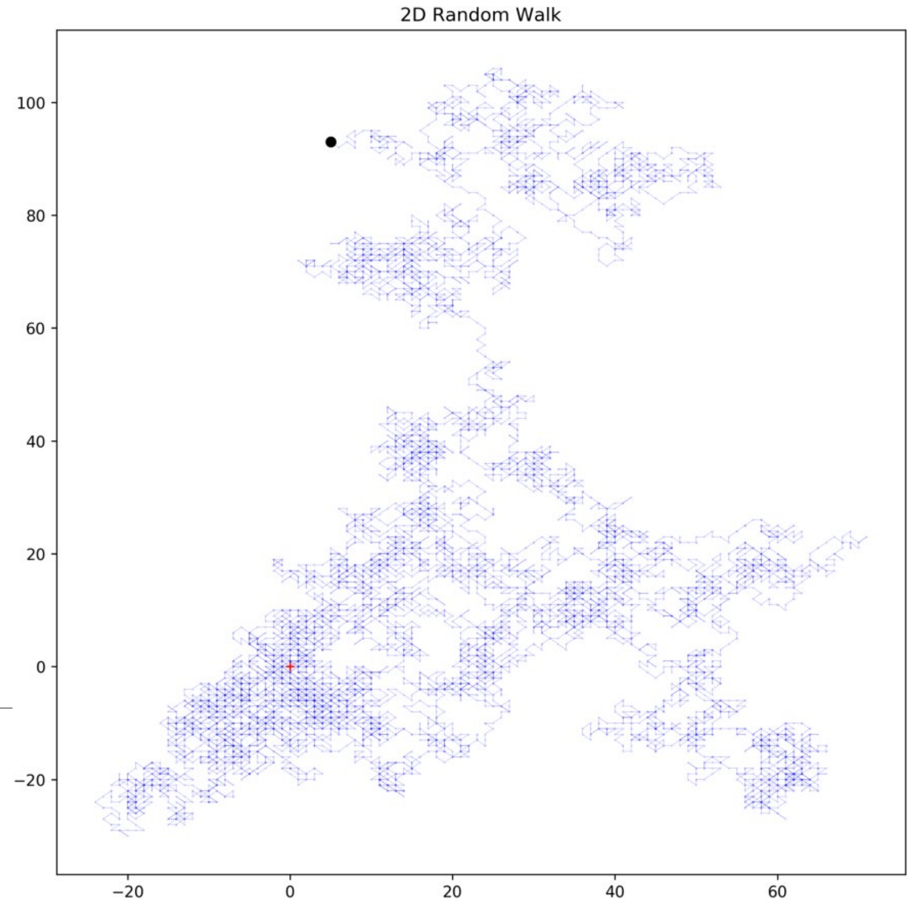
Simulation and Randomness

LECTURE9



Simulation of a random drunk walker over 10K steps

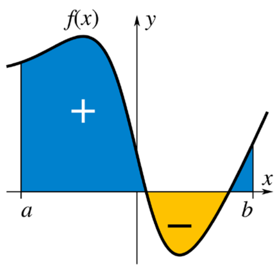
Organization of Lecture 9

- Simulation
- Numerical Integration using Uniform Data Points
- Randomness
- Monte Carlo Methods using Random Data Points
  - Monte Carlo Integration
  - Computing Pi
- Plotting Simulations (as we go along)

Simulation

- A term with broad interpretation
  - Refers to a mock event or model that mimics an underlying entity, real-world system or phenomenon
  - Used in variety of applications, including design of radiotheraphy treatments for cancer patients, construction, training for catastrophes, simulating wear and tear over the years on devices/structures.
- Definition narrows in computational science
  - Usually refers to a virtual “random” model to study an uncertain element.

