Survey of Scientific Computing

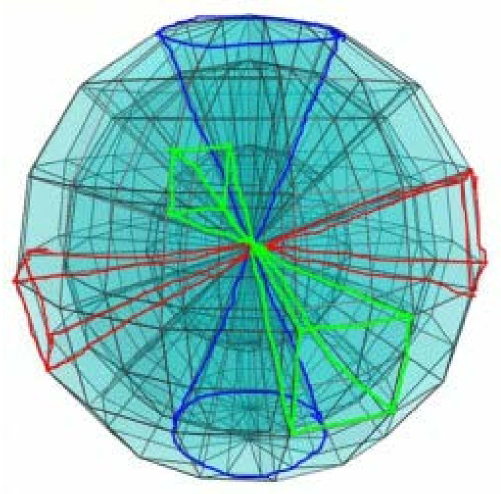
(SciComp 301)

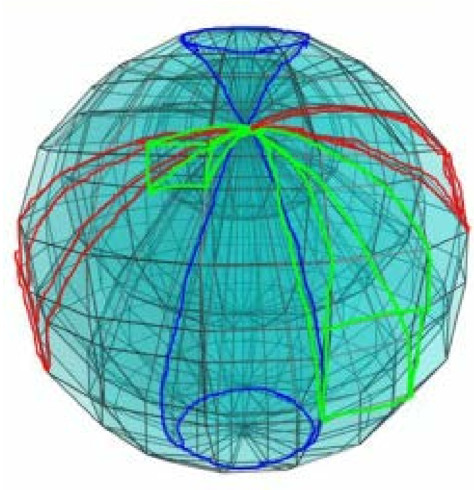
Dave Biersach
Brookhaven National Laboratory
dbiersach@bnl.gov

Session 20
Monte Carlo Method

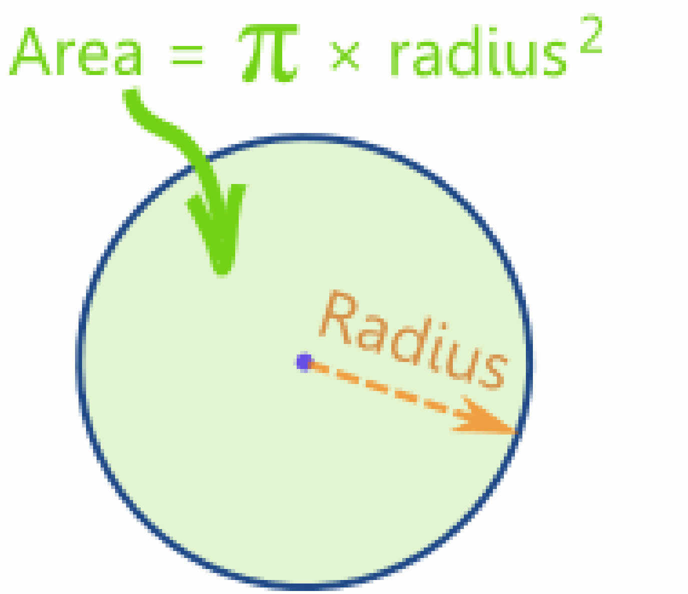
An *Interesting* Question

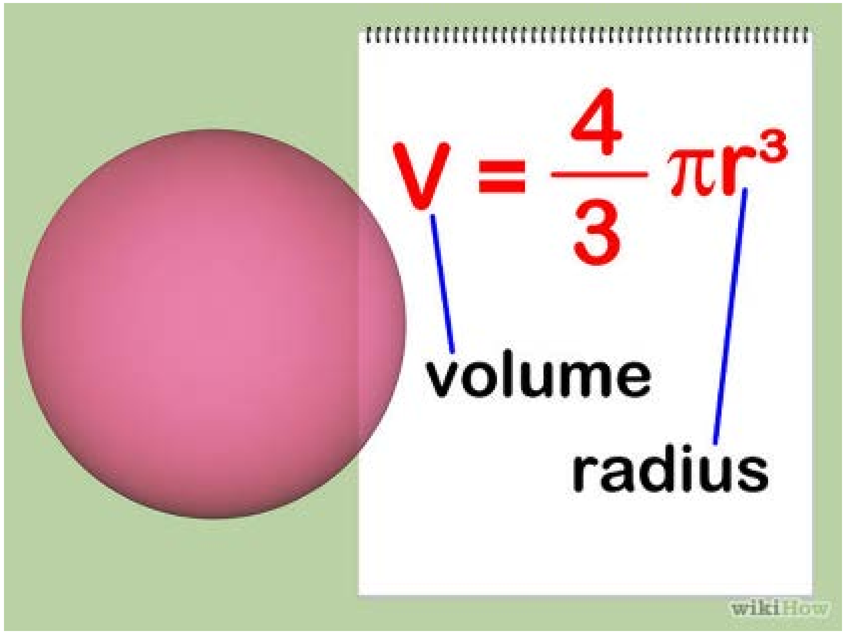
- What is the ***volume*** of a **four-dimensional** **unit** hypersphere?
  - What does a 4D sphere "look" like?
  - What is a "unit" sphere?
  - Where do I even start?
- Break down complex questions into simpler steps:
  - How can we calculate the area of a 2D circle?
  - How can we calculate the volume of a 3D sphere?
  - How do we move from 3D to 4D?



