$ Supplement Note 5: Calculation of π by Monte Carlo Methods

The value of π can also be calculated by means of the *Monte Carlo Method* using *random numbers*. Similar techniques have been extensively utilized in a broad areas of simulations.

The Monte Carlo simulation of the value of π uses the fact that the area of a unit circle ($r = 1$) is given by $r^2\pi = \pi$. In this approach N sets of points (x, y) which are *randomly and uniformly* distributed within the unit square , $0 \leq x, y \leq 1$, are generated by the *random number generator* package. The total number of points N_2 inside the unit circle in the first quadrant (i.e. $x^2 + y^2 \leq 1$ *whose area is* $\frac{\pi}{4}$) are then calculated. $\frac{N_2}{N}$ then provides an approximation to the ratio of the area of the unit circle in the first quadrant (area $= \frac{\pi}{4}$) to that of the square(area $= 1$). This leads to

$$\pi = \lim_{N\to\infty} 4 \times \frac{N_2}{N} \tag{1}$$

The above method can readily be extended to 3-dimension. The volume of a unit sphere ($r = 1$) is $\frac{4}{3}r^3\pi = \frac{4}{3}\pi$. *Its volume in the first quadrant is then equal to* $\frac{1}{6}\frac{3\pi}{4}$. N sets of *randomly* (and *uniformly*) distributed points (x, y, z) are generated within the unit cube $0 \leq x, y, z \leq 1$. The total number of points N_3 inside the unit sphere of the first quadrant , i.e. $x^2 + y^2 + z^2 \leq 1$ are then counted. $\frac{N_3}{N}$ then yields an approximation to the ratio of the volume of the unit sphere in the first quadrant (volume $= \frac{\pi}{8}$) to that of the unit cube (volume $= 1$). This leads to

$$\pi = \lim_{N\to\infty} 8 \times \frac{N_3}{N} \tag{2}$$

In general, the *statistical error* from such random variables varies as $\approx \frac{1}{\sqrt{N}}$. A python code *MonteCarloPi.py* for such simulation is listed below.

```
#MonteCarloPi.py
```

```
# Calculation of pi by Monte Carlo method
import numpy as np
n2, n3 ,PI = 0,0, np.pi
numt, piC , piS, Staterr = [], [], [], []
iprint = 5000
for ntest in range(1, 20000000):
    (x, y, z) = np.random.rand(3)
    x2, y2, z2 = x*x, y*y,z*z
    if x2 + y2  < 1: n2 += 1
    if x2 + y2+ z2  < 1: n3 += 1
    #if ntest % 500000 != 0: continue
    if ntest != iprint : continue
    iprint = int(1.5*iprint)
    p2 = 4*n2 /ntest
    p3 = 6*n3 /ntest
    err= 1/np.sqrt(2*ntest)
    print("{0:16.8f}, {1:16.8f}, {2:16.8f}".format(p2, p3, err))
    numt.append(ntest)
    piC.append(p2)
    piS.append(p3)
    Staterr.append(err)
import matplotlib.pyplot as plt
plt.plot(numt,piC,'+',  label= '$\pi$ from circle')
plt.plot(numt,piS,'*',  label= '$\pi$ from sphere')
plt.hlines(PI, 1, ntest-1, linestyles='dotted')
for x,err in zip(numt, Staterr):
    #plt.vlines(x, PI-err, PI+err, linestyles='dotted')
    plt.vlines(x, PI-err, PI+err)
plt.xlabel('number of random points')
plt.legend()
plt.show()
```