Steven Wray

Physics 4610

2/6/12

**Monte Carlo Integration**

***Introduction***

This project demonstrates the use of Monte Carlo methods to provide a numerical approximation for the evaluation of an integral.

***Activities***

1. (a) Write an MCI routine to determine the value of . How can be found from simple integration?

The area of a circle is , and so the area of a unit circle is . The function describes the upper half of the unit circle as varies between and . This gives the formula:

The Monte Carlo routine will be used to estimate this integral. All points are chosen randomly from the square . Each point is examined to determine if it lies in the region of integration. The ratio of points in the region of integration to total sample points gives a numerical estimate for the integral. The final value is multiplied by to approximate . Using only the first quadrant allows the points to be more densely packed and gives a more accurate estimate for a given sample size.

(b) Use a simple random number generator and note the seed value. Make a graph of as a function of random numbers generated. How many random numbers are needed for the estimate to be accurate to within three digits? Five digits?

A linear congruential generator was used to obtain random values in part (b). Graphing the value of as versus the log the of number of points evaluated shows:

In order to achieve a value of accurate to three digits, a total of 250,000 sample points were needed. Even after one billion sample points, the estimate was not accurate to within five digits. It was 3.14155. Using a sample size larger than one billion is not realistic given the constraints of this project.

(c) Repeat the calculations in part (b) using the Mersenne twister to generate random numbers.

The graph of versus the log of the number of points is:

Using the second algorithm, the estimate of was accurate to within three digits using only 50,000 points. As in part (b), the estimate was not accurate to within five digits after one billion sample points (3.14169).

2. (a) Write an MCI routine using the Mersenne twister to find the mass of an ellipsoid given by:

Using Monte Carlo integration with 400 million sample points gives an estimate of 213.7730 kg.

To find the analytic solution, note that the -values of the ellipsoid varies from -5 to +5. At a fixed value of , the cross-sectional of the ellipsoid is an ellipse satisfying:

From elementary calculus, this ellipse has area:

So the total volume of the ellipsoid is:

Since the ellipsoid has uniform density of , the mass given by the analytic approach is .

The absolute error of the Monte Carlo method is and its relative error is

.

(b) Repeat the calculation for the ellipsoid in part (a) when it is truncated by .

Monte Carlo integration with 400 million sample points gives a value of 82.8516 kg for the mass of the truncated ellipsoid.

The volume of this truncated ellipsoid is:

The corresponding mass is 82.8488 kg.

The absolute error of the Monte Carlo method is kg. The relative error is

.

(c) Repeat the calculation in part (b) for a truncated ellipsoid having variable density given by .

When using a Monte Carlo method on this case, the value given to each randomly generated point that is inside the ellipse must be weighted by the density of the shape at that point. Using a sample of 400 million points gives an estimate for the mass of 148.1902 kg.

I was not able to find an analytic solution to the density integral:

(d) Give an example of this type of calculation for which an analytic solution would not be possible.

Suppose the density of the truncated ellipsoid is described by the function . This function does not have an elementary antiderivative, so no analytic solution will be possible.

Applying the Monte Carlo method to determine the mass of this ellipsoid gives a numerical solution of kg.

***Conclusions***

The Monte Carlo approach is easy to implement with a modern programming language. It provides solutions to problems that would otherwise be unapproachable. In addition to learning the details of the Monte Carlo algorithm, this project also gave me some good practice programming in Python. That is one of my goals for this course.

The one problem I had was trying to determine the mass of the truncated ellipse. I could not find an easy approach to solve this integral analytically. I used Mathematica to get the value for exercise 2b; I was not able to find an analytic solution for 2c. I assume there is some trick to setting up these calculations that I did not discover.