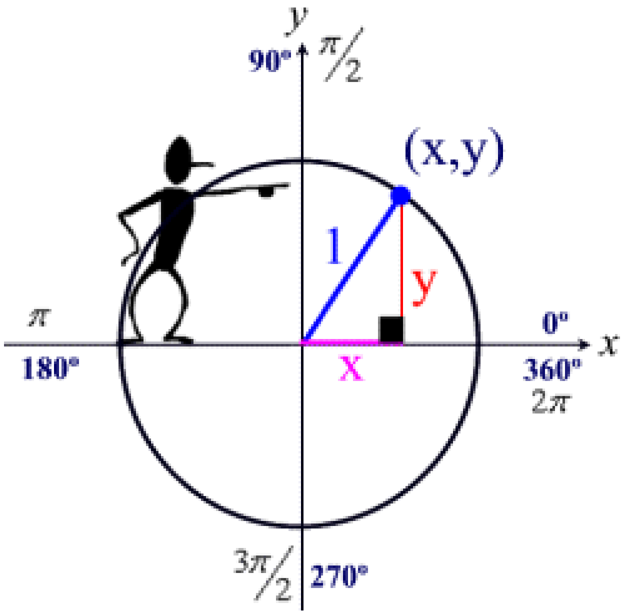


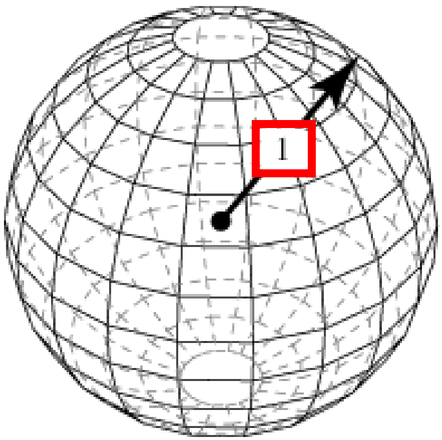
Area and Volume



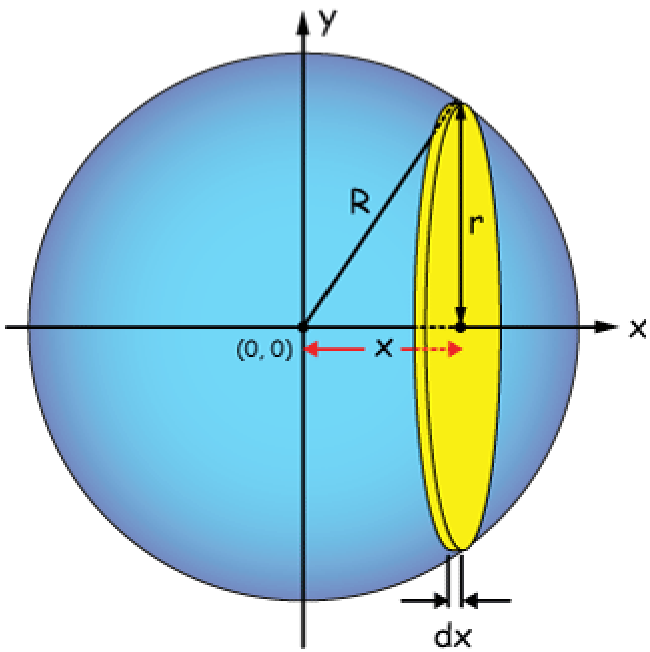


A **Unit** Circle and **Unit** Sphere

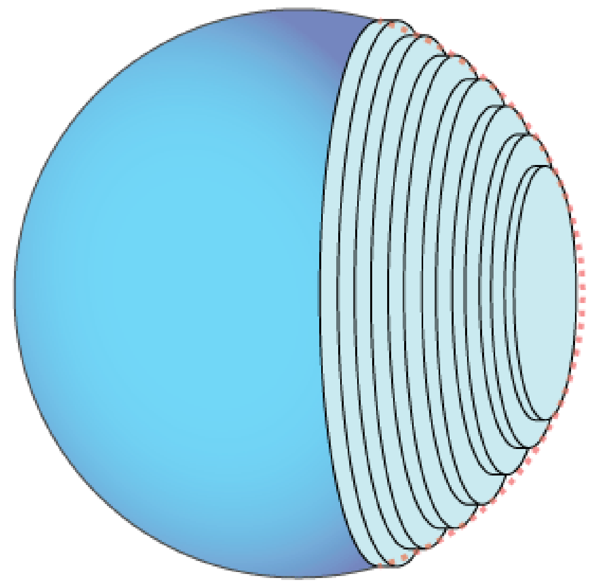




2-D Area → 3-D Volume

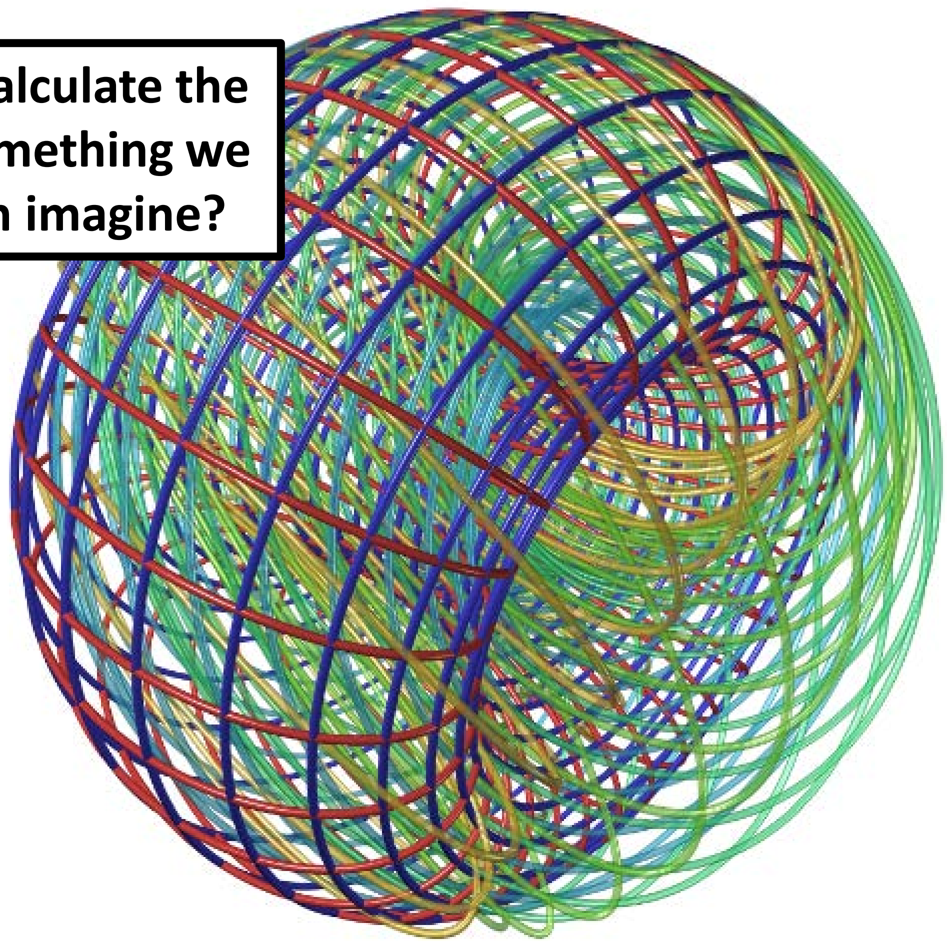


Volume of the disk:
$\pi r^2 \cdot dx = \pi(R^2 - x^2) \cdot dx$

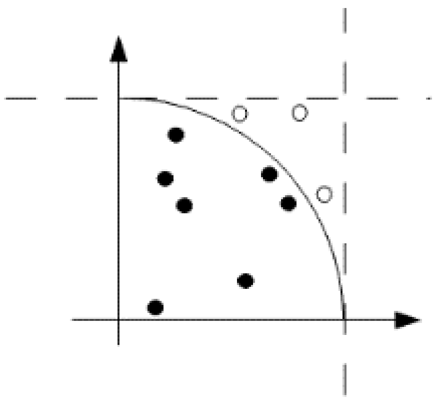


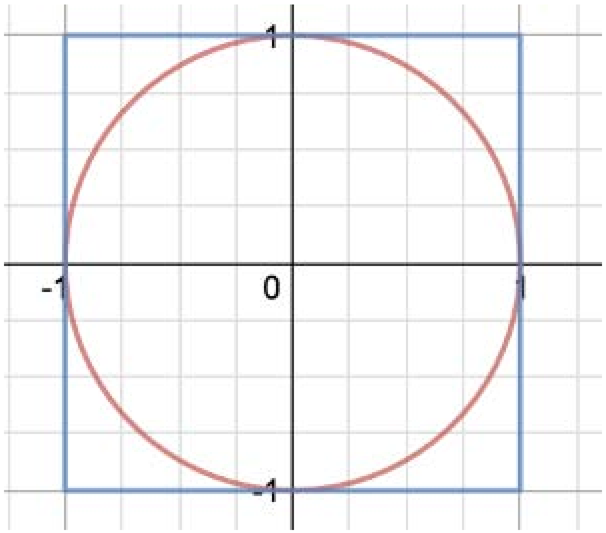
A 4-D Hypersphere

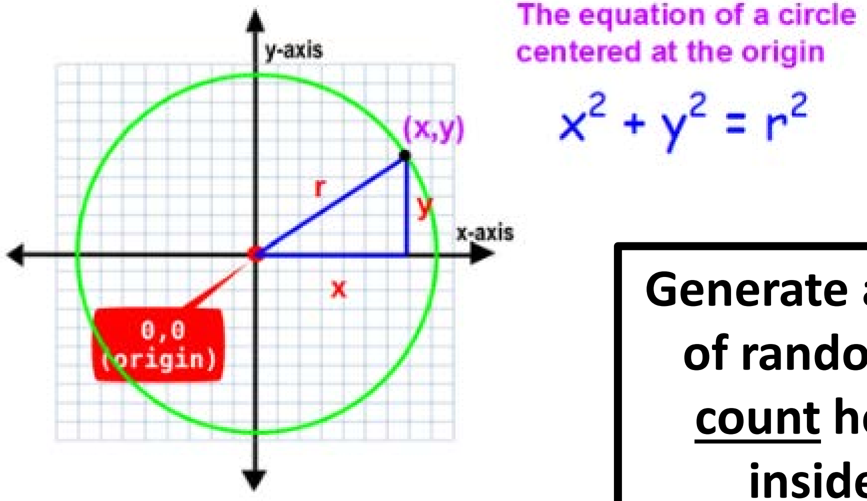
How do we calculate the volume of something we can not even imagine?

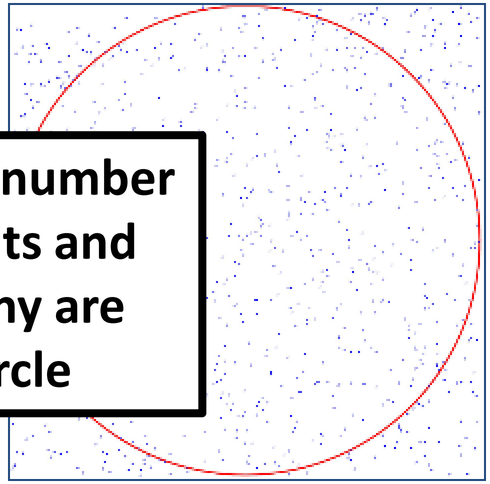


Area as a "Ratio" of Inside vs. Total Dots



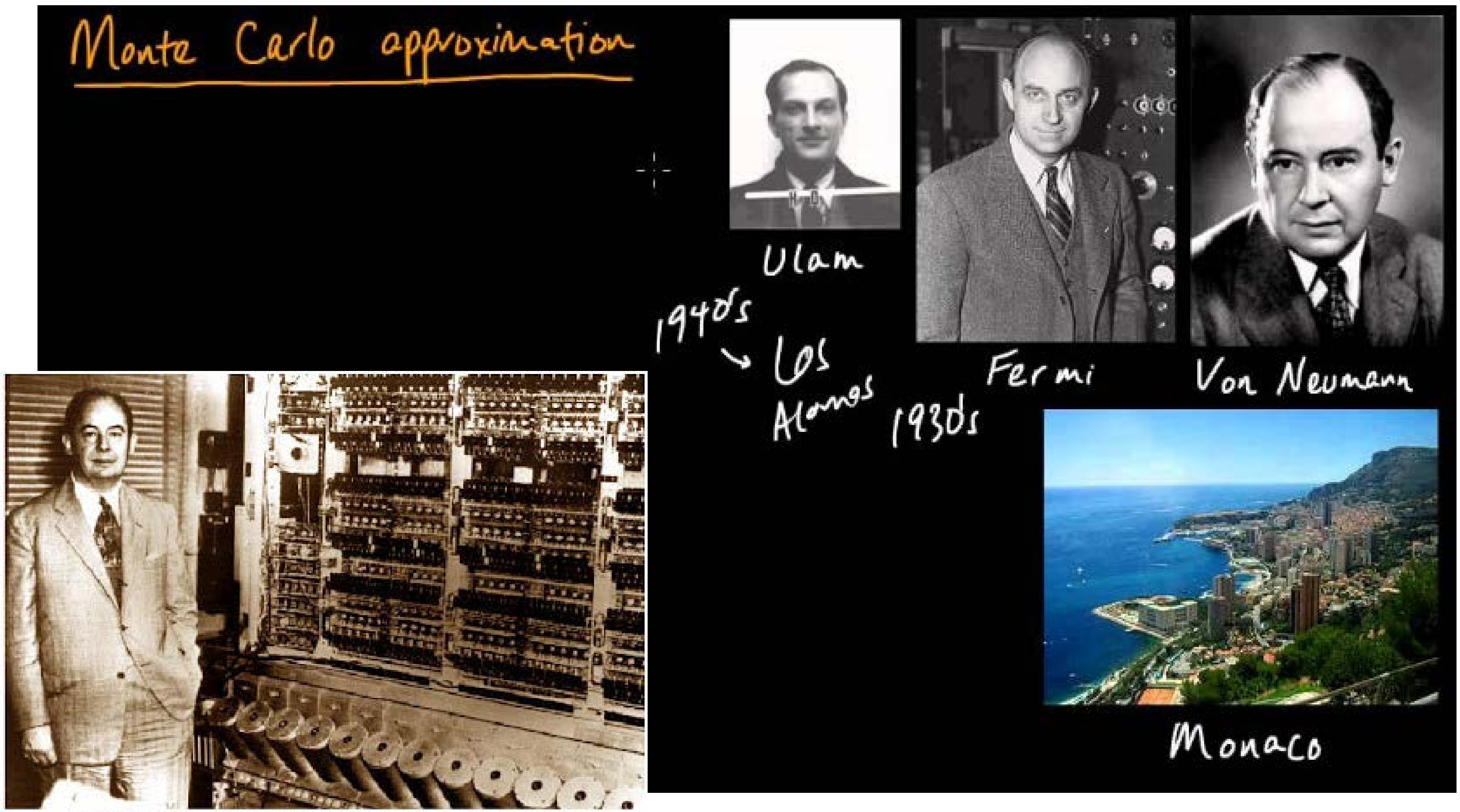






Generate a large number of random points and <u>count</u> how many are inside the circle

