# Lecture 1

## Monte Carlo Integration

## Contents

# 1 Motivation

> *"Monte Carlo is an extremely bad method; it should only be used when all alternative methods are worse" - Alan Sokal*

Monte Carlo (MC) is a very general technique.

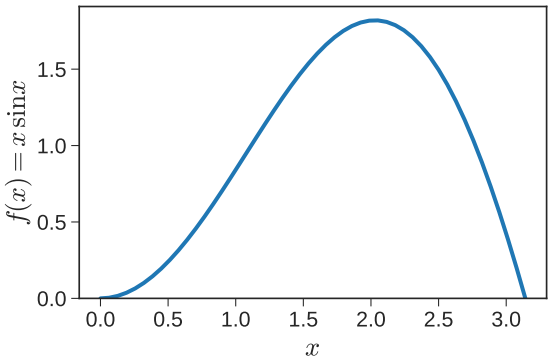
Instead of going from general $\rightarrow$ specific, let us do the exact opposite, and start with a specific example.[1] Consider a 1D integral,

$$I = \int_a^b f(x)\, dx. \tag{1}$$

This is still too general. Consider a *particular* 1D integral,

$$I = \int_0^\pi x \sin x \; dx. \tag{2}$$

Here $a = 0$, $b = \pi$ , and $f(x) = x \sin x$. The integrand looks like a roller-coaster.




It can be evaluated analytically, using integration by parts.[2]

$$I = \int_0^\pi x \sin x \; dx = -x \cos x + \sin x|_0^\pi = \pi.$$

# 2 Average and Integral

The average $\bar{f}$ of a 1D function $f(x)$ defined over a domain $a \leq x \leq b$ is defined as,

$$\bar{f} := \frac{\int_a^b f(x)dx}{\int_a^b dx} = \frac{\int_a^b f(x)dx}{b-a}. \tag{3}$$

From eqns 1 and 3,

$$\boxed{I = \bar{f}\,(b-a)} \tag{4}$$

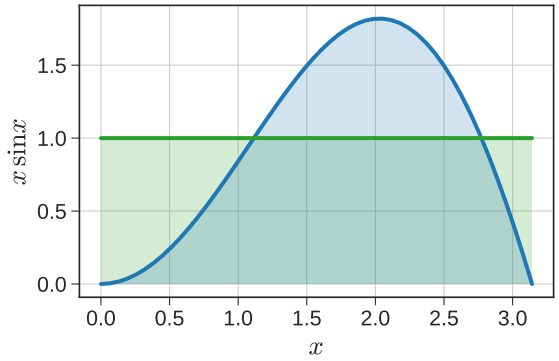
integral = average $\times$ domain size

[1] "show, then tell"

[2] Or a handy online calculator like Wolfram Alpha or integral-calculator

In 1D, $\bar{f}$ can be thought of as the mean "height" of the function. The integral is the area under the curve. The two are intimately related. Here is a key insight: **if the size of the domain is known, finding the integral is equivalent to finding the average value of the integrand**.

In our specific example, the average value is $\bar{f} = 1$, the size of the domain is $(\pi - 0) = \pi$, and the integral is $I = (1) \cdot \pi = \pi$. Graphically, the areas under the blue and green curves are equal:




Okay, so you might be wondering "where is Monte Carlo in all this?" Averages are the bridge between integrals and Monte Carlo.

## 2.1 Computing Averages

Suppose we are given a function $f(x)$ over a domain. How do we find the average?

We could sample $n$ points $\{x_1, x_2, ..., x_n\}$ on the domain, evaluate $f(x_i)$ at each of these points, and compute the average,

$$\bar{f} \approx \frac{\sum_{i=1}^{n} f(x_i)}{n}. \tag{5}$$

How do we go about picking the $n$ points? I can think of two methods right away:

Method 1

- pick equispaced points
- proxy for quadrature



```
def equispacedAverage(npts):
    xi = np.linspace(0, np.pi, npts)
    fi = xi * np.sin(xi)
    return np.mean(fi)
```

