INTRODUCTION TO CALCULUS

MATH 1A

Data project 4: Monte Carlo

4.1. Instead of the Riemann integral $S_n = \sum_{a \leq \frac{k}{n} < b} f(\frac{k}{n})\frac{1}{n}$ approximating the integral $\int_a^b f(x)\,dx$, we can also chose random numbers x_k and produce $S_n = \sum_{k=1}^n f(x_k)\frac{b-a}{n}$. Computing the integral as such is called a **Monte Carlo** integration. It turns out to be equivalent to the **Lebesgue integral**, a more sophisticated integral which is required in real analysis or probability theory.

4.2. How do we generate random numbers? There are two fundamental types. The first uses a **dynamical system** to generate pseudo random numbers. An example is to start with a **seed** $x_0 = 0.4$ (which could also depend on the time or other metrics in the computer) and then compute $g^j(x)$ where $g(x)$ is a chaotic map like $g(x) = 4x(1-x)$. The random number generator can then be seeded like $k = 234$. The random numbers x_j are then $x_j = g^{k+j}$ where j goes from 1 to n. This is not bad as we have seen in the last project that it even matters whether we take $g(x) = 4x(1-x)$ or $g(x) = 4x - 4x^2$. Here is code:

```
x0=0.4; k=234; n=10; g[x_]:=4x(1-x);
x1=Last[NestList[g,x0,k-1]]; NestList[g,x1,n-1]
```

4.3. Pseudo random numbers could also be generated (a bit more costly of course) from the digits of pi.

```
n=1000; X[k_]:=Mod[N[Pi*10^k,2n],1];
f[x_]:=Sin[x]; a=0; b=Pi; Sum[f[Pi*X[k]],{k,1,n}]*(b-a)/n
```

Problem 1. Use at least 20 digits of π to find a numerical approximation of the integral $\int_0^1 x^2\,dx$. Make groups of 2 digits to form numbers. Start with the digits after the decimal point. The first number $x_1 = 14/100$, the second $x_2 = 15/100$. Now write down the Riemann sum and give the result.

4.4. Problem 2. We want you to compute the area of the Mandelbrot set. You can do that by hand, printing out a version in which the coordinate axes are given and splitting it up into a grid, then counting. You can also use a machine. Here is some Mathematica code:

```
M=Compile[{x,y},Module[{z=x+I y,k=0}, While[Less[Abs[z],2]
  &&Less[k,1000],z=N[z^2+x+I y];++k];Floor[k/1000]]];
9*Sum[M[-2+3 Random[],-1.5+3 Random[]],{100000}]/100000
```

On an Online page like *https : //www.tutorialspoint.com/execute_python_online.php* you can also execute some Python code. If there is somebody in your group who knows how to run Python locally, you might get better results. The code should be pretty self-explanatory. If you are interested 1j denotes i. We shoot random points into a square of side length 3 (and area 9) and count the number of hits of the Mandelbrot set.

```
from numpy import random
import math

iterations = 200
samples = 1000

def M(x,y):
    z = x+y*1j
    k = 0
    while abs(z)<2 and k<iterations:
      z = z*z+ x+y*1j
      k += 1
    return math.floor(k/iterations)

sum = 0
n = 0
while n < samples:
   n += 1
   sum += M(-2 + 3*random.rand(), -1.5 +3*random.rand())

final = 9*sum/samples
print(final);
```

The Mandelbrot set can be generated quickest with the open source ray tracer Povray:

```
camera{location <-1,0,3> look_at <-1,0,-3> right <0,16/9,0>
up <0,0,1>} plane{z,0 pigment{mandel 200 color_map{[0 rgb 0]
[1/6 rgb <1,0,1>][1/3 rgb <1,0,0>][1 rgb 0]}} finish{ambient 1}}
```

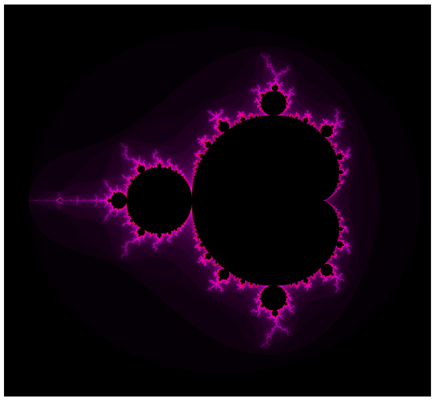

FIGURE 1. The Mandelbrot set.

4.5. Task: solve the two short problems. Author a 2 slide presentation or a document with two pages in which the two problems are solved, then submit this as a PDF. Please include the names of each of the group members who worked on the project.

OLIVER KNILL, KNILL@MATH.HARVARD.EDU, MATH 1A, HARVARD COLLEGE, SPRING 2021