Deterministic Numerical Integration



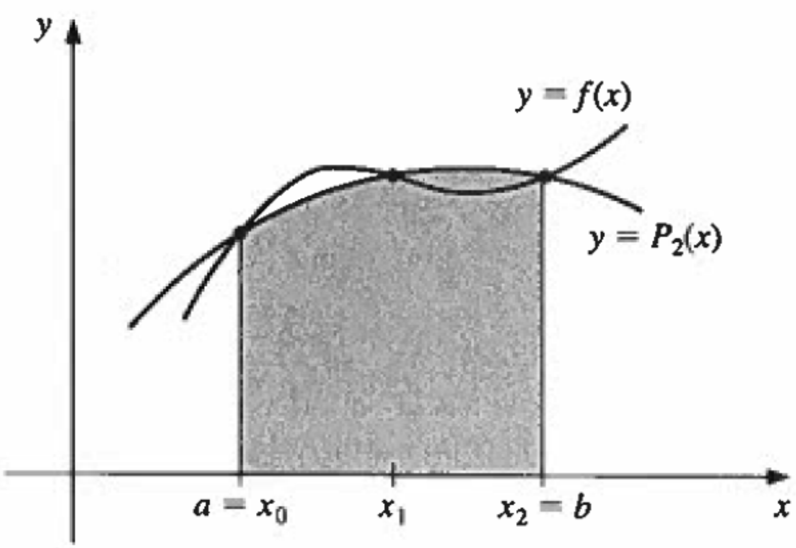
- Integration is a principal study of Calculus.
  - Differential equations and probability calculations solve *many* problems.
  - Hand calculation of integrals to find closed-form solutions can be intimidating
- Modern Computing provides a way to *estimate* a definite integral's value
  - Numerical integration or Quadrature
- Review a basic method called Simpson's Rule which uses a deterministic algorithm before tackling Monte Carlo Integration

Simpson's Rule

- Find the area under the curve f(x) from a to b, that is, $\int_a^b f(x)dx$.
- Approximate a polynomial function f(x) with a quadratic polynomial $P_2(x)$ with equally spaced points
  - $x_0 = a$, $x_1 = (a+b)/2$, $x_2 = b$.
- Derivation of the formula can be found

  in handout SimpsonsRule.pdf on D2L.
- Formula:

$$\int_a^b f(x)dx \approx \frac{(b-a)}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right].$$



- Notice we underestimate first half and overestimate second half in image.

Example

- Evaluate $\int_0^2 f(x)dx$ where $f(x) = \sin(x)$.
- On [0,2], Simpson's Rule has the form

$$\int_a^b f(x)dx \approx \frac{(2-0)}{6}\left[f(0) + 4f\left(\frac{0+2}{2}\right) + f(2)\right]$$

$$= \frac{1}{3}\,[f(0) + 4f(1) + f(2)]$$

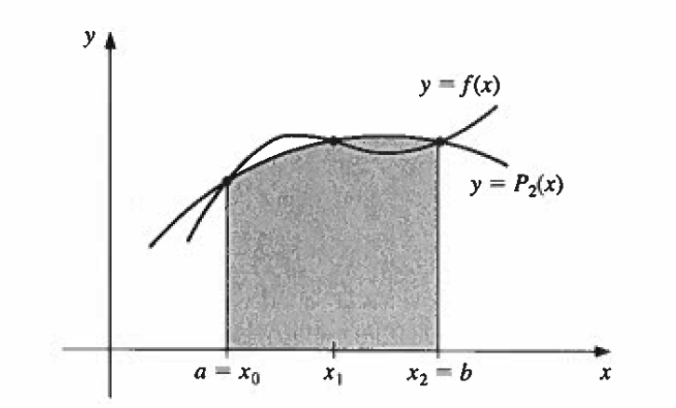
$$= \frac{1}{3}\,[0 + 4(0.8414171) + .90929]$$

$$= 1.425$$

- The exact value of $\int_0^2 f(x)dx$ where $f(x) = \sin(x)$ is 1.416

Simpson’s Rule

- Remember we underestimated first half and overestimated second half in the image below.



- We overestimated *f(x) = sin (x).*
- The error for Simpson’s rule is approximately

