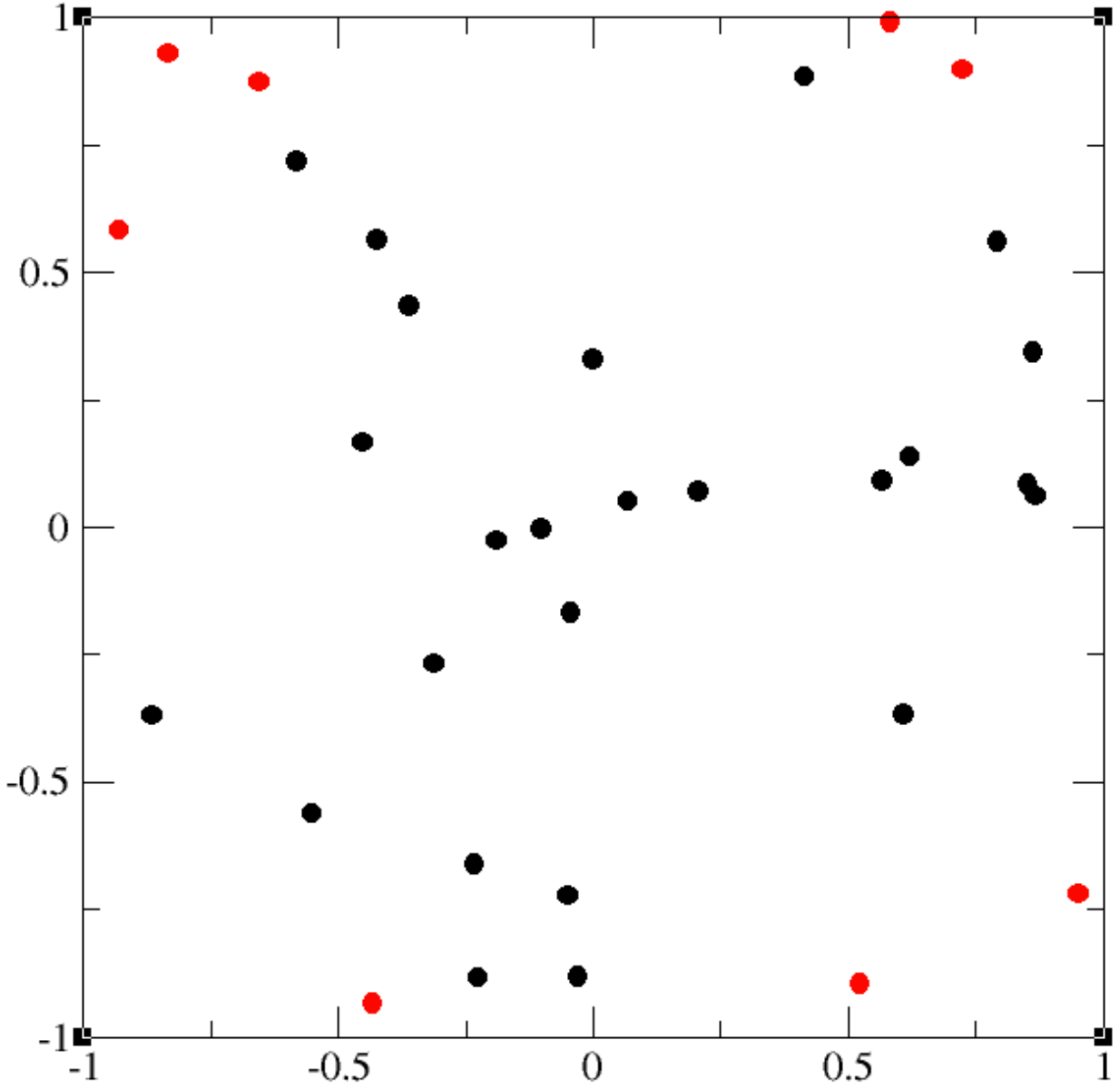
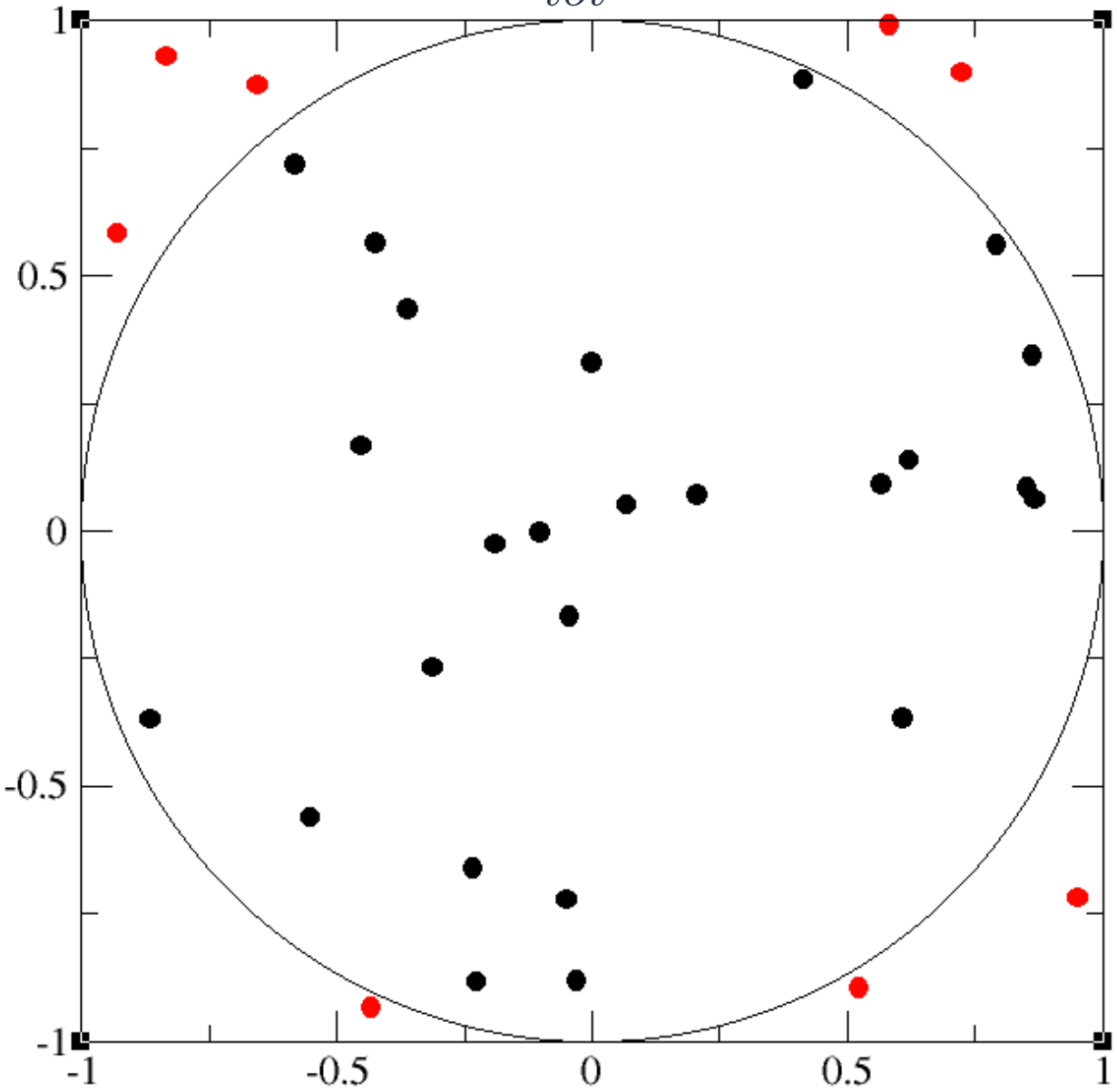
PHY-4810
COMPUTATIONAL PHYSICS