Method 2

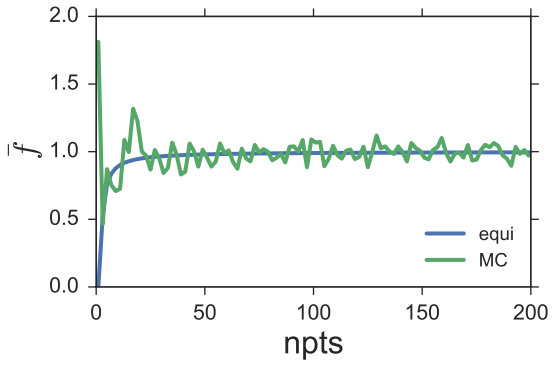
- pick points randomly (but uniformly)
- proxy for MC



```
def randomAverage(npts):
    xi = np.random.uniform(0, np.pi, npts)
    fi = xi * np.sin(xi)
    return np.mean(fi)
```

```
# using npts = 10 points.
print(equispacedAverage(10), randomAverage(10))
(0.890842865047, 0.812445848056)
```

Recall that the true $\bar{f} = 1$. So these averages look reasonable enough. Instead of using `npts = 10`, let us now systematically vary `npts` between 1 and 200.




We can make a few observations:

(i) As `npts` $\uparrow$, both estimates "converge" to $\bar{f} = 1$. Every nook and cranny of the domain gets explored as we increase the number of samples.

(ii) Method 1 converges more "systematically". Sample points spread out evenly. With method 2, there are pockets of the domain that are sparsely (or densely) sampled.

(iii) Method 2 requires more work (since we have to generate random numbers), and produces an inferior answer.

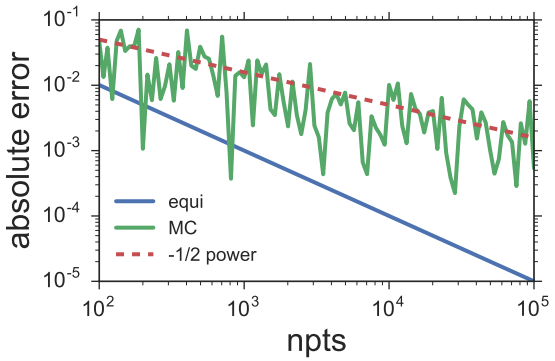
For 1D averages (integrals) Monte Carlo is a bad idea!

### 2.1.1 Convergence

Before we move on, let us do some post-mortem. As it turns out, some of the insights gleaned from studying this particular example are general. To be more quantitative, let us do a convergence study. This is a fancy way of saying "let's plot the absolute error,

$$\epsilon = |\bar{f}_{\text{true}} - \bar{f}_{\text{est}}|, \tag{6}$$

versus $n$ or `npts` on a log-log plot".




From the figure above, we find,

$$\epsilon_{\text{equi}} \sim \frac{1}{n}, \qquad \epsilon_{\text{mc}} \sim \frac{1}{\sqrt{n}}. \tag{7}$$

The manner in which the error decays with $n$ for both these cases is general. That is, it is not restricted to the particular integrand $f(x) = x \sin x$.

**Exercise**: Pick a different function, $f(x)$ and generate a similar convergence plot.

The integration method underlying method 1 (equispaced) belongs to a family of 1D integration methods called "Newton-Cotes",

$$I \approx \sum_{i=1}^{n} w_i f(x_i), \tag{8}$$

with $h = (b-a)/n$, and $x_i = a + (i-1)h$.

Method 1 is similar to the simplest member of this family, sometimes called the "rectangle rule" or "Riemann sum" with constant "weights" $w_i = h$. Other members of this family include trapezoidal and Simpson's rule, which can be thought of as weighted averages.

For trapezoidal rule $\epsilon \sim 1/n^2$, and for Simpson's rule $\epsilon \sim 1/n^4$. The dependence of the error on $n$ can been derived by considering Taylor series expansions.

On the other hand, the error in the MC estimate, $\epsilon \sim n^{-1/2}$ comes from a beautiful result from statistics called the central limit theorem. We will look at this in more detail later.

