Random Numbers
Random Walk

Computational Physics

Random Numbers
Random Walk
Outline

- Random Systems
- Random Numbers
- Monte Carlo Integration Example
- Random Walk
- Exercise 7 Introduction
Random Systems

• Deterministic Systems
  • Describe with equations
  • Exact solution

• Random or Stochastic Systems
  • Models with random processes
  • Describe behavior with statistics
Example

*Particles in a Box*

- Consider 1cm$^3$ box
  - $\sim 10^{19}$ particles
  - motion and collisions
- Not interested in detailed trajectories
- Model behavior as result of action of random processes
- Statistical Description of Results: e.g. probability of finding particle at particular location
Generation of Random Numbers

• Most computing systems and computer languages have a means to generate random numbers between 0 and 1.

• Sequence generated from recursive relationship: $x_{n+1} = (a \times x_n + b) \mod m$
  
  • need a "seed" to start the process
  • same sequence generated by each seed "pseudorandom"

• in real systems, sequences may repeat eventually. Caveat Emptor!
Python

*Generation of Random Numbers with RandomState object*

- RandomState object is used to generate “streams” of random numbers.

- Important Methods:
  - `rand()` generates a sequence of *uniformly* distributed random numbers.
  - `randn()` generates a sequence of *normally* distributed random numbers.
  - `seed(arg)` “seeds” the random number stream with a fixed value `arg`.
About RandomState...

- RandomState is a class in NumPy. We use an instance of this class to manage random number generation.
- Random numbers are generated by methods in the class (e.g. the `rand` or `randn` methods).
- Each instance of RandomState comes with its own specific random number stream.
  - The random number stream is initialized ("seeded") when you create a RandomState instance.
  - You can reset the "seed" using the `seed()` method.
  - We can use the RandomState objects to have different random streams or to reset a stream.
Examples with RandomState

```python
from numpy.random import RandomState

# an instance of the RandomState class
# used to make a stream of random numbers
r = RandomState()
print 'generate array of 5 random numbers - uniform dist.'
print r.rand(5)

# if we seed the RandomState with an integer
# we always get the same stream
r2 = RandomState(12345)  # a random stream
r3 = RandomState(12345)  # another one - same seed

# these give the same results!!
print 'check two streams with same seed - normal dist.'
print 'first: ', r2.randn(5)
print 'second: ', r3.randn(5)
```

Output:

generate array of 5 random numbers - uniform dist.
[ 0.98772701  0.72617043  0.9203089   0.45822263  0.15032224]

check two streams with same seed - normal dist.
first:  
[-0.20470766  0.47894334 -0.51943872 -0.5557303   1.96578057]
second:  
[-0.20470766  0.47894334 -0.51943872 -0.5557303   1.96578057]
Monte Carlo Integration

• Algorithm:
  • Select random number pair \((x,y)\) from uniformly distributed sample
  • Check whether \((x,y)\) is above or below \(f(x)\) curve.
  • Repeat

• Fraction of points below curve is fraction of area below curve.
Monte Carlo Integration

Example: \( f(x) = x^2 \)

```python
import numpy as np
from numpy.random import RandomState

def f(x):  # define function to be integrated
    return x**2

# create an instance of RandomState
r = RandomState()

# create random x and y arrays
n = input('number of monte carlo trials: ')
x = r.rand(n)
y = r.rand(n)

sum = 0.
for i in range(n):
    # compare y to f(x)
    if( y[i] < f(x[i]) ):
        sum = sum + 1.

print '{0:d} Monte Carlo trials'.format(n)
print 'Monte Carlo Answer: {0:10.7f}'.format(sum/n)
print 'Exact Answer: {0:10.7f}'.format(1./3.)
```
Random Walk
1D Random Walk

```python
import numpy as np
import matplotlib.pyplot as pl
from numpy.random import RandomState

n = 1000  # number of steps
r = RandomState()

p = np.zeros(n)
p[0] = 0.0

for i in range(n-1):
    if (r.rand() >= 0.5):
    else:
```

- Initialize array for number of steps
- Start at position = 0
- Loop through n-1 steps
- `rand` is uniformly distributed: 0->1
- Take forward step if > 0.5
- Take backward step if < 0.5
7 Random Walks
100 steps

One D Random Walks

Position

Step
Random Walk Results

10 Million Trials with 100 Steps

Random Walk Properties:

1. Zero Mean
2. Dispersion which increases with number of steps
Random Walk with 10,000,000 “Walkers” demonstrate that:

\[ \text{Mean Square Distance} = \text{Number of Random Steps} \]

\[ \text{Mean Square Distance} = \frac{1}{N} \sum_{i=1}^{N} d_i^2 \]
Mean Square Distance
= Number of Random Steps
Understanding this result...

\[
z_N = \Delta z_N + \Delta z_{N-1} + \ldots + \Delta z_1 = \sum_{i=-1}^{N} \Delta z_i
\]

\[
z_N^2 = \left( \sum_{j=1}^{N} \Delta z_j \right) \left( \sum_{k=1}^{N} \Delta z_k \right) = \sum_{j,k=1}^{N} \Delta z_j \Delta z_k
\]

\[
= \sum_{j=1, k=j}^{N} \Delta z_j^2 + \sum_{j,k=1, k\neq j}^{N} \Delta z_j \Delta z_k = N \ell^2 + \sum_{j,k=1, j\neq k}^{N} \Delta z_j \Delta z_k.
\]

\[
<z_N^2> = N \ell^2 + \left< \sum_{j,k=1, j\neq k}^{N} \Delta z_j \Delta z_k \right>
\]

\[
<z^2> = N \ell^2
\]
Exercise
2D Random Walk

Θ is random number: 0 - 2π

s is step size

\[ X_{\text{step}} = s \cos \Theta \]

\[ Y_{\text{step}} = s \sin \Theta \]
Things we want to demonstrate

- For an ensemble of particles:
  - Mean Square Displacement is proportional to number of steps.
  - Constant of proportionality depends on the step size.

- For track of a single particle:
  - Mean Square Displacement after N steps for a single particle track is the same as Mean Square displacement for an ensemble of particles.