LECTURE 4: INTRODUCTION TO RANDOM NUMBERS

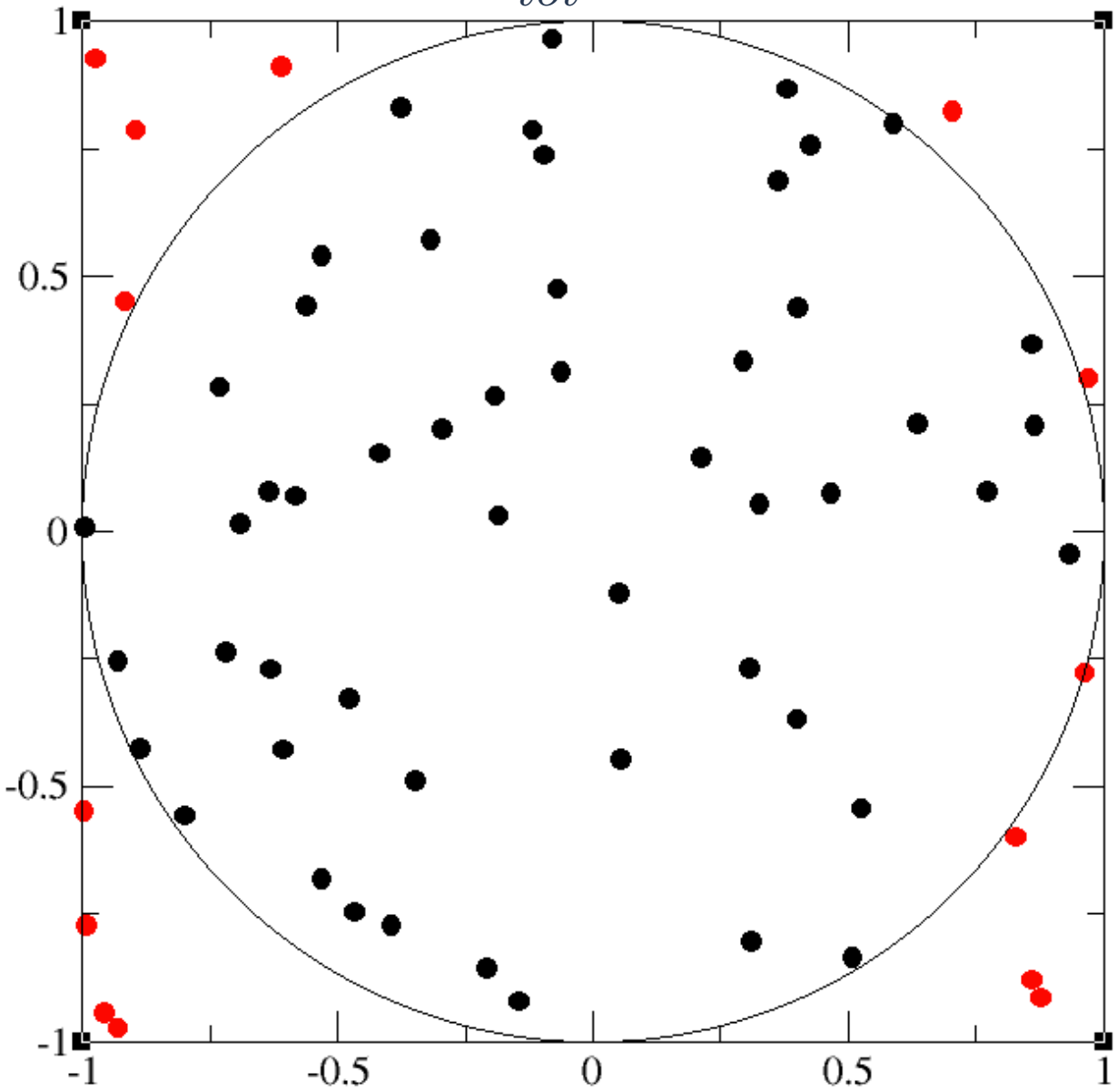


$x^2 + y^2 < 1 \rightarrow p \in in$ $x^2 + y^2 > 1 \rightarrow p \in out$

