Random Numbers and Generators

ECE 3340
Electrical and Computer Engineering
University of Houston
Dr. David Mayerich

Probability distributions

- Describes the likelihood of a variable to take on a value
- The probability that a variable x will take on a value between $[a, b]$ is defined by:

$$\Pr[a \leq x \leq b] = \int_a^b P(x)$$

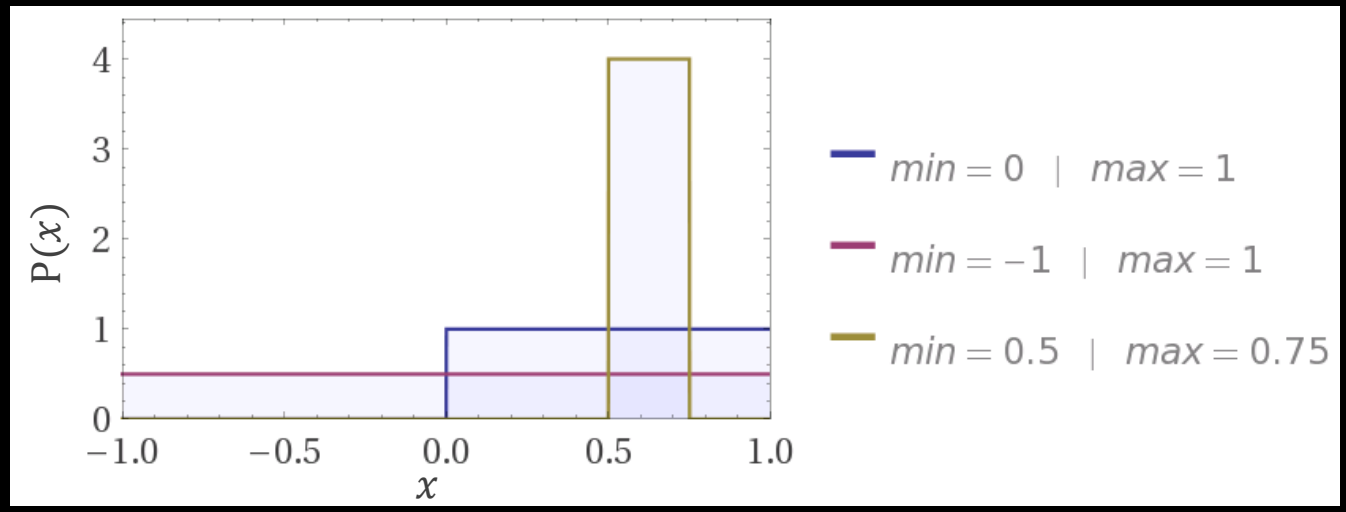
- Most common:
  - Normal (Gaussian)
  - Uniform

- Cumulative Distribution Function $C(x)$ – describes the likelihood that a variable will take on a value $< x$