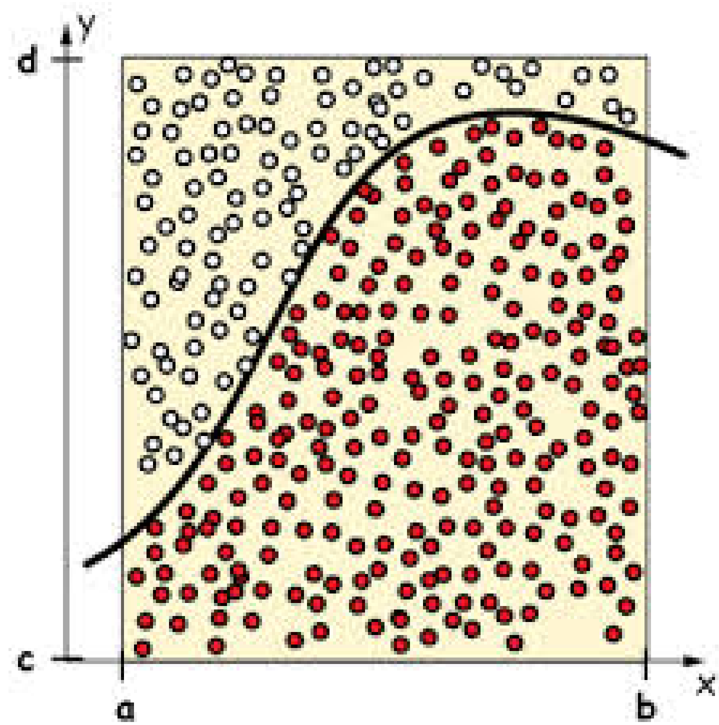
The Monte Carlo Method



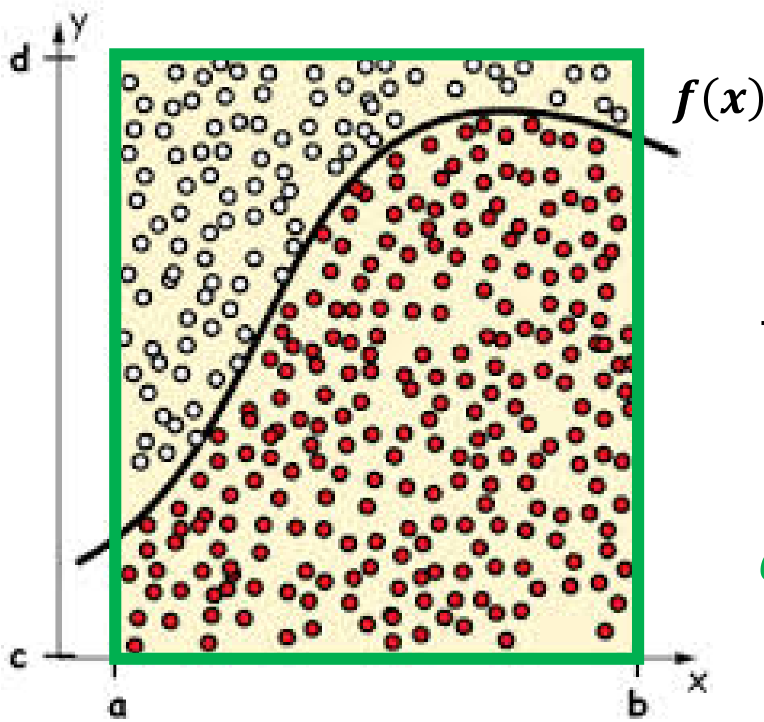
Johnny von Neumann [1903-1957] alongside the Maniac computer at the Institute for Advanced Studies, Princeton.

The Monte Carlo Method

- With Monte Carlo, we **randomly sample** points within a bounded space and count how many are below the curve
- The ratio of **inside** dots (those under the curve) vs. **total** dots leads to an estimate of the *integral*
- Monte Carlo is **non-deterministic** when a random number generator is used to create the sample points



The Monte Carlo Method



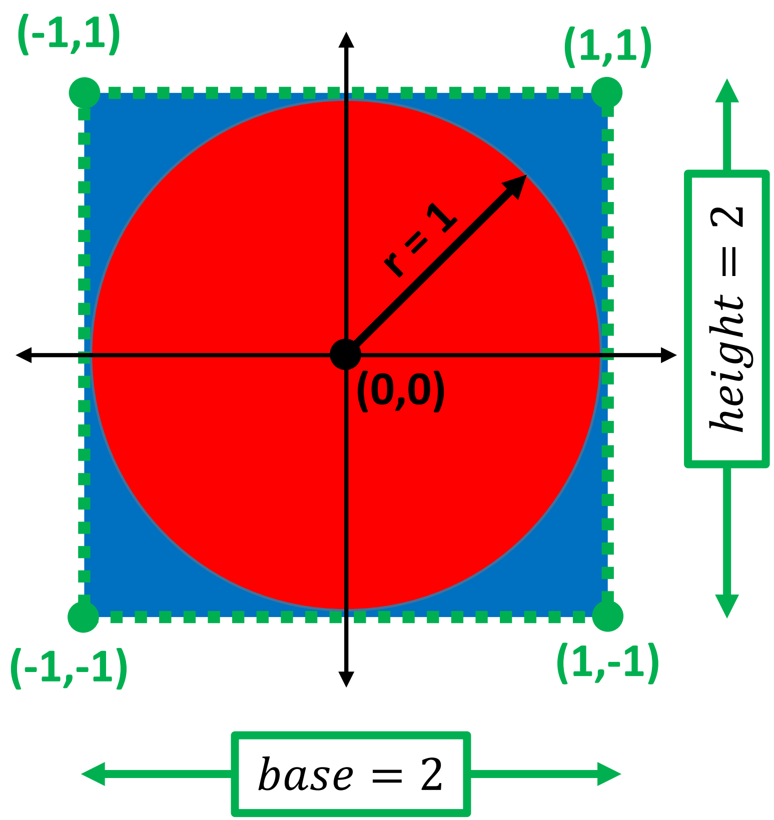
We don't know this area

$$\frac{dots_{inside}}{dots_{total}} = \frac{area_{\,curve}}{area_{sample}}$$

$$area_{sample} = (b - a) \times f(b)$$

$$area_{\,curve} = area_{sample} \times \frac{dots_{inside}}{dots_{total}}$$

The Monte Carlo Method



We don't know this area

$$\frac{dots_{inside}}{dots_{total}} = \frac{area_{\mathbf{circle}}}{area_{sample}}$$

$$\begin{aligned} area_{sample} &= base \times height \\ &= 2 \times 2 \\ &= 4 \end{aligned}$$

$$area_{\mathbf{circle}} = 4 \times \frac{dots_{inside}}{dots_{total}}$$

```cpp
void draw(SimpleScreen &ss)
{
    ss.DrawAxes();
    ss.DrawCircle(0, 0, 1, "green", 2);

    seed_seq seed{2017};
    default_random_engine generator{seed};
    uniform_real_distribution<double> distribution{-1, 1};

    const int iterations = 100000;
    int count{};

    ss.LockDisplay();

    for (int i{}; i < iterations; ++i)
    {
        double x = distribution(generator);
        double y = distribution(generator);

        if (x * x + y * y <= 1.0)
        {
            ss.DrawPoint(x, y, "red");
            count++;
        }
        else
            ss.DrawPoint(x, y, "blue");
    }

    ss.UnlockDisplay();

    double area = (double)count / iterations * 4.0;
    double err = (M_PI - area) / M_PI * 100;

    cout << "2D Circle Area PRNG" << endl
         << "Iterations = " << iterations << endl
         << "Est. Area  = " << area << endl
         << "Act. Area  = " << M_PI << endl
         << "Abs. % Err = " << abs(err) << endl;
}
```