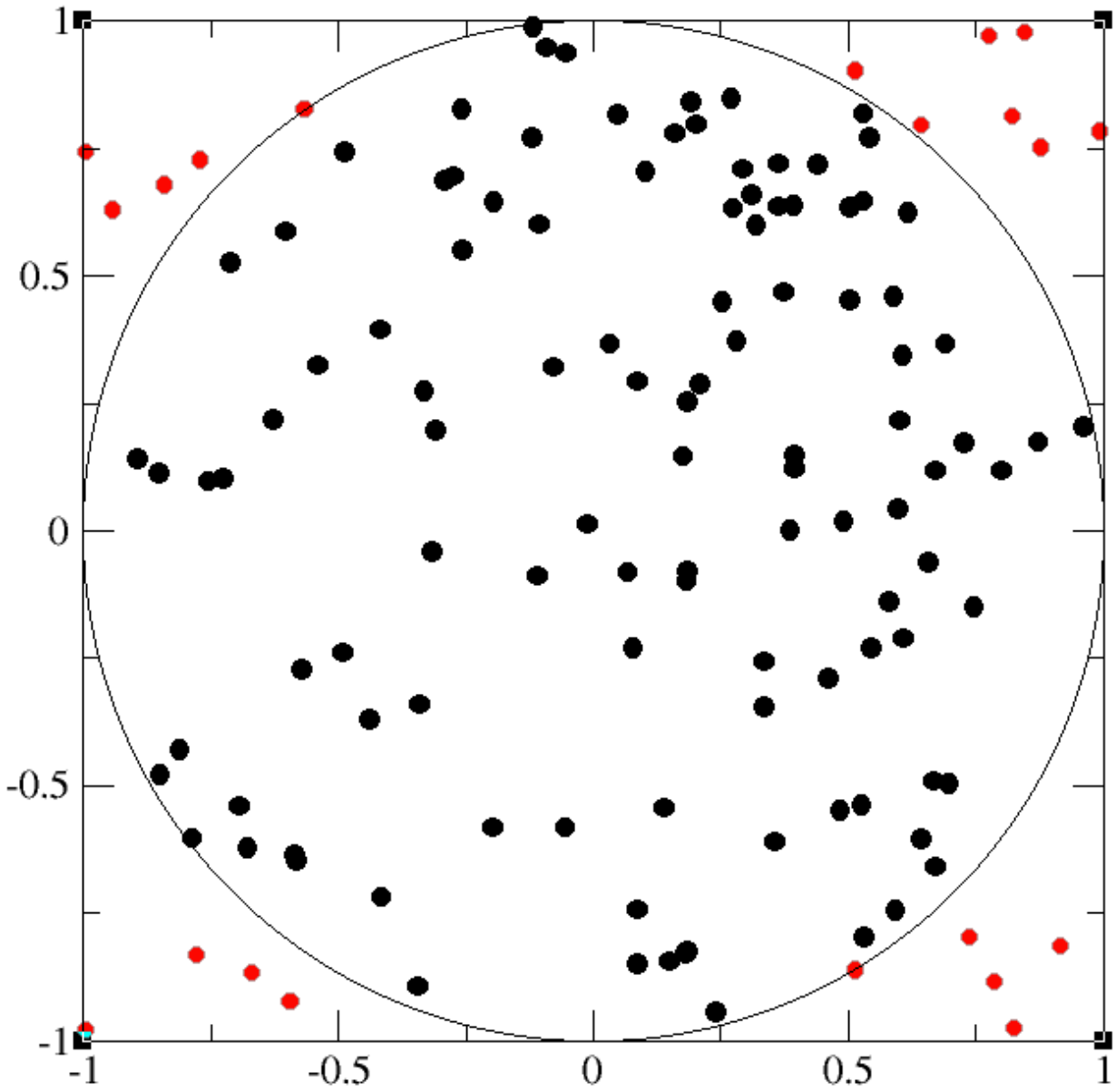
$$4 \times \frac{n_{in}}{n_{tot}} = 2.94$$



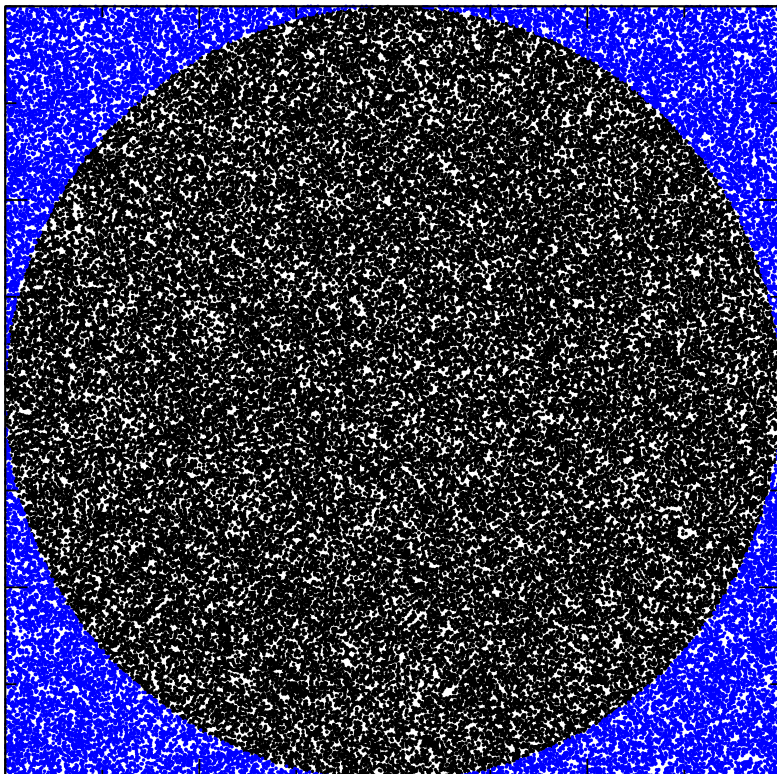
$$4 \times \frac{n_{in}}{n_{tot}} = 3.10$$