**APPENDIX**

***Source Code***

# MCI\_1b.py

# Python 3.2.2

# Steven Wray

# Physics 4610: Project 1/Exercise 1b

# Use Monte Carlo integration to approximate the value of pi

import math

# For this approximation, I will use a pseudorandom number generator that is

# simpler and less effective than the Mersenne twister. I have chosen a

# linear congruential generator.

#

# Given x\_n, the next value in the sequence of pseudorandom numbers is

# x\_(n+1) = A \* x\_n + C (mod M)

# Global variables for LCG:

A = 1664525

C = 1013904223

M = 2 \*\* 32

xi = 0 # Stores current element of the pseudorandom sequence

def randomSeed():

# Use the same random seed every time so the sequence can be reproduced

global xi

xi = 3814988170

def randomValue():

# Return a random number between 0 and 1

# Implemented as a subroutine to allow easy substitution of algorithms

global xi

xi = (A\*xi + C) % M

return xi/M

def main():

MAX\_EXPONENT = 9 # 10\*\*9 total points used for integral

POINTS\_PER\_DECADE = 4 # Current value of integral is output 4 times

# per decade to give dataset for graphing.

x = 0

y = 0

areaCounter = 0

totalPoints = 1

n = 0

k = 0

outputFile = open('Exercise1b.txt', 'w')

# Initialize random number generator

randomSeed()

for n in range(MAX\_EXPONENT):

for k in range(POINTS\_PER\_DECADE):

while totalPoints < 2.5 \* (k+1) \* 10\*\*n:

# Generate a random point that lies in the first quadrant

x = randomValue()

y = randomValue()

# Determine if that point lies within the region of integration. If so,

# defined by the function f(x) = sqrt(x\*\*2 + y\*\*2). If so,

# iterate the area counter.

if math.sqrt(x\*\*2 + y\*\*2) < 1:

areaCounter = areaCounter + 1

totalPoints = totalPoints + 1

# Calculate the current ratio of points in the region of integration

# to total points. Multiply by 4 to give an approximation of Pi.

# Output the result.

outputFile.write(str(totalPoints) + '\t')

outputFile.write(str(4 \* areaCounter/totalPoints) + '\n')

outputFile.close()

main()

# MCI\_1c.py

# Python 3.2.2

# Steven Wray

# Physics 4610: Project 1/Exercise 1c

# Use Monte Carlo integration to approximate the value of pi

import math

import random

# For part (c), I am using the Mersenne twister. This is the default random

# number generator in Python 3.

def randomSeed():

# Use the same random seed every time so the sequence can be reproduced

random.seed(3814988170)

def randomValue():

# Return a random number between 0 and 1

# Implemented as a subroutine to allow easy substitution of algorithms

return random.random()

def main():

MAX\_EXPONENT = 9 # 10\*\*9 total points used for integral

POINTS\_PER\_DECADE = 4 # Current value of integral is output 4 times

# per decade to give dataset for graphing.

x = 0

y = 0

areaCounter = 0

totalPoints = 1

n = 0

k = 0

outputFile = open('Exercise1c.txt', 'w')

# Initialize random number generator

randomSeed()

for n in range(MAX\_EXPONENT):

for k in range(POINTS\_PER\_DECADE):

while totalPoints < 2.5 \* (k+1) \* 10\*\*n:

# Generate a random point that lies in the first quadrant

x = randomValue()

y = randomValue()

# Determine if that point lies within the region of integration. If so,

# defined by the function f(x) = sqrt(x\*\*2 + y\*\*2). If so,

# iterate the area counter.

if math.sqrt(x\*\*2 + y\*\*2) < 1:

areaCounter = areaCounter + 1

totalPoints = totalPoints + 1

# Calculate the current ratio of points in the region of integration

# to total points. Multiply by 4 to give an approximation of Pi.

# Output the result.

outputFile.write(str(totalPoints) + '\t')

outputFile.write(str(4 \* areaCounter/totalPoints) + '\n')

outputFile.close()

main()

# MCI\_2a.py

# Python 3.2.2

# Steven Wray

# Physics 4610: Project 1/Exercise 2a

# Use Monte Carlo integration to find the mass of the ellipsoid

# 2x\*\*2 + 3y\*\*2 + z\*\*2 = 25

# with density rho = 1.000 [kg/m\*\*3]

#

# Since the ellipsoid has a uniform density of 1, we only need to calulate the

# volume of the ellipsoid.

import math

import random

def main():