$$\frac{f^4\,(\varepsilon)}{2880}\,(b-a)^5$$

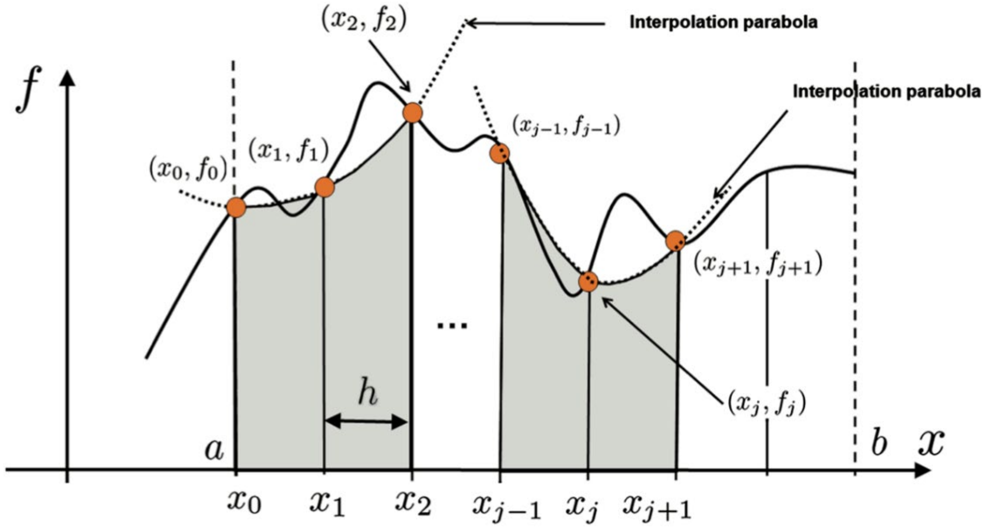
- For *f(x) = sin(x)*, *f* ^{4}*(x) = sin(x)*, and the error is

$$\frac{\max|sin\,(\varepsilon)|}{2880}\,(2-0)^5 \;<\; 0.0111$$

Composite Simpson's Rule

- Allows for any number of uniform intervals.

$$\int_a^b f(x)dx \approx \frac{h}{3}\left[f(x_0) + 4\left(\sum_{i=1, i \text{ odd}}^{n-1} f(x_i)\right) + 2\left(\sum_{i=2, i \text{ even}}^{n-2} f(x_i)\right) + f(x_n)\right].$$



Python Code for f(x) = sin(x)

```
import numpy as np

a = 0
b = 2
n = 11
h = (b - a) / (n - 1)
x = np.linspace(a, b, n)  #returns array of evenly spaced values
f = np.sin(x)

C_simp = (h/3) * (f[0] + 2*sum(f[:n-2:2]) \
                  + 4*sum(f[1:n-1:2]) + f[n-1])

print ("Integral of f(x) = sin(x) on interval [", a, ",", b, "]")
print("Result ~ " , C_simp)
```

Examples

- https://replit.com/@CSREPLIT/CompositeSimpsonRule#main.py

- https://replit.com/@CSREPLIT/CompositeSimpsonRule2#main.py
- https://replit.com/@CSREPLIT/CompositeSimpsonsRule3#main.py

Randomness

- Probability theory and statistics study random phenomena
  - Our focus will be in terms of random samples

- Probability theory – about origin and production of random samples
  - Draw random samples from appropriate probability distributions
    - Simulate raw data for model testing
    - Split raw data into testing and training sets (machine learning)

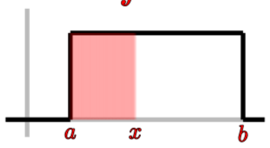
- Statistics – about studying properties of already collected random samples

Common Distributions

Probability Distributions

Continuous

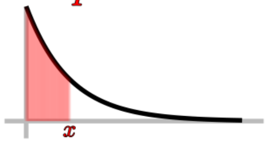
Uniform



$$\mu = \frac{a+b}{2} \qquad \sigma = \sqrt{\frac{(b-a)^2}{12}}$$

$$P(X < x) = \frac{x-a}{b-a}$$

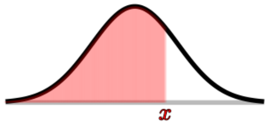
Exponential



$$\mu = \frac{1}{\gamma} \qquad \sigma = \frac{1}{\gamma}$$

$$P(X < x) = 1 - e^{-\gamma x}$$

Normal



$$z = \frac{x-\mu}{\sigma}$$

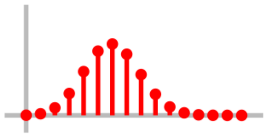
$P(X < x) \Rightarrow$ Use Z-Chart

Key

- γ = *rate parameter*
- z = *z-score*
- p = *probability of success*
- n = *# of trials*
- N = *population size*
- K = *# of success states*

