

Random Numbers and Random Processes

Computational Physics

Random Numbers and Random Processes

Outline

- Random Systems
- Random Number Generation
- Monte Carlo Integration

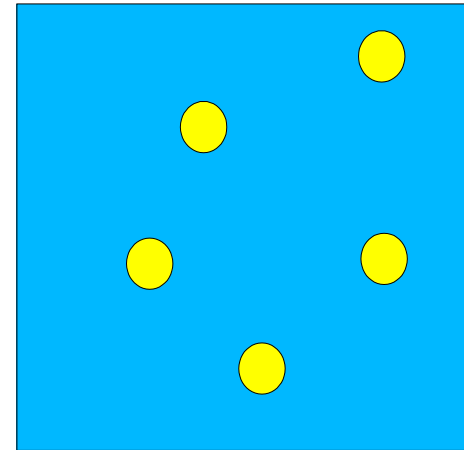
Random Systems

- 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



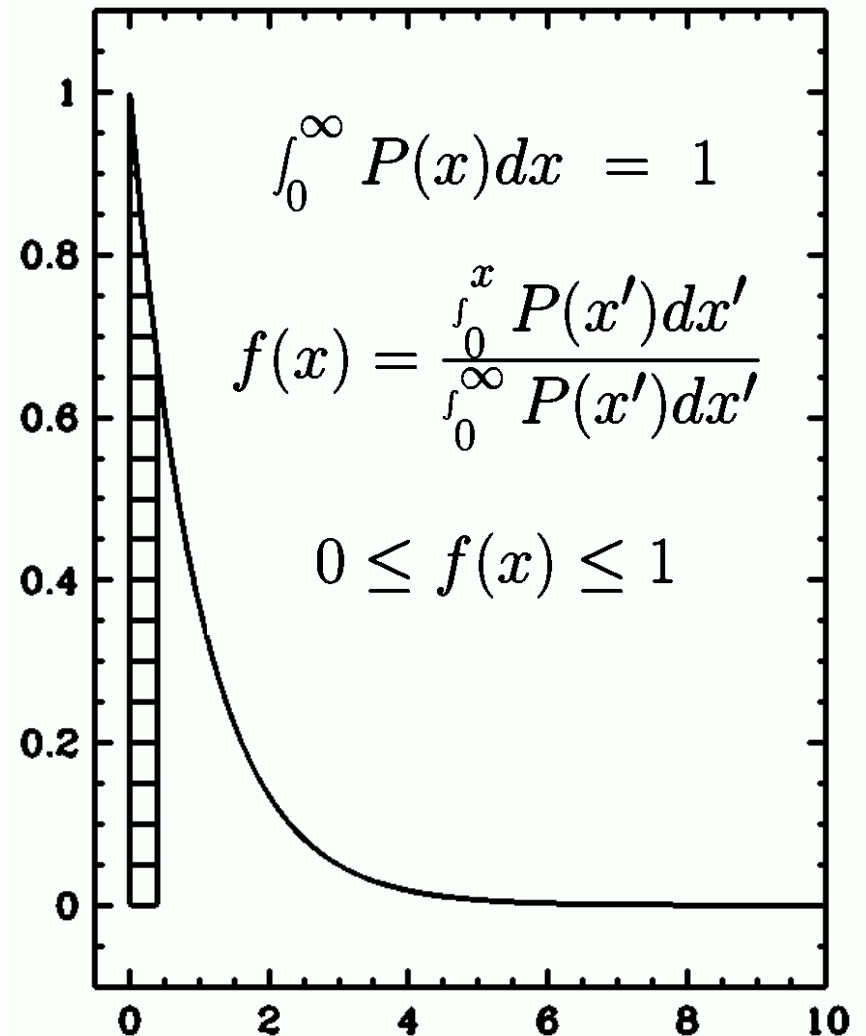
Generation of Random Numbers

- Most computing systems and computer languages have a means to generate random numbers between 0 and 1.
- Sequence generated from recursive relationship: $x_{n+1} = (a x_n + b) \text{ mod } m$
 - need a "seed" to start the process
 - same sequence generated by each seed
"pseudorandom"
 - in real systems, sequences may repeat eventually. Caveat Emptor!

Creating New Distributions

Transformation Method

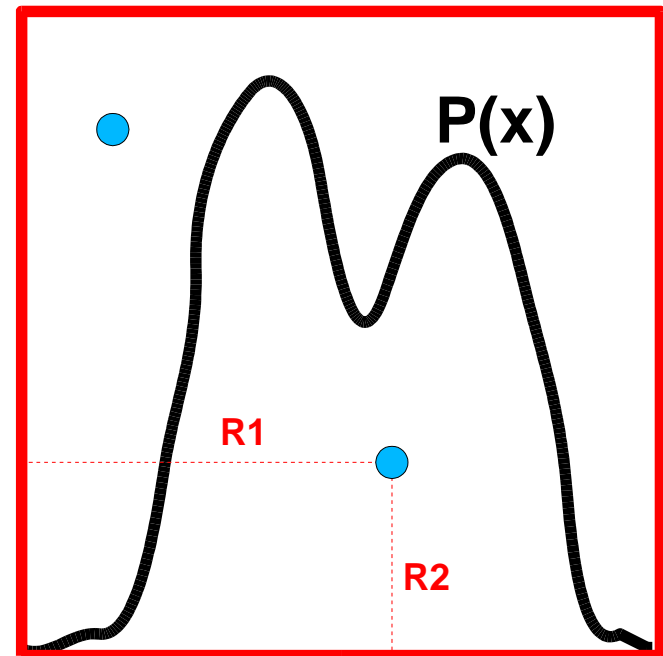
- Equal areas have equal probabilities:
| $P(x) dx$ | = | $P(y) dy$ |
- Consider:
 - $P(y) = e^{-y}$
 - $f = 1 - e^{-y}$
- Select f from uniform distribution, then:
 - $y = -\ln(1-f)$



