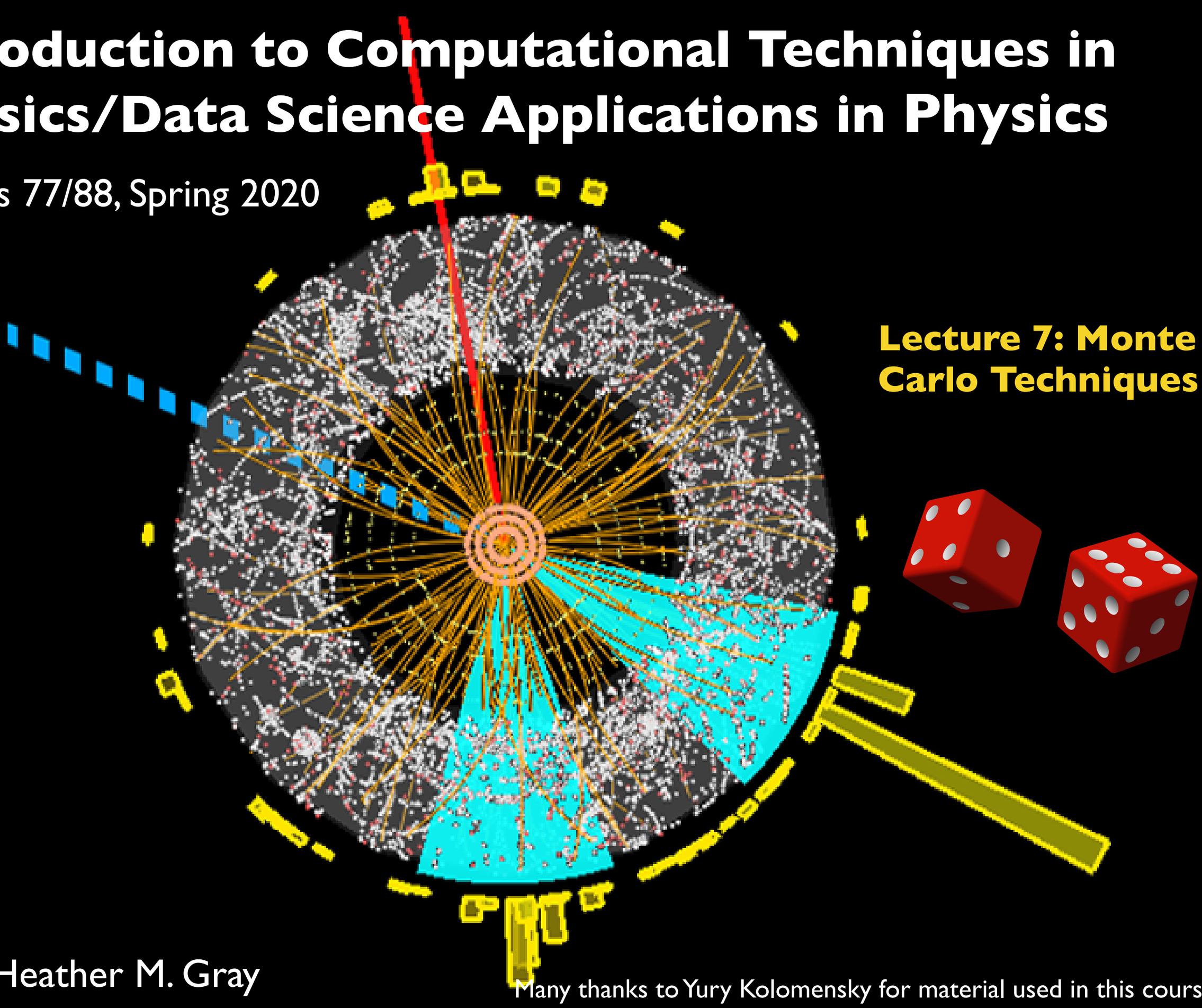
Introduction to Computational Techniques in Physics/Data Science Applications in Physics

Physics 77/88, Spring 2020



Lecture 7: Monte Carlo Techniques

Prof. Heather M. Gray

Many thanks to Yury Kolomensky for material used in this course

Outline

- Introduction
- Random number generators
- Random number distributions
- Applications: integration

Monte Carlo Simulation

- technique
  - Applicable when an to a problem is too or
  - Applicable when the problem has
    - e.g. an of a
  - Involves repeated of the to obtain approximate (usually) result
- Applications
  - Most
  - Engineering, finance:
  -
  - And of course,