Discrete

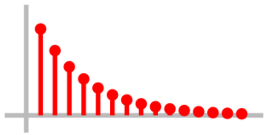
Binomial



$$\mu = n \cdot p \qquad \sigma = \sqrt{n \cdot p \cdot (1-p)}$$

$$P(X = x) = \binom{n}{x} p^x (1-p)^{n-x}$$

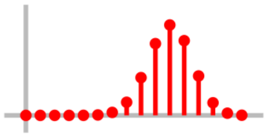
Geometric



$$\mu = \frac{1}{p} \qquad \sigma = \frac{\sqrt{1-p}}{p}$$

$$P(X = x) = (1-p)^{x-1} p$$

Hypergeometric



$$\mu = n\frac{K}{N} \qquad \sigma = \sqrt{n\frac{K(N-K)(N-n)}{N^2(N-1)}}$$

$$P(X = x) = \frac{\binom{K}{x}\binom{N-K}{n-x}}{\binom{N}{n}}$$

Sample Draws

- We can draw samples from distributions in Python
  - Can also calculate statistical measures (mean, standard deviation, skewness, covariance)
  - Example: Pearson correlation coefficient function shows there is not a linear relationship between population and beer consumption.

- Support comes from modules: statistics, numpy.random, pandas, and scipy.stats

Generating Random Numbers

- `import numpy.random` for all major probability distributions
- For reproducibility purposes, you can start with the same seed

```
import numpy.random as rnd
rnd = np.random.RandomState(2021)    #new way to do this
```

- Functions to generate random numbers: return numpy array of shape or size.

  rnd.uniform(low=0.0, high=1.0, size=None) or rnd.rand(shape)

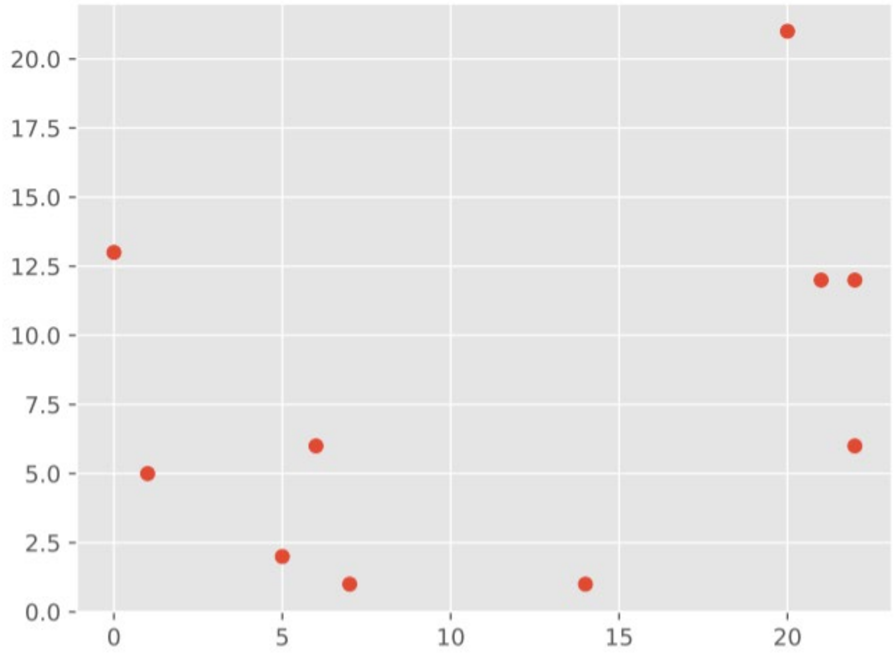
  rnd.randint(low, high=None, size=None)

  rnd.normal(loc=0.0, scale=1.0, size=None) or rnd.randn(shape)

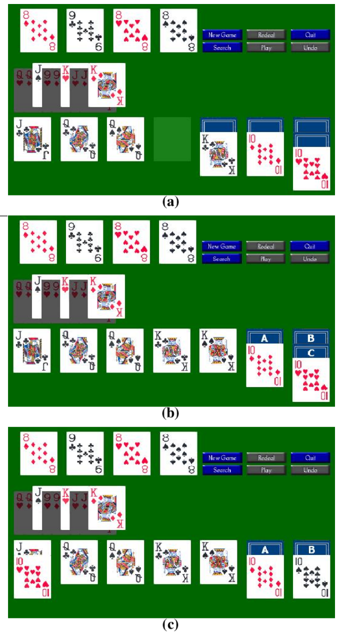
  rnd.binomial(n, p, size=None) #for predictive sets in machine learning