Creating New Distributions

Rejection Method

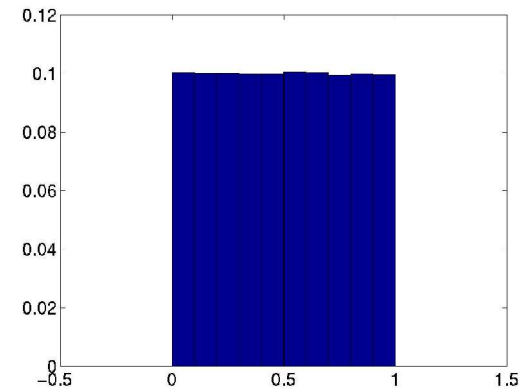
- Draw box around function
- Generate UD random number in x : $R1$
- Generate second UD random number in y : $R2$
- If $P(R1) > R2$: keep the point.



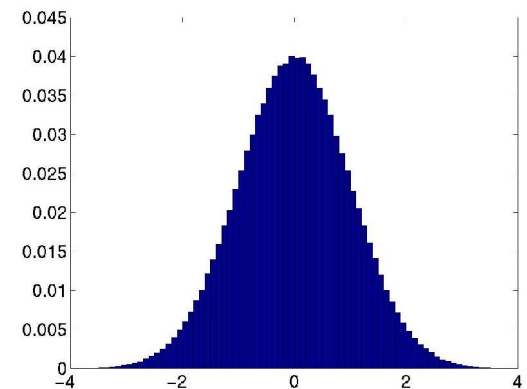
MATLAB

Generation of Random Numbers

- Random number functions
 - **rand** generates a sequence of *uniformly* distributed random numbers.
 - **randn** generates a sequence of *normally* distributed random numbers.



rand



randn

MATLAB

Initializing the Seed

- Initialize random number generator with "seed".
- Same seed will generate same set of random numbers.
- Can initialize with number derived from time to avoid same sequence.

```
% set state with seed=12345  
rand('state', 12345);
```

```
% print random number  
rand
```

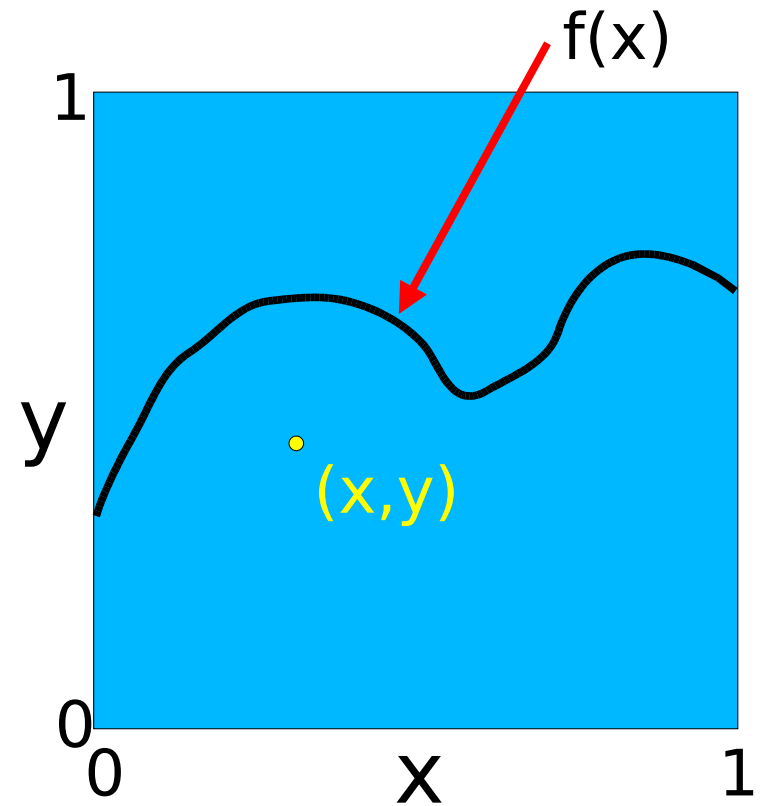
```
% reset with same seed  
rand('state', 12345);
```

```
% next call to rand gives  
% same value as above  
rand
```

```
% derive seed from time  
rand('state',sum(100*clock));
```

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: $f(x) = x^2$

```
n = 100000;  
  
% generate n random numbers x,y  
x = rand(n, 1);  
y = rand(n, 1);  
  
sum = 0.0;  
for i=1:n  
  
    % compare y to f(x)  
    if( y(i) < x(i)*x(i) )  
        sum = sum + 1.0;  
    end  
  
end  
  
% print the result normalized by  
% number of points generated  
  
sum/n
```

creates 1D array with n uniformly distributed random numbers

initialize "sum" to 0 to accumulate number of points below curve

compare each point to function and add to sum of points below curve

number of points below curve out of total number is fraction of area