## 3 2D Integrals

Let us now consider the classic problem of finding the area of a unit circle using Monte Carlo.[3]

Finding the area is an integration problem. Similar to the 1D example, we can get at the integral by finding the average value of the integrand over the domain. And as before, the average can be computed using method 1 or method 2.

It is useful to pose the integration problem formally.

$$I = \int\limits_{x_1^2+x_2^2\leq 1} dx_1 dx_2 = \int_{-1}^{1}\int_{-1}^{1} g(x_1, x_2)\, dx_1 dx_2, \tag{9}$$

where,

$$g(x) = \begin{cases} 1, \text{ if } x_1^2 + x_2^2 \leq 1 \\ 0, \text{ otherwise.} \end{cases} \tag{10}$$

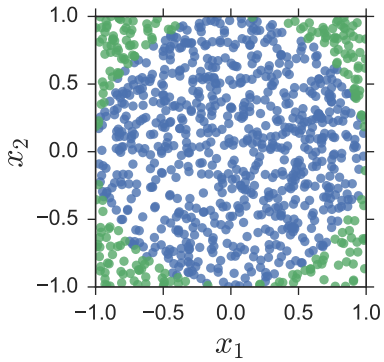
Thus, the area of the circle is the average value $\bar{g}$ times the size of the domain, which in this case is $[-1,1]^2 = 2 \times 2 = 4$. Thus, area of circle, $I = 4\bar{g}$.

For this problem, the true area is $\pi r^2 = \pi$, since radius = 1. Thus, $I = \pi$ and $\bar{g} = \pi/4$.

Consider the python program `MCdarts` in sec A.1. Let us throw 1000 darts, and get an estimate of the area.

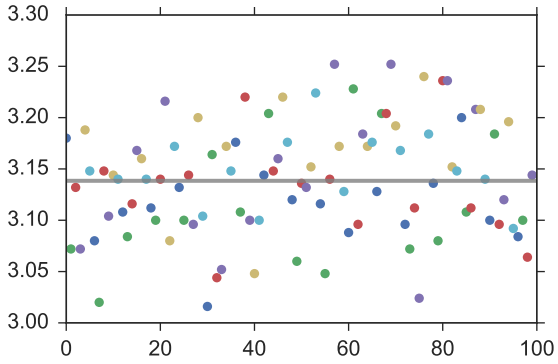
```
print("area", 4*MCdarts(1000, True))
area 3.148
```

Sometimes we get lucky!




[3] This is also sometimes called "integration by darts"

Due to its random nature, the MC "answer" is not fixed. If we run the calculation multiple times, we get a different answer each time.[4] If we repeat the experiment of throwing 1000 darts many (=100) times, we get something like this.



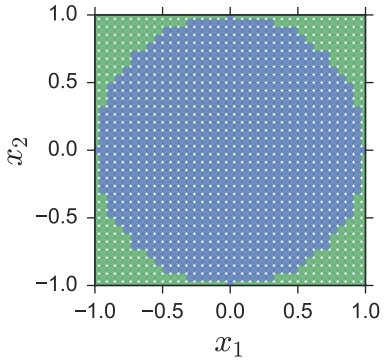

Note the considerable variation in the estimates. They gray line denotes the true average, $\pi$. The estimates are roughly within $\pm 5\%$ of this value. Eventually, we shall have to contend with this inherent variability in MC. For now, let us simply bookmark this as an important feature to revisit later.

## 3.1 Method 1 v/s Method 2 in 2D

Now, let us compute the area of a circle by throwing darts systematically on a "grid". The goal is to sample $g(x_1, x_2)$ at all of these points to find $\bar{g}$. The python program `equiDarts` in sec A.1 can be used to do this.

Let us throw 1000 darts, and get an estimate of the area. Note $\sqrt{1000} \approx 35$ along 1 dimension,

```
print("area", 4*equiDarts(1000, True))
area 2.94204081633
```

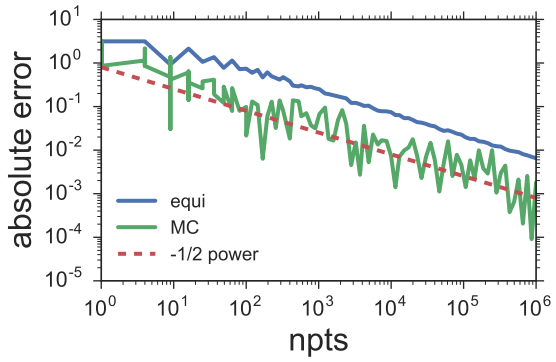



Note that this estimate is worse than most of the MC answers, even when you consider the variability within different MC runs.

