Introduction to Monte Carlo Methods

5 May 2020

PHY2004W KDSMIL001

Contents

1 Introduction and Aim **1**

2 Activity **1**

3 Appendix **3**

1 Introduction and Aim

In this assignment we investigated the Monte Carlo method, a way of simulating systems by use of random numbers. We revisited some data from a previous assignment and created our own version of the data to analyse.

2 Activity

- **Histogramming Data**
  Firstly, we had a look back at some data from CP1, namely Activity1Data.txt, a list of 60 values generated with a certain mean ($\mu = 40$) and standard deviation ($\sigma = 2$) using a Monte Carlo method. We plotted these values on a histogram (Appendix 1) and plotted the gaussian based on a μ and σ calculated from the data itself. We also plotted the expected gaussian given the μ and σ used to generate the data.

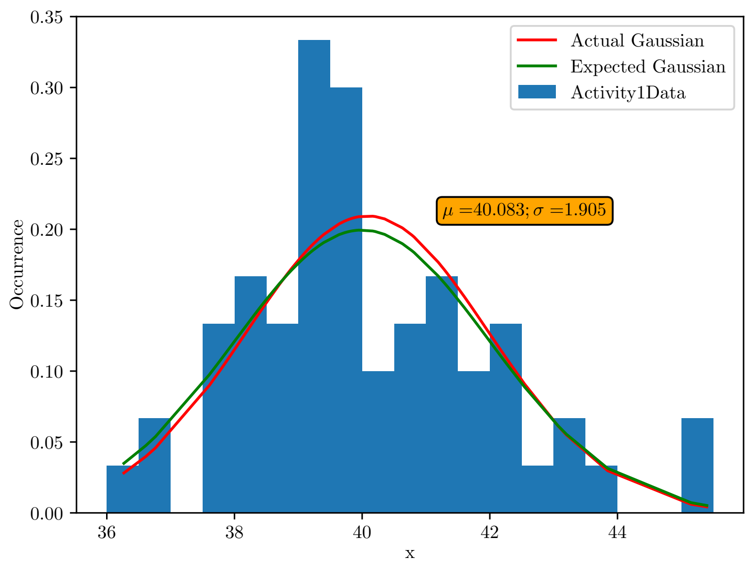

Figure 1: Activity1Data Histogram and Gaussians

Before scaling, the histogram was much larger than the gaussian as the area under a gaussian distribution is always 1, whereas the histogram doesn't necessarily have an area of 1. To resolve this and make the graph more readable, we set the density argument to True when creating the histogram, which normalises it, giving us the graph above.
Now, looking at the Actual Gaussian compared to the Expected Gaussian in

Figure (1), we see that they're a bit different, in fact the actual gaussian peaks slightly higher than the expected one. This is probably because the method of generating data isn't perfect so we can expect slight variations from dataset to dataset. We will investigate this further in the next section.

- **Generating Random Numbers**
  Next, we created some of our own data to analyse in the same way (Appendix 2). Below is an example of the file we created to analyse.

```
Random numbers drawn from Gaussian with mu = 40.0 and sigma = 2.0
1 40.4
2 40.941
3 41.158
```

Randomised Data

Doing exactly the same analysis as before, we have the plots below, an example of 2 different datasets created from the same initial conditions.

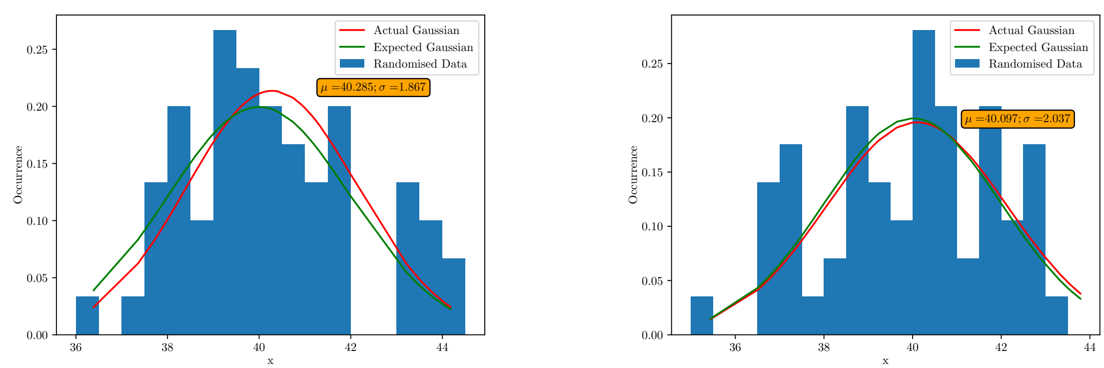

Figure 2: Randomised Data Histogram and Gaussians

As can be seen, the mean still sits at around 40 but due to the different distributions of the random numbers used to create the data it's never exactly 40.

