Monte Carlo

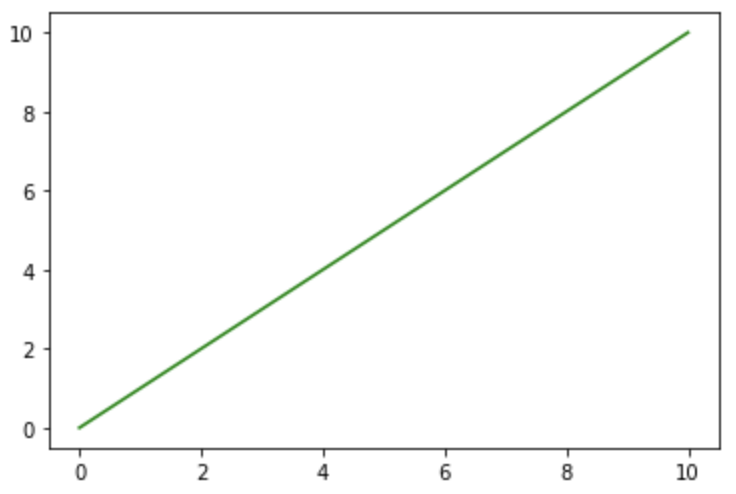
In this article we will go through an alternative means to integration via the monte Carlo method, but first we must understand how and why it works.

Background.

To demonstrate this, we will use the function  the integration of that function Given  would be

The would be = 50

This is self-evident in the following graph.



Now we have some context of what we are trying to do let’s introduce the monte Carlo simulation.

A formal definition of simulation is:

“A Monte Carlo simulation is a probabilistic model involving an element of chance. Hence, such a simulation is not deterministic but is probabilistic or stochastic, the results of each execution can vary from those of other runs.”

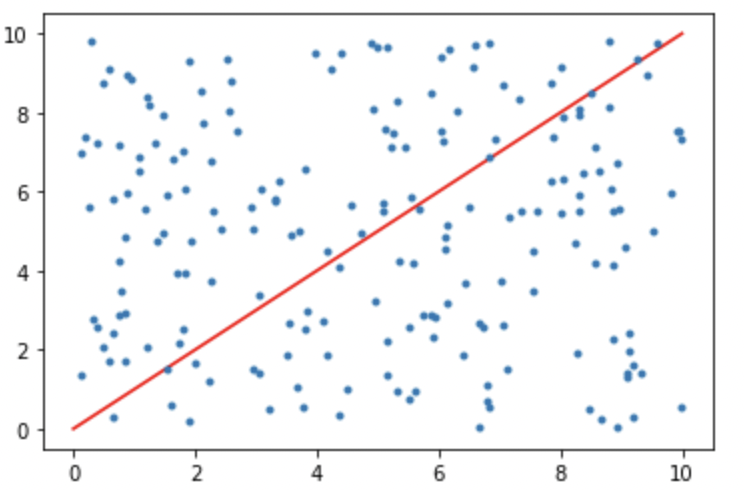
During World War II, physicists on the Manhattan Project developed the concept of Monte Carlo simulation. Scientists knew the behaviour of one neutron, but they did not have a formula for how a system of neutrons would behave. Although they needed to understand such behaviour to construct dampers and shields for the atomic bomb, experimentation was too time consuming and dangerous. John von Neumann and Stanislaus Ulam developed the technique of Monte Carlo simulation to solve the problem.

Back to the simulation.

Given we throw random “darts” at out area we would intuitively know that in a perfect world half would be above the line and half would be below.

Using this logic, we can calculate the area as:

Thus looking like this:



Now if we use the formula we would get:

Area of rectangle = 100

Darts below = 4955

Number of darts = 10,000

Thus, giving us:

As for approximation it’s not far off from the true area. This is more prevalent with higher draw count.

Now for the code

The libraries we will need.



Declare darts above and below as 0



Co-ordinates for the physical points of the draws. It will come in use later.



Now for the actual simulation draws are set to 0

Graphical user interface, text, application, email

Description automatically generated

We create a loop which iterates x number of times, in this case we set it as 10,000

Next, we generate uniform random numbers according to area space we want. (In this case it’s from 0-10)

Graphical user interface, text, application, email

Description automatically generated

Next, remember that counter and lists we created, will now add our random draws to them in lots of 50, as to not fill our screen with points. However, still maintaining the visual process.

Graphical user interface, text, application, email

Description automatically generated

Here we decide in which the point lands, whether it be in above or below.

The rdnx is random number generated by the uniform distribution.

Currently its set on: , however if it would be changed appropriately such as:

rdnx\*\*2 which would be

Graphical user interface, text, application, email

Description automatically generated

Lastly is to plot the function:

Shape, rectangle

Description automatically generated

Chart, scatter chart

Description automatically generated

Summary, this is a first principal approach to the monte Carlo method. Although this algorithm would have not much use as there are sophisticated libraries out there such as PYMC3. Its interesting to see how it works and how it can solve integration problems.

Chart, scatter chart

Description automatically generated

Shiflet, A.B. and Shiflet, G.W. (2014) *Introduction to computational science: Modeling and simulation for the Sciences*. Princeton (N.J.): Princeton University Press.