Example:

```
from random import randint as rnd
import numpy as np
import matplotlib, matplotlib.pyplot as plt

#To reproduce results, uncomment statement below
#rnd = np.random.RandomState(2021)

#select good looking plot sorted
matplotlib.style.use("ggplot")

xvalue = np.empty((0, 0))
yvalue = np.empty((0, 0))
n = 25
for x in range (0,10):
  xvalue = np.append(xvalue, rnd.randint(0, n-1))
  yvalue = np.append(yvalue, rnd.randint(0, n-1))

plt.scatter(xvalue, yvalue)
plt.savefig("RandomValues.pdf")
```



Monte Carlo Methods using Random Data Points



(a) (b) (c)

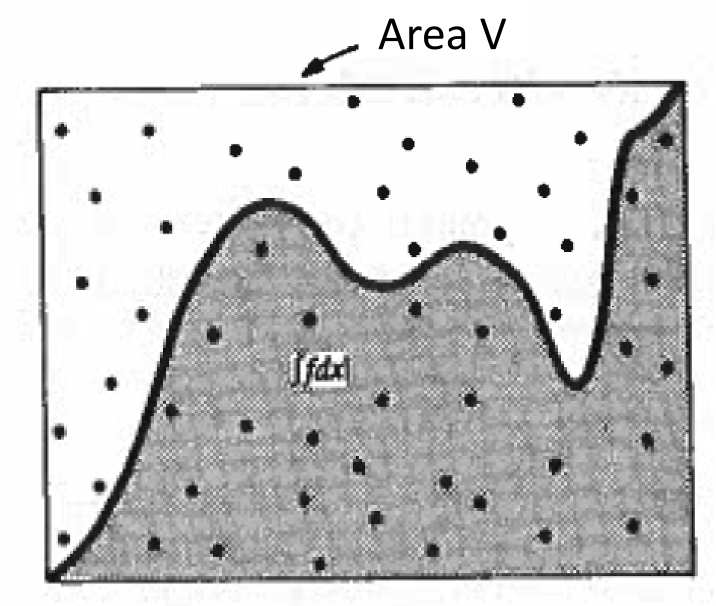
- Monte Carlo simulations
  - Originated with the virtual simulations of atomic physics
  - And from predicting outcomes of the card game Solitaire
  - Hence code name “Monte Carlo” by the military
- Essential idea
  - Assess a quantity by randomly sampling from a population of possibilities
  - If sample represents larger population, we gain confidence in predicted outcome
- Developed side-by-side with computing, since computers can simulate games quicker than a human can play them

Monte Carlo (MC) Integration

- Suppose you want to integrate a function f over a region W, maybe one that is not easy to sample randomly because it is a complicated shape.
- Find a less complicated region V, which includes the region W, which can easily be sampled. Make V enclose W as close as possible.
- Define *g* to be equal to *f* for points in W and equal to 0 for points outside of W (but still inside V).
- Pick N points, uniformly randomly $x_0, x_1, \ldots\ x_{N-1}$ in region V.

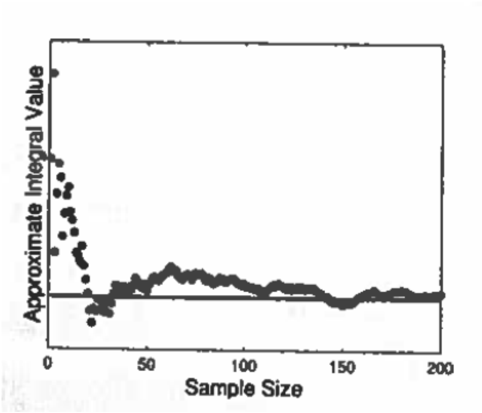
Visualize this

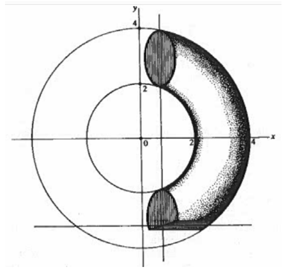
- Random points are chosen within area V. The integral of the function *f* is estimated as the area of V multiplied by the fraction of random points that fall below the curve *f*.
- How do know which points fall below the curve?
  - If the y-value is less than f(x).