2-D Area as a “Ratio” of Dots

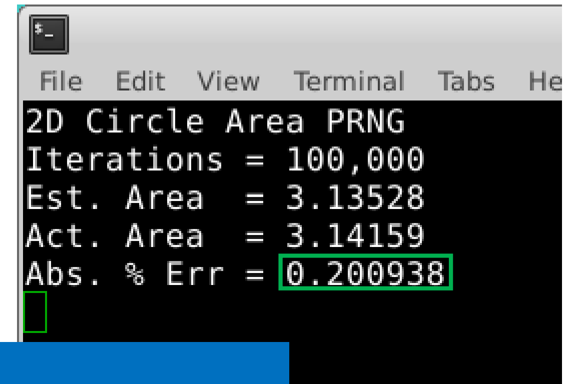
Open Lab 1

$$\frac{dots_{inside}}{dots_{total}} = \frac{area_{circle}}{area_{square}}$$

$$area_{square} = 2 \times 2 = 4$$

$$area_{circle} = \frac{count}{iterations} \times 4$$

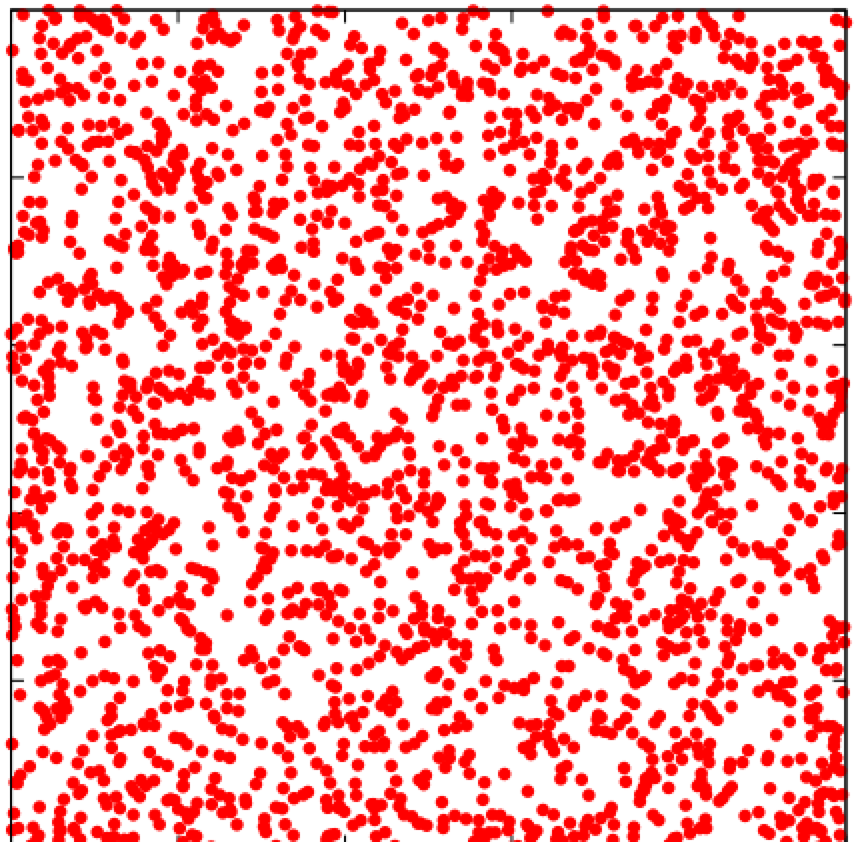
2-D Area as a “Ratio” of Dots



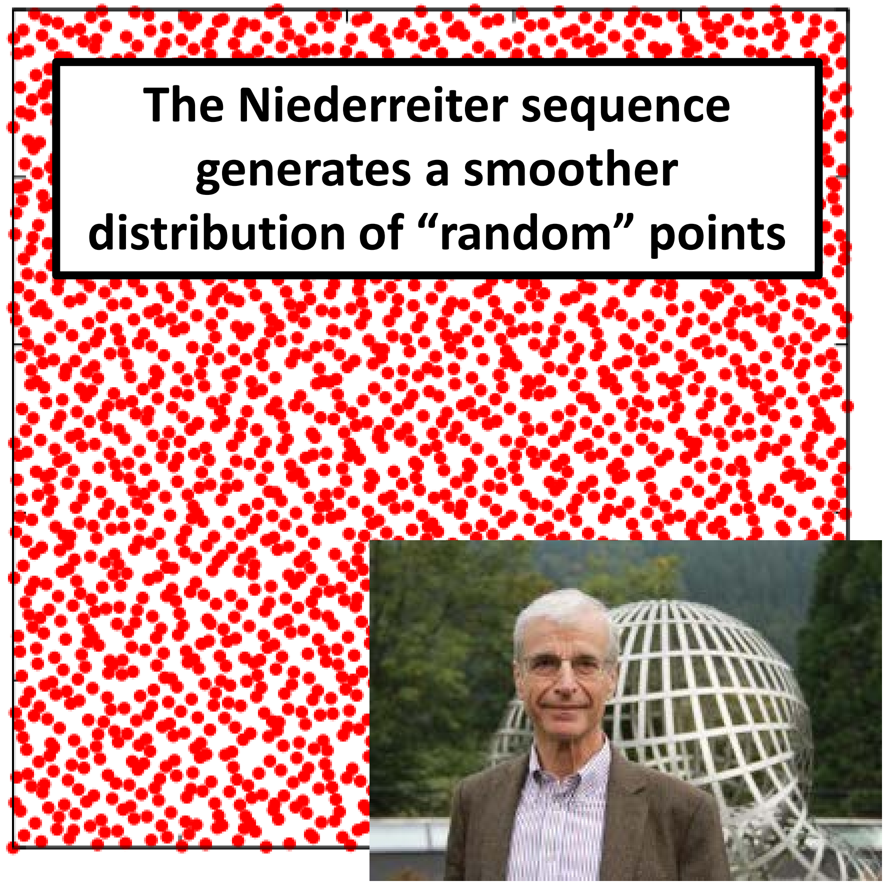
Run Lab 1

Comparing “Random” Number Generators

Standard PRNG

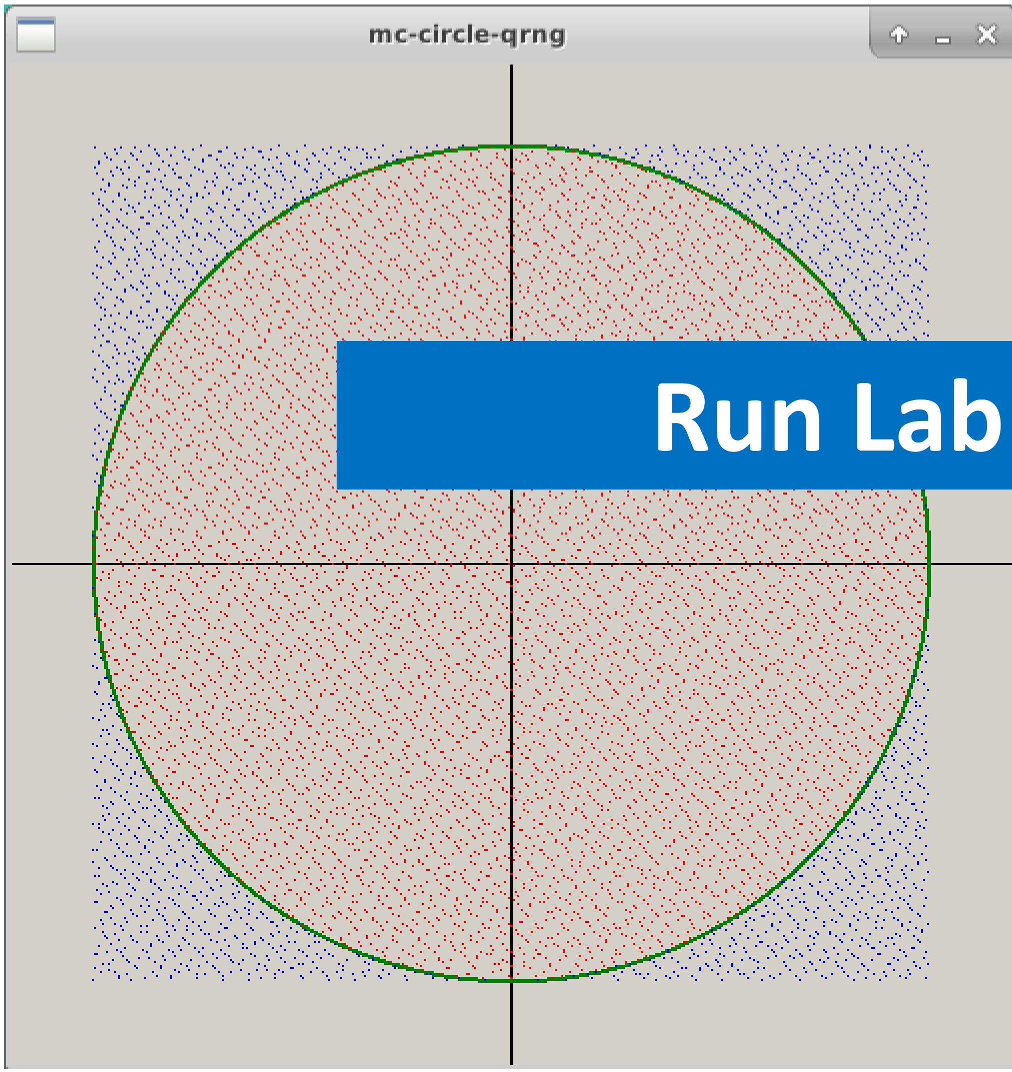


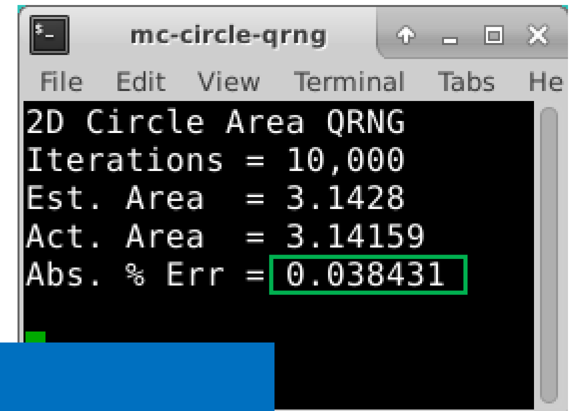
Niederreiter QRNG



The Niederreiter sequence generates a smoother distribution of “random” points

Improved 2-D Monte Carlo Estimator

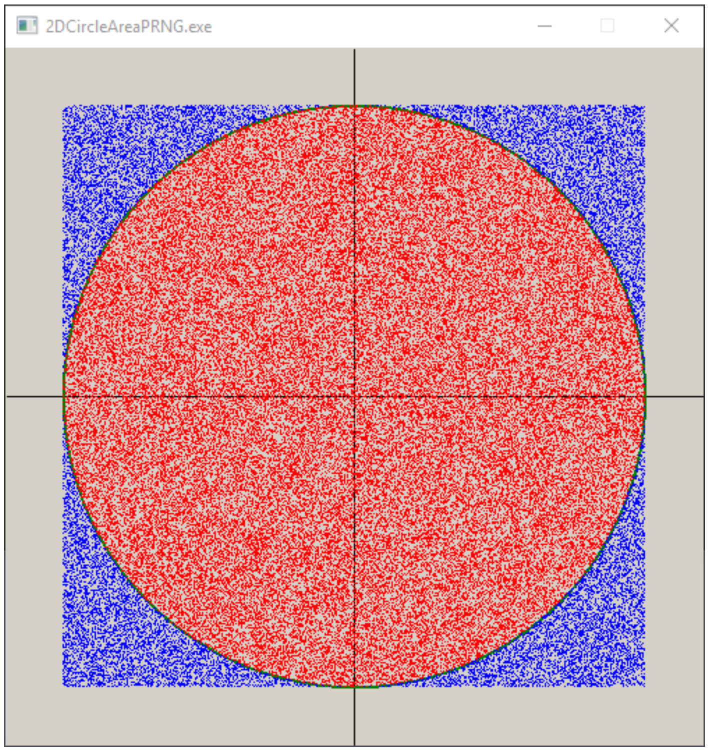




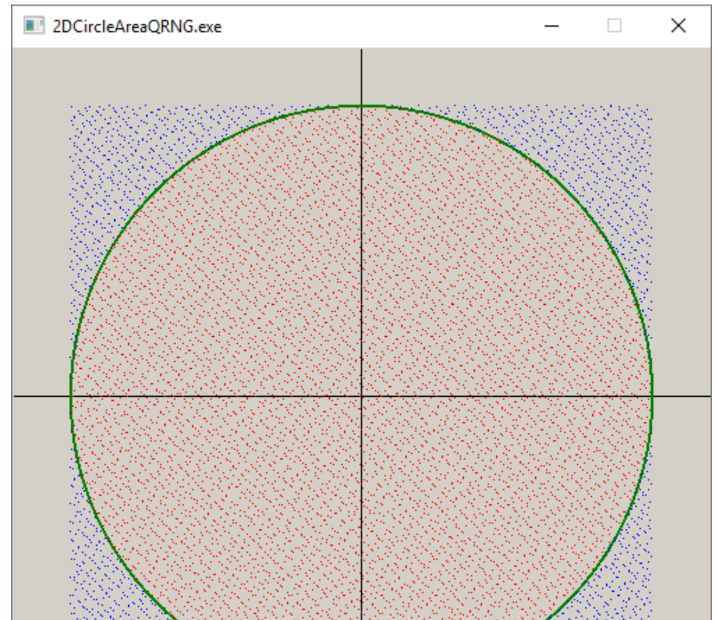
Run Lab 2

Improved 2-D Monte Carlo Estimator

PRNG % Error = .2009

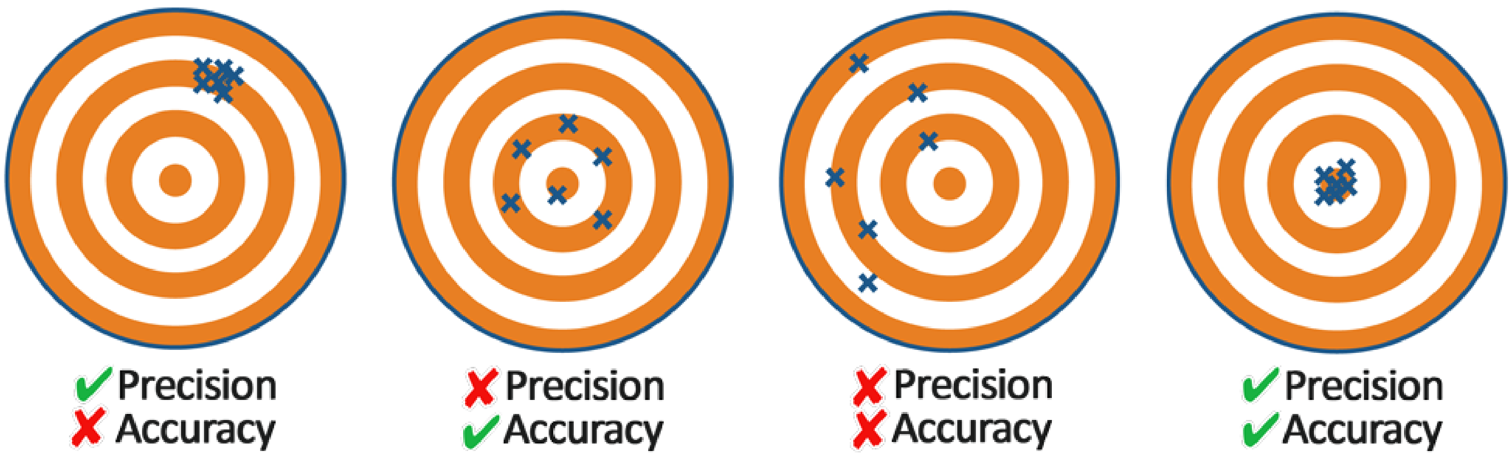


QRNG % Error = .0384



The Niederreiter sequence provides a 6X increase in accuracy of the estimate using 10X fewer points!