### 3.1.1 Convergence

Let us systematically vary the number of darts, and compare methods 1 and 2 as before.

[4] This is in constrast to method 1, which is deterministic.




We can make the following observations for this specific example, which hold for any 2D integration problem in general:

(i) The error in both methods $\epsilon \sim n^{-1/2}$.

(ii) Since the function $g(x)$ is no longer smooth near the edges, $\epsilon_{\text{equi}}$ no longer decays as smoothly as it did in 1D, especially initially.

(iii) $\epsilon_{\text{mc}}$ is lower than $\epsilon_{\text{equi}}$ (unlike in 1D)

In fact, later we shall show, independent of dimension

$$\epsilon_{\text{mc}} \sim n^{-1/2} \tag{11}$$

For *any* systematic "quadrature" method,[5]

$$\epsilon_{\text{quad}} \sim n^{-p_{1D}/d}, \tag{12}$$

where $p_{1D}$ is the convergence of the method for 1D problems. Thus, the convergence rate of such quadrature methods worsens as dimensionality $d$ increases.[6] This is the **curse of dimensionality**.

## 4 Curse of Dimensionality

Many important computational problems in science and engineering are very high-dimensional. The number of dimensions can easily exceed $10^6$. Systematic quadrature methods essentially grind to a halt.

For such problems, MC is the only known general method. Often, particular features of such high-dimensional problems play into other strengths of MC, such as complex boundaries.

For the most complex high-dimensional problems, vanilla Monte Carlo can become inefficient. In such cases one has to resort to Markov Chain Monte Carlo (MCMC).

That will be the main topic of our class.

[5] rectangle, trapezoidal, Simpson's have $p_{1D}$ = 1, 2, and 4, respectively. This explains the $\epsilon_{\text{equi}} \sim n^{-1/2}$, observed in the figure above.

[6] Note that automatic quadrature methods for $d$-dimensional integration in standard numerical libraries like `nquad` in `scipy.integrate` generalize 1D quadrature methods (`quad`, in this case) to multiple dimensions

# 5 Summary

(i) integral = average $\times$ domain size;

(ii) For 1D averages, MC is a bad idea;

(iii) For higher dimensional integrals, standard quadrature methods suffer from the curse of dimensionality;

(iv) For higher dimensional integrals, the error in MC decreases as the square root of the number of samples due to the central limit theorem;

(v) estimates from MC are noisy

# 6 Problems

## 6.1 Thought Questions

(i) How can you determine an integral using Newton-Cotes or Monte Carlo when the domain is infinite? Suggest change of variables to map,

a. $[-\infty, \infty] \to [-1, 1]$

b. $[0, \infty] \to [0, 1]$

c. $[0, \infty] \to [-1, 1]$

(ii) How can you perform a convergence study when the true value of an integral is not known analytically?

(iii) Under what circumstances is standard quadrature superior to Monte Carlo integration?

(iv) What happens when you apply the equispaced or MC method to estimate an integral with an integrable singularity like,

$$I = \int_0^1 \frac{1}{\sqrt{x}}\, dx = 2. \tag{13}$$

## 6.2 Numerical Problems

(i) Infinite Domains

Consider an integral like,

$$I = \int_0^\infty e^{-x}\, dx = 1.$$

How would you evaluate the integral using $n = 1000$ Monte Carlo samples?[7]

**Trick**: One method of attack is to transform variables. Consider the transformation,

$$z = \frac{1}{1+x},$$

to reformulate the integral so that it has a finite domain. Describe how you would deal with any potential singularities.

[7] Note that the problem with infinite domains exists for Newton-Cotes types methods as well.