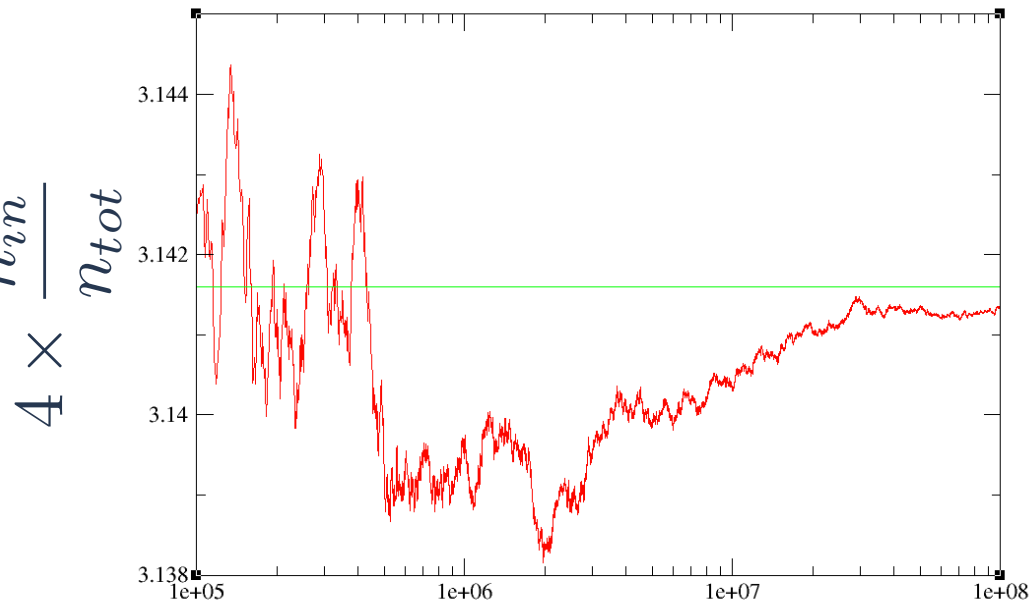
$$4 \times \frac{n_{in}}{n_{tot}} = 3.34$$



$$4 \times \frac{n_{in}}{n_{tot}} \approx 4 \times \frac{\pi R^2}{(2R)^2} = \pi$$



$$4 \times \frac{n_{in}}{n_{tot}} = 3.143$$



Number to trials

RANDOM NUMBER AND CONSTANT OF NATURE

- **By taking clever advantage of the uniform property of random distributions, we can make them do some very interesting things for us. For instance, since we know that an integral is nothing more than the area under a curve, we can find the area of that curve by using random numbers. By picking a large number of random numbers, and counting how many lie under the curve we are interested in, we can compare the number of hits to the number of total points to estimate the area.**

- **One common example of this technique is finding the area of a unit circle by placing random points within a square that encompasses the circle, and counting the number of points that end up inside the circle. By using the calculated area, and using our formula for the area of a circle $A = \pi r2$, we can find π by using random numbers and a little geometry.**

ANOTHER EXAMPLE (CREDIT TO CHRIS PERSICHILLI, 2012)

- We can also calculate another fundamental mathematical constant, Euler's number e, in a similar fashion.
- We do this by examining the curve 1/x in the interval 1 < x < 2. We can use very much the same technique, using random numbers to find the area under the curve, to calculate the integral.
- We know from elementary calculus that the integral has the exact value

$$\int_1^2 \frac{1}{x}\, dx = \ln(2)$$

FINDING e

- If we let A be the area that we calculated in out random number integration so that