Examples

- Let's start with some examples in the python notebook
  -
  -
  -

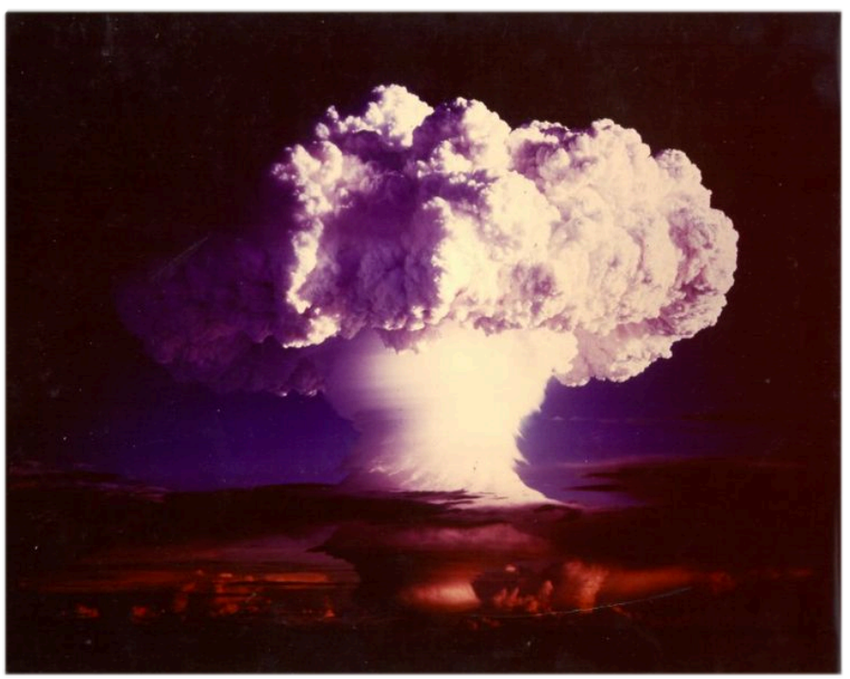
From: https://en.wikipedia.org/wiki/Coin_flipping#/media/File:Coin_Toss_(3635981474).jpg

From: https://en.wikipedia.org/wiki/Dice

From: https://en.wikipedia.org/wiki/File:AcetoFive.JPG

More examples

- Professional packages
  - Particle physics:
    - JETSET
    - Pythia
    - GEANT
  - Many ray-tracing/optics packages
  - First (documented) use: Manhattan project
- Possible Final Projects
  - Tournament simulator (e.g. March Madness, NHL playoffs, etc)
  - Program a game ( )
    - Incorporate some
      - e.g. betting odds, strategy