3 Appendix

Appendix 1: CP4_2a

```python
from matplotlib import pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
from numpy import cos, pi, sin, sqrt, exp, random
import matplotlib
matplotlib.use('pgf')
matplotlib.rcParams.update({
    'pgf.texsystem': 'pdflatex',
    'font.family': 'serif',
    'text.usetex': True,
    'pgf.rcfonts': False,
})
# Creates a numpy array to hold the data
data = np.zeros(60)
i = 0
# Reads the data and puts it into the array
f = open(r'PHY2004W Computational\CP4\Activity1Data.txt', 'r')
header = f.readline()
for line in f:
    line = line.strip()
    columns = line.split()
    data[i] = float(columns[1])
    i += 1
f.close()
data.sort()
# A function to compute a Gaussian
def gaussian(x, mu, sigma):
    return (1/(sigma*sqrt(2*pi)))*exp(-((x-mu)**2)/(2*(sigma**2)))
# Computes values for the gaussian plot
dataMu = np.mean(data)
dataSigma = sqrt(np.var(data))
xplot = data
yplot = gaussian(xplot, dataMu, dataSigma)
gaussianMax = gaussian(dataMu, dataMu, dataSigma)
trueGaussian = gaussian(xplot, 40, 2)
# Defines some things for the histogram plotting
binwidth = 0.5
bins = np.arange(np.floor(min(data)), np.floor(max(data))+1, binwidth)
# Creates and plots the histogram
hist = plt.hist(data, bins, density=True, label='Activity1Data')
plt.xlabel('x')
plt.ylabel('Occurrence')
plt.draw()
# Plots the gaussian
gaussian = plt.plot(xplot, yplot, 'r-', label='Actual Gaussian')
```

```
trueGaussian = plt.plot(xplot, trueGaussian, 'g-', label='Expected Gaussian')
# Plots the graph
plt.legend()
plt.annotate(r'$\mu=$'+str(round(dataMu, 3))+'$; \sigma=$'+str(round(dataSigma, 3)), (dataMu, gaussianMax), textcoords='offset points', bbox=dict(facecolor='orange', edgecolor='black', boxstyle='round'))
# plt.show()
plt.savefig('PHY2004W Computational\CP4\Activity1DataHist.pgf')
```

Appendix 2: CP4_2b

```
from matplotlib import pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
from numpy import cos, pi, sin, sqrt, exp, random
import matplotlib
# matplotlib.use('pgf')
matplotlib.rcParams.update({
    'pgf.texsystem': 'pdflatex',
    'font.family': 'serif',
    'text.usetex': True,
    'pgf.rcfonts': False,
})
# Creates random data to use, with specific mean and standard deviation and writes it to a file
data = random.normal(40, 2, 60)
writeFile = open(r"PHY2004W Computational\CP4\RandomisedData.txt", 'w+')
writeFile.write("Random numbers drawn from Gaussian with mu = 40.0 and sigma = 2.0\n")
c = 0
for i in data:
    c+=1
    writeFile.write(str(c)+"  "+str(round(i, 3))+"\n")
writeFile.close()
# Creates a numpy array to hold the data
data2 = np.zeros(60)
i = 0
# Reads the data and puts it into the array
f = open(r"PHY2004W Computational\CP4\RandomisedData.txt", 'r')
header = f.readline()
for line in f:
    line = line.strip()
    columns = line.split()
    data2[i] = float(columns[1])
    i += 1
f.close()
data2.sort()
```

```python
# A function to compute a Gaussian
def gaussian(x, mu, sigma):
    return (1/(sigma*sqrt(2*pi)))*exp(-((x-mu)**2)/(2*sigma**2))
# Computes values for the gaussian plot
dataMu = np.mean(data2)
dataSigma = sqrt(np.var(data2))
xplot = data2
yplot = gaussian(xplot, dataMu, dataSigma)
gaussianMax = gaussian(dataMu, dataMu, dataSigma)
trueGaussian = gaussian(xplot, 40, 2)
# Defines some things for the histogram plotting
binwidth = 0.5
bins = np.arange(np.floor(min(data2)), np.floor(max(data2))+1, binwidth)
# Creates and plots the histogram
hist = plt.hist(data2, bins, density=True, label='Randomised Data')
plt.xlabel('x')
plt.ylabel('Occurrence')
plt.draw()
# Plots the gaussian and the expected gaussian
gaussian = plt.plot(xplot, yplot, 'r-', label='Actual Gaussian')
trueGaussian = plt.plot(xplot, trueGaussian, 'g-', label='Expected Gaussian')
# plt.show()
plt.legend()
plt.annotate(r'$\mu=$'+str(round(dataMu, 3))+'$; \sigma=$'+str(round(dataSigma, 3)), (dataMu, gaussianMax), textcoords='offset points', bbox=dict(facecolor='orange', edgecolor='black', boxstyle='round'))
plt.savefig('PHY2004W Computational\CP4\RandomisedDataHist1.pgf')
```

Appendix 3: CP4_3

```python
from matplotlib import pyplot as plt
import numpy as np
from scipy.optimize import curve_fit
from numpy import cos, pi, sin, sqrt, exp, random
import matplotlib
#matplotlib.use('pgf')
matplotlib.rcParams.update({
    'pgf.texsystem': 'pdflatex',
    'font.family': 'serif',
    'text.usetex': True,
    'pgf.rcfonts': False,
})

numRuns = 1000
N = 100000
approxPi = np.zeros(numRuns)
for c in range(numRuns):
```

```
    xs = random.uniform(-1, 1, N)
    ys = random.uniform(-1, 1, N)
    inCircle = 0
    for i in range(N):
        if sqrt((xs[i]**2) + (ys[i]**2)) <= 1:
            inCircle += 1

    approxPi[c] = ((inCircle/N)*4)

print(np.mean(approxPi), sqrt(np.var(approxPi)))
approxPi.sort()
def gaussian(x, mu, sigma):
    return (1/(sigma*sqrt(2*pi)))*exp(-((x-mu)**2)/(2*sigma**2))
dataMean = np.mean(approxPi)
dataSigma = sqrt(np.var(approxPi))
xplot = approxPi
yplot = gaussian(xplot, dataMean, dataSigma)
trueGaussian = gaussian(xplot, 40, 2)

binwidth = 0.0001
bins = np.arange(np.floor(min(approxPi)), np.floor(max(approxPi))+1, binwidth)
# Creates and plots the histogram
plt.hist(approxPi, bins, density=True)
plt.xlabel('x')
plt.ylabel('Occurrence')
plt.draw()
plt.xlim(3.1, 3.2)

plt.plot(xplot, yplot, 'r-')
plt.show()
```