Computer Project 1

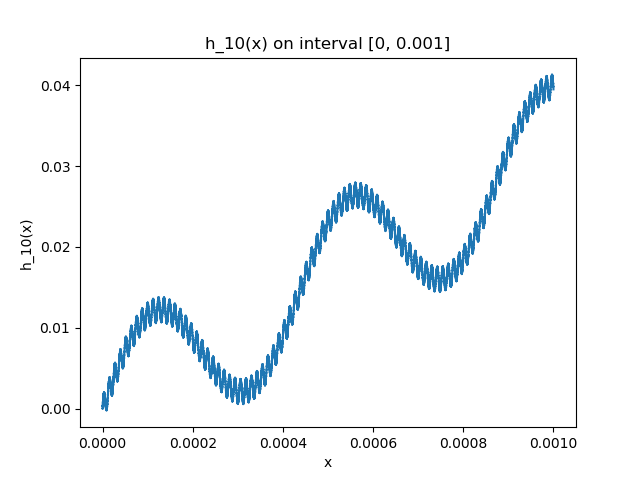
Nicolas Perez

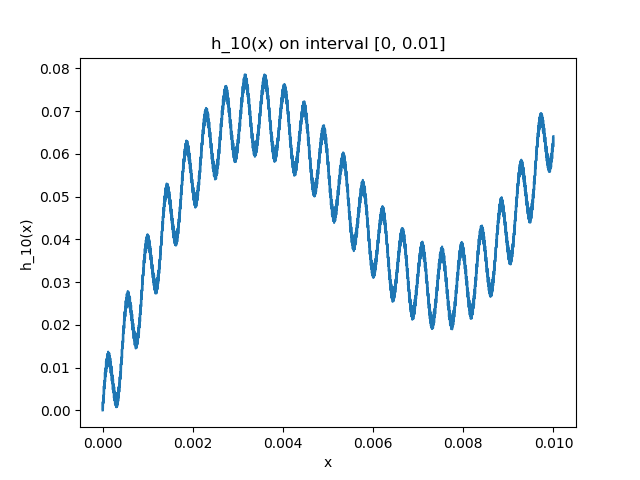
Prof. Sergey Lototsky

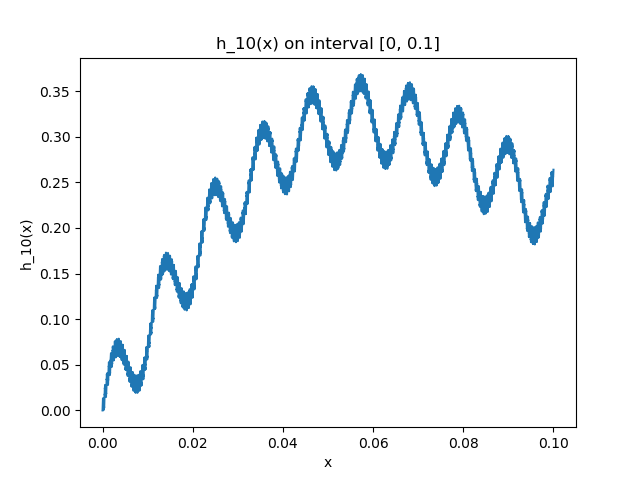
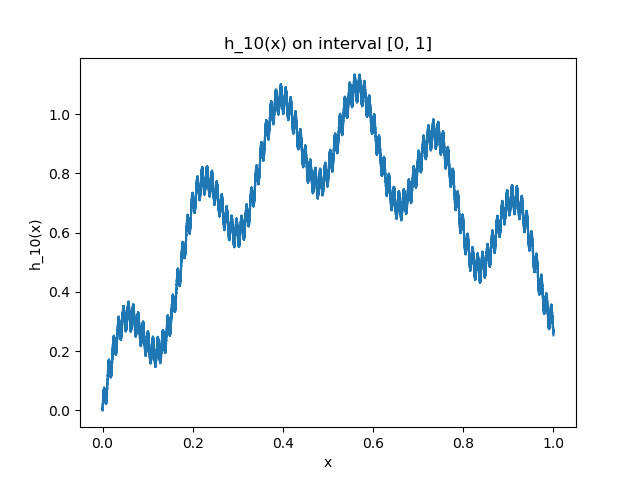
3/12/2020

**Problem 1**

**Part 1**

Graphs:





Printout of program:

import matplotlib.pyplot as plt

import numpy as np

from math import factorial, sin

import pandas as pd

def h10(x: int):

count = 0

for i in range(1, 11):

count += sin((factorial(i) \*\* 2) \* x) / factorial(i)

return count

def plot\_h10(interval: tuple, steps: int):

"""

interval: (tuple) containing boundaries for x

steps: (int) number of datapoints to generate

"""

result = {}

samples = np.linspace(interval[0], interval[1], num=steps)

for x in samples:

result[x] = h10(x)

result = pd.Series(result)

fig = plt.figure()

ax = plt.subplot(111)

ax.plot(result)

ax.set\_xlabel('x')

ax.set\_ylabel('h\_10(x)')

ax.set\_title('h\_10(x) on interval [{}, {}]'.format(interval[0], interval[1]))

fig.savefig('h10\_[{}\_{}].png'.format(interval[0], interval[1]))

def main():

# testing functions

plot\_h10((0, 1), 10000)

plot\_h10((0, 0.1), 10000)

plot\_h10((0, 0.01), 10000)

plot\_h10((0, 0.001), 10000)

plot\_h10

if \_\_name\_\_ == '\_\_main\_\_':

main()