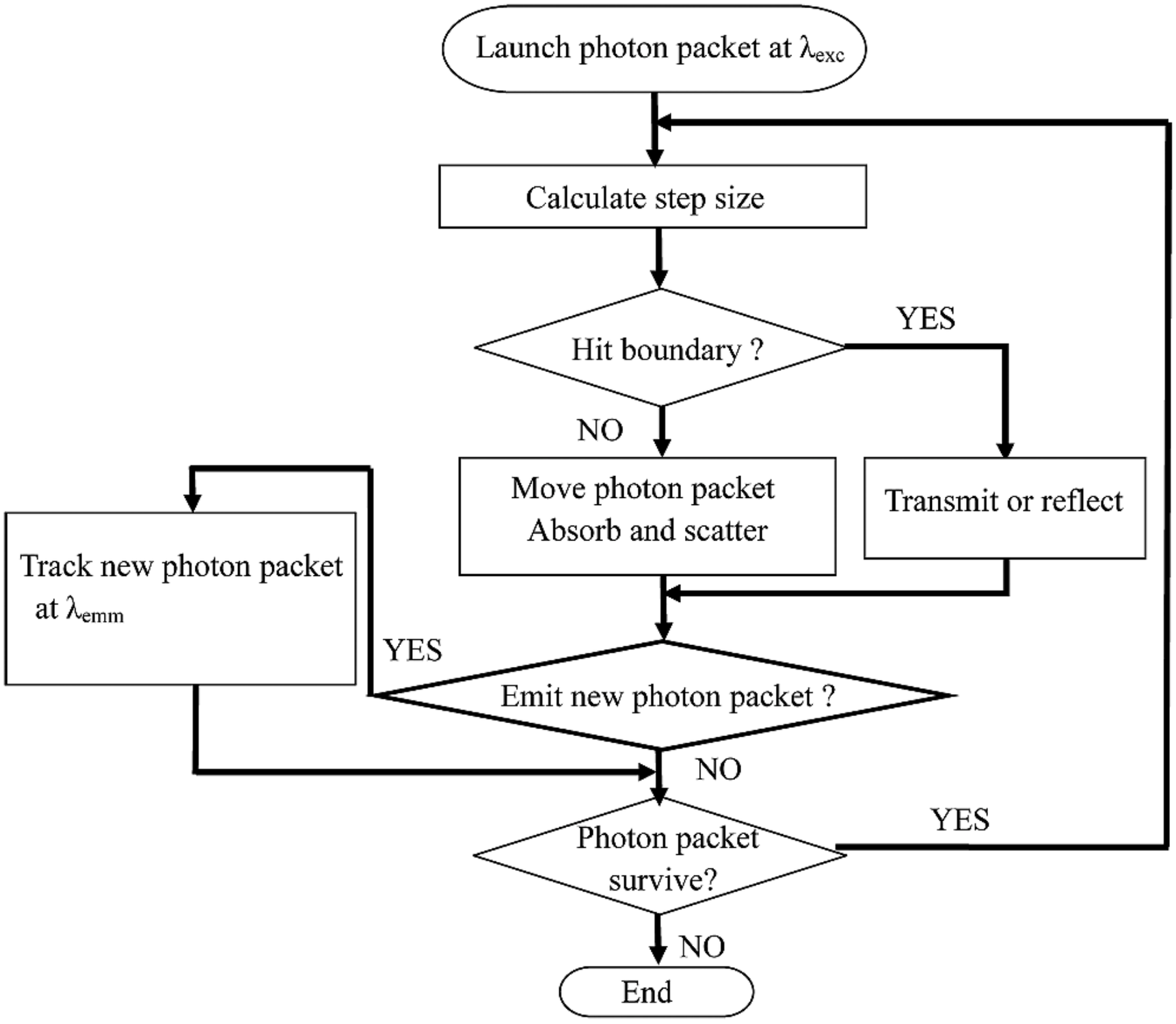
The Monte Carlo Technique

- What Monte Carlo simulation can not do:
  - Predict of a (measurement, process, …) or of
- What Monte Carlo simulation can do:
  - Compute for a outcome
  - Determine probability , correlations between , … that resemble those observed in the
- Precision limited by:
  - sampling ( )
  - Modeling of the
    - e.g. in describing (systematics)

Overview of the technique

- Split the process into , such that probability density functions for each could be
- For each step:
  - Identify and their
  - Generate
  - Evaluate outcome of the step
  - Repeat for the
  - At the end,
  - Repeat , build up a of

Example



C. Zhu and Q. Liu, "Review of Monte Carlo modeling of light transport in tissues"
J. Biomed. Opt. 18(5), 050902 (May 10, 2013). doi:10.1117/1.JBO.18.5.050902

Random Number Generators

- All MC simulations rely on a
  - Computer , so not
    - Although generators are possible (e.g. Johnson's noise generator — a cute 111A project)
- The most basic distribution is
  - distributions can be by a transform of a distribution (see below)
- Requirements:
  - Generate with a very long
  - Small within the
  - "Seeded" to produce
  -

Simplest: Linear Congruent Generator

- $x_{n+1} =$
- fixed by
- depends on
- of at most
  - So make c , a & b
  - E.g. RANDU: a= b= , c=
    - Period of numbers
    - Produces random numbers within the
- Modern generator: Mersenne Twister
  - Implemented in Python, ROOT (TRandom3)
    - Period of

- Back to the jupyter notebook

Generating Arbitrary Distributions

- Most frequently, you want a distributed according to some (predefined)
- Since most have a random number generator, derive from a random number :
  -
- Two basic techniques
  - method
  - von Neumann's method

Inverse Transform Method

- Theorem:
  - If is chosen with , and is a of , then is a random variable distributed between [ ]
  - $F(a) =$
- Therefore can implement with :
  - Generate u:
    - $u \in$ $, f(u) =$
  - Invert :
    - $x =$

