Jacob Dennis

26 January 2016

Physic 411

Prof. Emanuel Gull

Homework 2

**Problem 1**

*Code*

import math

import numpy as np

import matplotlib.pyplot as plt

def TrueSinInt(a, b):

return math.cos(a) - math.cos(b)

def RectIntegration(f, a, b, N):

h = (b - a) / N

I = 0.0

for x in np.arange(a, b, h):

I += h \* f(x)

return I

def TrapIntegration(f, a, b, N):

h = (b - a) / N

I = 0.5 \* (f(a) + f(b)) \* h

for i in np.arange(1.0, N):

I += f(a + h \* i) \* h

return I

def SimpsonIntegration(f, a, b, N):

h = (b - a) / N

I = h \* (f(a) + f(b)) / 3.0

for i in np.arange(1.0, N, 2.0):

I += 4.0 \* f(a + i \* h) \* h / 3.0

for i in np.arange(2.0, N, 2.0):

I += 2.0 \* f(a + i \* h) \* h / 3.0

return I

def BooleIntegration(f, a, b, N):

h = (b - a) / N

I = h \* 2.0 / 45.0 \* (7.0 \* f(a) + 7.0 \* f(b))

for i in np.arange(1.0, N, 5.0):

I += 32.0 \* f(a + i \* h) \* h \* 2.0 / 45.0

for i in np.arange(2.0, N, 5.0):

I += 12.0 \* f(a + i \* h) \* h \* 2.0 / 45.0

for i in np.arange(3.0, N, 5.0):

I += 32.0 \* f(a + i \* h) \* h \* 2.0 / 45.0

for i in np.arange(4.0, N, 5.0):

I += 14.0 \* f(a + i \* h) \* h \* 2.0 / 45.0

return I

def LogErrorCalculation(IApproxFunc, TrueI, f, a, b, NList):

logErrors = []

loghRange = []

I = TrueI(a, b)

for N in NList:

I\_N = IApproxFunc(f, a, b, N)

loghRange.append(math.log10(abs(b - a) / N))

if (I\_N == I):

logErrors.append(0.0)

else:

logErrors.append(math.log10(abs(I - I\_N)))

return [loghRange, logErrors]

NRange = np.arange(1.0, 1000.0)

NRangeEvens = np.arange(2.0, 1000.0, 2.0)

NRangeFives = np.arange(5.0, 1000.0, 5.0)

RectErrorsandHs = LogErrorCalculation(RectIntegration, TrueSinInt, math.sin, 0.0, math.pi, NRange)

TrapErrorsandHs = LogErrorCalculation(TrapIntegration, TrueSinInt, math.sin, 0.0, math.pi, NRange)

SimpErrorsandHs = LogErrorCalculation(SimpsonIntegration, TrueSinInt, math.sin, 0.0, math.pi, NRangeEvens)

BooleErrorsandHs = LogErrorCalculation(BooleIntegration, TrueSinInt, math.sin, 0.0, math.pi, NRangeFives)

plt.clf()

plt.plot(RectErrorsandHs[0], RectErrorsandHs[1], label = 'Rectangular Rule')

plt.plot(TrapErrorsandHs[0], TrapErrorsandHs[1], label = 'Trapezoidal Rule')

plt.plot(SimpErrorsandHs[0], SimpErrorsandHs[1], label = 'Simpson Rule')

plt.plot(BooleErrorsandHs[0], BooleErrorsandHs[1], label = 'Boole\'s Rule')

plt.xlabel('log(h)')

plt.ylabel('log(error)')

plt.title('Errors of Various Integration Methods')

plt.legend(loc = 0)

plt.savefig('Homework 2 problem 1.png')

print 'Rectangular Integration: error = h ^', sum(np.asarray(RectErrorsandHs[1]) / np.asarray(RectErrorsandHs[0])) / len(RectErrorsandHs[1]) # Essentially taking the average slope

print 'Trapezoidal Integration: error = h ^', sum(np.asarray(TrapErrorsandHs[1]) / np.asarray(TrapErrorsandHs[0])) / len(TrapErrorsandHs[1])

print 'Boole Integration: error = h ^', sum(np.asarray(BooleErrorsandHs[1]) / np.asarray(BooleErrorsandHs[0])) / len(BooleErrorsandHs[1])

print 'Simpson Integration: error = h ^', sum(np.asarray(SimpErrorsandHs[1]) / np.asarray(SimpErrorsandHs[0])) / len(SimpErrorsandHs[1])