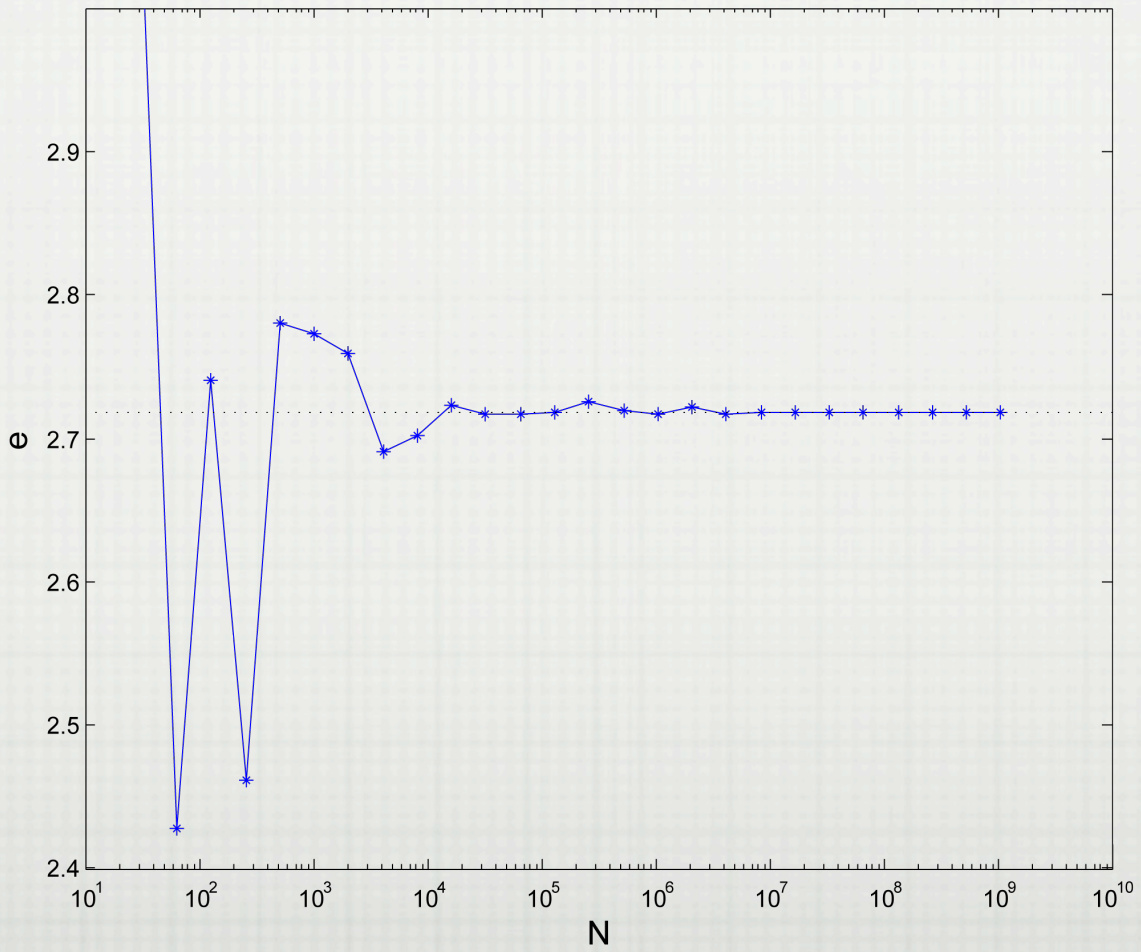
$$A \approx \ln(2)$$

- We define e as the base of the natural logarithm, so that

$$e^A = 2$$

$$\boxed{e = 2^{1/A}}$$

FINDING e



ON THE MENU TODAY

- ☐ **Random Numbers: what are they?**
- ☐ **How do computers *generate* random numbers?**
- ☐ **How can you define and assess the *quality* of random numbers?**

RANDOM SEQUENCES: DEFINITION

- We define a sequence of numbers r_1, r_2, . . . as random if there are **no correlations** among the numbers in the sequence.
- Yet randomness does not necessarily mean all numbers in the sequence are **equally likely** to occur.
- If all numbers in a sequence are equally likely to occur, then the sequence is **uniform**.

RANDOMNESS

- To illustrate, 1, 2, 3, 4, . . . is uniform but probably not random, while 3, 1, 4, 2, 3, 1, 3, 2, 4, . . . may be random but does not appear to be uniform.
- In addition, it is possible to have a sequence of numbers that, in some sense, are random but have very short range correlations, for example, $r_1, (1 - r_1), r_2, (1 - r_2), r_3, (1 - r_3) \cdot \cdot \cdot$.

RANDOMNESS

- **Mathematically, the likelihood of a random number occurring is described by a distribution function *P(r)*.**
  - **This means the probability of finding r_i in the interval *[r, r + dr] is P(r)dr.***
  - **Uniform means *P(r)* is constant**
- ***The standard random-number generator on computers generates uniform distributions (P = 1) between 0 and 1***
- **Numbers can also be generated non-uniformly and still be random**

(PSEUDO) PHILOSOPHICAL QUESTION

Computers are deterministic, therefore how could they produce true randomness?

ANSWER?

- **They can't...**

PSEUDO-RANDOM NUMBERS

- **Determinism:**
  - **Although it may be a bit of work, if we know r_m and its preceding elements, it is always possible to figure out r_{m+1}.**
  - **For this reason, computers generate "pseudo"-random numbers.**

THE LINEAR CONGRUENT OR POWER RESIDUE METHOD TO GENERATE RANDOM NUMBERS

- Pick a seed r_1, two constants *a* and *c*, and an interval [0,M-1]
- To generate a pseudo-random sequence of numbers $\{r_1, r_2, \ldots, r_k\}$ over interval [0, M – 1]:
  - Multiply the previous random number r_{i-1} by the constant *a*
  - Add on another constant *c*
  - Take the modulus by *M* as the next random number r_i

$$r_i \overset{\text{def}}{=} (a\, r_{i-1} + c) \bmod M$$

  - The value for r_1 (the seed) is supplied by the user
- Where is randomness built in?
  - Answer: Randomness is built in the roundoff error produced during mod

EXAMPLE

- if c=1, a=4, M=9, and you supply r_1 =2, then you obtain the sequence
- $r_1 = 2$
  - $r_2 = (4 \times 2 + 1) \bmod 9 = 0$
  - $r_3 = (4 \times 0 + 1) \bmod 9 = 1$
  - $r_4 = (4 \times 1 + 1) \bmod 9 = 5$
  - $r_5 = (4 \times 5 + 1) \bmod 9 = 3$
  - $r_6 = 4$
  - $r_7 = 8$
  - $r_8 = 6$
  - $r_9 = 7$
- In the normalized [0,1] range: 0.00, 0.111, 0.555,...

