Jacob Dennis

Physics 411

28 March 2016

Prof. Gull

Homework 7

**Lorentzian Distribution**

*Code*

import random

import numpy as np

def LorentzianCDF(x):

return 1.0 / np.pi \* np.arctan(x) + 0.5

def inverseLorentzian(z):

return np.tan(np.pi \* z - np.pi / 2.0)

ZRange = []

for i in range(1000000):

ZRange.append(random.random())

ZRange = np.array(ZRange)

RandLorentzDist = inverseLorentzian(ZRange)

under1ran = 0

for i in np.arange(1000000):

if abs(RandLorentzDist[i]) < 1.0:

under1ran += 1

print 'Fraction of the randomly generated distribution that lies in the range abs(x) < 1:', under1ran/1000000.0

print 'Fraction of the actual distribution that lies in the range abs(x) < 1:', (LorentzianCDF(1.0) - LorentzianCDF(-1.0)) / LorentzianCDF(1000000.0)

*Results*

Fraction of the randomly generated distribution that lies in the range abs(x) < 1: 0.499806

Fraction of the actual distribution that lies in the range abs(x) < 1: 0.500000159155

**Monte Carlo Integration with Error Bars**

*Code*

import numpy as np

import random

import matplotlib.pyplot as plt

N = 2.0\*\*np.arange(4.0, 21.0)

IntegralResults = []

Errors = []

for n in N:

samplePoints = []

for i in range(int(n)):

samplePoints.append(random.uniform(1.0, 2.0))

tempSum = 0.0

for x in samplePoints:

tempSum += np.log2(x) / n

IntegralResults.append(tempSum)

Errors.append(1.0/n)

IntegralResults = np.asarray(IntegralResults)

Errors = np.asarray(Errors) / 2

print 'Integral Estimates Errors'

for i in range(len(IntegralResults)):

print '{:f} {:f}'.format(IntegralResults[i], 2.0 \* Errors[i])

plt.figure(1)

plt.clf()

plt.errorbar(np.arange(4.0, 21.0), IntegralResults, xerr = None, yerr = Errors)

plt.xlabel('log\_2(N)')

plt.ylabel('Integral Estimates')

plt.title('Monte Carlo Integration')

plt.savefig('Homework 7 - Monte-Carlo.png')

plt.close(1)

*Results*

Integral Estimates Errors

0.464328 0.062500

0.662811 0.031250

0.597246 0.015625

0.553640 0.007812

0.570491 0.003906

0.563624 0.001953

0.550997 0.000977

0.571274 0.000488

0.559023 0.000244

0.552831 0.000122

0.556412 0.000061

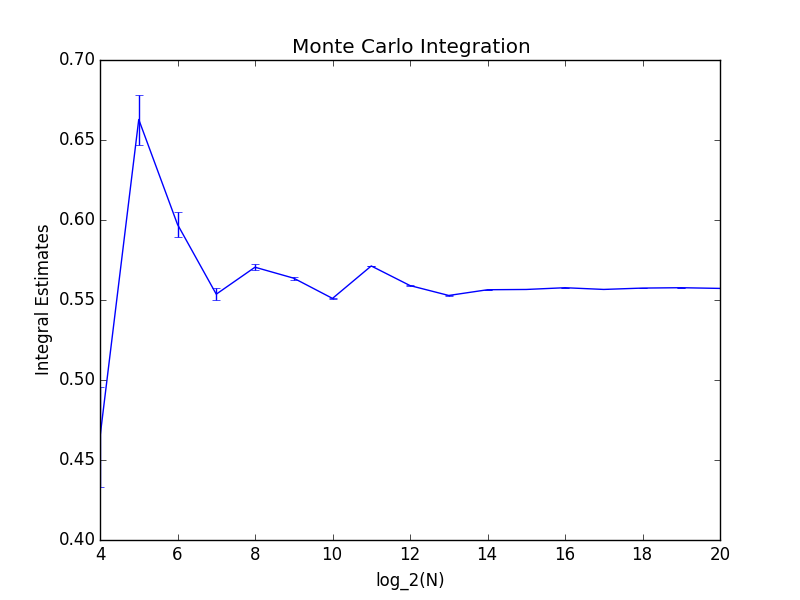
0.556585 0.000031

0.557678 0.000015

0.556607 0.000008

0.557452 0.000004

0.557679 0.000002

****0.557243 0.000001

**Bound States in a Potential Well (Version 2)**

*Code*

import numpy as np

import matplotlib.pyplot as plt

def V(c, x):