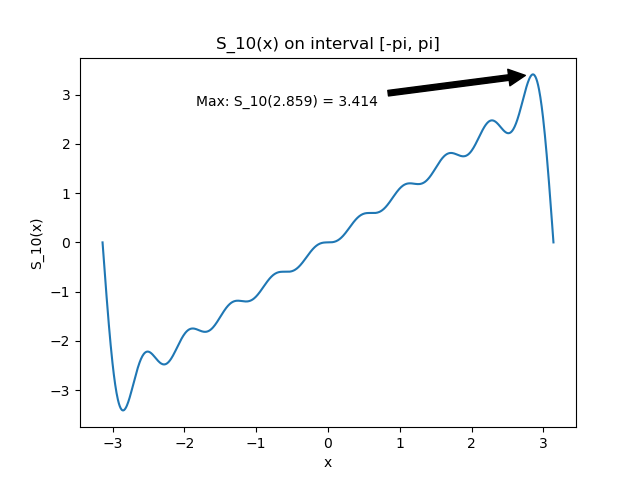
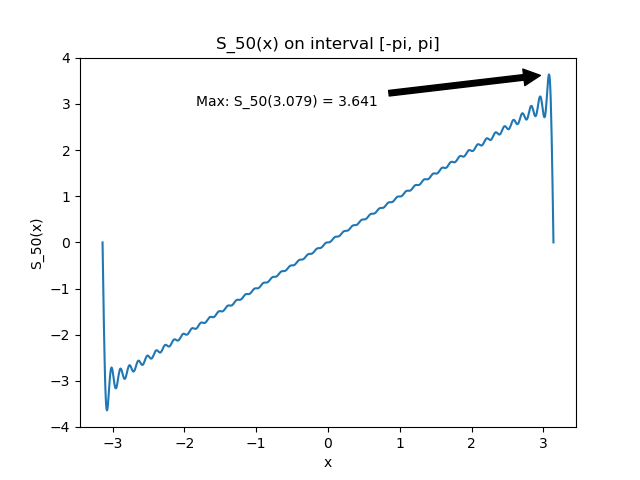
**Part 2**

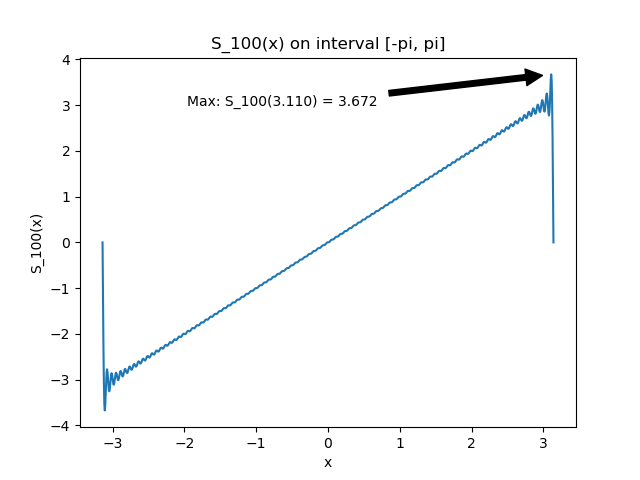
1. The series converges absolutely for all values of x (by comparison test with 1/k!), hence it is defined in the interval.
2. The series converges uniformly and thus is continuous

**Problem 2**

1. Graphs:

­­





1. Program printout

In the program below, the coefficients b­­k have been computed in the function “compute\_bk”. In this function I used the python library “scipy” and it’s “integrate.quad” function.

import numpy as np

import matplotlib.pyplot as plt

from scipy import integrate

def main():

# 1 Plot graphs of S\_n(x) for n = 10, 50, 100

vals = [10, 50, 100]

for n in vals:

plot\_S(n, "S\_{}.png".format(str(n)))

"""

x (float): value to compute fourier Series

n (int): number for approximation

"""

def S\_n(x, n):

result = 0

for k in range(1, n+1):

result += compute\_bk(k)\*np.sin(k\*x)

return result

"""

steps (int): number of steps to approximate integral

k (int): index of coefficient

"""

def compute\_bk(k):

i = integrate.quad(lambda x: x\*np.sin(k\*x), -np.pi, np.pi)[0]

return (1/np.pi) \* i

"""

n (int): n approximation of fourier series

filename (str): name of file to save plot

"""

def plot\_S(n, filename):

# divide interval:

x = np.linspace(-np.pi, np.pi, 1000)

s = np.vectorize(lambda x: S\_n(x, n))

y = s(x)

p = (x[np.argmax(y)], y.max())

fig = plt.figure()

ax = plt.subplot(111)

ax.plot(x,y)

ax.annotate('Max: S\_{}({:.3f}) = {:.3f}'.format(n, p[0], p[1]), xy = p,

xycoords='data', xytext=(0.6, 0.9), textcoords='axes fraction',

arrowprops=dict(facecolor='black', shrink=0.05),

horizontalalignment='right', verticalalignment='top'

)

ax.set\_xlabel('x')

ax.set\_ylabel('S\_{}(x)'.format(str(n)))

ax.set\_title('S\_{}(x) on interval [-pi, pi]'.format(str(n)))

fig.savefig(filename)

if \_\_name\_\_ == '\_\_main\_\_':

main()

1. Gibbs phenomenon

We can see from the graphs above that the limit will go to 0.