$$r_i \overset{\text{def}}{=} (a\, r_{i-1} + c) \bmod M$$

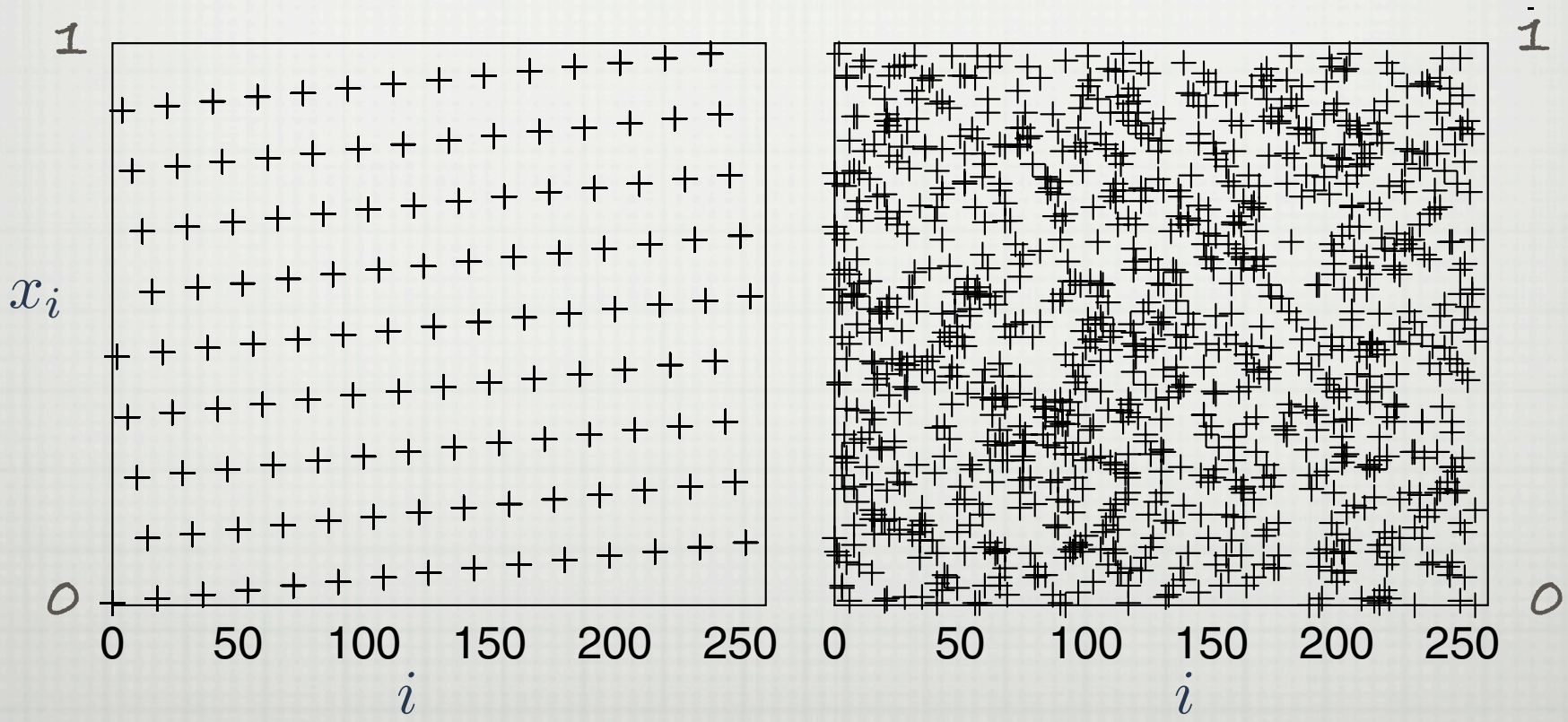
- The rule produces integers in the range *[0, M − 1]*, but not necessarily every integer.
- When a particular integer comes up a second time...
  - ...the whole cycle repeats.
- In order to obtain a longer sequence, *a* and *M* should be large numbers (and avoiding overflows!)

- Your computer probably has random-number generators that are better than the one you will compute with the power residue method.

- You may check this out in the manual or help pages and then test the generated sequence.

ASSESSING RANDOMNESS AND UNIFORMITY: 1. PLOT x_i AS A FUNCTION OF i.

Observe how there appears to be a uniform distribution between 0 and 1 and no particular correlation between points



ASSESSING RANDOMNESS AND UNIFORMITY: 2. EVALUATE THE K_{TH} MOMENT OF THE RANDOM-NUMBER DISTRIBUTION

$$\frac{1}{N}\sum_{i=1}^{N} x_i^k \simeq \int_0^1 dx\, x^k P(x) + O(1/\sqrt{N}) \simeq \frac{1}{k+1}$$

PLOT THIS AS A FUNCTION OF K:

$$\sqrt{N}\left|\frac{1}{N}\sum_{i=1}^{N} x_i^k - \frac{1}{k+1}\right|$$

ASSESSING RANDOMNESS AND UNIFORMITY: 3. DETERMINES THE NEAR-NEIGHBOR CORRELATION

$$C(k) = \frac{1}{N}\sum_{i=1}^{N} x_i x_{i+k} \qquad (k = 1, 2, \ldots)$$

$$\frac{1}{N}\sum_{i=1}^{N} x_i\, x_{i+k} \;\simeq\; \int_0^1 dx \int_0^1 dy\, xy\, P(x,y) = \tfrac{1}{4}$$

ASSESSING RANDOMNESS AND UNIFORMITY: 4. PREPARE A SCATTER PLOT

- **Prepare a scatter plot of ($x_i = r_{2i}$, $y_i = r_{2i+1}$) for many i values. If your points have noticeable regularity, the sequence is not random. If the points are random, they should uniformly fill a square with no discernible pattern.**