return c \* (x\*\*2 - x)

def FVector(r, x, c, E):

psi = r[0]

psi\_prime = r[1]

fpsi = psi\_prime

fpsi\_prime = (V(c, x) - E) \* psi

return np.array([fpsi, fpsi\_prime])

def RungeKutta(r0, x0, xf, N, f, c, E):

h = (xf - x0) / N

R = np.zeros((N + 1, 2))

R[0] = r0

X = np.arange(x0, xf, h)

for i in range(len(X)):

k1 = h \* f(R[i], X[i], c, E)

k2 = h \* f(R[i] + 0.5 \* k1, X[i] + 0.5 \* h, c, E)

k3 = h \* f(R[i] + 0.5 \* k2, X[i] + 0.5 \* h, c, E)

k4 = h \* f(R[i] + k3, X[i] + h, c, E)

R[i + 1] = R[i] + (k1 + 2 \* k2 + 2 \* k3 + k4) / 6.0

return np.flipud(np.rot90(R))

def RegulaFalsi(a, b, tol, f, c):

if np.sign(f(c, a)) == np.sign(f(c, b)):

print 'No root present.'

return None

xGuesses = []

IterationDifference = abs(a - b)

while (IterationDifference >= tol):

xNew = (f(c, b) \* a - f(c, a) \* b) / (f(c, b) - f(c, a))

xGuesses.append(xNew)

if (np.sign(f(c, a)) == np.sign(f(c, xNew))):

IterationDifference = abs(a - xNew)

a = xNew

elif (np.sign(f(c, b)) == np.sign(f(c, xNew))):

IterationDifference = abs(b - xNew)

b = xNew

elif xNew == 0.0:

return xGuesses

return xGuesses[-1]

def FindAValue(a, A, tol):

for i in range(len(A)):

if abs(A[i] - a) < tol:

return i

return None

def FunctionofB(c, b):

E = V(c, b)

xRange = np.linspace(0.0, 1.0, 1000)

r0\_left = np.asarray([1.0, np.sqrt(-2.0 \* E)])

r0\_right = np.asarray([1.0, np.exp(-np.sqrt(-2.0 \* E))])

PhiL = RungeKutta(r0\_left, 0.0, 1.0, 1000, FVector, c, E)

PhiR = RungeKutta(r0\_right, 0.0, 1.0, 1000 , FVector, c, E)

bIndex = FindAValue(b, xRange, .001)

return PhiL[1][bIndex] / PhiL[0][bIndex] - PhiR[1][bIndex] / PhiR[0][bIndex]

#Finding the ground energy for c = 1

groundEnergyPosition = RegulaFalsi(0.0, 0.5, .005, FunctionofB, 1.0)

print 'Ground State energy:', V(1.0, groundEnergyPosition)

excitedEnergyPosition = RegulaFalsi(groundEnergyPosition, 0.5, .005, FunctionofB, 1.0)

if excitedEnergyPosition == None:

print 'No more bound states.'

#Finding the energy for multiple cs

cRange = range(1, 102, 5)

plt.figure(0)

plt.clf()

xRange = np.linspace(0.0, 1.0, 1000)

for c in cRange:

print 'c:', c

groundEnergyPosition = RegulaFalsi(0.0, 0.5, .01, FunctionofB, c)

groundE = V(c, groundEnergyPosition)

print 'Ground State energy:', groundE

if (c % 20.0) == 1.0:

plt.plot(xRange, RungeKutta(np.asarray([1.0, np.sqrt(-2.0 \* groundE)]), 0.0, 1.0, 1000, FVector, c, groundE)[0][:-1],

label = 'c ={}'.format(c))

else:

plt.plot(xRange, RungeKutta(np.asarray([1.0, np.sqrt(-2.0 \* groundE)]), 0.0, 1.0, 1000, FVector, c, groundE)[0][:-1])

lastEnergyPosition = groundEnergyPosition

n = 1

while lastEnergyPosition != None:

nextEnergyPosition = RegulaFalsi(lastEnergyPosition, 0.5, .01, FunctionofB, c)

lastEnergyPosition, nextEnergyPostion = nextEnergyPosition, lastEnergyPosition

if lastEnergyPosition != None:

print 'Excited Energy {}: {}'.format(n, V(c, lastEnergyPosition))

n += 1

print 'No more bound states for this energy level'

plt.xlabel('x')

plt.ylabel('psi')

plt.title('Ground State Energies as a function of c')

plt.savefig('Homework 7 - Ground State Energies as a Function of C.png')

