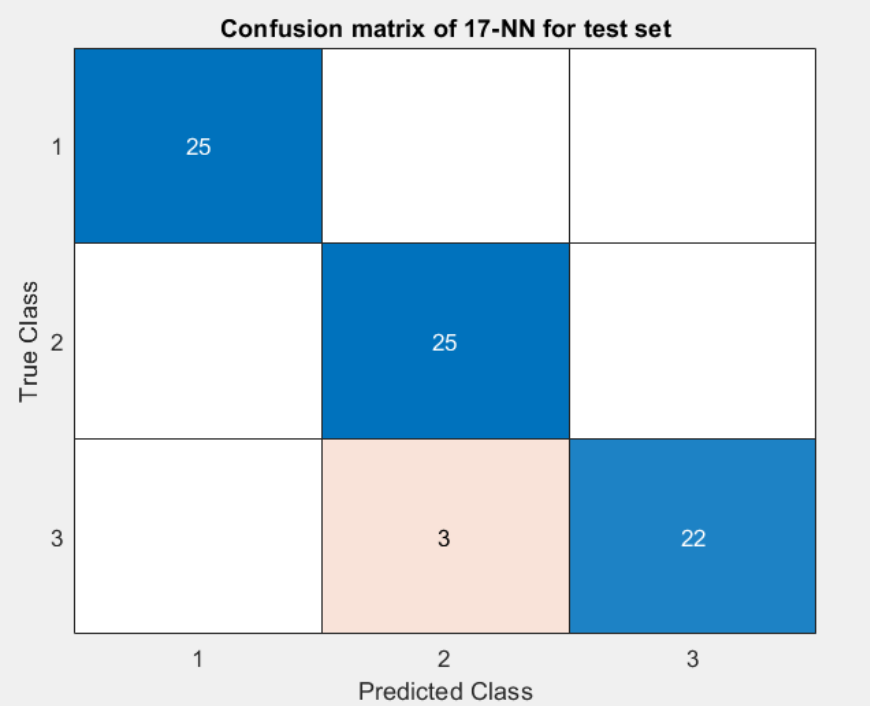
k=9


Confusion matrix of 9-NN for test set
True Class
Predicted Class
1
2
3
25
24
1
3
22

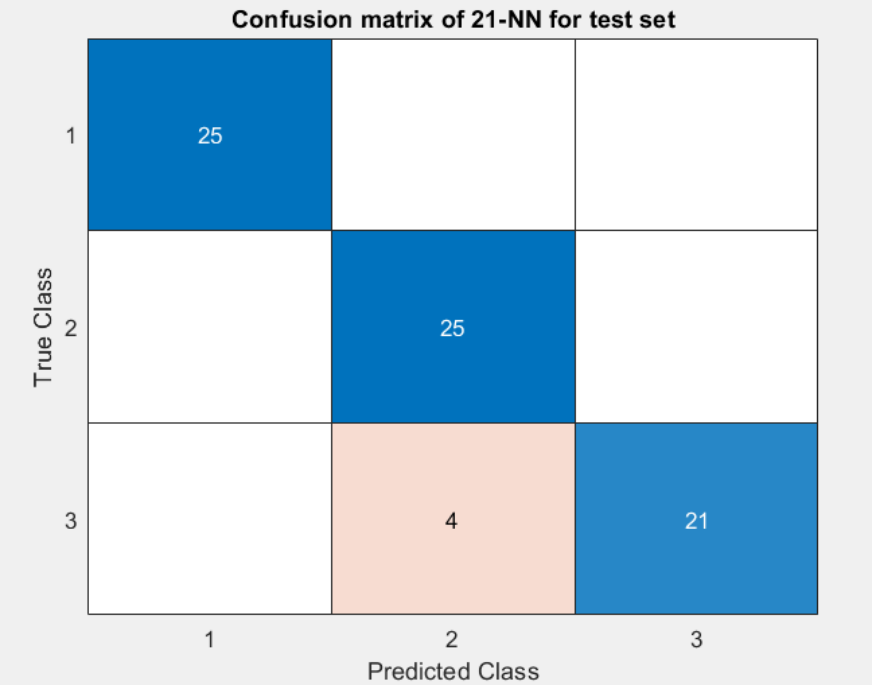
k=13


Confusion matrix of 13-NN for test set
True Class
Predicted Class
1
2
3
25
24
1
3
22