ASSESSING RANDOMNESS AND UNIFORMITY: MORE TESTS

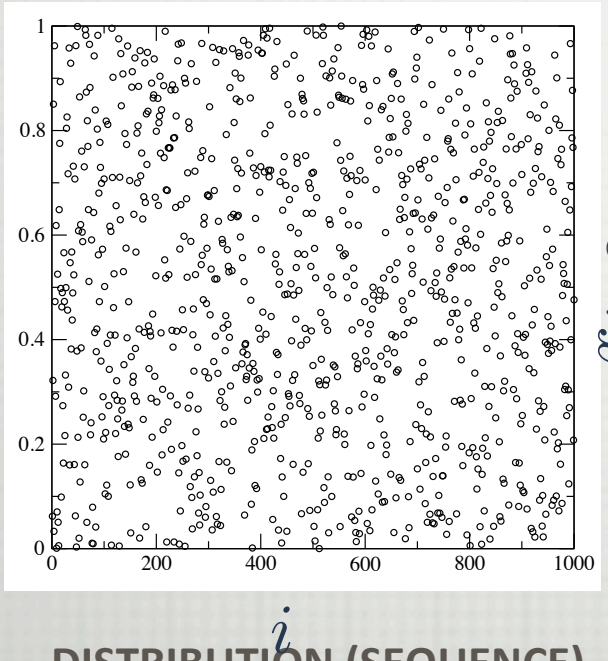
- **Run your calculation or simulation with the sequence r_1, r_2, r_3,..., and then again with the sequence $(1-r_1)$, $(1-r_2)$, $(1- r_3)$, ... Because both sequences should be random, if your results differ beyond statistics, then your sequence is probably not random.**
- **Yet another test is to run your simulation with a sequence of true random numbers from a table and compare it to the results with the pseudo- random-number generator. In order to be practical, you may need to reduce the number of trials being made.**

ASSESSING RANDOMNESS AND UNIFORMITY

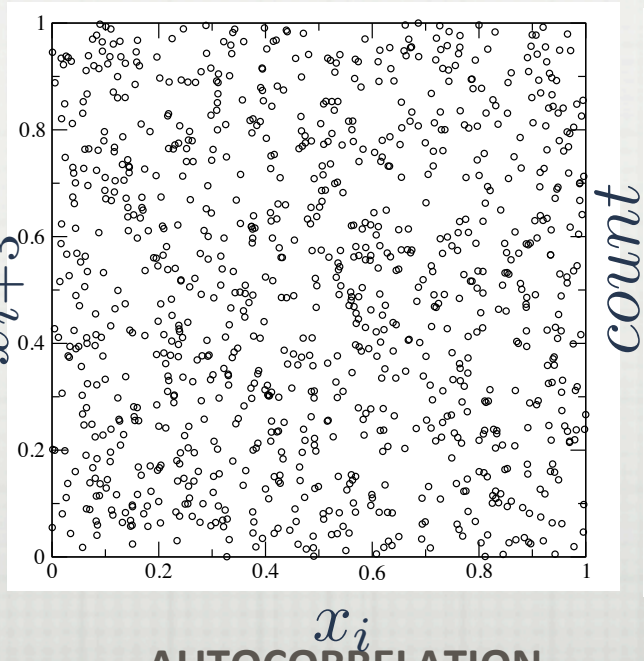
- 1000 points (fortran built in generator)
- 4th moment: 0.196873 (0.2)
- Autocorrelation Test: 0.245859 (0.25)

$$\frac{1}{N}\sum_{i=1}^{N} x_i^k \approx \frac{1}{k+1}$$

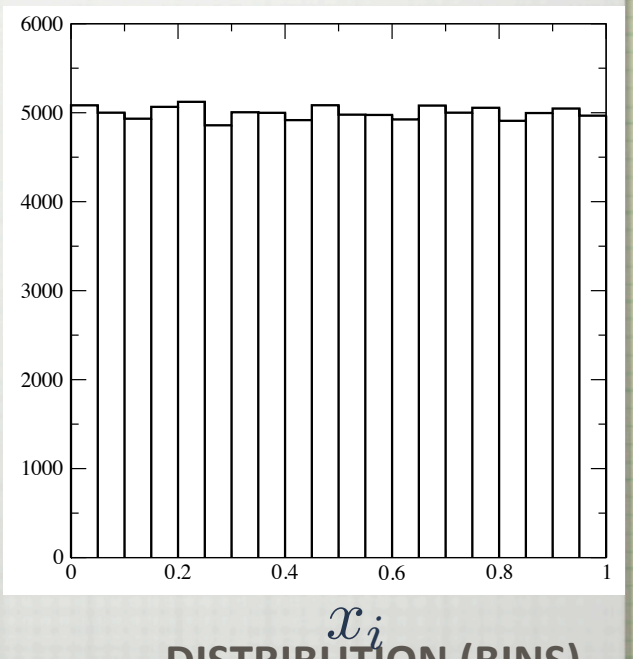
$$C(k) = \frac{1}{N}\sum_{i=1}^{N} x_i x_{i+k} \approx \frac{1}{4}$$



DISTRIBUTION (SEQUENCE)



AUTOCORRELATION



DISTRIBUTION (BINS)

HOW TO USE COMPUTER RNG?

- The sequence of numbers returned by rand() are called random because they satisfy statistical tests for randomness, e.g, uniform distribution, correlation between sequential numbers is zero, no apparent patterns). But of course they really can't be truly random (whatever that means) because computers are deterministic. Therefore they are more properly called pseudorandom numbers.

- **For a given seed (starting value), the sequence of numbers that rand() returns will always be the same.** Because the starting point for the pseudorandom sequence can easily be varied and because the sequence is very long (perhaps billions before the sequence repeats), these pseudorandom numbers are as good as random.

- Having the same sequence generated each time can be useful for debugging, but it isn't very useful when you're actually using the program. To generate a different random sequence, it's necessary to set the seed that starts the sequence. The srand() function takes a positive integer parameter which tells where to start the sequence.