*Results*

Rectangular Integration: error = h ^ 2.36662662475

Trapezoidal Integration: error = h ^ 2.36662662475

Boole Integration: error = h ^ 0.231160143161

Simpson Integration: error = h ^ 5.02503671127

**Problem 2**

*Code*

import math

import numpy as np

import matplotlib.pyplot as plt

def func(x):

return 1.0 / (1.0 + 25.0 \* x\*\*2.0)

def LagrangeInterp(X, xdata, ydata):

L = []

for x in X:

LSumTerm = 0.0

for i in range(len(xdata)):

LProductTerm = ydata[i]

for j in range(len(xdata)):

if j != i:

LProductTerm \*= (x - xdata[j])/(xdata[i] - xdata[j])

LSumTerm += LProductTerm

L.append(LSumTerm)

return L

x1 = np.arange(0.0, 5.0, 0.1)

xdata1 = [0.0, 1.0, 3.0, 4.0]

ydata1 = [0.0, 3.0, 3.0, 4.0]

plt.clf()

plt.plot(xdata1, ydata1, 'ro', label = 'Data')

plt.plot(x1, LagrangeInterp(x1, xdata1, ydata1), 'b', label = 'Interpolation Function')

plt.title('Problem 2 - Lagrange Interpolation, Graph 1')

plt.xlabel('x')

plt.ylabel('f(x)')

plt.legend()

plt.savefig('Homework 2 Problem 2 Plot 1.png')

x2 = np.arange(-2.0, 2.0, 0.1)

xdata2 = np.arange(-1.0, 1.4, 0.4)

ydata2 = func(xdata2)

yRange = func(x2)

plt.clf()

plt.plot(x2, yRange)

plt.plot(x2, LagrangeInterp(x2, xdata2, ydata2))

plt.savefig('Homework 2 Problem 2 Plot 2.png')

xdata3 = np.arange(-1.0, 1.1, 0.1)

ydata3 = func(xdata3)

plt.clf()

plt.plot(x2, yRange)

plt.plot(x2, LagrangeInterp(x2, xdata3, ydata3))

plt.savefig('Homework 2 Problem 2 Plot 3.png')

*Results*

(see attached graphs)

**Problem 3**

*Code*

import math

import numpy as np

import matplotlib.pyplot as plt

def inverse(x):

return 1.0 / x

def TrapIntegration(f, a, b, h):#computes a trapezoidal integral for a given h, a, and b

N = (b - a) / h

I = 0.5 \* (f(a) + f(b)) \* h

for i in np.arange(1.0, N):

I += f(a + h \* i) \* h

return I

def AdaptiveTrapIntegration(f, a, b, h0, m):

I0 = TrapIntegration(f, a, b, h0)

I = [I0]

h = [h0]

for i in range(1, int(m)):

h.append(h[i - 1] / 2.0)

fSum = 0.0

N = (b - a) / h[i]

for j in np.arange(1.0, N, 2.0):

fSum += f(a + h[i] \* j)

Inext = 0.5 \* I[i - 1] + h[i] \* fSum

I.append(Inext)

return I

def Romberg(I):

R = [[I[0]], [I[1]]]

m = len(IntList)

R[1].append(R[1][0] + 1.0 / (4.0\*\*m - 1.0) \* (R[1][0] - R[0][0]))

for i in range(2, m):

Rtemp = [I[i]]

for j in range(i):

Rtemp.append(Rtemp[j] + 1.0 / (4.0\*\*m - 1.0) \* (Rtemp[j] - R[i - 1][j]))

R.append(Rtemp)

return R

IntList = AdaptiveTrapIntegration(inverse, 1.0, 2.0, 1.0, 5.0)

RombergList = Romberg(IntList)

Errors = []

hValues = []

for i in range(len(RombergList)):

Errors.append(math.log10(abs(math.log(2.0) - RombergList[i][i])))

hValues.append(math.log10(1.0 / (2.0\*\*i)))

plt.clf()

plt.plot(hValues, Errors)

plt.xlabel('log(h)')

plt.ylabel('log(Error)')

plt.title('Problem 3 - Romberg Integration')

plt.savefig('Homework 2 problem 3.png')

*Results*

(See attached graphs)