(ii) Direct MC in High Dimensions

The volume of a $n$-dimensional hypercube is $V_C = L^n$ (think line $\rightarrow$ square $\rightarrow$ cube).

The volume of a $n$ dimensional hypersphere of radius $R$ is

$$V_S = \frac{\pi^{n/2}}{\Gamma(\frac{n}{2}+1)} R^n,$$

where $\Gamma(x)$ is the "gamma function". Consider finding the volume of a hypersphere of radius $R$ by throwing darts inside a hypercube (in principle, this is far simpler than it sounds!) of edge length $L = 2R$.[8] The fraction of darts that land in the sphere is,

$$\frac{V_S}{V_C} = \frac{\pi^{n/2}}{2^n \Gamma(\frac{n}{2}+1)}.$$

Plot this ratio as a function of $n$ to understand why MC might not be a silver bullet.

(iii) Complicated Boundaries 1

Consider a domain $\mathcal{D}$ inscribed by a cardiod, which in polar coordinates is given by,

$$r = 2(1+\cos\theta), \quad 0 \leq \theta < 2\pi$$

(a) Find the area enclosed by $\mathcal{D}$.

(b) Find the average distance, $\bar{d} = \sqrt{x^2+y^2}$, of points $(x, y)$ inside the domain from the origin (0,0).

(c) Find a line $x = k$, which cuts the area in half.

Perform 10 independent simulations, each with $n = 10^5$ independent darts.

Report the mean and standard deviation of the 10 independent runs.

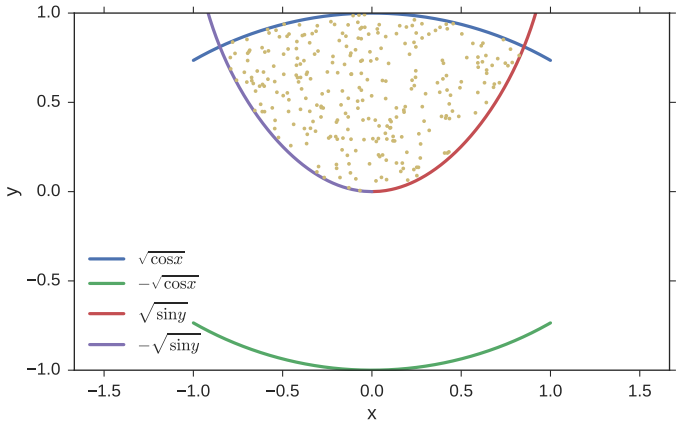
(iv) Complicated Boundaries 2

Using Monte Carlo integration, solve the following two-dimensional integral:

$$I = \frac{\iint\limits_{x^2<\sin(y), y^2<\cos(x)} e^{\sqrt{x^2+y^2}} dxdy}{\iint\limits_{x^2<\sin(y), y^2<\cos(x)} dxdy}$$

The region over which the integral needs to be calculated looks something like this:

[8] You don't have to do an MC simulation




(v) Integrals with Peaks

Compute the following sharply peaked integral,

$$I = \int_{-1}^{1}\int_{-1}^{1} 100 \exp\left(-10(x_1^2 + x_2^2)\right) dx_1 dx_2,$$

(a) Estimate the integral using different number of sample points $n$

(b) Error estimation: For suitably large value of $n$, repeat the calculation (with different seeds for random numbers), and report the mean and standard deviation of the replicates.

(vi) Complicated Boundaries 3

The radius of gyration $R_G$ of an object is the average distance of points from its center-of-mass, $\mathbf{r}_{cm}$.

Consider a heart-shaped domain $\mathcal{D} \in \mathbf{R}^2$ enclosed by the curve:

$$\mathbf{h}(t) = \left(16\sin^3 t,\ 13\cos t - 5\cos 2t - 2\cos 3t - \cos 4t\right),\ \ 0 \le t < 2\pi.$$


Assuming $\mathbf{r} = (x, y)$ is a point in $\mathbf{R}^2$, numerically estimate the following:

(a) The area of $\mathcal{D}$, given by,

$$A = \int_{\mathbf{r}\in\mathcal{D}} dx\ dy.$$

(b) The center of mass,

$$\mathbf{r}_{cm} = \frac{1}{A} \int_{\mathbf{r} \in \mathcal{D}} \mathbf{r} \, dx \, dy.$$