Example

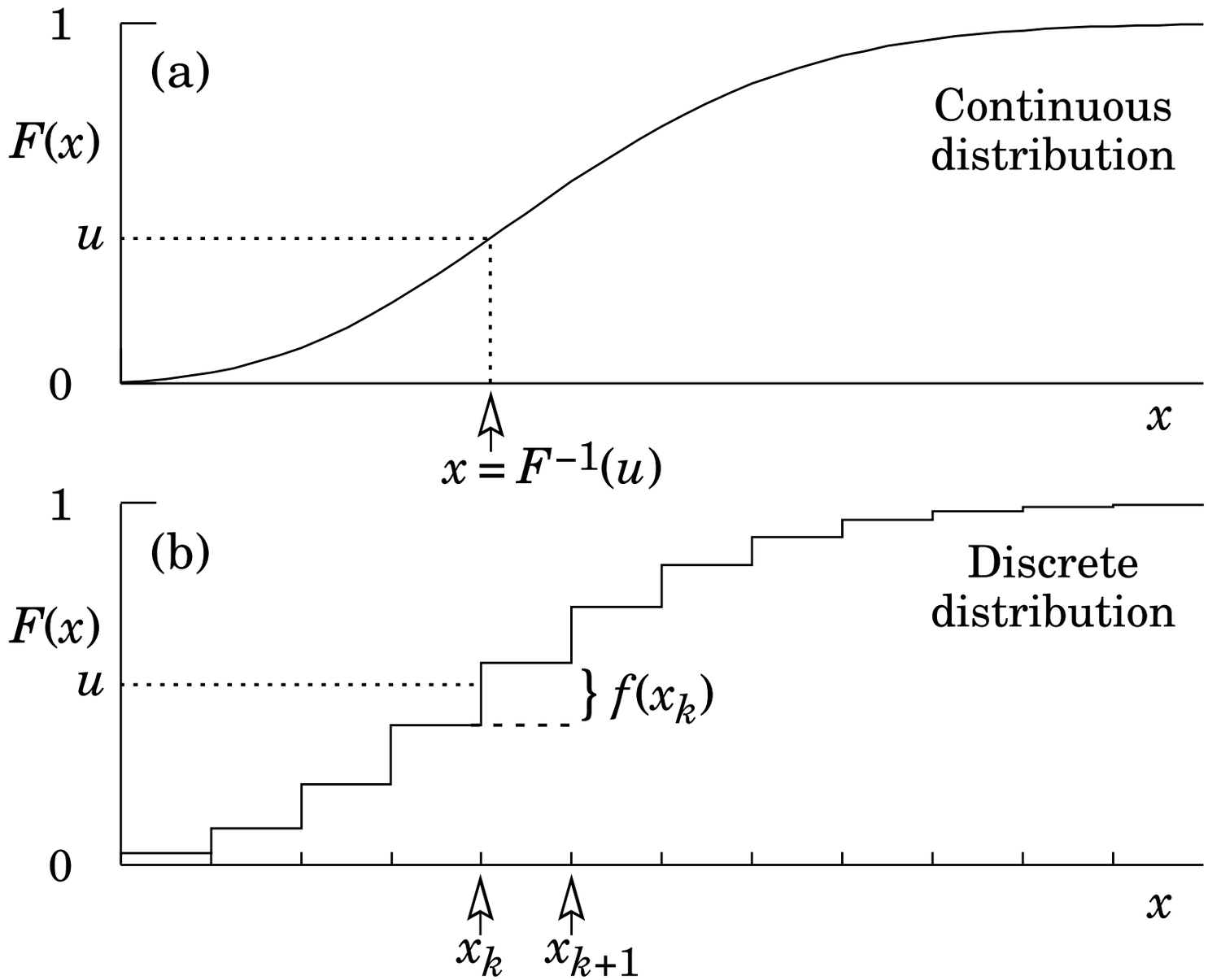
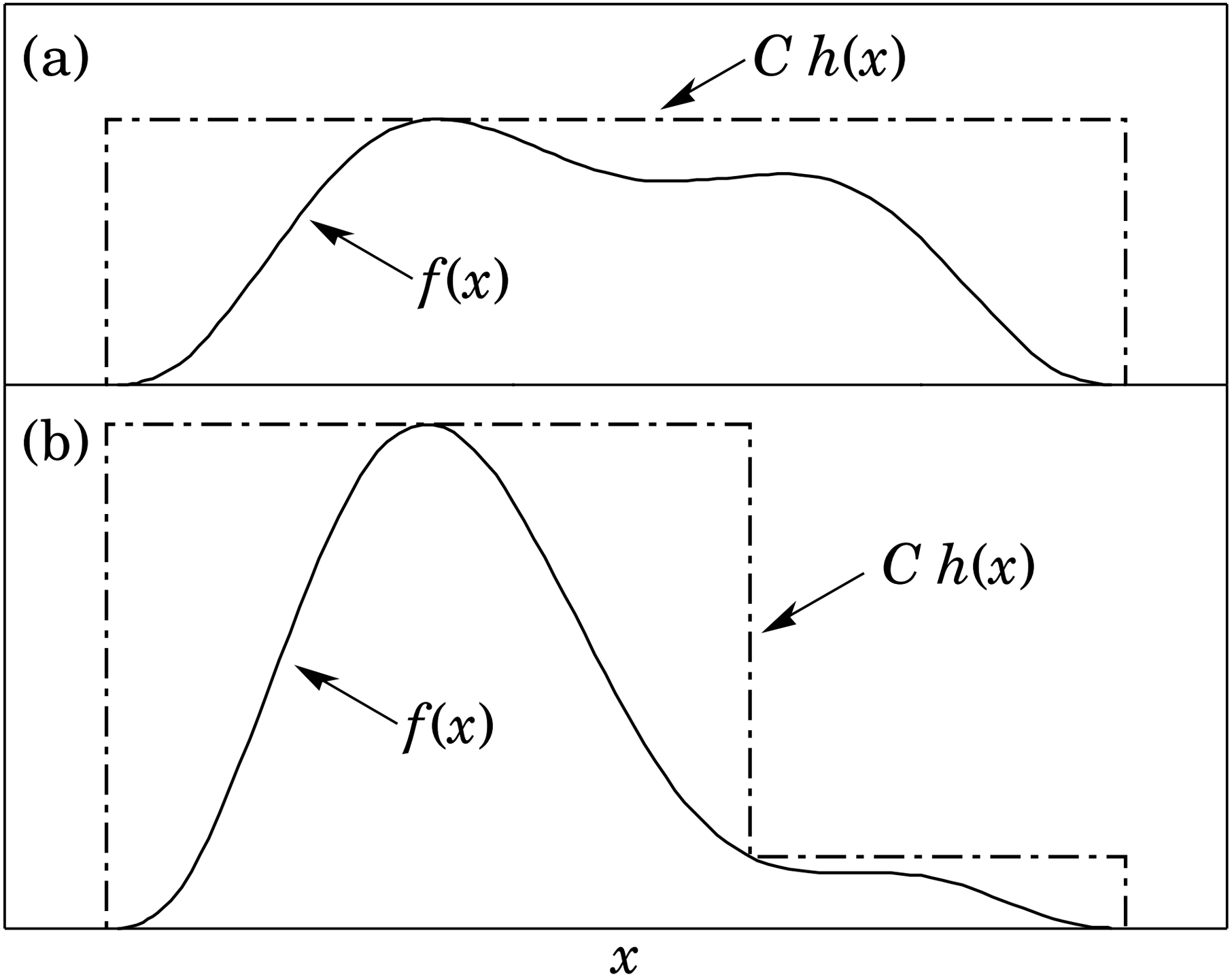