Accuracy vs. Precision



Accuracy = **A**im

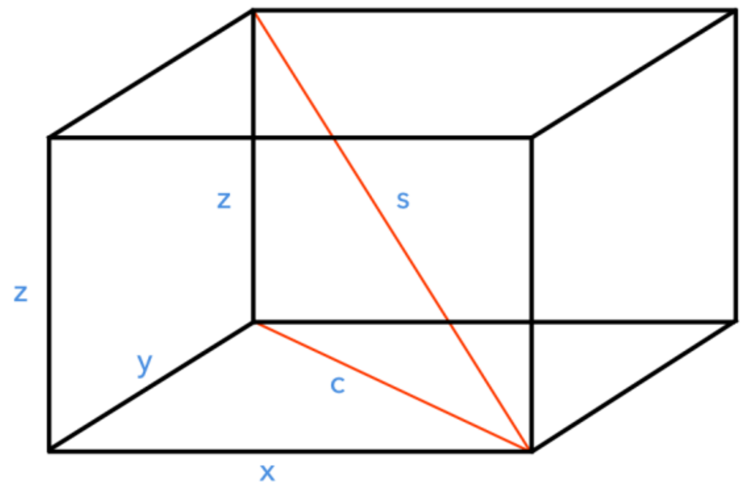
- Hitting what you are pointing at

Precision = **Consistency**

- Digits of Precision ≠ Accuracy
- More digits → less *variability*

Moving to Higher Dimensions

The Pythagorean Distance is a **metric** that is true in all **orthogonal** spaces of any dimension



$$c^2 = x^2 + y^2 \qquad s^2 = c^2 + z^2$$

$$\therefore s^2 = x^2 + y^2 + z^2$$

3-D Unit Sphere Volume Estimator

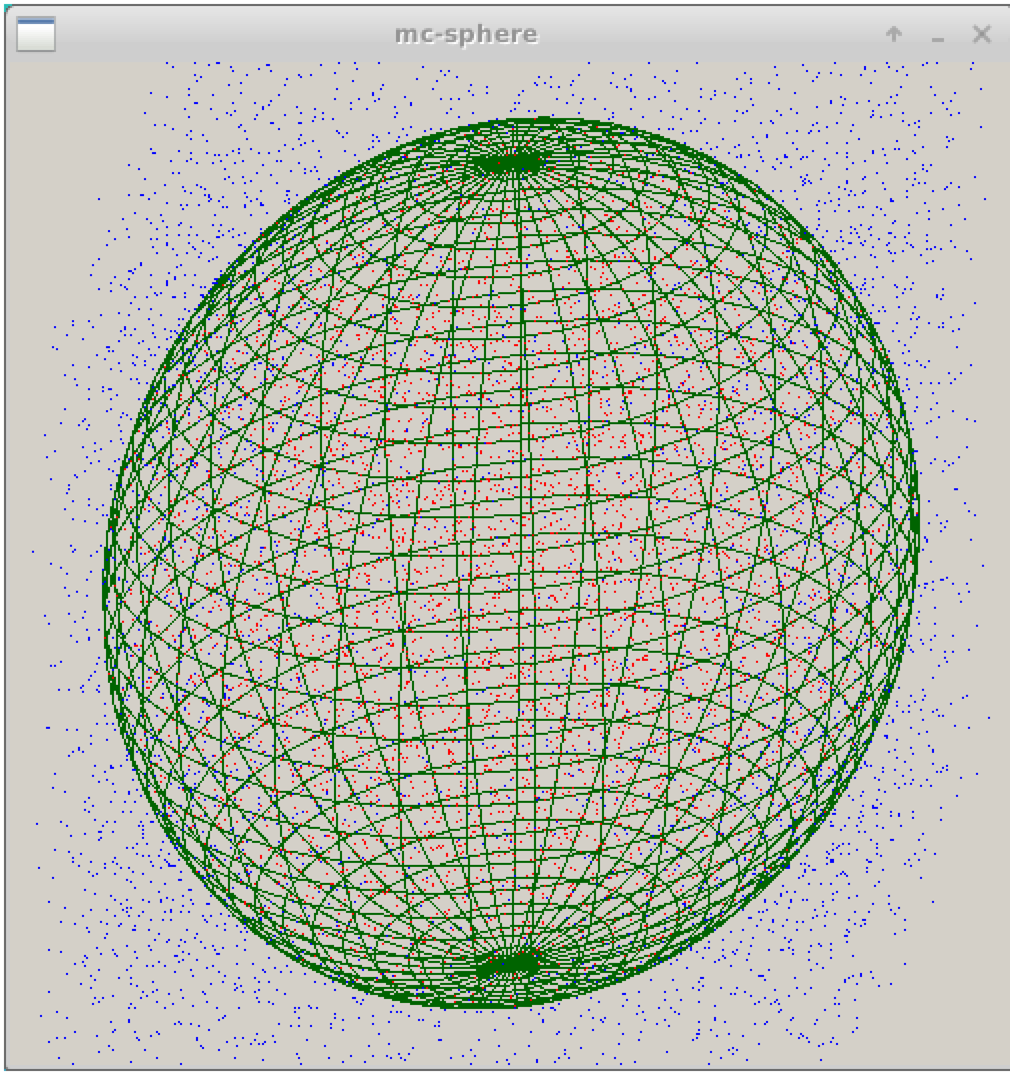
```
Niederreiter2 qrng;
double r[3];
int seed{};

const int iterations = 10000;
int count{};

ss.LockDisplay();

for (int i{};i < iterations;++i) {
    qrng.Next(3, &seed, r);

    double x = r[0] * -2.0 - 1.0;
    double y = r[1] * -2.0 - 1.0;
    double z = r[2] * -2.0 - 1.0;
    if (x*x + y*y + z*z <= 1.0) {
        ss.DrawPoint3D(x, y, z, "red");
        count++;
    }
    else
        ss.DrawPoint3D(x, y, z, "blue");
}

ss.UnlockDisplay();

double estVol = (double)count / iterations * 8;
double actVol = 4.0 / 3.0 * M_PI;
double err = (actVol - estVol) / actVol * 100;

cout << "3D Sphere Volume QRNG" << endl
    << "Iterations = " << iterations << endl
    << "Est. Volume  = " << estVol << endl
    << "Act. Volume  = " << actVol << endl
    << "Abs. % Error = " << abs(err) << endl << endl;
```

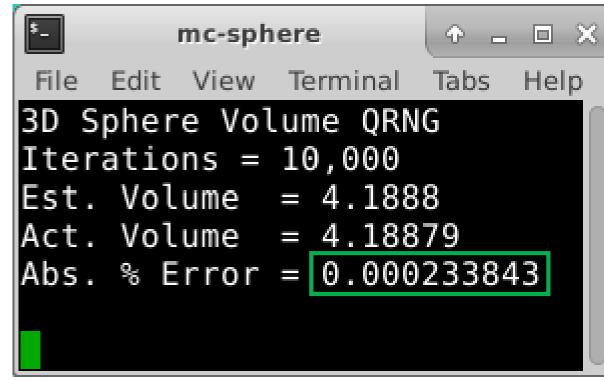
$$\frac{dots_{inside}}{dots_{total}} = \frac{volume_{sphere}}{volume_{cube}}$$

$$volume_{cube} = 2 \times 2 \times 2 = 8$$

$$volume_{sphere} = \frac{count}{iterations} \times 8$$

3-D Unit Sphere Volume Estimator





3

$$= \frac{4\pi}{3}$$

The Halton Sequence

```
double Halton(int n, int p)
{
    double h = 0;
    double f = 1;
    for (int i = n; i > 0; i /= p) {
        f = f / p;
        h += f * (i % p);
    }
    return h;
}
```

Mathematical Proceedings of the Cambridge Philosophical Society

Mathematical Proceedings of the Cambridge Philosophical
/ Issue 02 April 1965, pp 497-498

DOI: http://dx.doi.org/10.1017/S0305004100004059 (About
online: 24 October 2008

Table of Contents - 1965 - Volume 61, Issue 02

Buy This Article $45.00 / £30.00 | Rent This Article Now for 24 Hours $5.99 / £3.99 / €4.49 | Request Permissions | < Previous Abstract | Ne

Research Article

On the relative merits of correlated and importance sampling for Monte Carlo integration

John H. Halton[a1]

[a1] Brookhaven National Laboratory, Upton, New York

Accommodating the 4^{th} Dimension

```
int main()
{
    int iterations = int(1e7);

    double count = 0;

    for (int i = 0; i < iterations; i++)
    {
        double x = Halton(i, primes[0]);
        do
        do

        double distance = x * x + y * y + z * z;

        if (distance <= 1.0)
            count++;
    }

    double volume = count / iterations * 8;

    cout << fixed << setprecision(4)
         << volume << endl;

    return 0;
}
```

We need to update this code to include the **4^{th}** ...on!

