Ay 190 Assignment 15

Monte Carlo Methods

March 12, 2013

1 Measuring π with a MC Experiment

One can estimate the value of π by Monte Carlo methods by choosing N points at random from a square of length ℓ and tracking the number of points n that lie inside the circle of radius $\ell/2$ inscribed within the box. As the ratio of the areas of the circle and the square is $\pi(\ell/2)^2/\ell^2 = \pi/4$ and the number of points within each shape should be proportional to each shape's respective area, π can be estimated as $4n/N$.

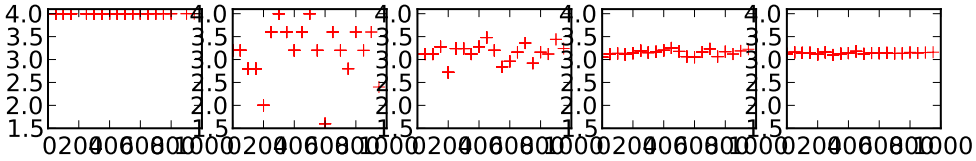

Figure 1: Selection of estimates of π calculated using the procedure above. From left to right, 1, 10, 100, 1000, and 10,000 samples were used for each estimate of π.

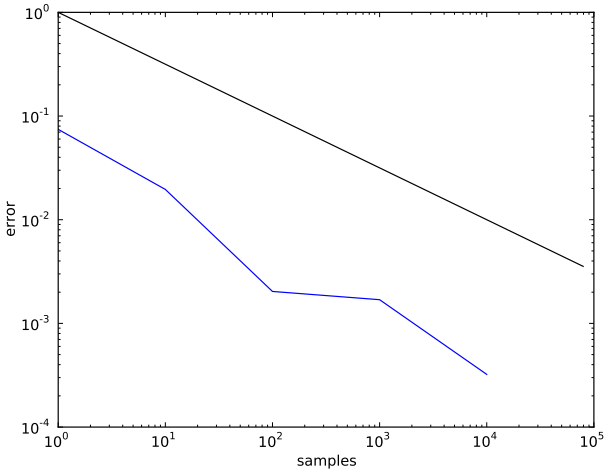

Figure 2: Convergence of estimates of π calculated using Monte Carlo methods. The results of a thousand iterations of the procedure were averaged to compute the estimate of π for a given number of samples. The estimate converages as $1/\sqrt{N}$ as expected (denoted by the black line).

2 Random Walk (Diffusion) in 1D

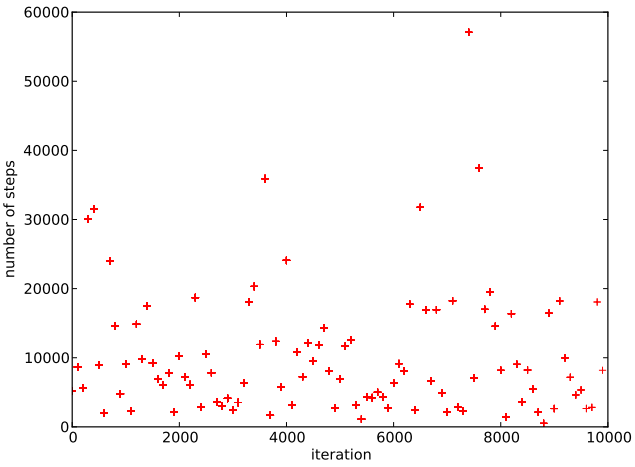

Figure 3: Number of steps required for a particle originally at the origin to reach edge of a 1D domain at [-1,1], for step sizes of 0.01 and the ability to move in either the negative or positive directions at each step. An average of 10,075 steps were required, close to the expected value of 10,000.