Figure 39.1: Use of a random number u chosen from a uniform distribution (0,1) to find a random number x from a distribution with cumulative distribution function $F(x)$.

http://pdg.lbl.gov

Accept-Reject Method

- If is an , or difficult to compute function, then finding may be
- Let's suppose however that is and , such that for any we can find an easily generated and a constant
  -
- Accept-reject method:
  - Generate according to , and
  - If , accept ;
  - Otherwise and


(a)
$C\,h(x)$
$f(x)$
(b)
$C\,h(x)$
$f(x)$
x

Algorithms

- Exponential decay
  - $f(t) =$
  - Generate: $u \in$ $,f(u) =$
  - Define
    - $\alpha =$
    - $\beta =$
      - Then $t =$
- Gaussian distribution
  - If and are on , then
    - $z_1 =$
    - $z_2 =$
- are and with mean and $\sigma =$

Algorithms

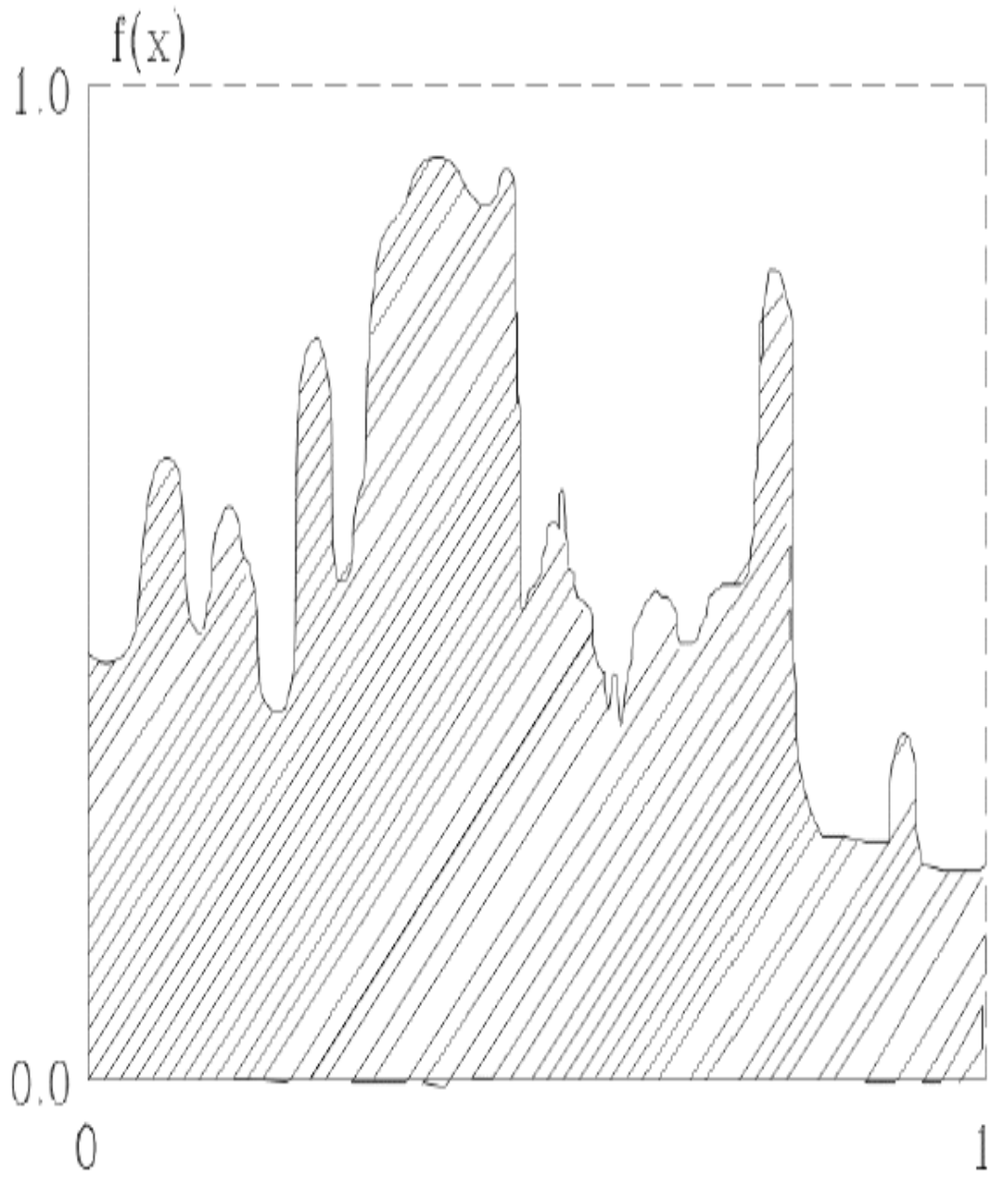
- Poisson Distribution
  - Iterate until a choice
    - Begin with $k =$ and set $A =$
    - Generate
    - Replace A with
    - If A < (where is the parameter)
      - accept $n_k =$ and
- Else increment by and generate a new and repeat
  - Always start with the value of left from the