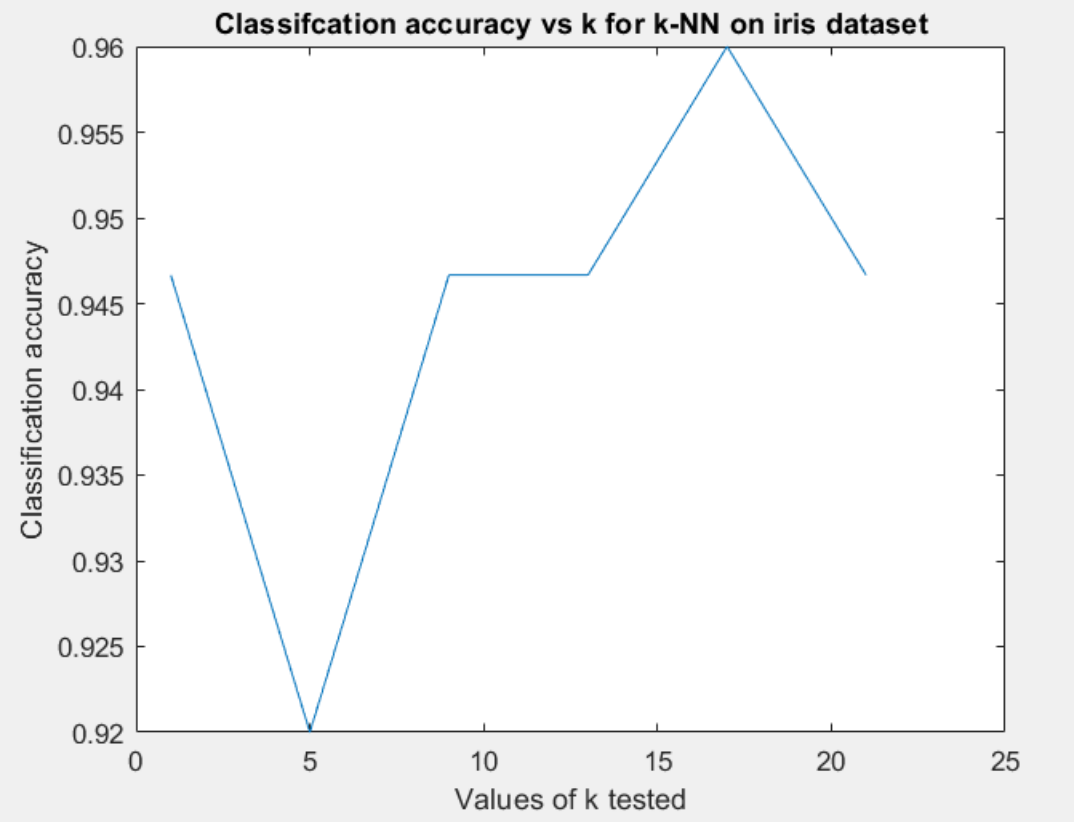
k=17


Confusion matrix of 17-NN for test set
True Class
Predicted Class
1
2
3
25
25
3
22

k=21


Confusion matrix of 21-NN for test set
True Class
Predicted Class
1
2
3
25
25
4
21

b)



Discussion:

The graph above shows that the optimal value of k for classification accuracy peaks at a certain value but isn't necessarily always the largest or smallest k. As k increases, the classification accuracy can fluctuate. A small value of k is computationally inexpensive but may result in misclassification which is sensitive to noise. A large value of k is computationally expensive, but usually performs better until a certain inflexion point (k=17 in our example case). K value that is too large or too small may not accurately capture the underlying patterns of the features. Testing various values of k until the best classification accuracy is found is a good idea it seems.

Code used for problem 3:

```
D = importdata('iris_data.txt');
D_training = [D(1:25, :); D(51:75,:); D(101:125, :)];
D_test = [D(26:50, :); D(76:100,:); D(126:150, :)];


k = 21;
knn_prediction = zeros(75, 1);
for x=1:75
    euclid_dist = zeros(75, 1);
    for i=1:75
        euclid_dist(i) = norm(D_test(x,1:4) - D_training(i,1:4));
    end
    [M, I] = mink(euclid_dist, k);
    neighbor_list = zeros(k, 1);
    for z=1:k
        neighbor_list(z) = D_training(I(z), 5);
    end
    frequency_list = histc(neighbor_list, [1,2,3]); %find the frequency of each class among neighbors
    max_frequency = max(frequency_list);
    index_max_frequency = find(frequency_list == max_frequency);
    %knn_prediction(x) = index_max_frequency(1);
    knn_prediction(x) = index_max_frequency(randperm(length(index_max_frequency), 1));
end

% C_knn = confusionmat(D_test(:,5), knn_prediction);
% confusionchart(C_knn);
% title('Confusion matrix of 21-NN for test set');

k_list = [1, 5, 9, 13, 17, 21];
accuracy = [71/75, 69/75, 71/75, 71/75, 72/75, 71/75];
plot(k_list, accuracy);
xlabel('Values of k tested');
ylabel('Classification accuracy');
title('Classifcation accuracy vs k for k-NN on iris dataset');
```

Problem 4)

SBS:

Input: $Y = \{y_j | j = 1, \dots, D\}$ //total available samples

Output: $X_k = \{x_k | k = 1, \dots, k, x_j \in Y\}$ //desired suboptimal subset features k

Initialization: $k = D; X_k = Y$ //Start with the full set of features

Termination: Stop when k equals the number of features desired

Step 1 (Exclusion) //Calculate the criterion function $J(X)$ for each feature removed and choose best subset. Repeat until subset contains k features

$$x^- = \arg\max_{x \in X_k} J(X_k - x)$$

$$X_{k-1} = X_k - x^-$$

$$k = k - 1$$

go to **Step 1**

SBFS:

Input: $Y = \{y_j | j = 1, \dots, D\}$ //total available samples

Output: $X_k = \{x_k | k = 1, \dots, k, x_j \in Y\}$ //suboptimal subset of k features

Initialization: $k = D; X_k = Y$ //Start with the full set of features

Termination: Stop when k equals the number of features desired

Step 1 (Exclusion) //Calculate the criterion function $J(X)$ for each feature removed and choose best subset.

$$x^- = \arg\max_{x \in X_k} J(X_k - x)$$

$$X_{k-1} = X_k - x^-$$

$$k = k - 1$$

Step 2 (Conditional Inclusion) //Perform SFS if the corresponding subsets are better than previously evaluated

$$x^+ = \arg\max_{x \in Y - X_k} J(X_k + x)$$

$if \quad J(X_k - \{x^+\}) > J(X_{k+1}) \quad then$

$$X_{k+1} = X_k + x^+$$

$$k = k + 1$$

go to **Step 2**

$else$

go to **Step 1**