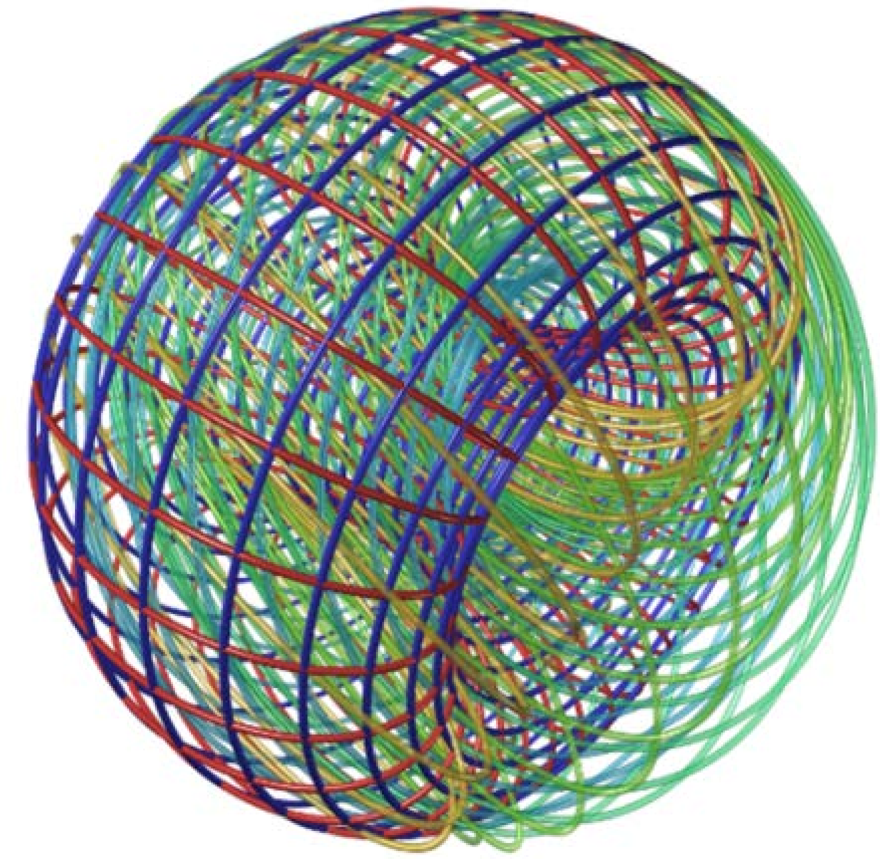
Open and edit Lab 4

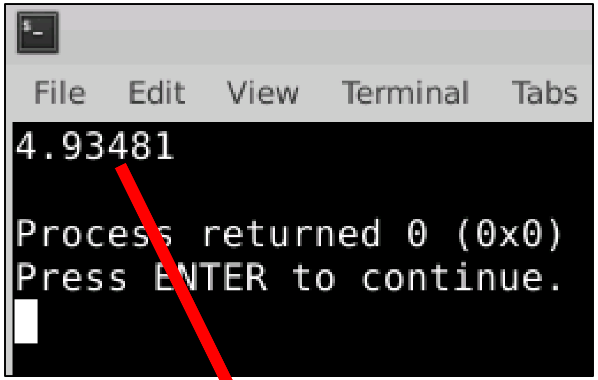
Accommodating the 4th Dimension

```
int main()
{
    int iterations = int(1e7);

    double count = 0;

    for (int i = 0; i < iterations; i++)
    {
        double x = Halton(i, primes[0]);
        double y = Halton(i, primes[1]);
        double z = Halton(i, primes[2]);
        double w = Halton(i, primes[3]);

        double distance = x * x + y * y + z * z + w * w;

        if (distance <= 1.0)
            count++;
    }

    double volume = count / iterations * 16;

    cout << volume << endl;

    return 0;
}
```

Add all the code in **red** then run the application

What is the content of a 4-D unit hypersphere?





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

We <u>can</u> calculate the volume of something we can not even *imagine!*

What lurks beyond the 4^{th} dimension?

```
int main()
{
    int iterations = int(1e7);
    for (int dimension{2}; dimension < 13; ++dimension)
    {
        double count{};
        for (int i{}; i < iterations; ++i)
        {
            double distance = 0;
            for (int d{}; d < dimension; ++d)
            {
                double v = Halton(i, primes[d]);
                distance = distance + v * v;
                if (distance > 1.0)
                    break;
            }
            if (distance <= 1.0)
                count++;
        }
        double volume = count / iterations * pow(2, dimension);
        cout << fixed << right << setw(2) << dimension << ", "
             << setprecision(4) << volume
             << endl;
    }

    return 0;
}
```

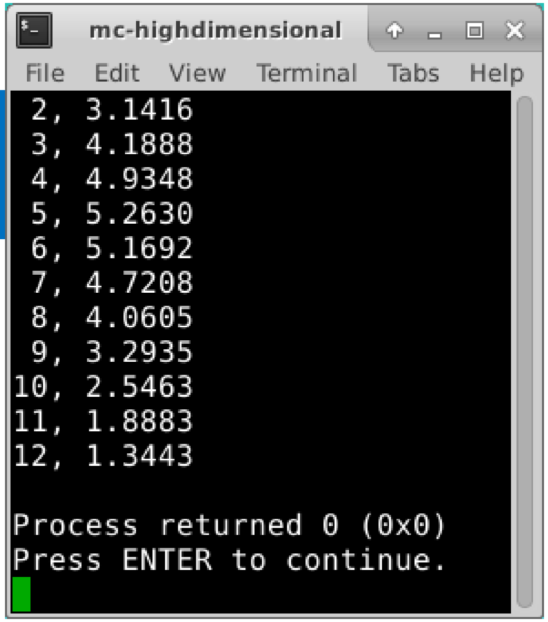
This code will estimate the volume up to the **12th** dimension!

$$volume_{sphere} = \frac{count}{iterations} \times 2^{dimensions}$$

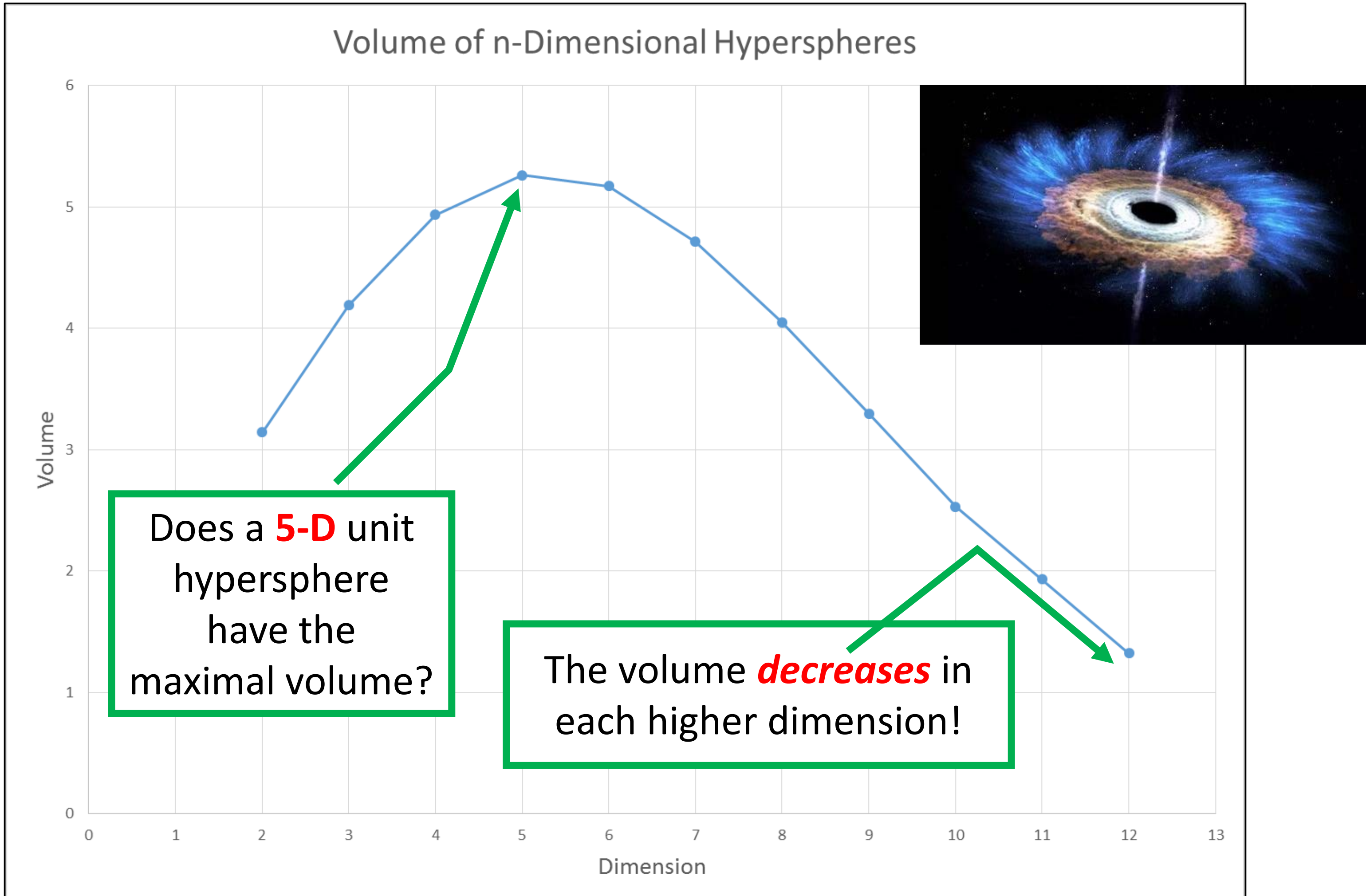
What lurks beyond the 4th dimension?

```
int main()
{
    int iterations = int(1e7);
    for (int dimension{ 2 }; dimension < 13; ++dimension) {
        double count{};
        for (int i{}; i < iterations; ++i) {
            double distance = 0;
            for (int d{}; d < dimension; ++d) {
                double v = Halton(i, primes[d]);
                distance = distance + v * v;
                if (distance > 1.0)
                    break;
            }
            if (distance <= 1.0)
                count++;
        }
        double volume = 
        cout << fixed <<
            << setprecis
            << endl;
    }

    return 0;
}
```

Run Lab 5



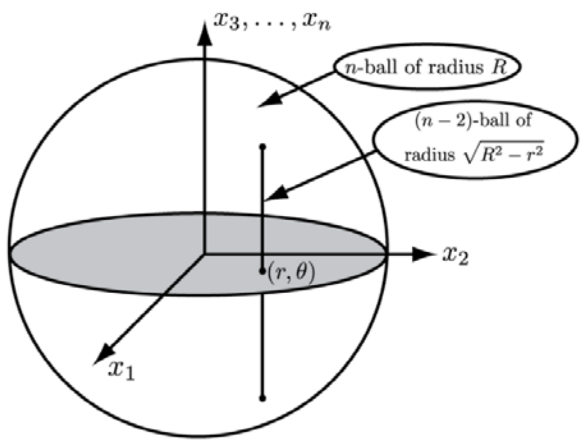
What lurks beyond the 4th dimension?