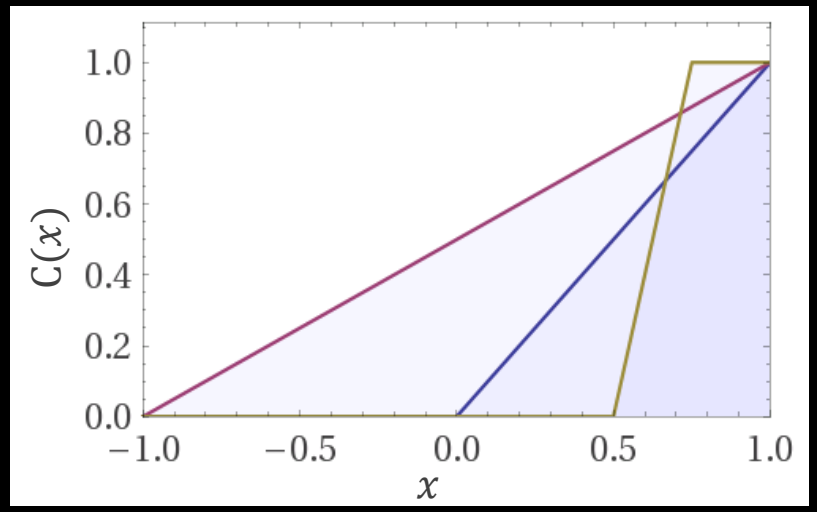
$$C(x) = \int P(x)$$

$$\Pr[a \leq x \leq b] = C(b) - C(a)$$

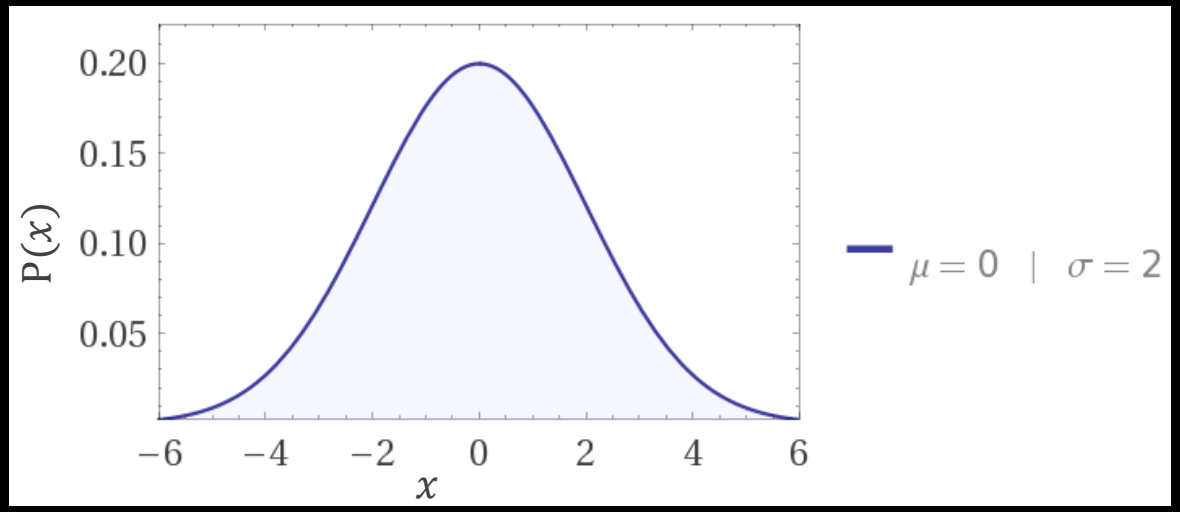
Uniform Distribution



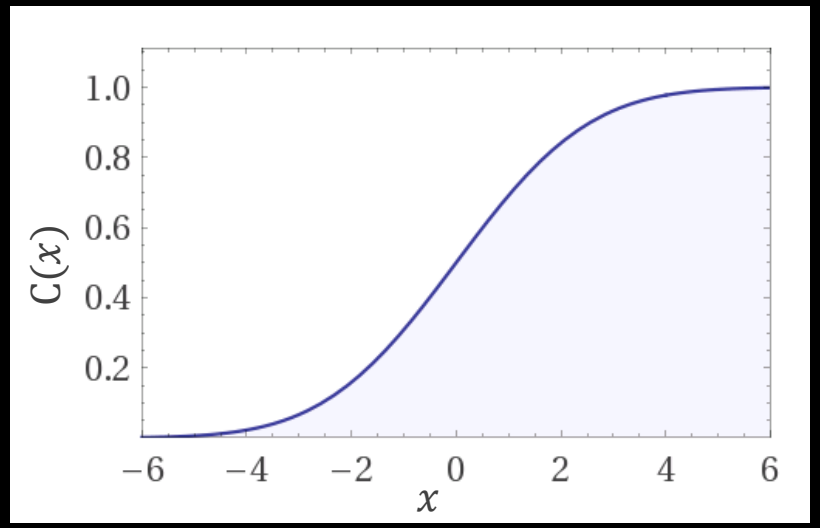
- Corresponding Cumulative Distribution:



Gaussian Distribution



- Corresponding Cumulative Distribution:



Common distributions

- Uniform

$$P(x) = \begin{cases} 0, & x < a \\ \dfrac{1}{b-a}, & a \leq x \leq b \\ 0, & x > b \end{cases}$$

where $[a, b]$ is the desired interval for the distribution

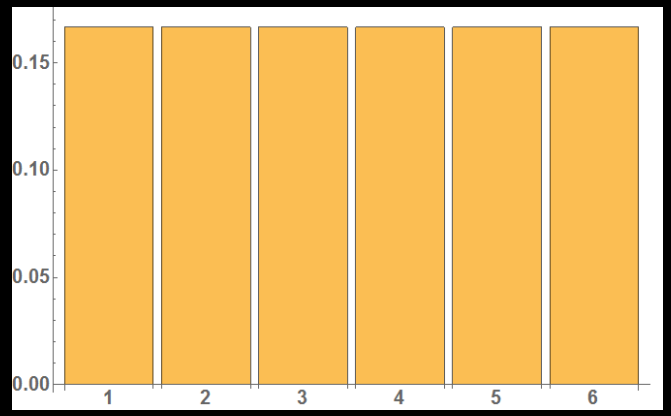
- Normal

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

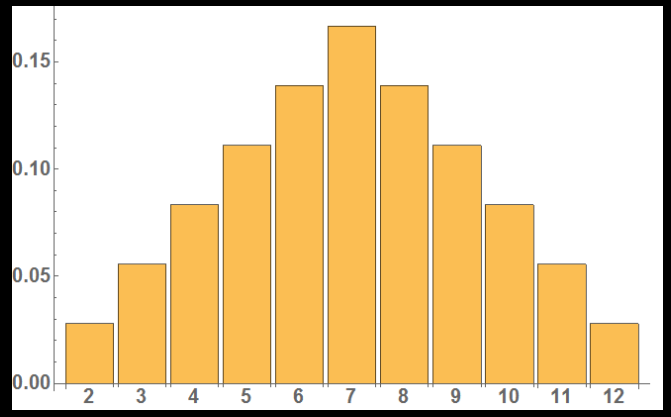
where μ is the mean and σ is the standard deviation

Dice

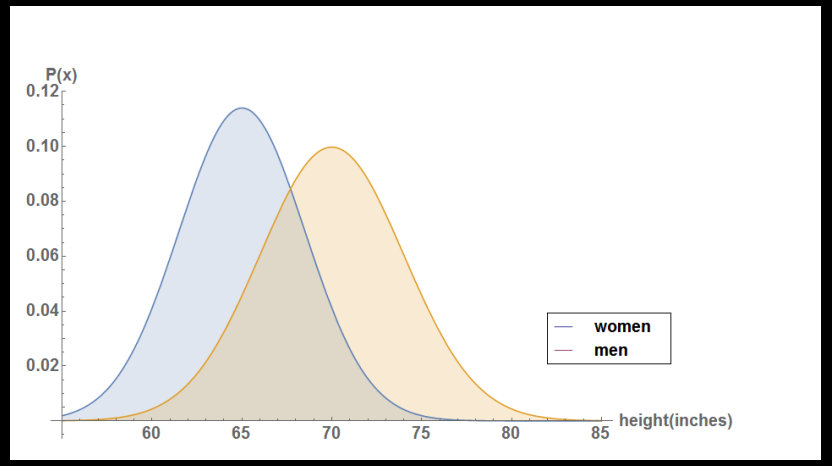
- PDF for a perfect 6-sided die



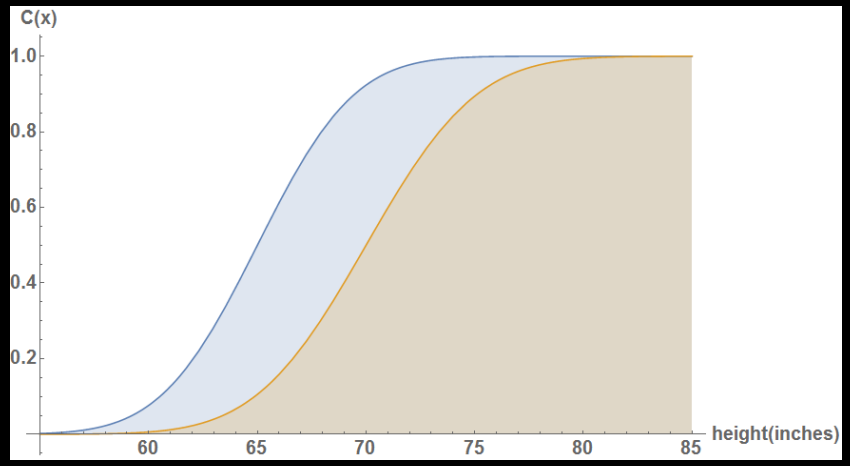
- PDF for rolling twice



Human Height



PDF



CDF

- My wife and I are both 5’7” (67”)
  - How Likely are we to encounter someone of the same sex that is shorter than us?
- Me: 22.6%
- Her: 71.6%

Random Numbers

- We often want to generate values for x that obey some distribution $P(x)$
- This is generally referred to as *random number generation*
  - The algorithm used is known as a random number *generator*
- Random numbers are used *extensively* in numerical methods and algorithms
  - Simulation – particularly of noisy systems
  - Sorting and searching algorithms (randomized quicksort)
  - Probabilistic algorithms – Monte-Carlo methods
  - High-dimensional integration
  - Computer graphics, video games (procedural animation)
  - Cryptography
  - Machine learning

Linear congruential generator

- Uniform pseudo-random number generator

- Recurrence relation:

$$X_{n+1} = (aX_n + c) \bmod m$$

where

$m > 0$ is the modulus

$0 < a < m$ is the multiplier

$0 \leq c < m$ is the increment

X_0 is the seed

LCG (Examples)

- Demonstrate a linear congruential generator with the following characteristics. How many numbers do you get without repeats?