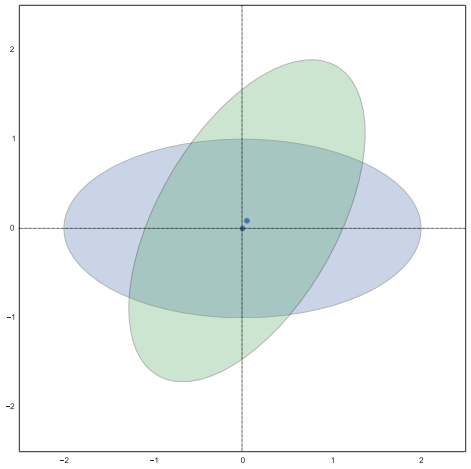
(c) The radius of gyration,

$$R_G^2 = \frac{1}{A} \int_{\mathbf{r} \in \mathcal{D}} (\mathbf{r} - \mathbf{r}_{cm})^2 \, dx \, dy,$$

where $(\mathbf{r} - \mathbf{r}_{cm})^2$ is the square of the Euclidean distance between $\mathbf{r}$ and the center of mass.

(vii) Intersection of Domains

Consider an ellipse $E_1$ with semi-major axis $a$ = 2 , and semi-minor axis $b$ = 1, centered at the origin $(0, 0)$, and oriented with its major axis along the $x$-axis. Consider a second ellipse $E_2$, which has the same shape as $E_1$, but is oriented at an angle $\theta = \pi/3$ with respect to the $x$-axis. Furthermore, the center of $E_2$ is at a distance $\delta = 0.1$ away from the origin.




Find the area of overlap between $E_1$ and $E_2$ using Monte Carlo.

(viii) Consider the probability distribution,

$$\pi(x, y) = \frac{1}{C} \exp\left[-\frac{1}{2}\left(x^2 y^2 + x^2 + y^2 - 8x - 8y\right)\right],$$

where $-\infty < x, y < \infty$, and $C \approx 20216.34$ is a normalization constant.

Find the expected value of $x$, $E[x]$, which is given by the integral,

$$E[x] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x \, \pi(x, y) \, dx \, dy.$$

to an accuracy of $10^{-3}$.

# A Appendix

## A.1 Python Code

### A.1.1 2D Monte Carlo Integration of a Unit Circle

```
def MCdarts(npts, isPlot=False):
    """input : #darts, visualize
       output: average value of g"""
    x1 = np.random.uniform(-1,1,size=npts)
    x2 = np.random.uniform(-1,1,size=npts)

    cond = x1**2 + x2**2 <= 1

    if isPlot: # toggle to visualize darts
        plt.plot(x1[cond],x2[cond],'o')
        plt.plot(x1[~cond],x2[~cond],'o')
        plt.xlim(-1,1); plt.ylim(-1,1)
        plt.xlabel('$x_1$', fontsize=18)
        plt.ylabel('$x_2$', fontsize=18)
        plt.axes().set_aspect('equal')

    return np.sum(cond)/float(npts)
```

### A.1.2 2D Equispaced Integration of a Unit Circle

```
def equiDarts(npts, isPlot=False):
    """total number of points are npts = ngrid*ngrid"""

    ngrid = int(np.sqrt(npts))

    x = np.linspace(-1, 1, ngrid)
    x1, x2 = np.meshgrid(x,x)

    cond = x1**2 + x2**2 <= 1

    if isPlot:
        plt.plot(x1[cond],x2[cond],'o', mec='none', alpha=0.8)
        plt.plot(x1[~cond],x2[~cond],'o', mec='none', alpha=0.8)
        plt.xlim(-1,1)
        plt.ylim(-1,1)
        plt.xlabel('$x_1$')
        plt.ylabel('$x_2$')
        plt.gca().set_aspect('equal')

        plt.tight_layout()
        plt.show()
```

```
npts = ngrid*ngrid

y = np.sum(cond,axis=0)
y = np.sum(y)

return y/float(npts)
```