A Recurrence Relation

The ***volume*** *of an* ***n-ball*** *is proportional to the unit ball for that dimension*

$$\boxed{V_n(R) = V_n(1)R^n}$$



We can compute $V_n(1)$ by integrating the $\boldsymbol{n-2}$ ball over a unit disk using polar coordinates

$$V_n(1) = \int_0^1 \int_0^{2\pi} V_{n-2}(1)\left(\sqrt{1-r^2}\right)^{n-2} r\, d\theta\, dr$$

$$= V_{n-2}(1) \int_0^1 r(1-r^2)^{\frac{n-2}{2}}\, \theta\Big|_0^{2\pi}\, dr$$

$$= 2\pi V_{n-2}(1) \int_0^1 r(1-r^2)^{\frac{n-2}{2}}\, dr$$

$$\boxed{V_n(1) = \frac{2\pi}{n} V_{n-2}(1)}$$

A Recurrence Relation

	$V_n(1) = \frac{2\pi}{n} V_{n-2}(1)$	$V_n(R) = V_n(1) R^n$
By definition	$V_o(1) = 1$	$V_o(R) = 1$
$1 - (-1) = 2$	$V_1(1) = 2$	$V_1(1) = 2R$
Lab 2	$V_2(1) = \frac{2\pi}{2}(1) = \pi$	$V_2(1) = \pi R^2$
Lab 3	$V_3(1) = \frac{2\pi}{3}(2) = \frac{4}{3}\pi$	$V_3(1) = \frac{4}{3}\pi R^3$
Lab 4	$V_4(1) = \frac{2\pi}{4}(\pi) = \frac{\pi^2}{2}$	$V_4(1) = \frac{\pi^2}{2} R^4$

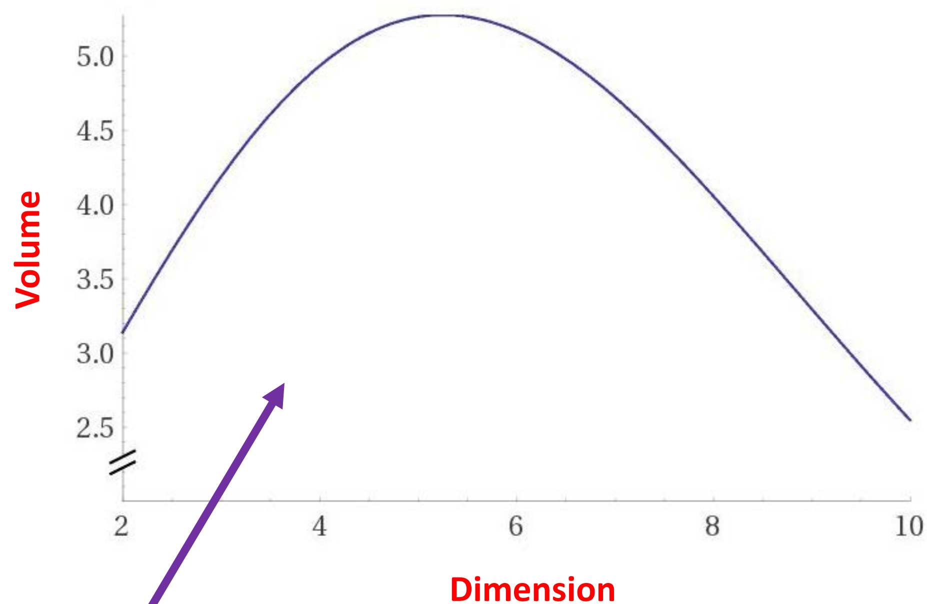
Volume via the Gamma Function

$$n_{even} \rightarrow V_n(R) = \frac{2^{\frac{n}{2}} \pi^{\frac{n}{2}} R^n}{2 \cdot 4 \cdot 6 \cdots n}$$

$$n_{odd} \rightarrow V_n(R) = \frac{2^{\frac{n+1}{2}} \pi^{\frac{n-1}{2}} R^n}{1 \cdot 3 \cdot 5 \cdots n}$$

$$\boxed{V_n(R) = \frac{\pi^{\frac{n}{2}} R^n}{\Gamma\left(\frac{n}{2} + 1\right)}}$$

Volume of Unit Hypersphere



Because we can evaluate $\boldsymbol{\Gamma(x)}$ at every point in $\mathbb{R}$ we can now determine the volume of a unit hypersphere in *any* dimension

Volume via the Gamma Function

$$V_n(R) = \frac{\pi^{\frac{n}{2}} R^n}{\Gamma\left(\frac{n}{2} + 1\right)}$$

$\Gamma(n) = (n-1)!$

$n! = \Gamma(n+1)$

$$V_2(R) = \frac{\pi R^2}{\Gamma\left(\frac{2}{2} + 1\right)} = \frac{\pi R^2}{\Gamma(2)} = \frac{\pi R^2}{(2-1)!} = \boxed{\pi R^2}$$

$$V_3(R) = \frac{\pi^{\frac{3}{2}} R^3}{\Gamma\left(\frac{3}{2} + 1\right)} = \frac{\pi R^3}{\Gamma\left(\frac{5}{3}\right)} = \frac{\pi^{\frac{3}{2}} R^3}{\left(\frac{3\sqrt{\pi}}{4}\right)} = \pi^{\frac{3}{2}} R^3 \left(\frac{4}{3\sqrt{\pi}}\right) = \boxed{\frac{4}{3}\pi R^3}$$

$$V_4(R) = \frac{\pi^{\frac{4}{2}} R^4}{\Gamma\left(\frac{4}{2} + 1\right)} = \frac{\pi^2 R^2}{\Gamma(3)} = \frac{\pi^2 R^2}{(3-1)!} = \boxed{\frac{\pi^2 R^4}{2}}$$

Challenges of Numerical Analysis

$$V_{12}(1) = \frac{\pi^{\frac{12}{2}} 1^{12}}{\Gamma\left(\frac{12}{2}+1\right)} = \boxed{1.335262}$$

Did we use enough dots?

mc-highdimensional

File Edit View Terminal Tabs Help

```
 2, 3.1416
 3, 4.1888
 4, 4.9348
 5, 5.2630
 6, 5.1692
 7, 4.7208
 8, 4.0605
 9, 3.2935
10, 2.5463
11, 1.8883
12, 1.3443

Process returned 0 (0x0)
Press ENTER to continue.
```

From Lab 5

Curse of dimensionality

The curse of dimensionality refers to various phenomena that arise when analyzing and organizing data in high-dimensional spaces (often with hundreds or thousands of dimensions) that do not occur in low-dimensional settings such as the three-dimensional physical space of everyday experience. The expression was coined by Richard E. Bellman when considering problems in dynamic optimization.[1][2]

There are multiple phenomena referred to by this name in domains such as numerical analysis, sampling, combinatorics, machine learning, data mining, and databases. The common theme of these problems is that when the dimensionality increases, the volume of the space increases so fast that the available data become sparse. This sparsity is problematic for any method that requires statistical significance. In order to obtain a statistically sound and reliable result, the amount of data needed to support the result often grows exponentially with the dimensionality.

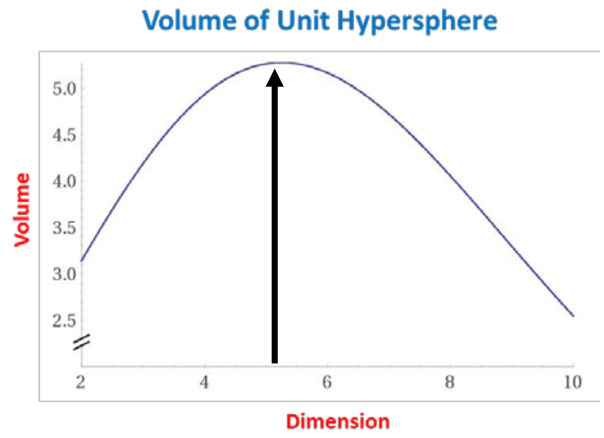
Volume via the Gamma Function

$$V_n(R) = \frac{\pi^{\frac{n}{2}} R^n}{\Gamma\left(\frac{n}{2} + 1\right)}$$

As the Gamma function can extends its domain to include $\boldsymbol{n} \in \mathbb{R}$, we can use this analytic solution to compute the volume of hyperspheres having **fractional** (non-integer) dimensions!

$$V_{\mathbf{7.89}}(5.12) = \frac{\pi^{\frac{7.89}{2}} 5.12^{7.89}}{\Gamma\left(\frac{7.89}{2} + 1\right)} = \mathbf{1{,}633{,}106.2809}$$

It appears $V_n(1)$ has a maximum somewhere *between* **4** and **6** dimensions. Does a <u>fractional</u> dimension contain the largest unit hypersphere?



Open Lab 6 – nball-volume

```
// Euler's Gamma integrand
inline double f(double x, double s)
{
    return pow(x, s - 1) * exp(-x);
}

// Find Gamma using Simpson's integration
double gamma(double s)
{
    double a{0};
    double b{1e3};
    int intervals = 1e5;

    double dx{(b - a) / intervals};
    double sum{f(a, s) + f(b, s)};
    a += dx;
    for (int i{1}; i < intervals; ++i, a += dx)
        sum += f(a, s) * (2 * (i % 2 + 1));
    return (dx / 3) * sum;
}

// Find volume of unit ball
// See https://en.wikipedia.org/wiki/N-sphere
double v(double x)
{
    double halfx = x / 2.0;
    return pow(M_PI, halfx) / gamma(halfx + 1)
}
```