$m = 7$

$a = 3$

$c = 1$

$x_0 = 1$

$$X_{n+1} = (aX_n + c) \bmod m$$

$x_0 = 1$

$x_1 = 4$

$x_2 = 13 \bmod 7 = 6$

$x_3 = 19 \bmod 7 = 5$

$x_4 = 16 \bmod 7 = 2$

$x_5 = 0$

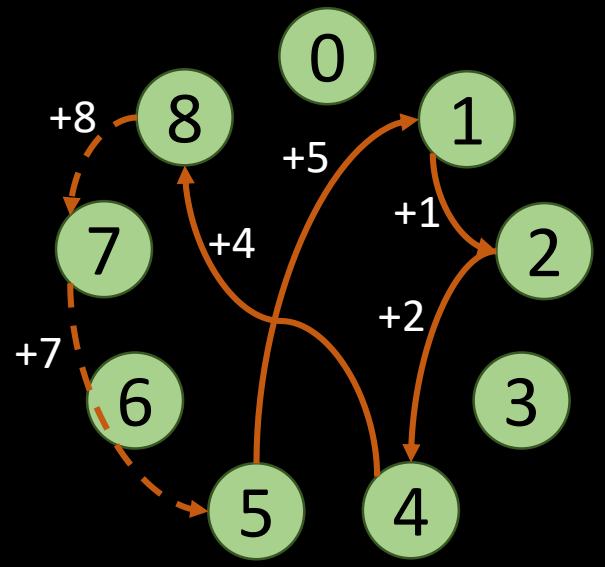
$x_6 = 1$

Visualization of an MCG

- If $c = 0$, the RNG is a multiplicative linear congruential generator (MLCG)
  - c and m must be co-prime
- Assume an MLCG with the following properties:

$$m = 9$$
$$a = 2$$
$$x_0 = 1$$

1) $x_0 = 1$
2) $x_1 = 2$
3) $x_3 = 4$
4) $x_4 = 8$
5) $x_5 = 7$
6) $x_6 = 5$
7) $x_7 = x_0$



Maximizing period for an MLCG

- An MLCG can get a maximum period of $m - 1$ if
  - x_0 is coprime to m
  - a is a primitive root modulo m

- We would like to maximize our ability to seed (select x_0)
- Select a large prime number as the modulus m
  - allow us to use all possible values for x_0
- A Mersenne prime fits nicely into a 32-bit unsigned integer

$$\mathrm{M}_{31} = 2^{31} - 1$$

- Park and Miller "minimal standard generator" uses a primitive root modulo M_{31} of $a = 7^5 = 16807$.
- This represents the current C++ standard.

Maximizing period for an LCG

- An LCG can achieve a period of at most m
- Hull-Dobell Theorem
  - c and m are coprime
  - $a - 1$ is divisible by all prime factors of m
  - $a - 1$ is a multiple of 4 if m is a multiple of 4

- No dependence on x_0
- Use $m = 2^n$ so that mod can be computed using bitwise operations:

$$x \bmod m = x \text{ AND } (\text{NOT } m)$$

or by only keeping the n least significant bits:

$$18 \bmod 8 = 10010_2 \bmod 1000_2 = 010_2 = 2$$
$$27 \bmod 8 = 11011_2 \bmod 1000_2 = 011_2 = 3$$
$$27 \bmod 16 = 11011_2 \bmod 10000_2 = 1011_2 = 11$$

  - This can be done by casting an unsigned integer in C\C++

Problems with LCGs and MLCGs

- They are deterministic
  - Knowing the algorithm allows prediction of the sequence
- Periodic sequences are generated
- When $m = 2^{\mathrm{n}}$, lower-order bits have lower period:
  - least significant bits have periods of at most 2, 4, 8, etc.

Problems with LCGs and MLCGs

- In high-dimensional spaces, the points exist in planes
  - In a $(k-1)$ dimensional space, there will be at most $m^{1/k}$ planes

- If a series of 3-dimensional points are produced using a 32-bit LCG, how many hyper-planes do these numbers lie on? How about 10D?

$$\left(2^{32}\right)^{\frac{1}{4}} = \sqrt[4]{2^{32}} \approx 256$$

$$\left(2^{32}\right)^{\frac{1}{11}} \approx 7.51$$

- In general, don't use these for encryption or (particularly high-dimensional) scientific applications

Anecdote from Numerical Recipes

“One of us recalls as a graduate student producing a ‘random’ plot with only 11 planes and being told by his computer center’s programming consultant that he had misused the random number generator ... This set back our graduate education by at least a year!”

-Numerical Recipes, 2007

Problems with LCGs and MLCGs

- Creating various distributions generally requires higher precision than the LCG

- Efficiently generating a uniform distribution on $[0, 1]$ using a 32-bit generator requires at least 32 bits in the mantissa

- IEEE 754-1985 specifies
  - 32-bit float has a 23-bit mantissa
  - 64-bit float (double) has a 52-bit mantissa

- Most common alternative is the *Mersenne twister* algorithm
  - Extremely large period
- Cryptographic algorithms often rely on random input
  - User-provided mouse movements in TrueCrypt

Uniform RNG in the range $[0, 1)$

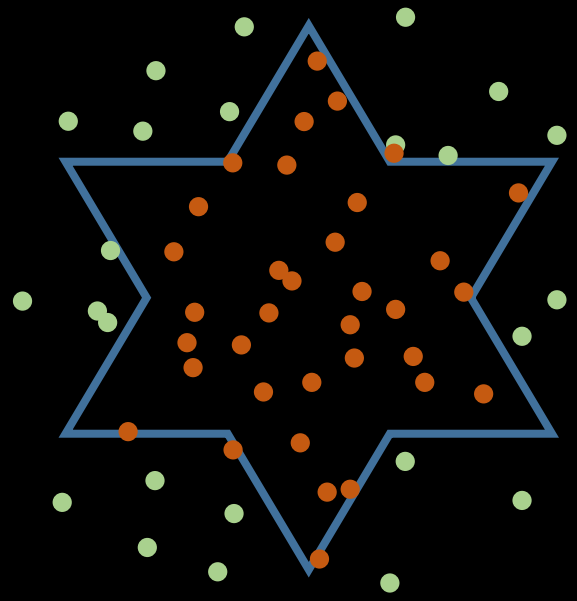
integer array $(l_i)_{0:n}$;
real array $(x_i)_{1:n}$;