TOTAL\_POINTS = 4 \* 10\*\*8 # Total points used for integral

# We will only sample points in the first octant. By symmetry, the

# total volume of the ellipsoid is 8 times the volume contained in the

# first octant.

x = 0

y = 0

z = 0

X\_MAX = 5 / math.sqrt(2)

Y\_MAX = 5 / math.sqrt(3)

Z\_MAX = 5

volume = 0

BOX\_VOLUME = X\_MAX \* Y\_MAX \* Z\_MAX

areaCounter = 0

n = 0

# Initialize random number generator

random.seed(3814988170)

for n in range(TOTAL\_POINTS):

# Generate a random point that lies in the first octant

x = X\_MAX \* random.random()

y = Y\_MAX \* random.random()

z = Z\_MAX \* random.random()

# Determine if that point lies within the region of integration.

# If so, iterate the area counter.

if 2 \* x\*\*2 + 3 \* y\*\*2 + z\*\*2 <= 25:

areaCounter = areaCounter + 1

# Calculate the ratio of points in the region of integration to total

# sample points. Multiply by the volume of the sample area, then

# by 8 to give the volume of the entire ellipsoid. Output the result.

volume = 8 \* BOX\_VOLUME \* areaCounter/TOTAL\_POINTS

outputFile = open('Exercise2a.txt', 'w')

outputFile.write(str(volume) + '\n')

outputFile.close()

main()

# Python 3.2.2

# MCI\_2b.py

# Steven Wray

# Physics 4610: Project 1/Exercise 2b

# Use Monte Carlo integration to find the mass of the ellipsoid

# 2x\*\*2 + 3y\*\*2 + z\*\*2 = 25

# which is truncated by z = +/-2 and x = -1

# and has density rho = 1.000 [kg/m\*\*3]

#

# Since the form has a uniform density of 1, we only need to calulate the

# volume.

import math

import random

def main():

TOTAL\_POINTS = 4 \* 10\*\*8 # Total points used for integral

# We will only sample points in the quadrant described by y >= 0 and

# z >=0. By symmetry, the total volume of the form is 4 times the volume

# contained in this quadrant.

x = 0

y = 0

z = 0

X\_MIN = -1

X\_MAX = 5 / math.sqrt(2)

Y\_MAX = 5 / math.sqrt(3)

Z\_MAX = 2

volume = 0

BOX\_VOLUME = (X\_MAX - X\_MIN) \* Y\_MAX \* Z\_MAX

areaCounter = 0

n = 0

# Initialize random number generator

random.seed(3814988170)

for n in range(TOTAL\_POINTS):

# Generate a random point that lies in the sample area

x = (X\_MAX - X\_MIN) \* random.random() + X\_MIN

y = Y\_MAX \* random.random()

z = Z\_MAX \* random.random()

# Determine if that point lies within the region of integration.

# If so, iterate the area counter.

if 2 \* x\*\*2 + 3 \* y\*\*2 + z\*\*2 <= 25:

areaCounter = areaCounter + 1

# Calculate the ratio of points in the region of integration to total

# sample points. Multiply by the volume of the sample area, then

# by 4 to give the volume of the entire truncated ellipsoid. Output

# the result.

volume = 4 \* BOX\_VOLUME \* areaCounter/TOTAL\_POINTS

outputFile = open('Exercise2b.txt', 'w')

outputFile.write(str(volume) + '\n')

outputFile.close()

outputFile.close()

main()

# MCI\_2c.py

# Python 3.2.2

# Steven Wray

# Physics 4610: Project 1/Exercise 2b

# Use Monte Carlo integration to find the mass of the ellipsoid

# 2x\*\*2 + 3y\*\*2 + z\*\*2 = 25

# which is truncated by z = +/-2 and x = -1

# and has density rho = x\*\*2 [kg/m\*\*3]

import math

import random

def main():