3 Random Walk (Diffusion) in 2D

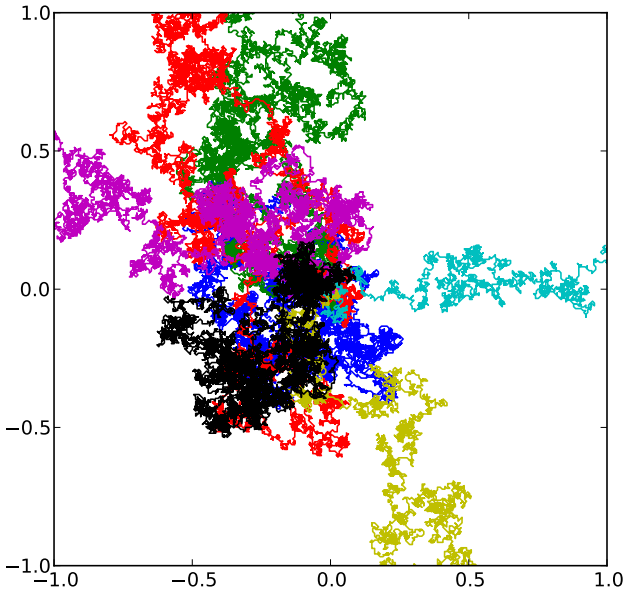

Figure 4: Paths of 7 particles originally at the origin constrained to move in a 2D domain in one of 8 directions (multiple of 45 degrees) with a step size of 0.01 at each step. After 10,000 steps, the particles were an average distance of 0.969 away from the starting point, close to the expected value of 1.

Appendices

The Python modules and scripts used in this assignment include

A The Poisson Equation Solvers: `set15.py` (see page 3-8)

A The MC Scripts

```
# set15.py
# created 3/12/13 by stacy kim

from pylab import *
from numpy import *
import sys


# MEASURING PI WITH AN MC EXPERIMENT ---------------------------------------------
niter=1000
nN=5
results=[]
ion()
for i in range(nN):
    N=10**i
    tot=0
    xx=[]
    yy=[]
    subplot(5,5,i+1)
    xlim([0,niter])
    ylim([1.5,4.1])
    for j in range(niter):
        random.seed(j)
        pairs = random.rand(N,2)*2-1 # x,y both \el [-1,1]
        n=sum([1 if (x*x+y*y)**0.5 < 1.0 else 0 for x,y in pairs])
        tot+=n
        if j % 50 == 0:
            xx+=[j]
            yy+=[4.*n/N]
            plot(xx,yy,'r+')
            draw()
    est_pi=4.*tot/N/niter
    print 'average estimate after',niter,'trials of',N,'samplings:',est_pi
    results.append(est_pi)

ioff()
savefig('sample_pi.pdf')
show()

clf()
x=pow(10,arange(0,5,0.1))
```

```
plot(x,[1/sqrt(val) for val in x],'k')
plot([10**i for i in range(nN)],abs(array(results)-pi))
xlabel('samples')
ylabel('error')
xscale('log')
yscale('log')
savefig('pi_mc_convg.pdf')
show()

"""
# RANDOM WALK (DIFFUSION) IN 1D ------------------------------------------------

step_size=0.01
all_steps=[]
niter=10000
ion()
xlim([0,niter])
xlabel('iteration')
ylabel('number of steps')
for i in range(niter):
    random.seed(i)
    steps=0
    path=0
    while abs(path) < 1:
        path+=step_size*(-1 if random.rand() < 0.5 else 1)
        steps+=1
    all_steps+=[steps]
    if i % 100 == 0 and i >= 100:
        plot(range(0,i+100,100),[all_steps[j] for j in range(0,i+100,100)],'r+')
        draw()

ioff()
savefig('walk_1D.pdf')
show()
print sum(all_steps)/float(niter)


# RANDOM WALK (DIFFUSION) IN 2D ------------------------------------------------

stepsize=0.01
nsteps=10000
niter=7
path=[]
for i in range(niter):
    random.seed(i)
    steps=[array([0,0])]
    for j in range(nsteps):
        th=pi/4*int(8*random.rand())
        steps.append(steps[-1]+stepsize*(-1 if random.rand() < 0.5 else 1)*array([cos(th),sin(th)]))
    path.append(steps)
```

```
path=array(path)
print 'average distance',sum([sqrt(p[-1,0]**2+p[-1,1]**2) for p in path])/niter

fig=figure()
ax=fig.add_subplot(111,aspect='equal')
for i in range(niter): plot(path[i,:,0],path[i,:,1])
xlim([-1,1])
ylim([-1,1])
savefig('2D_paths.pdf')
show()
"""
```