$l_0 \leftarrow$ any integer $1 < l_0 < 2^{31} - 1$
for $i = 1$ **to** n
    $l_i \leftarrow (l_{i-1} \cdot 7^5) \bmod (2^{31} - 1)$
    $x_i \leftarrow l_i / (2^{31} - 1)$ //cast at higher precision!
end for

Uniform sampling of shapes

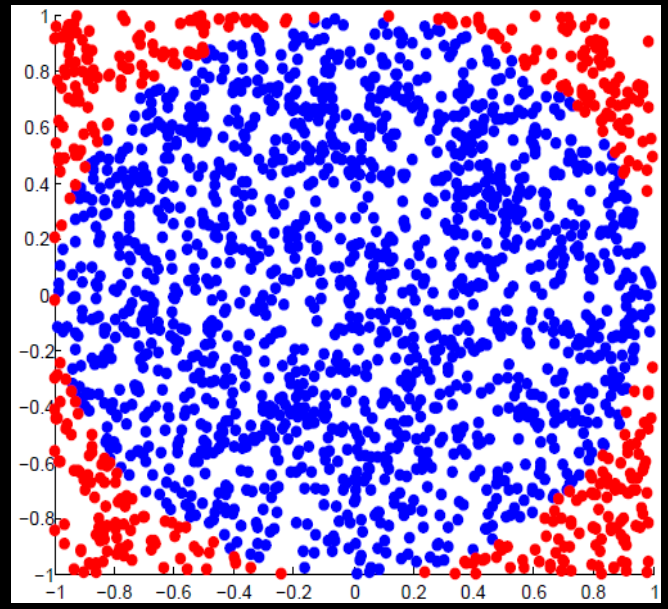
- We often want to calculate a random sample of points on an object or surface
  - Light falling on a solar panel
  - Function values in polar coordinates

- Rejection sampling
  - Sample a box circumscribing the shape
  - Exclude samples outside the shape
  - Inefficient, but useful when shapes are difficult to parameterize



Sampling hyperspheres

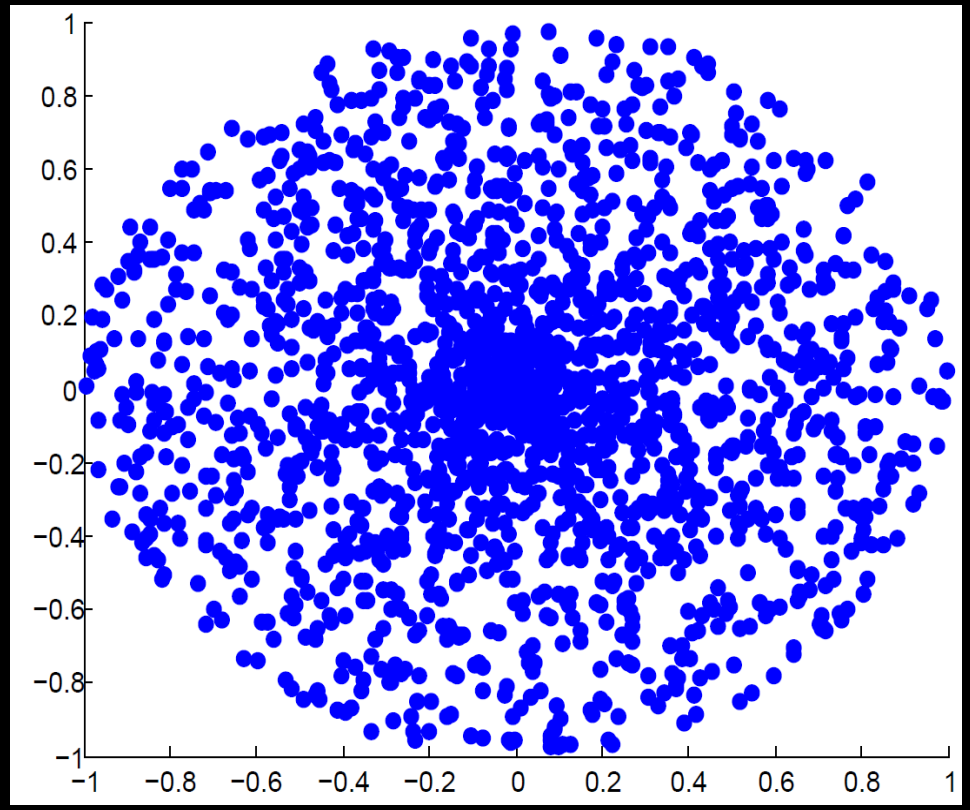
- Sampling volumes of hyperspheres is extremely common
  - Disks, spheres, etc.
- Rejection sampling becomes less efficient in higher dimensions
  - Curse of dimensionality



Sampling hyperspheres

- Sample using spherical coordinates?

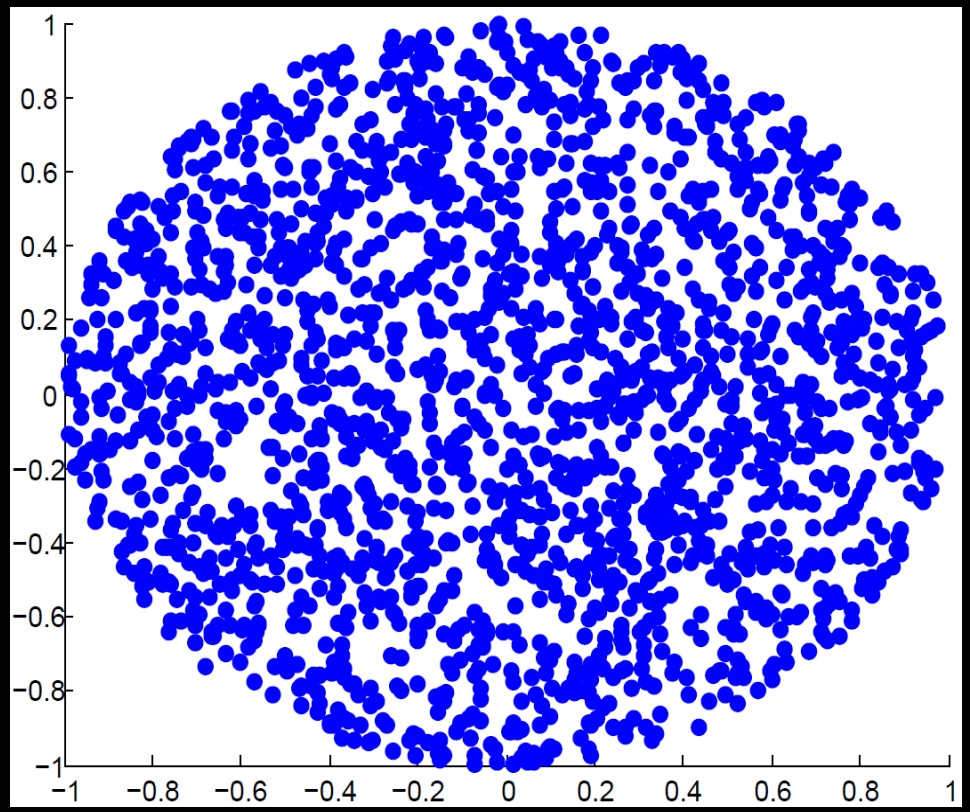
$$r \leftarrow [0, 1]$$
$$\theta \leftarrow [0, 2\pi]$$
$$x = r \cos\theta$$
$$y = r \sin\theta$$



