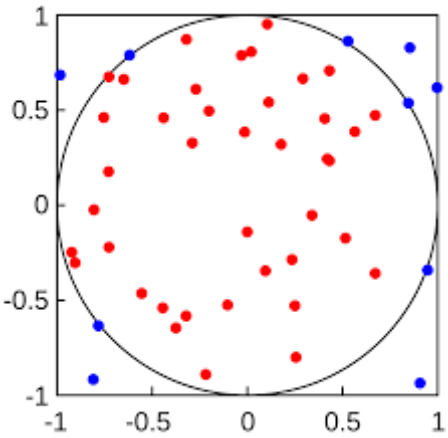
Montecarlo Integration



Objectives:

Compute $\int_\Omega f(\mathbf{x})d\mathbf{x}$ with $\Omega \subset \mathbb{R}^n$ using a MonteCarlo Algorithm:

$\int_\Omega f(\mathbf{x})d\mathbf{x} \simeq \frac{|\Omega|}{n}\sum_{i=0}^{n-1} f(\mathbf{x}_i)$

- Start with $\Omega = \{\prod_{i=1}^{n}[a_i, b_i]\}$: the domain of integration is an hyper-rectangle in n dimensions. Keep the dimensions as a parameter. Use a uniform distribution of points.
- Consider domains of integration of simple shapes, like an hypersphere, where you can reject points easily;

More complex stuff
- More complex domains: a general polygon (in 2D) or even polytopes in nD. Here you need to design a way to reject points outside the domain (think about it).
- Use different type of sampling: stratified sampling, importance sampling (Metropolis-Hastings). See [1]

Random numbers in C++ and a possible parallel implementation

In C++ random numbers are created by using a **random engine** and a **distribution**
A random engine is a source of random integer numbers generated according to a seed given to the constructor (a part a special engine called random_device). Once you have created the engine, you choose the distribution and by calling the distribution object bassing the engine you create a random number according to the given distribution. For exameple, to print 10 double random numbers with uniform distribution between -1 and 1 you do

```
#include <random>
#include <iostream>
std::random_device rd;
auto seed = rd(); // generate seed with randoem device
```

```
std::default_random_engine eng(seed);//use default random engine
std::uniform_real_distribution<double> dis(-1.0, 1.0);
for (auto i=0 ;i<10;++i)
   std::cout<< dis(eng)<<std::endl;
```

If you want to generate random numbers in parallel, a possibility is to let each process/thread use a local generator and distribution. In this case, it may be useful to use std::seed_seq to generate a good sequence of seeds, instead of using the random_device in each process. See https://en.cppreference.com/w/cpp/numeric/random/seed_seq for the use of random_seq.

References

[1] https://en.wikipedia.org/wiki/Monte_Carlo_integration
[2] Quinn_Ch310.pdf
[3] RandomNumbersAndDistributions.pdf