Comparison of Simpson's Rule to Monte Carlo Integration

- Simpson's Rule – deterministic quadrature estimation
- Monte Carlo integration – stochastic estimation
- To compute $\int_0^{pi/4} \sin(x)\,dx$
  - Simpson's rule can approximate it within guaranteed accuracy of 10^{-8} with 17 function evaluations
  - Need many more for MC Integration *and* not sure what size of sample to use





  - However, in higher dimensions, MC out performs Simpson's Rule

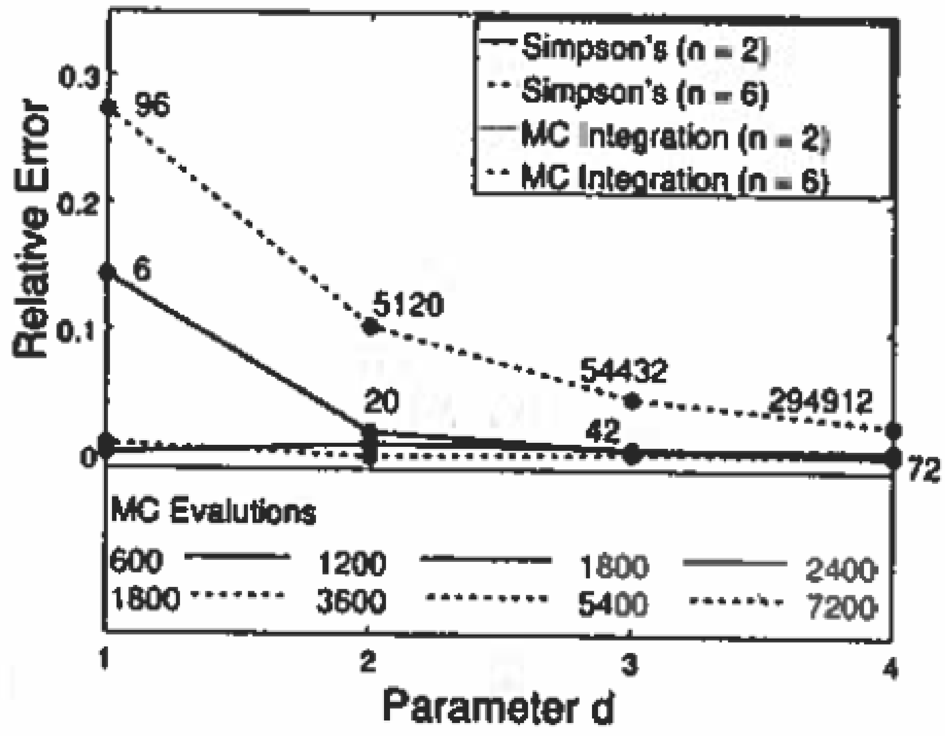


Fig. 6.6 A comparison between Simpson's method and Monte Carlo integration. The integral being evaluated is in (6.8). The Monte Carlo approximations are generated by 30 uniform samples of D of size $10dn$.