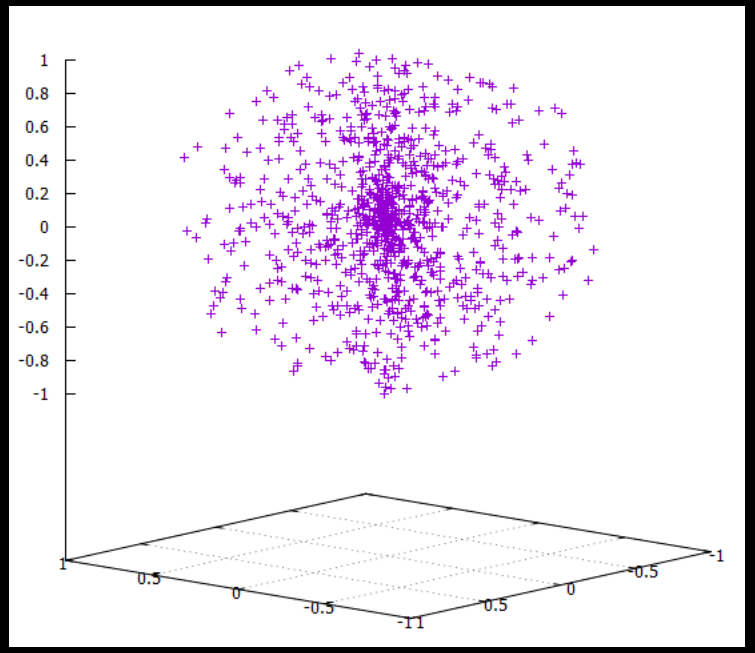
Sampling hyperspheres

- Scale the distance from the center by the root of the dimension d

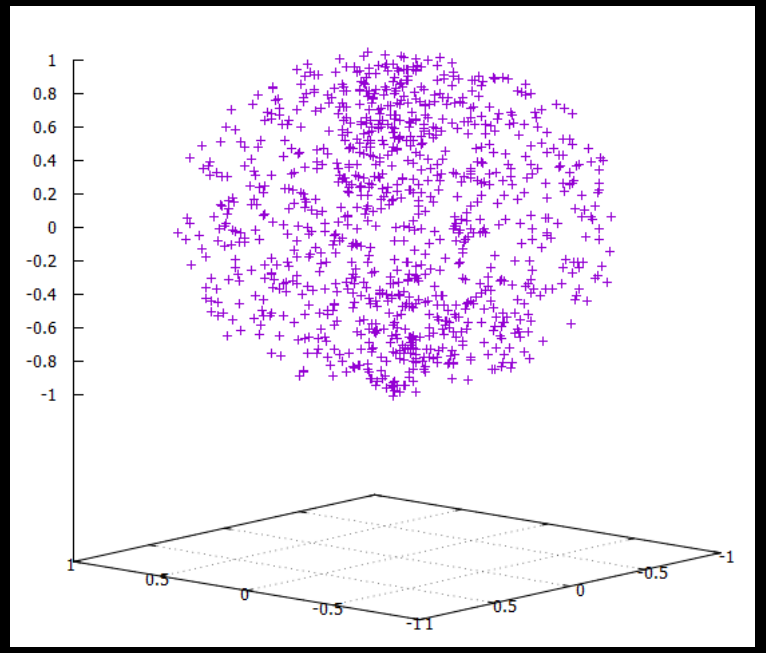
$r \leftarrow [0, 1]$
$\theta \leftarrow [0, 2\pi)$
$x = \sqrt[d]{r} \cos\theta$
$y = \sqrt[d]{r} \sin\theta$



Sampling hyperspheres



$r \leftarrow [0, 1]$
$\theta \leftarrow [0, 2\pi]$
$\phi \leftarrow [0, \pi]$
$x = r \cos\theta \sin\phi$
$y = r \sin\theta \sin\phi$
$z = r \cos\phi$



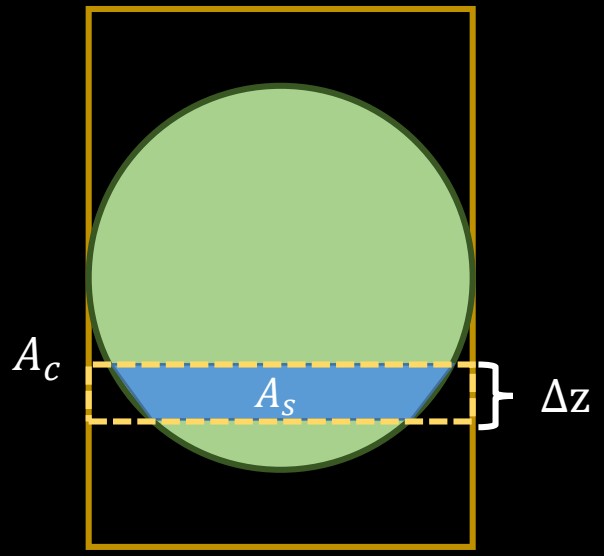
$r \leftarrow [0, 1]$
$\theta \leftarrow [0, 2\pi]$
$\phi \leftarrow [0, \pi]$
$x = \sqrt[3]{r} \cos\theta \sin\phi$
$y = \sqrt[3]{r} \sin\theta \sin\phi$
$z = \sqrt[3]{r} \cos\phi$

Sampling a solid angle

- Rely on Archimedes' Theorem
  - The area of a sphere equals the area of every right circular cylinder circumscribed about the sphere (excluding the bases).
  - The axial projection of any measurable region on a sphere on the right circular cylinder circumscribed about the sphere preserves area.
  - For any given Δz in cylindrical coordinates (r, θ, z):