*Results*

For c = 1:

Ground State energy: -0.166918538329

No root present.

No more bound states.

For C from 1 to 101:

c: 1

Ground State energy: -0.17574678053

No root present.

No more bound states for this energy level

c: 6

Ground State energy: -0.209717853621

No root present.

No more bound states for this energy level

c: 11

Ground State energy: -0.217640732892

No root present.

No more bound states for this energy level

c: 16

Ground State energy: -0.288048612574

No root present.

No more bound states for this energy level

c: 21

Ground State energy: -0.305530720412

No root present.

No more bound states for this energy level

c: 26

Ground State energy: -0.322969973182

No root present.

No more bound states for this energy level

c: 31

Ground State energy: -0.341122127243

No root present.

No more bound states for this energy level

c: 36

Ground State energy: -0.359260995802

No root present.

No more bound states for this energy level

c: 41

Ground State energy: -0.37909554268

No root present.

No more bound states for this energy level

c: 46

Ground State energy: -0.399403154507

No root present.

No more bound states for this energy level

c: 51

Ground State energy: -0.421105738604

No root present.

No more bound states for this energy level

c: 56

Ground State energy: -0.444358677789

No root present.

No more bound states for this energy level

c: 61

Ground State energy: -0.469745602835

No root present.

No more bound states for this energy level

c: 66

Ground State energy: -0.496946436633

No root present.

No more bound states for this energy level

c: 71

Ground State energy: -0.526583397072

No root present.

No more bound states for this energy level

c: 76

Ground State energy: -0.329099815509

No root present.

No more bound states for this energy level

c: 81

Ground State energy: -0.340976665545

No root present.

No more bound states for this energy level

c: 86

Ground State energy: -0.353568790445

No root present.

No more bound states for this energy level

c: 91

Ground State energy: -0.368042078077

No root present.

No more bound states for this energy level

c: 96

Ground State energy: -0.382426305688

No root present.

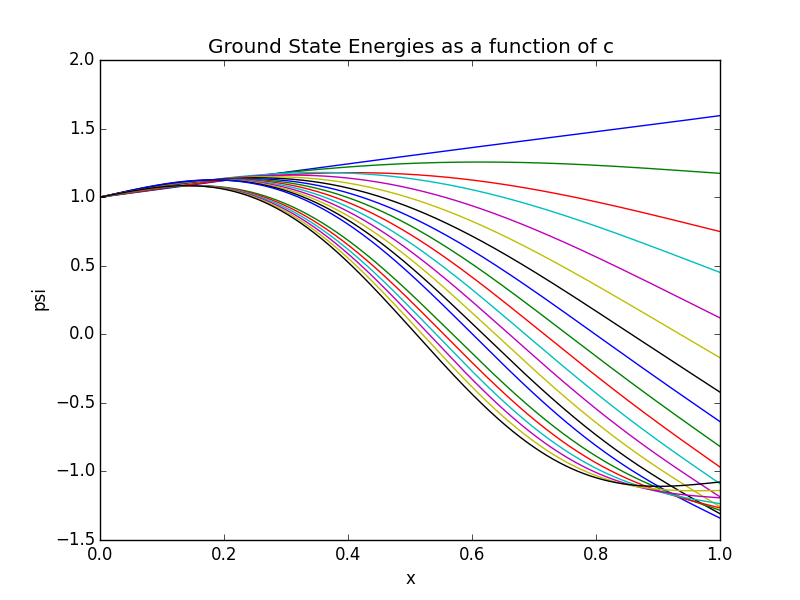
No more bound states for this energy level

c: 101

Ground State energy: -0.399060642755

No root present.

No more bound states for this energy level

****