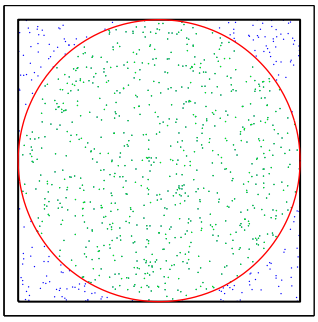
Notes on Monte Carlo integration

Let's consider a circle of radius $r = 1$ inside a square of side $l = 2$. If we generate a uniform distribution of points that covers the square, the ratio between the area of the circle A_c and the area of the square A_s, is equal to the ratio between the number of points that fall inside the circle N_c (i.e. that are less than r from the center) and the total number of points generated N:

$$\frac{A_c}{A_q} \simeq \frac{N_c}{N} \Rightarrow A_c \simeq \frac{N_c}{N} A_q \simeq \frac{N_c}{N} l^2$$



The following Python script is a function that implements the Monte Carlo integration to compute the surface of a circle.

```
import numpy as np

def montecarloIntegral(radius,center,N):
    yMin = center[1] - radius
    yMax = center[1] + radius
    xMin = center[0] - radius
    xMax = center[0] + radius

    areaRect = (xMax - xMin) * (yMax - yMin)

    X = np.random.uniform(xMin,xMax,N)
    Y = np.random.uniform(yMin,yMax,N)

    distance = np.sqrt((X-center[0])**2 + (Y-center[1])**2)
    pointsInside = np.sum(distance < radius)

    return areaRect * pointsInside / N
```

We can use the same tecnique also to compute the area of a more complex closed curves or to compute the integral of a function. For example:

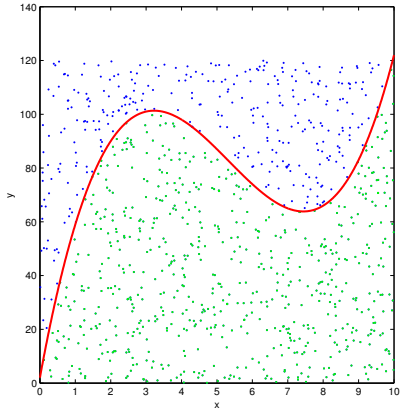
$$\int_a^b f(x)\mathrm{d}x$$

First of all, it is necessary to define a rectangle that contains all the curve and to compute its area A_r. After that, we generate N random points, from a uniform distribution, that cover all the rectangle. Then we have to count how many points N_f fall under the curve.

A random point (x_r, y_r) falls under the curve $f(x)$ if $y_r < f(x_r)$.

Finally the approximation of the integral is:

$$\int_a^b f(x)\mathrm{d}x \simeq \frac{N_f}{N} A_r$$


140
120
100
80
60
40
20
0
y
0
1
2
3
4
5
6
7
8
9
10
x