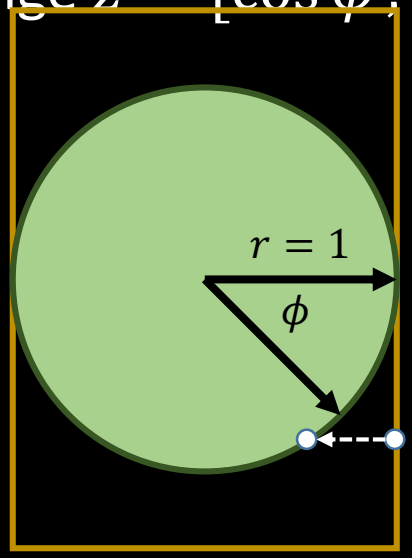
$$A_s = A_c$$

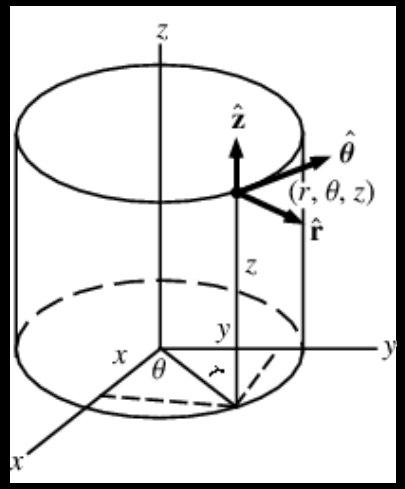
The surface area of the sphere in this band is equal to the surface area of the cylinder in this band

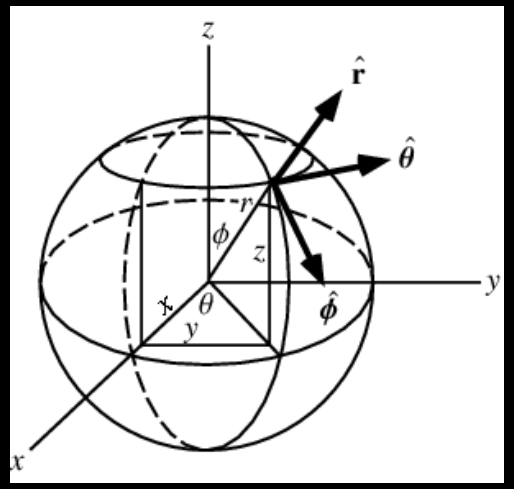


Sampling a solid angle

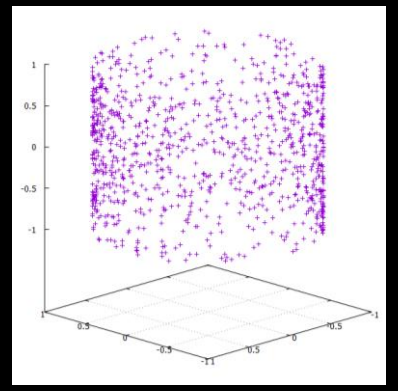
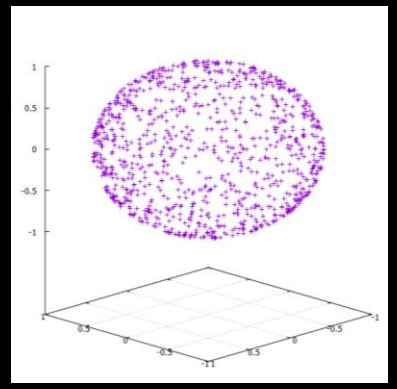
- Another method for uniform sampling of a sphere using Archimedes' Theorem (rely on polar and cylindrical cords.)
  - Create two uniform random variables $z = [0, 1]$, and $\theta = [0, 2\pi]$
    - These will sample the surface of a cylinder of length 1
  - Project these points onto the unit sphere $(r = 1, \theta, \phi)$
    - $r = 1, \phi = \cos^{-1} z$
  - Advantage: pick a solid angle ϕ and limit the random numbers to sampling in the range $z = [\cos\phi\ , 1]$



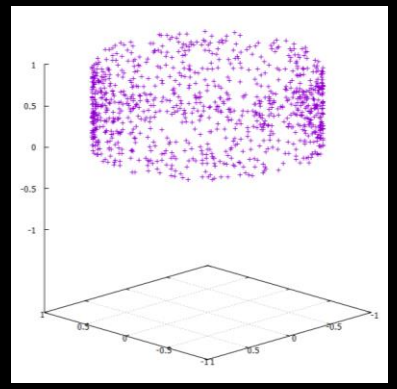
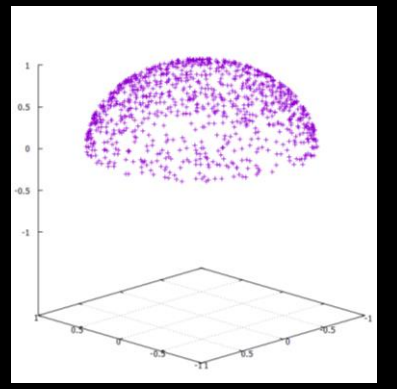




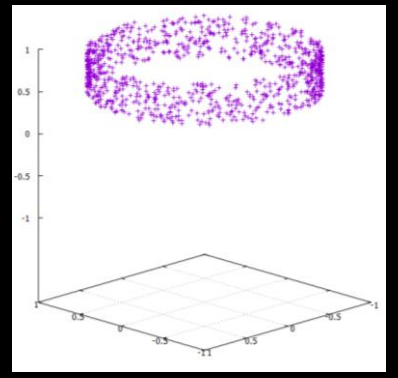
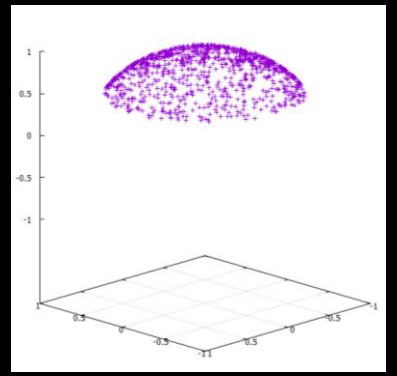
Sampling a solid angle

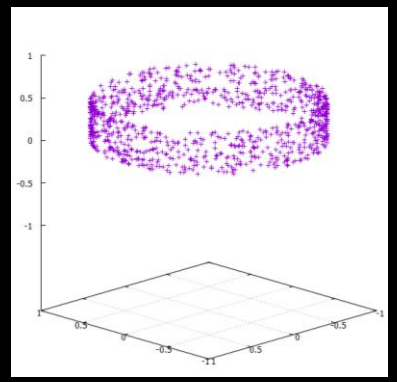
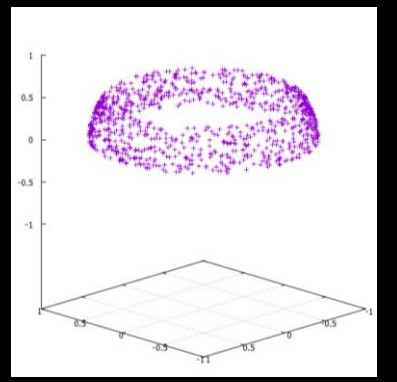
$z = [-1, 1]$
$\phi = [0, \pi]$

