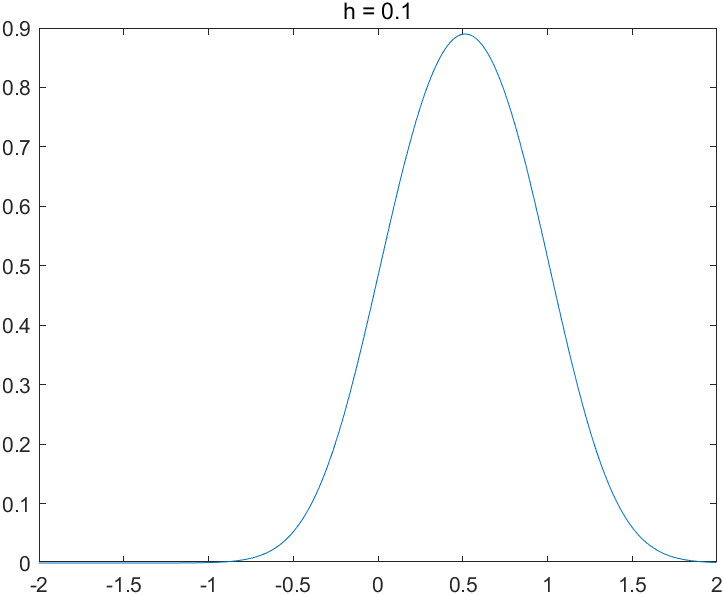
CS 596 hw 4

Name: Yan Gu NetID: yg369

**Problem 1**

图片包含 地图, 文字

描述已自动生成



图片包含 文字, 地图

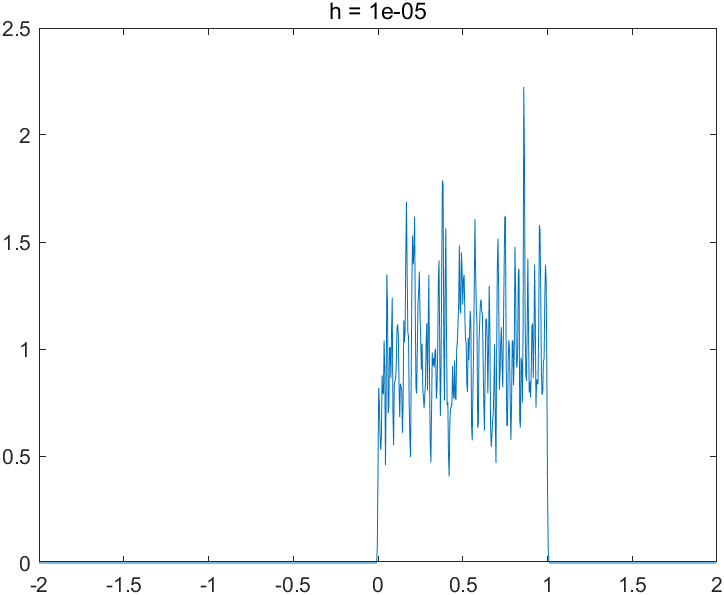
描述已自动生成

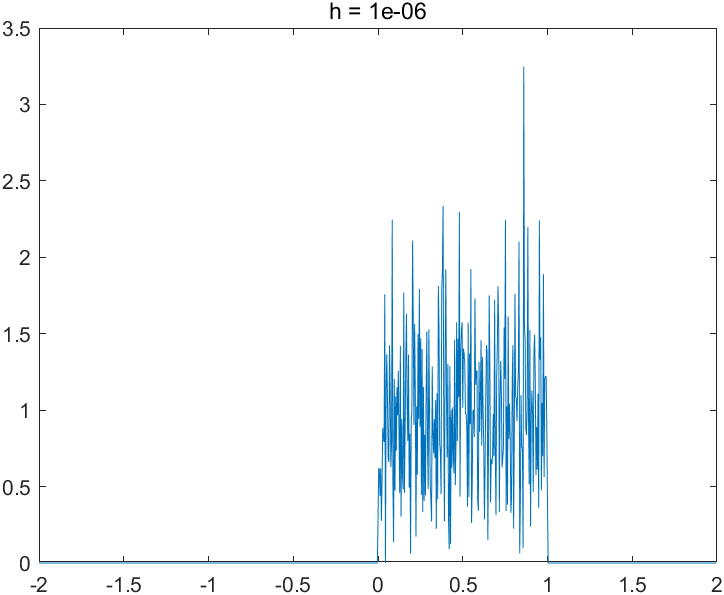
图片包含 文字, 地图

描述已自动生成

图片包含 文字

描述已自动生成





图片包含 文字, 地图

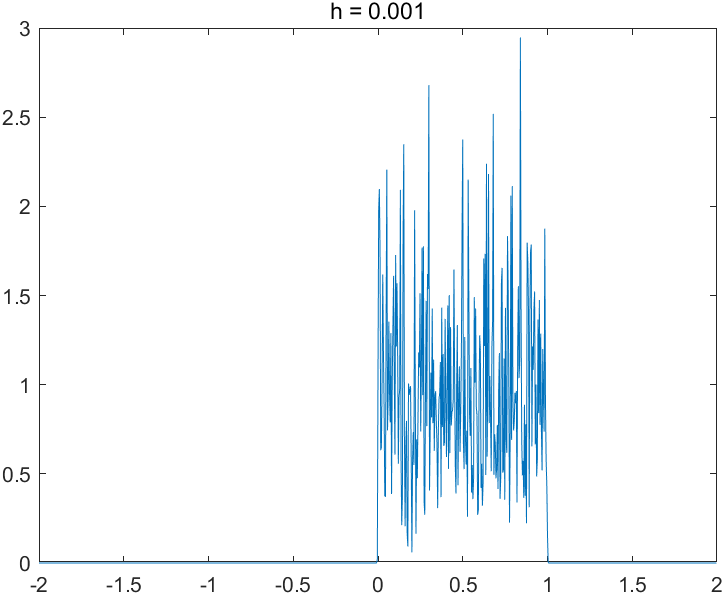
描述已自动生成

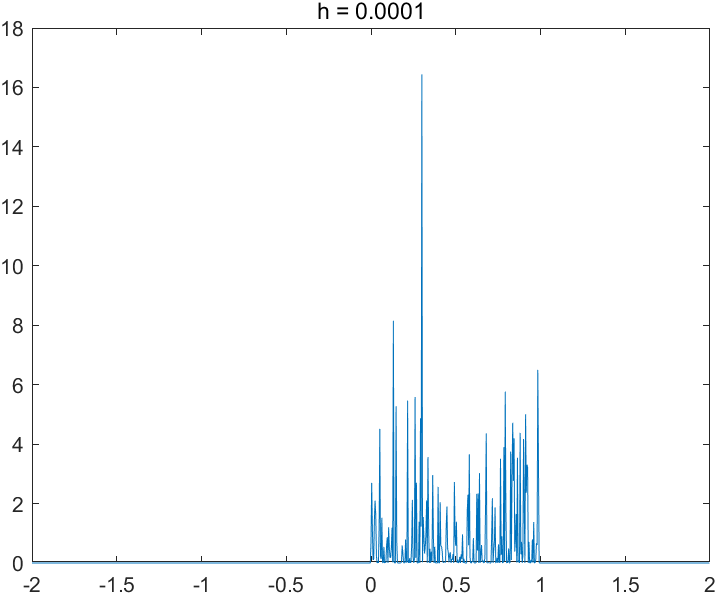
图片包含 文字, 地图

描述已自动生成

图片包含 文字

描述已自动生成





图片包含 屏幕截图

描述已自动生成

图片包含 屏幕截图

描述已自动生成

**Matlab Code:**

**Problem 1 a)**

r = rand(1000, 1);

x = [-2:0.004:1.996];

h = 1;

while h > 1e-6

y = [];

for i = 1:1000

temp = 0;

for j = 1:1000

temp = temp + 1 / sqrt(2 \* pi \* h) \* exp(-1 / (2 \* h) \* ((x(i) - r(j)) ^ 2));

end

temp = temp / 1000;

y(i) = temp;

end

plot(x, y);

title(['h = ',num2str(h)]);

h = h / 10;

end

**Problem 1 b)**

r = rand(1000, 1);

r = transpose(r);

x = [-2:0.004:1.996];

h = 1;

while h > 1e-6

y = [];

for i = 1:1000

temp = 0;

for j = 1:1000

temp = temp + 1 / (2 \* h) \* exp(-1 / h \* abs(x(i) - r(j)));

end

temp = temp / 1000;

y(i) = temp;

end

plot(x, y);

title(['h = ',num2str(h)]);

h = h / 10;

end

**Problem 2**

In the hand-writing scripts.

In the hand-writing scripts.

