**Project: Monte Carlo integration**

# Part I
# Assignment

In many applications, you need to evaluate the integral of $f$ over a multidimensional region $V$ (typically a hypercube):

$$I = \int_V f(x)dx$$

where $x$ denotes some scalar or multidimensional variable. Monte Carlo integration involves a trick: defining some sampling distribution $p_s$ , which can never be zero if $f = 0$ and which you can draw samples from (described below). Using $p_s$, we rewrite the integral as the expectation of $f/p_s$ relative to the distribution $p_s$:

$$I = \int_V f(x)/p_s(x)p_s(x)dx \equiv \langle f/p_s \rangle_s \tag{1}$$

As you recall from class, the true expectation value of a random variable $g$ relative to $p_s$ can be well estimated using the sample mean $\bar{g} = \sum_k g_k/N$, where $g_k = g(x_k)$ is the value on some set of $N$ independent identically distributed (iid) samples $x_1 \dots x_N$ from $p_s$. Therefore, if we have a set of samples $x_k$ from $p_s$, we know the distribution of the sample mean of $f(x_k)/p_s(x_k)$ will be close to the true mean (which is $I$), with a variance $V(f/p_s)/N$:

$$\bar{I} \equiv \frac{1}{N}\sum_k f(x_k)/p_s(x_k) \tag{2}$$

$$\langle I \rangle_s = I \tag{3}$$

$$V(I)_s \simeq V(f/p_s)/N \tag{4}$$

In other words, we can **compute the integral value and an error estimate**, just by evaluating $f/p_s$ on a set of random numbers!

### Code examples

The `scipy.stats` library provides many pre-implemented distributions: you can create a random variable $x$, draw samples from it, and evaluate the sampling distribution $p_s$. As a concrete example, we can define a uniform distribution on $[-3, 3]$ and a normal distribution (restricted to the range [-3,3]) by the following two definitions

```
import numpy as np
import scipy.stats
distrib0 = scipy.stats.truncnorm(-3,3,loc=0,scale=1)
distrib1 = scipy.stats.uniform(loc=-3,scale=6)
```

We can perform one-dimensional Monte Carlo integration with the following code snippet:

```
# One dimensional example
def integrate_me(f, distrib, npts=100):
  x = distrib.rvs(npts)
  ps = distrib.pdf(x)
  f = f(x)
  mu = np.mean(f/ps)
  err = np.std(f/ps)/np.sqrt(npts)
  return mu,err
```

### Getting started questions

**Integral estimates 0: Uniform, 1d**: Using exactly $N = 10^3$ evaluations, estimate the following integrals (both value and accuracy), using a *uniform* distribution. (In some cases, extra credit may be awarded if you compare your answer against the exact value for the integral, with even more extra credit if you compare your answer for the accuracy with the expected value for that accuracy).

1. On $[-10, 10]$, $f(x) = 1$. Discuss your error estimate.
2. On $[-1, 1]$, $f(x) = P_\ell(x)^2$ for $\ell = 0, 1, 2, 3, 4, 5$ (do all 6). The function $P_\ell(x)$ can be produced in python with `scipy.special.eval_legendre(ℓ,x)`
3. On $[-10, 10]$, $f(x)$ is the standard normal PDF with mean of 0 and variance 1.
4. On $[-10, 10]$, $f(x)$ is a weighted sum of two normal distributions $p_1(x)$ and $p_2(x)$ with means $-3$ and 3 and standard deviations 1 and 3 respectively, using

$$f(x) = 0.7p_1(x) + 0.3p_2(x) \tag{5}$$

**Integral estimates 1: Gaussian, 1d**: Using exactly $N = 10^3$ evaluations, repeat parts (a,c,d) above (the standard normal PDF) but use a sampling distribution $p_s(x)$ which is the (appropriately truncated) standard normal distribution. What is your expected answer and variance? Discuss how the uncertainty estimate compares with the uncertainty

### Main project: Supporting tools

For the main project, you will need to change your code to work in two dimensions.
You may be interested in the following code blocks and examples

```
# Example of 2d distribution object
class MyTwoDUniform(object):
  def __init__(self, bounds=None):
      self.bounds =np.array(bounds)
  def rvs(self,npts):
    my_out = np.empty( (len(self.bounds),npts))
    for dim in np.arange(len(self.bounds)):
      my_out[dim] = np.random.uniform(low=self.bounds[dim][0], high=self.bounds[dim][1], size=npts)
    return my_out.T
  def pdf(self,x):
    V = np.prod([self.bounds[:,1]- self.bounds[:,0]])
    return np.ones(x.shape[0])/V

# Example of integration
my2d = MyTwoDUniform(bounds=[[-3,3],[-3,3]])
from scipy.stats import multivariate_normal
mu = np.array([1,-0.5])
cov = np.array([[ 1.  , -0.1 ],       [-0.1 ,  0.05]])
def f(x1, x2):
    x = np.array([x1, x2]).T
    return multivariate_normal.pdf(x, mu, cov)
integrate_me(lambda x: f(*(x.T)), my2d)
```

### Main project

**Rosenbrock integral**: Over the interval $[-5, 5]^2$, compute the integral of the rosenbrock function

$$\int L dx_1 dx_2 = \int dx_1 dx_2 \exp -((1 - x_1)^2 + 100(x_2 - x_1^2)^2) \tag{6}$$

Describe both the integral value, your estimate of your uncertainty, the number of function evaluations needed to produce your answer.

```
def ln_f(x1, x2):
    minus_lnL = np.array(np.power((1.-x1), 2) + 100.* np.power((x2-x1**2),2),dtype=float)
    return - minus_lnL
```

### Extra credit 0

Repeat the project, but use a (truncated?) normal distribution for the sampling distribution, with some mean and covariance chosen to improve sampling efficiency. Explain what parameters you selected for your sampling distribution and why, ideally using figures (e.g., contour plots of your distribution, superimposed with plots of the rosenbrock function).

### Extra credit 1

Perform the following integrals by Monte Carlo integration, mostly drawn from Gao et al arxiv:2001.05486, to a relative accuracy of $10^{-2}$, carefully explaining what sampling distributions you adopted and how many evaluations you performed:

1. High-dimensional unimodal gaussian: Integrate the standard (multivariate) normal distribution in 15 dimensions, over the hypercube range $[-5, 5]^{15}$.
2. The function $f(x) = 1$ if $|x| \leq 1$ and 0 otherwise, for $2, 3, 4$ dimensions. I recommend you integrate over the hypercube $[-1, 1]^n$ for $n$ the dimension.
3. Camel function: The superposition of two standard-normal-like functions in $n$ dimensions, over the **unit hypercube** $[0, 1]^n$ with $\alpha = 0.2$. Perform your integral in $n = 2, 4, 8, 16$ dimensions.

$$f(x) = \frac{1}{2}(\alpha\sqrt{\pi})^{-n}[\exp[-\alpha^{-2}\sum_k (x_k - 1/3)^2] - \alpha^{-2}\sum_k (x_k - 2/3)^2]$$

4. Polynomial: Also over the **unit hypercube**, perform the following integral in $n = 2, 4, 8, 16, 32, 64$ dimensions

$$f(x) = \sum_{k=1}^{n} -x_k^2 + x_k$$