$z = [0, 1]$
$\phi = \left[0, \frac{\pi}{2}\right]$

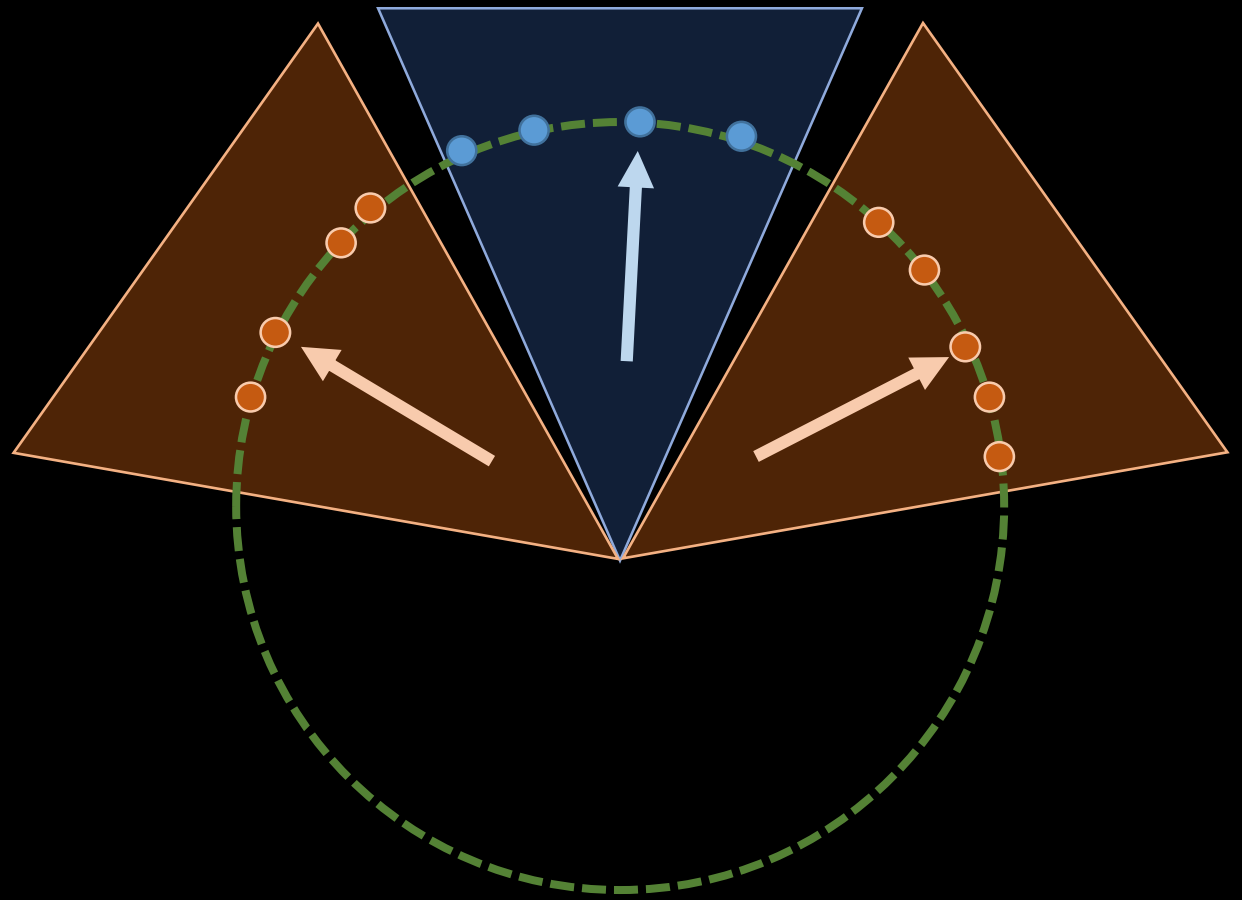
$z = \left[\frac{1}{2}, 1\right]$
$\phi = \left[0, \frac{\pi}{3}\right]$

$z = \left[0, \frac{1}{2}\right]$
$\phi = \left[\frac{\pi}{3}, \frac{\pi}{2}\right]$

Sampling a solid angle

- Samples on the sphere surface can also represent vectors
  - Used to compute random vectors within a solid angle



Generating Normal Distributions

- Box-Muller transform – generate a normal distribution from uniform random variables
- Requires two random integer variables x_0 and x_1