- This code uses Simpson's Rule to calculate Euler's Gamma function:

$$\Gamma(s) = \int_0^\infty \boxed{x^{s-1} e^{-x}\, dx}$$

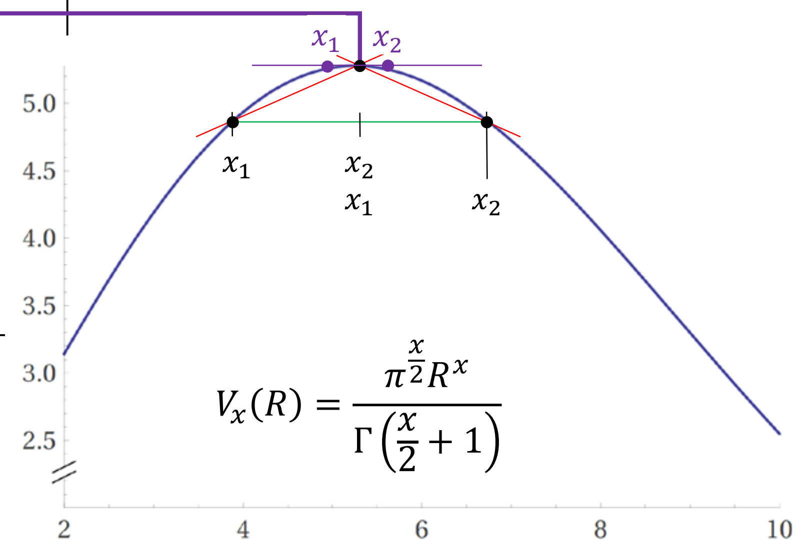
- The code uses Gamma to find the volume of **unit** hyperspheres with **fractional** dimensions

$$\boxed{V_x(1) = \frac{\pi^{\frac{x}{2}}}{\Gamma\left(\frac{x}{2}+1\right)}}$$

View Lab 6 – nball-volume

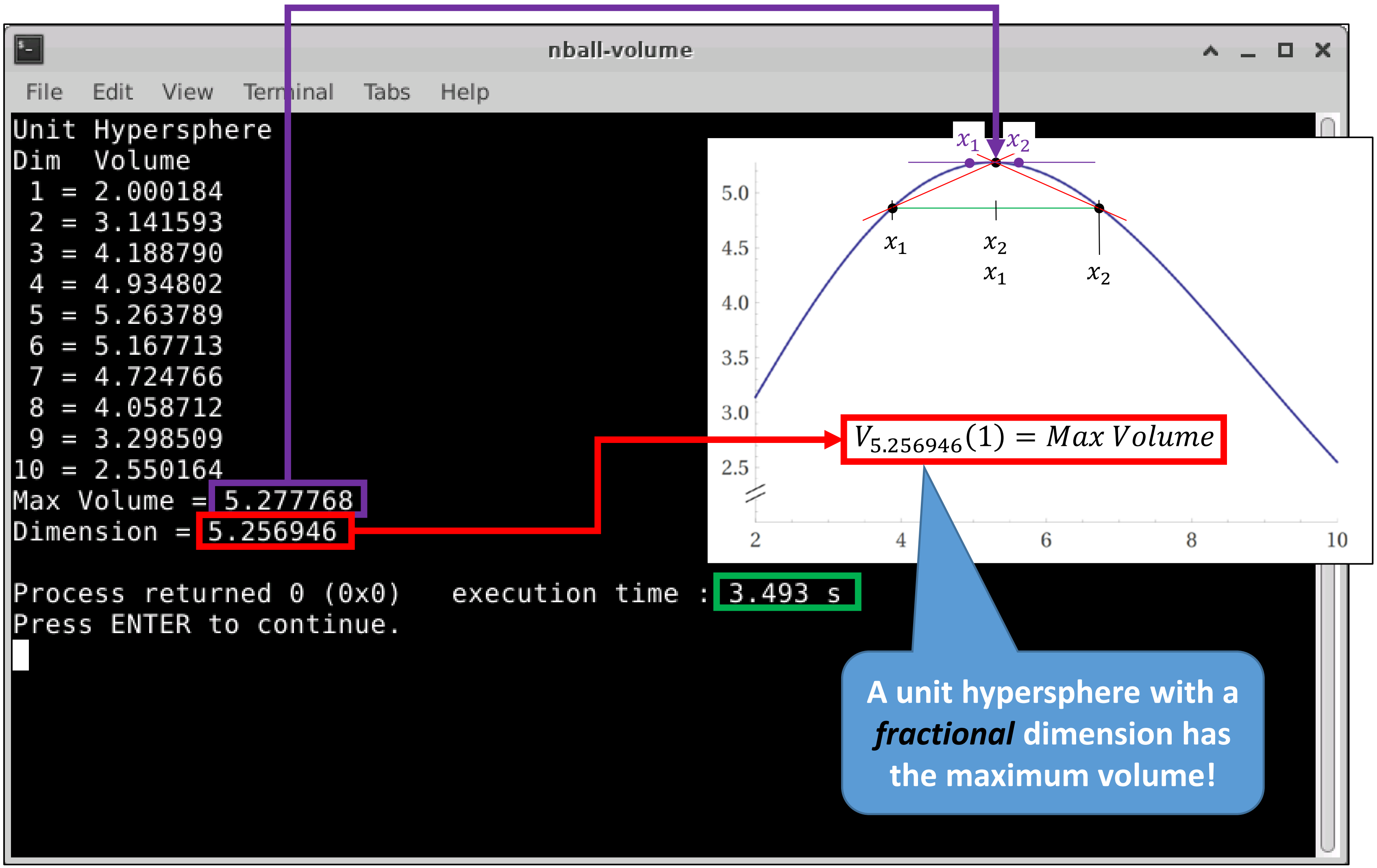
```cpp
// This returns the x and y values for the maximum value of a function
// when both the analytic & numerical first derivative are unavailable.
// Note: This only works for concave down (negative second derivative) functions
tuple<double, double> find_max(double x1, double x2, double epsilon)
{
    double max = numeric_limits<double>::min();
    while (true)
    {
        double dx = (x2 - x1) / 97;
        double y1 = v(x1);
        double y2 = v(x1 + dx);
        while (y1 < y2)
        {
            x1 = x1 + dx;
            y1 = v(x1);
            y2 = v(x1 + dx);
        }
        if (abs(max - y1) < epsilon)
            break;
        x1 = x1 - dx;
        x2 = x1 + dx;
        max = v(x1);
    }
    return tuple<double, double>(x1, v((x1 + x2) / 2));
}
```

A **tuple<>** is a compound type that can hold **two** data items $\langle x_{max} | y_{max} \rangle$



This ***hill climbing*** algorithm steps across the curve in **decreasing** brackets until the maximum point is located

Run Lab 6 – nball-volume



A unit hypersphere with a ***fractional*** **dimension has the maximum volume!**

Edit Lab 6 – nball-volume

```
26 // Find volume of unit ball
27 // See https://en.wikipedia.org/wiki/N-sphere
28 double v(double x)
29 {
30     double halfx = x / 2.0;
31     return pow(M_PI, halfx) / tgamma(halfx + 1);
32 }
```

Edit line # 33 to call the C++ tgamma() function "the true gamma"

Run Lab 6 – nball-volume

nball-volume

File Edit View Terminal Tabs Help

```
Unit Hypersphere
Dim  Volume
 1 = 2.000000
 2 = 3.141593
 3 = 4.188790
 4 = 4.934802
 5 = 5.263789
 6 = 5.167713
 7 = 4.724766
 8 = 4.058712
 9 = 3.298509
10 = 2.550164
Max Volume = 5.277768
Dimension = 5.256946

Process returned 0 (0x0)   execution time : 0.030 s
Press ENTER to continue.
```

```
26 // Find volume of unit ball
27 // See https://en.wikipedia.org/wiki/N-sphere
28 double v(double x)
29 {
30     double halfx = x / 2.0;
31     return pow(M_PI, halfx) / tgamma(halfx + 1);
32 }
```

Edit line # 33 to call the C++ tgamma() function "the true gamma"

The built-in **tgamma()** is **11,500%** faster than our Simpson's integration!

Euler's Gamma Function

https://dlmf.nist.gov/5

Gamma gets its own chapter!

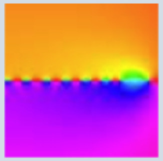
Digital Library of Mathematical Functions

Index
Notations
Search
Help?
Citing
Customize
Annotate

About the Project
NIST National Institute of Standards and Technology U.S. Department of Commerce

Chapter 5 Gamma Function

R. A. Askey
Department of Mathematics, University of Wisconsin, Madison, Wisconsin.
R. Roy
Department of Mathematics and Computer Science, Beloit College, Beloit, Wisconsin.

$$\Gamma(x)\Gamma(1-x) = \frac{\pi}{\sin \pi x}$$

In C++ use tgamma()

The Power Of Monte Carlo Integration

$$
\begin{aligned}
\mathbf{F}^{(n)} = {} & \frac{\mu}{8\pi} \int_{r_1}^{r_2} \int_{s_1}^{s_2} \int_{y_1}^{y_2} \left(\frac{2}{R_a^3} + \frac{3a^2}{R_a^5} \right) \{ (\mathbf{R} \times \mathbf{b}) (\mathbf{t} \cdot \mathbf{n}) + \mathbf{t} [(\mathbf{R} \times \mathbf{b}) \cdot \mathbf{n}] \} \\
& \times \left(\frac{r - r_1}{r_2 - r_1} \frac{s - s_1}{s_2 - s_1} \right) \mathrm{d}s \, \mathrm{d}r \, \mathrm{d}y \\
& - \frac{\mu}{4\pi (1 - \nu)} \int_{r_1}^{r_2} \int_{s_1}^{s_2} \int_{y_1}^{y_2} \left(\frac{1}{R_a^3} + \frac{3a^2}{R_a^5} \right) [(\mathbf{R} \times \mathbf{b}) \cdot \mathbf{t}] \, \mathbf{n} \left(\frac{r - r_1}{r_2 - r_1} \frac{s - s_1}{s_2 - s_1} \right) \mathrm{d}s \, \mathrm{d}r \, \mathrm{d}y \\
& + \frac{\mu}{4\pi (1 - \nu)} \int_{r_1}^{r_2} \int_{s_1}^{s_2} \int_{y_1}^{y_2} \frac{1}{R_a^3} \{ (\mathbf{b} \times \mathbf{t}) (\mathbf{R} \cdot \mathbf{n}) + \mathbf{R} [(\mathbf{b} \times \mathbf{t}) \cdot \mathbf{n}] \} \left(\frac{r - r_1}{r_2 - r_1} \frac{s - s_1}{s_2 - s_1} \right) \mathrm{d}s \, \mathrm{d}r \, \mathrm{d}y \\
& - \frac{\mu}{4\pi (1 - \nu)} \int_{r_1}^{r_2} \int_{s_1}^{s_2} \int_{y_1}^{y_2} \frac{3}{R_a^5} [(\mathbf{R} \times \mathbf{b}) \cdot \mathbf{t}] (\mathbf{R} \cdot \mathbf{n}) \, \mathbf{R} \left(\frac{r - r_1}{r_2 - r_1} \frac{s - s_1}{s_2 - s_1} \right) \mathrm{d}s \, \mathrm{d}r \, \mathrm{d}y .
\end{aligned}
$$

Now you know…

- Monte Carlo integration uses **random sampling**
  - The method calculates the ratio of the points below the curve to the total number of points – **the final ratio is the "area"**
  - It may require <u>billions of samples</u> to provide a reasonable estimate
  - It may be the *only way* to take the integral of a very complex function
- What you are taught cannot be the limit of your knowledge
  - The volume of a 4-D unit hypersphere $= \boldsymbol{\frac{\pi^2}{2}}$
  - In infinite dimensions the volume of **all** hyperspheres is zero!
  - A <u>*fractional*</u> **5-dimensional** unit sphere has **maximum** volume
  - Mother Nature *never* said dimensions have to be integers!