图片包含 文字, 地图

描述已自动生成

图片包含 文字, 地图

描述已自动生成

图片包含 文字, 地图

描述已自动生成

图片包含 文字, 地图

描述已自动生成

图片包含 文字, 地图

描述已自动生成图片包含 文字, 地图

描述已自动生成图片包含 文字, 地图

描述已自动生成图片包含 文字, 地图

描述已自动生成

图片包含 文字, 地图

描述已自动生成图片包含 文字, 地图

描述已自动生成图片包含 文字, 地图

描述已自动生成图片包含 文字, 地图

描述已自动生成

图片包含 文字, 地图

描述已自动生成图片包含 文字, 地图

描述已自动生成图片包含 文字, 地图

描述已自动生成图片包含 文字, 地图

描述已自动生成

**Matlab Code:**

**Problem 2**

clear;

cla;

cla reset;

m = load('hw4-2data.mat');

% c = circle, s = stars

s = m.stars;

c = m.circles;

% plot the point

scatter(s(:,1) ,s(:,2), 'red','+');

hold on;

scatter(c(:,1),c(:,2),'blue','d');

hold on;

[szs, szs\_] = size(s);

[szc, szc\_] = size(c);

sz = szc + szs

K = zeros(sz, sz);

points = vertcat(s, c);

h = [0.01, 0.1, 1, 10];

lambda = [0.01, 0.1, 1, 10];

for hi = 1:4

for lambdai = 1:4

% plot the point

scatter(s(:,1) ,s(:,2), 'red','+');

hold on;

scatter(c(:,1),c(:,2),'blue','d');

hold on;

% First, calculate the kernel function, it is the 42\*42 matrix.

for i = 1:sz

for j = 1:sz

K(i,j) = exp(-1 / h(hi) \* ((points(i,1) - points(j,1)) ^ 2 + (points(i,2) - points(j,2)) ^ 2));

end

end

% calculate gamma,b

% gamma = [a1, a2, a3, ..., ai, b1, b2, ..., bi]

% b for update

Gamma = zeros(sz,sz);

b = zeros(sz,1);

% stars

for i=1:sz/2

b = b + K(:,i);

Gamma = Gamma + K(:,i) \* K(i,:);

end

% circles

for i=sz/2+1:sz

b = b - K(:,i);

Gamma = Gamma + K(:,i) \* K(i,:) ;

end

Gamma = Gamma + lambda(lambdai) \* K;

res = Gamma \ b;

Gamma = res;

m = linspace(-1, 1, 100);

n = zeros(100);

syms m n

L = 0;

for i=1:sz

L = L + Gamma(i) \* exp(-((m - points(i,1)) ^ 2 + (n - points(i,2)) ^ 2) / h(hi));

end

% show alpha and beta

Gamma

line = fimplicit(L, [-1.25, 1.25]);

line.Color = 'black';

title(['h = ',num2str(h(hi)), ', {\lambda} = ', num2str(lambda(lambdai))]);

% clear scatter;

end

end

**Problem 3**

I choose .

As far as I see, for larger , the convergence rate is larger, but for smaller , the final error is better, I think I cannot tell whether one is better than another but in my opinion I think the small error (small ) is better if we have enough computational resources.

The for 0.01, The for 0.005

图片包含 文字, 地图

描述已自动生成

**Python Code:**

**Problem 3**

import random

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.scale as sc

# numpy.random.normal(loc=0.0, scale=1.0, size=(m, n, k))

LENGTH = 5

THETA\_STAR = np.array([25, 22, 16, 132, 66])

MU = 0.01

def generate\_samples(t, length):

    X = np.random.normal(size=(t, length))

    W = np.random.normal(loc=0.0, scale=0.1, size=t)

    # print(X)

    # print(W)

    Y = np.array([np.dot(THETA\_STAR.transpose(), X[i]) + W[i] for i in range(t)])

    return Y, X, W

def lms(Y, X, mu):

    theta = np.array([0.0 for \_ in range(5)])

    e = []

    for t in range(Y.size):

        temp = theta

        theta = temp + np.dot((mu \* (Y[t] - np.dot(temp.transpose(), X[t]))), X[t])

        # theta = theta + np.dot((MU \* (Y[t] - np.dot(theta.transpose(), X[t]))), X[t])

        e.append(np.linalg.norm(theta - THETA\_STAR) \*\* 2)

    return e, theta

if \_\_name\_\_ == "\_\_main\_\_":

    t = 2600

    Y, X, W = generate\_samples(t, LENGTH)

    iter\_ = [i for i in range(Y.size)]

    # c is for question c) with higher mu and d with mu / 2 in the question d)

    error\_c, theta\_c = lms(Y, X, MU)

    error\_d, theta\_d = lms(Y, X, MU/2)

    print(theta\_c)

    # print(min(error\_c), error\_c[-1])

    print(theta\_d)

    # print(min(error\_d), error\_d[-1])

    plt.figure(figsize=(13,8))

    plt.plot(iter\_, error\_c, color='r', linestyle='-', label="mu = 0.01")

    plt.plot(iter\_, error\_d, color='b', linestyle='-.', label="mu = 0.005")

    plt.ylabel('||theta\_T - theta\_star||^2')

    plt.xlabel('No. of iterations')

    plt.semilogy(error\_c)

    plt.semilogy(error\_d)

    plt.legend()

    plt.show()

    print("end")