Applications: MC Integration

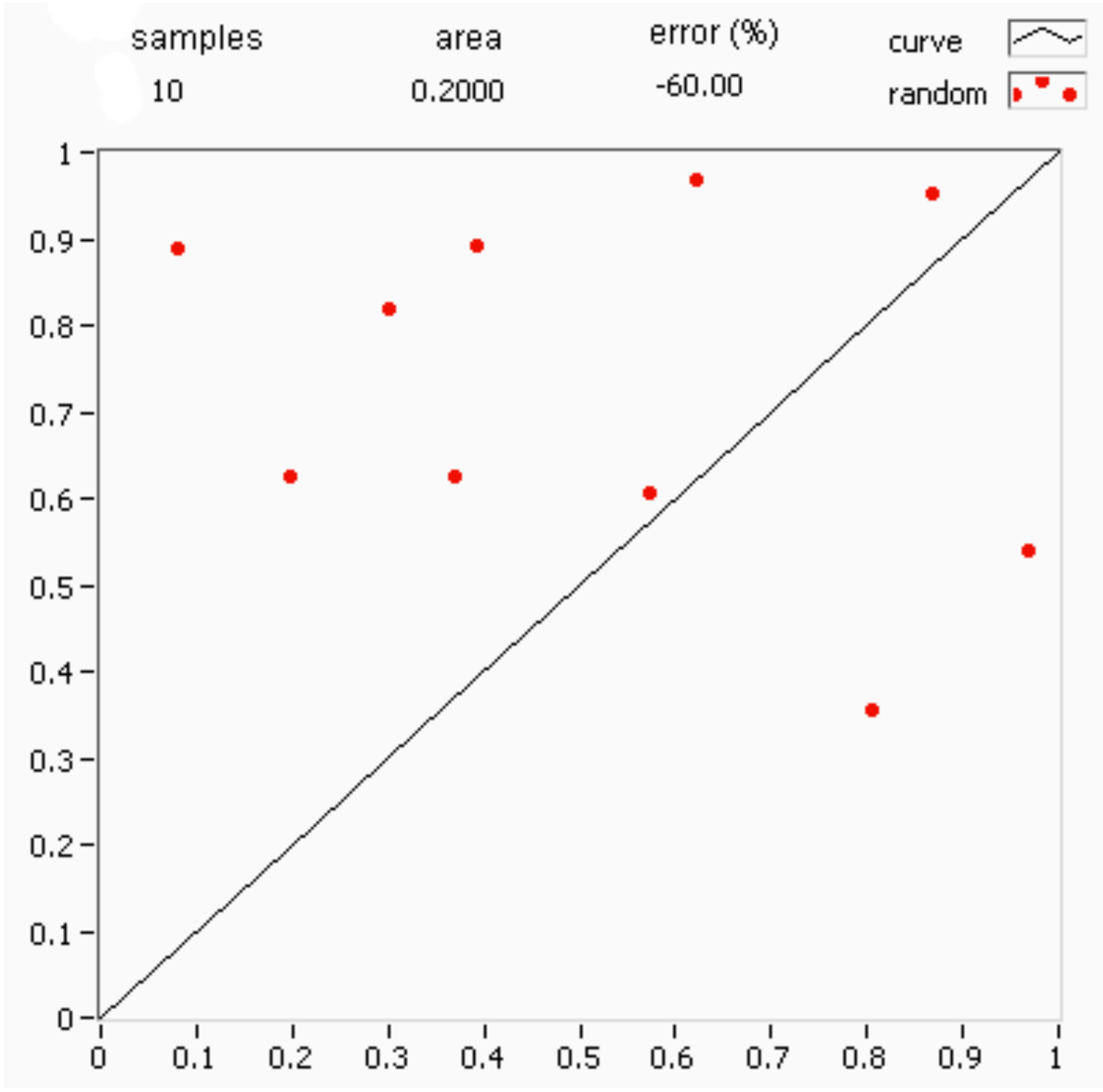
- Suppose we want to compute a
  - $Z = \int$
- Generate a (e.g. distributed in space)
- If is within , increment sum
- Repeat times
- Uncertainty on typically scales as