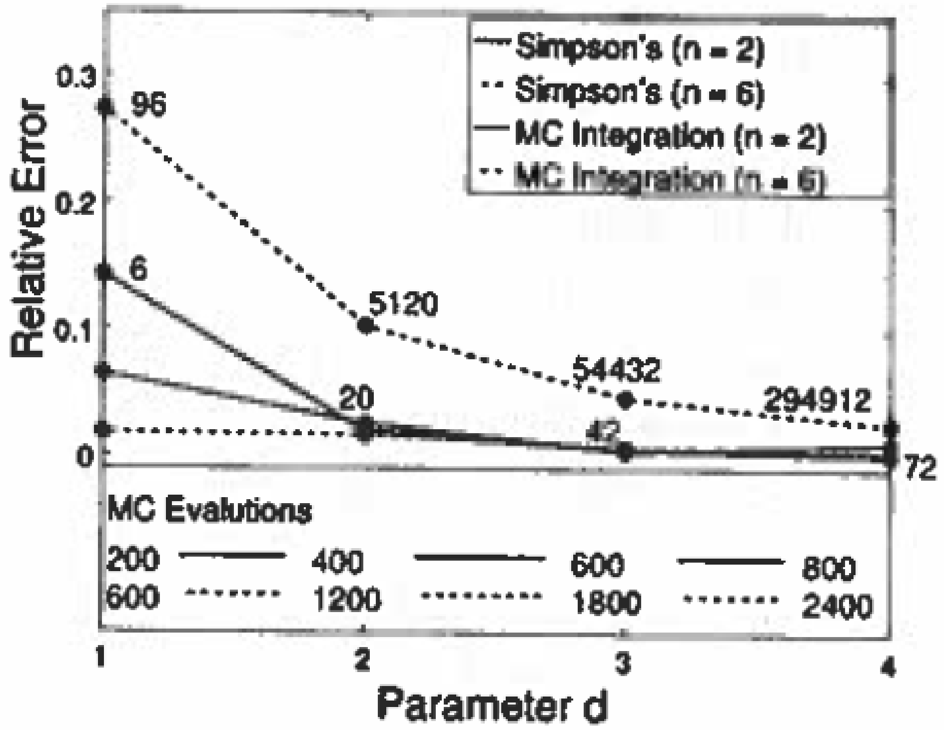
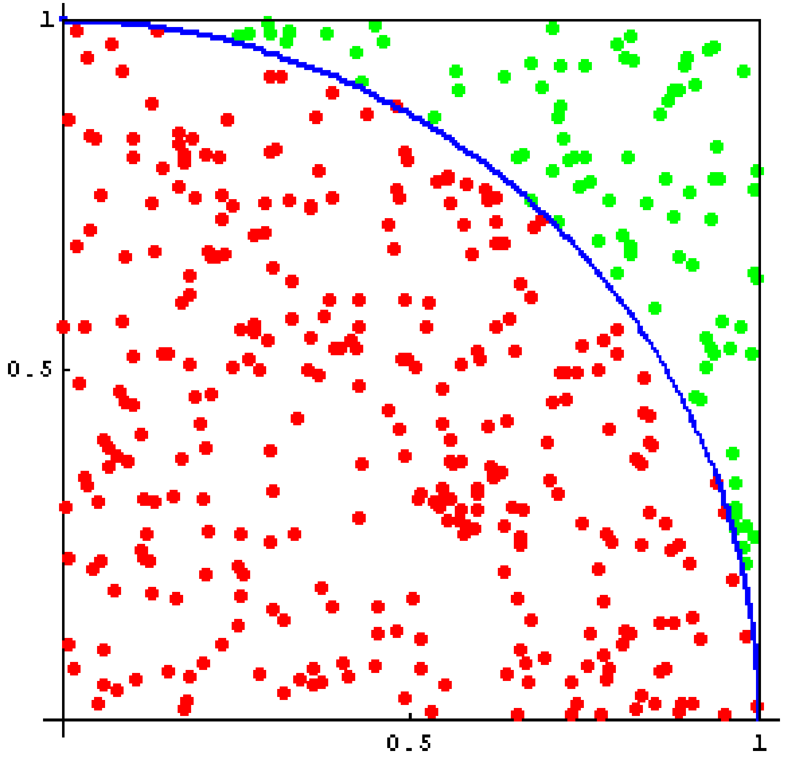


Fig. 6.7 A comparison between Simpson's method and Monte Carlo integration. The integral being evaluated is in (6.8). The Monte Carlo approximations are generated by single uniform samples of D of size $100dn$.

Computing ∏

- Compute ∏ by estimating the area of **¼ of a circle**.
  - Area of Circle = $\prod * r^2$, where r is the radius
  - $\prod = area/r^2$
  - Circle is defined as $x^2+y^2 = r^2$
  - Unit Circle is $x^2+y^2 = 1$
  - So if r < 1, point is inside



Python program to calculate PI

https://replit.com/@CSREPLIT/ComputePI#main.py

```
from random import random as rand
from math import sqrt

#number of random points to Generate
N = 10000

#initialize count of points inside
count = 0

#generate points and count inside
for i in range (N):
  x= rand()
  y= rand()
  radius = sqrt(x**2 + y**2)
  if radius < 1:
    count += 1

#calculate pi
print ("Esimate of PI is ", 4*count/N)
print ("Pi is ", 3.1415926535, "...")
```

Assignment

- Will involve numerical integration of several functions
  - Using Composite Simpson's Rule
  - Using Monte Carlo Technique

- Functions are on page 112 of the Numerical Integration Handout in D2L.
- Write two programs to do two of the functions. Choose any two except for (e) sin(x), since I did that one.
  - Create a table similar to Table 4.1 where you show the exact value for function integral and the results from the two techniques we discussed.
  - Create a plot of the curve and the MC plots. (See my Composite #3 example or ComputePI example for plotting a scatter plot and a curve)