TOTAL\_POINTS = 4 \* 10\*\*8 # Total points used for integral

# We will only sample points in the quadrant described by y >= 0 and

# z >=0. By symmetry, the total volume of the form is 4 times the volume

# contained in this quadrant.

x = 0

y = 0

z = 0

X\_MIN = -1

X\_MAX = 5 / math.sqrt(2)

Y\_MAX = 5 / math.sqrt(3)

Z\_MAX = 2

volume = 0

mass = 0

weightedMass = 0

BOX\_VOLUME = (X\_MAX - X\_MIN) \* Y\_MAX \* Z\_MAX

areaCounter = 0

n = 0

# Initialize random number generator

random.seed(3814988170)

for n in range(TOTAL\_POINTS):

# Generate a random point that lies in the sample area

x = (X\_MAX - X\_MIN) \* random.random() + X\_MIN

y = Y\_MAX \* random.random()

z = Z\_MAX \* random.random()

# Determine if that point lies within the region of integration.

# If so, iterate the area counter and add the density of the point

# to the total mass.

if 2 \* x\*\*2 + 3 \* y\*\*2 + z\*\*2 <= 25:

areaCounter = areaCounter + 1

weightedMass = weightedMass + x\*\*2

# Calculate the ratio of points in the region of integration to total

# sample points. Multiply by the volume of the sample area, then

# by 4 to give the volume of the entire truncated ellipsoid. The current

# weighted mass assumes that each sample point has a volume of 1 m\*\*3.

# Adjust to reflect the volume of the form as calculated by the Monte

# Carlo method. Output the result.

volume = 4 \* BOX\_VOLUME \* areaCounter/TOTAL\_POINTS

mass = weightedMass \* volume / TOTAL\_POINTS

outputFile = open('Exercise2c.txt', 'w')

outputFile.write(str(mass) + '\n')

outputFile.close()

outputFile.close()

main()

# MCI\_2c.py

# Python 3.2.2

# Steven Wray

# Physics 4610: Project 1/Exercise 2b

# Use Monte Carlo integration to find the mass of the ellipsoid

# 2x\*\*2 + 3y\*\*2 + z\*\*2 = 25

# which is truncated by z = +/-2 and x = -1

# and has density rho = e\*\*(sin x) [kg/m\*\*3]

import math

import random

def main():

TOTAL\_POINTS = 4 \* 10\*\*8 # Total points used for integral

# We will only sample points in the quadrant described by y >= 0 and

# z >=0. By symmetry, the total volume of the form is 4 times the volume

# contained in this quadrant.

x = 0

y = 0

z = 0

X\_MIN = -1

X\_MAX = 5 / math.sqrt(2)

Y\_MAX = 5 / math.sqrt(3)

Z\_MAX = 2

volume = 0

mass = 0

weightedMass = 0

BOX\_VOLUME = (X\_MAX - X\_MIN) \* Y\_MAX \* Z\_MAX

areaCounter = 0

n = 0

# Initialize random number generator

random.seed(3814988170)

for n in range(TOTAL\_POINTS):

# Generate a random point that lies in the sample area

x = (X\_MAX - X\_MIN) \* random.random() + X\_MIN

y = Y\_MAX \* random.random()

z = Z\_MAX \* random.random()

# Determine if that point lies within the region of integration.

# If so, iterate the area counter and add the density of the point

# to the total mass.

if 2 \* x\*\*2 + 3 \* y\*\*2 + z\*\*2 <= 25:

areaCounter = areaCounter + 1

weightedMass = weightedMass + math.exp(math.sin(x))

# Calculate the ratio of points in the region of integration to total

# sample points. Multiply by the volume of the sample area, then

# by 4 to give the volume of the entire truncated ellipsoid. The current

# weighted mass assumes that each sample point has a volume of 1 m\*\*3.

# Adjust to reflect the volume of the form as calculated by the Monte

# Carlo method. Output the result.

volume = 4 \* BOX\_VOLUME \* areaCounter/TOTAL\_POINTS

mass = weightedMass \* volume / TOTAL\_POINTS

outputFile = open('Exercise2d.txt', 'w')

outputFile.write(str(mass) + '\n')

outputFile.close()

outputFile.close()

main()