1-dimensional MC

- Suppose we want to find the under the for $y =$ for a given
- Now let's assume we are at the picture of
- Each will land on a point (,), where and
- If the value of , then we count it as a
- Repeat the for a large number of
- The under the is the /



Example



https://en.wikipedia.org/wiki/Monte_Carlo_method

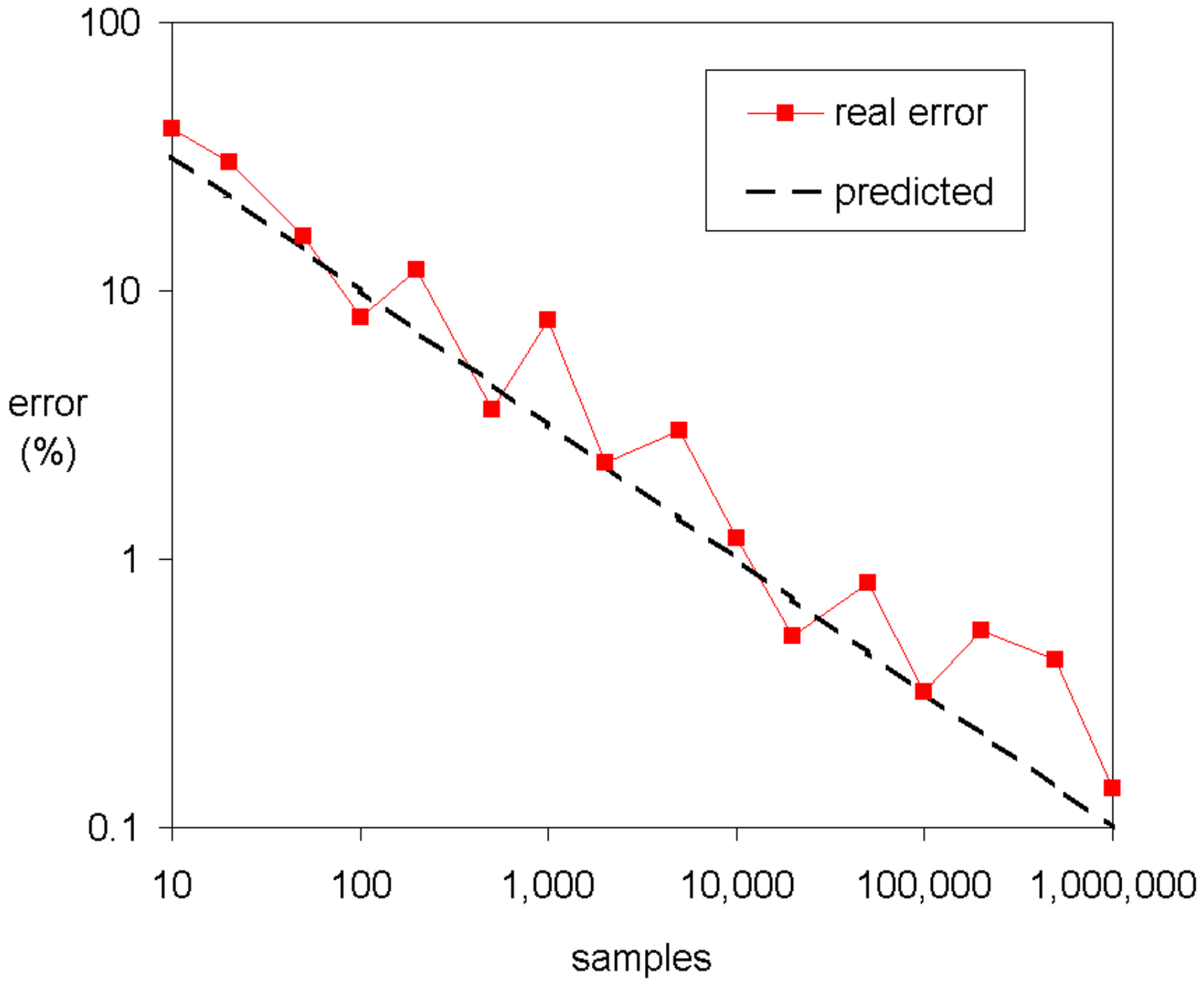
Error Estimate for MC Integrals

- Uncertainties for follow the distribution
  - Let the under the curve be , and the populated by is
  - Let be the number of points , and the number of points .
  - Then the estimator of is:
    - $\hat{S} =$
- Define the of as
  - $\epsilon =$