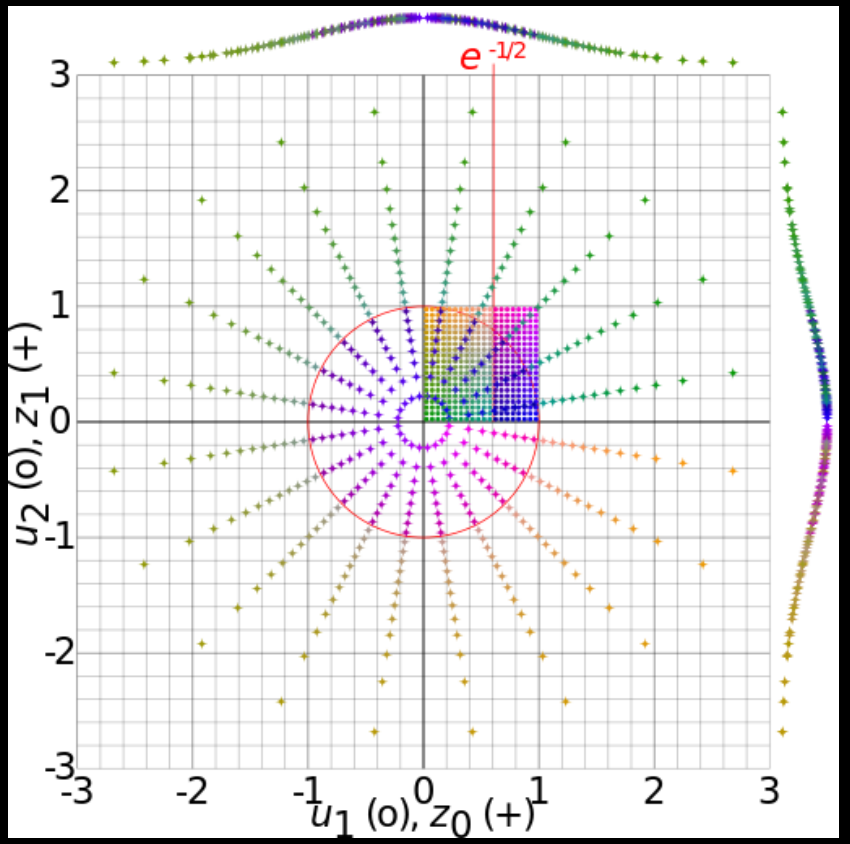
$$z_0 = r\cos\theta$$

$$z_1 = r\sin\theta$$

where

$$r = \sqrt{-2\ln x_0}$$

$$\theta = 2\pi x_1$$



Monte-Carlo Methods

Monte-Carlo Methods

- Estimate the results of functions using randomized samples

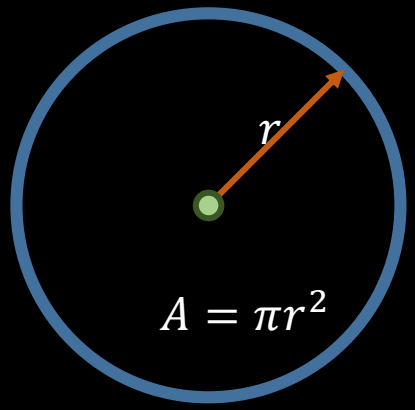
- Estimate the value of π using random values
  - We know that the area of a circle is given by

$$A = \pi r^2$$

  - Therefore

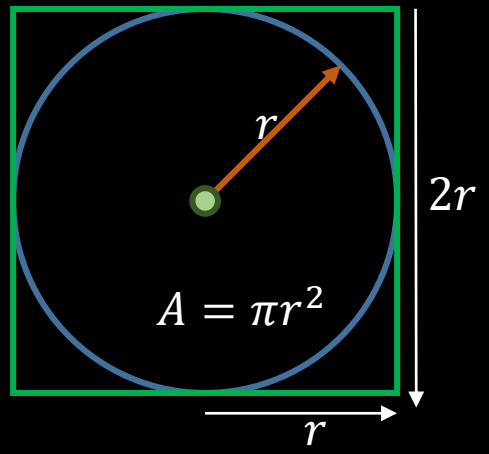
$$\pi = \frac{A}{r^2}$$

- If we want to calculate π, we need the area and radius of some circle



Monte-Carlo Methods

- Inscribe this circle into a square with a width and height of $2r$



- The area of the circle relative to the area of the square A_s is

$$A = A\left(\frac{A_s}{A_s}\right) \rightarrow A = A_s\left(\frac{A}{A_s}\right)$$

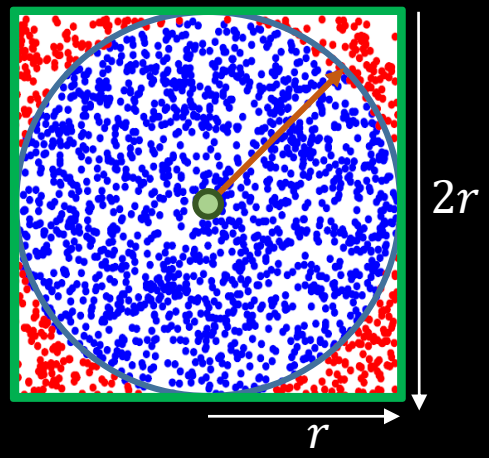
  - Since the area of the square is $A_s = 2r \cdot 2r = 4r^2$, the area of the circle is

$$A = 4r^2 t$$

where $t = \frac{A}{A_s} = \frac{\text{points inside the circle}}{\text{points inside the square}}$

Monte-Carlo Methods

- Sample a 2D uniform distribution on $x, y = [0, 2r]$



- The ratio $t = \frac{A}{A_S} = \frac{N_{in}}{N}$ where N is the number of random samples and

$$N_{in} = \begin{cases} 1, & \sqrt{x^2 + y^2} < r \\ 0, & \text{otherwise} \end{cases}$$

- π is calculated using

$$A = \pi r^2 = 4r^2 \left(\frac{N_{in}}{N}\right)$$

$$\pi = 4\left(\frac{N_{in}}{N}\right)$$

Convergence of π using MC

- Test the accuracy of π as a function of N using Matlab

N	Monte-Carlo	Uniform
$N = 10$	$\pi = 2.4$	$\pi = 2.4$
$N = 100$	$\pi = 3.28$	$\pi = 2.68$
$N = 1000$	$\pi \approx 3.116$	$\pi = 3.016$
$N = 10000$	$\pi \approx 3.155$	$\pi = 3.1$
$N = 100000$	$\pi \approx 3.1402$	$\pi = 3.1293$

- Convergence is slightly faster than regular sampling in 2D

Monte-Carlo Integration

- The definite integral

$$\int_0^1 f(x)dx$$

can be approximated using Monte-Carlo methods

- If the domain of integration is $x = [0, 1]$, we are essentially asking for the *average* value of $f(x)$ in $x = [0, 1]$.
- We can estimate the average by taking a set of N random values $x = \{x_1, x_2, \cdots, x_n\}$ and calculating the mean:

$$\int_0^1 f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

Monte-Carlo Integration

- Computing the average value in the domain $x = [a, b]$ is the equivalent of solving

$$\frac{1}{b-a}\int_a^b f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

- Therefore, the integral on any domain of integration can be approximated by

$$\int_a^b f(x)dx \approx \frac{b-a}{N}\sum_{i=1}^{N} f(x_i)$$

- In general, multiply the mean function value by the *area* of the integration domain.
- So, for a multi-dimensional integral

$$\int_{y_1}^{y_2}\int_{x_1}^{x_2} f(x,y)dxdy \approx \frac{(x_2-x_1)(y_2-y_1)}{N}\sum_{i=1}^{N} f(x_i,y_i)$$

Advantages of Monte-Carlo

- If regular sampling requires N evaluations for a 1D function, it will require N^d evaluations for similar accuracy in d dimensions.
  - Another aspect of the curse of dimensionality
- Monte-Carlo sampling provides faster convergence
  - Error behaves as $\mathrm{O}\left(\frac{1}{\sqrt{N}}\right)$
  - So $N = 16$ provides an error bound of $O\left(\frac{1}{4}\right)$ and $N = 4 \cdot 16 = 64$ provides an error bound of $O\left(\frac{1}{8}\right)$
  - Quadrupling the samples halves the error, no matter the number of dimensions

Sampling a hypersphere surface

- One other common problem is sampling a solid angle
  - Particle impacts – micrometeorites, dust particles
  - Radiation simulations – shielding, solar panel models, radiation detectors
- Uniform sampling on a hypersphere surface
  - Generate normally distributed random variables $\boldsymbol{x} = [x_1, x_2, \cdots x_n]$ with $\mu = c$, where c is the sphere center
  - Project:

$$\boldsymbol{s} = \frac{1}{\|\boldsymbol{x}\|}\boldsymbol{x}$$