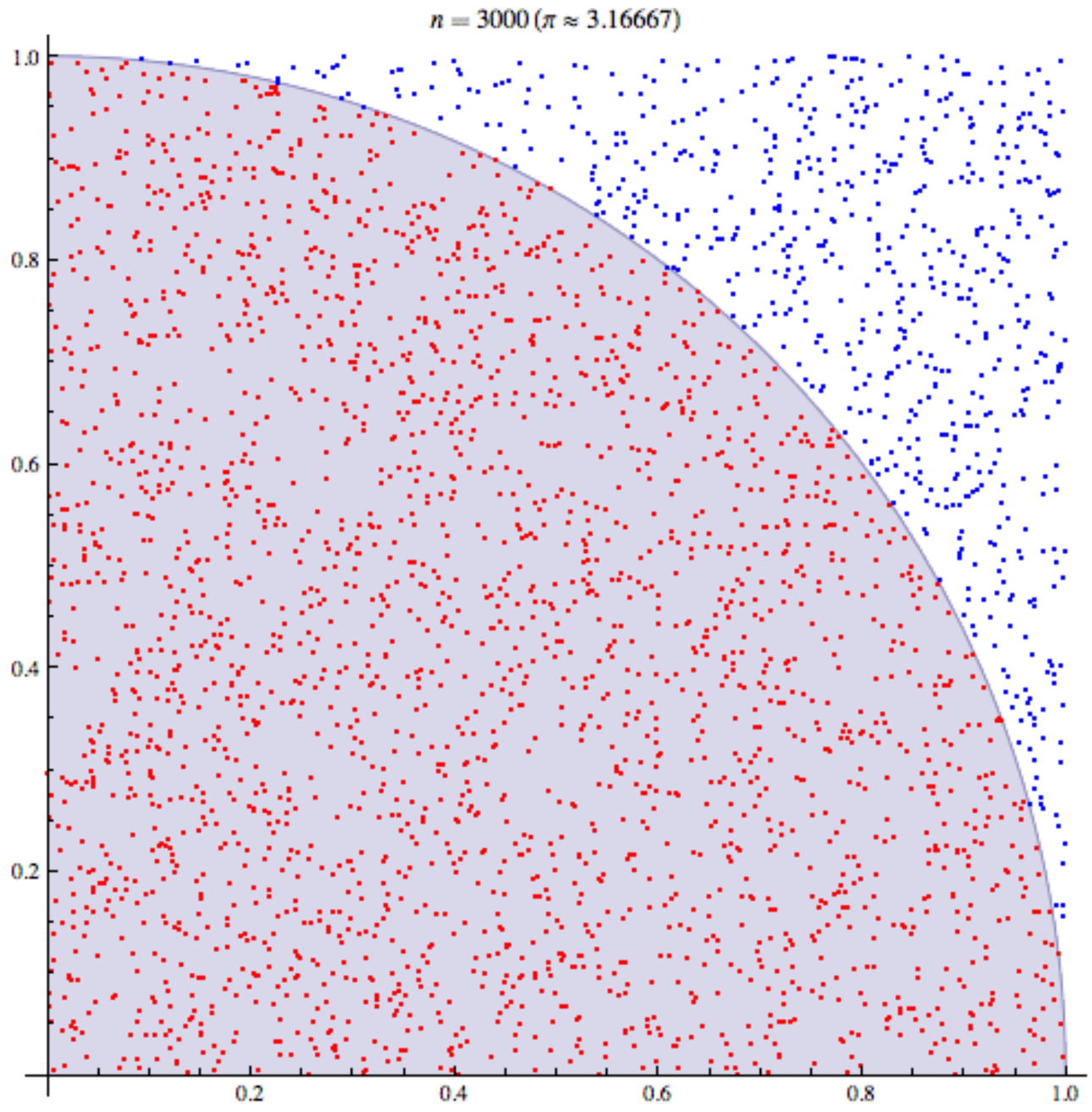
Error Estimate for MC Integrals

- is a variable, which follows a distribution (“choose out of ”).
- Estimator of is therefore also a
- Its is given by
  - $\sigma(\hat{\epsilon}) =$
- You can show that in the limit
  - $\sigma(\hat{\epsilon}) \approx$
  - $\sigma(\hat{S}) =$

Error Estimate



Example: Compute π by MC



https://en.wikipedia.org/wiki/Monte_Carlo_method