SAME SEED…SAME SEQUENCE

- Using the time as a seed - **srand(time(0))**
- Use the time() function as follows:
  - srand(time(0)); // Initialize random number generator.
  - at the beginning of the program to initialize the random seed.
  - time(0) returns the integer number of seconds from the system clock. This will almost always be a different value.
- Use the following include files.
  - #include <ctime> // For time()
  - #include <cstdlib> // For srand() and rand()

PHILOSOPHICAL QUESTION

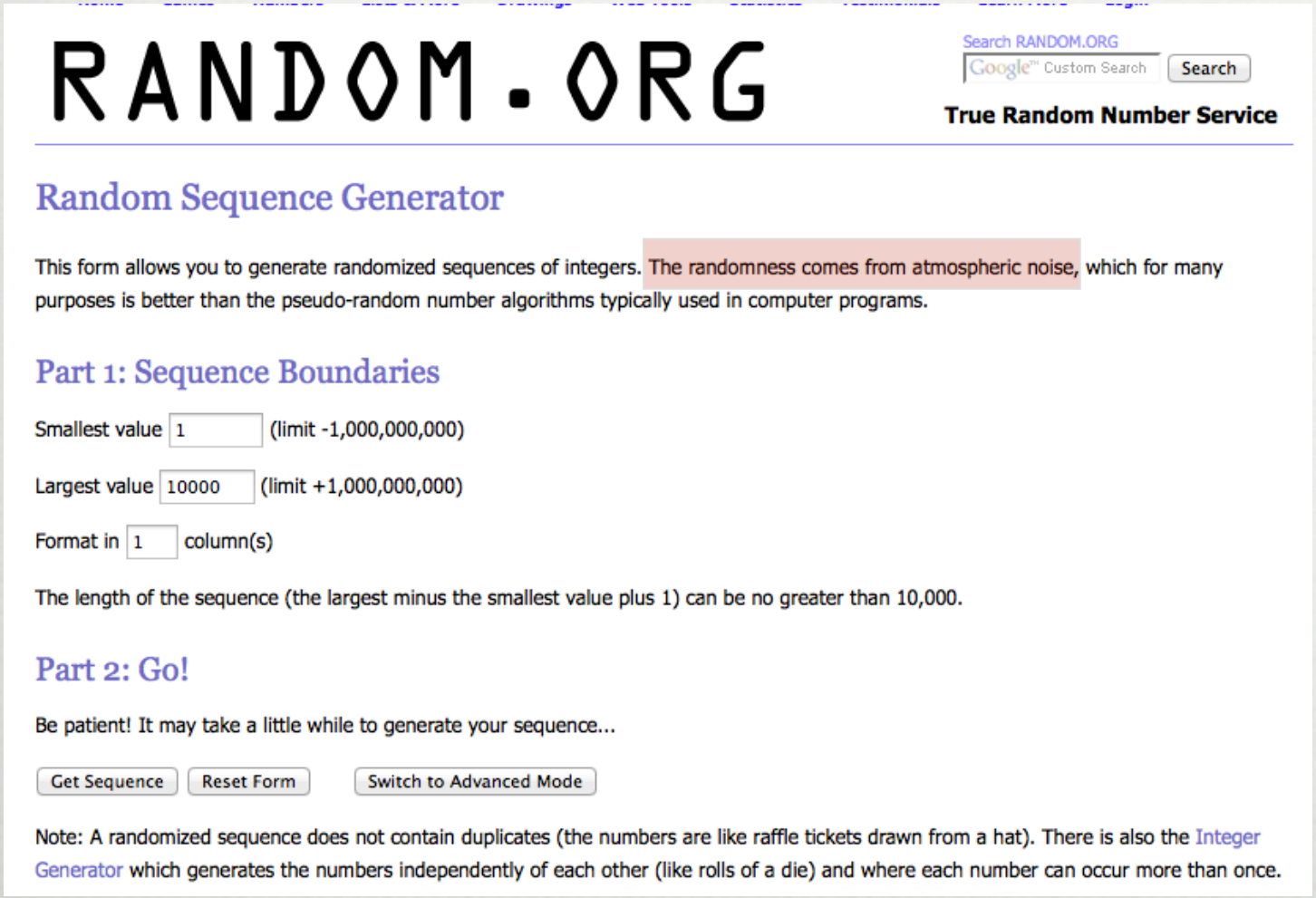
Computers are deterministic, therefore how could they produce true randomness?

TRUE RANDOM NUMBER

- RANDOM.ORG offers true random numbers to anyone on the Internet. The randomness comes from atmospheric noise.
- A physical random number generator can be based on an essentially random atomic or subatomic physical phenomenon whose unpredictability can be traced to the laws of quantum mechanics.
- Sources of entropy include radioactive decay, thermal noise, shot noise, avalanche noise in Zener diodes, clock drift, the timing of actual movements of a hard disk read/write head, and radio noise.
- Recently, a team at Bar-Ilan University in Israel has been able to create a physical random bit generator at a 300 Gbit/s rate, making it the fastest ever
- Various imaginative ways of collecting this entropic information have been devised. One technique is to run a hash function against a frame of a video stream from an unpredictable source.

EXAMPLE OF NON-PSEUDO DISTRIBUTION

RANDOM.ORG



RANDOM.ORG

Search RANDOM.ORG
Google™ Custom Search Search

True Random Number Service

Random Sequence Generator

This form allows you to generate randomized sequences of integers. The randomness comes from atmospheric noise, which for many purposes is better than the pseudo-random number algorithms typically used in computer programs.

Part 1: Sequence Boundaries

Smallest value 1 (limit -1,000,000,000)

Largest value 10000 (limit +1,000,000,000)

Format in 1 column(s)

The length of the sequence (the largest minus the smallest value plus 1) can be no greater than 10,000.

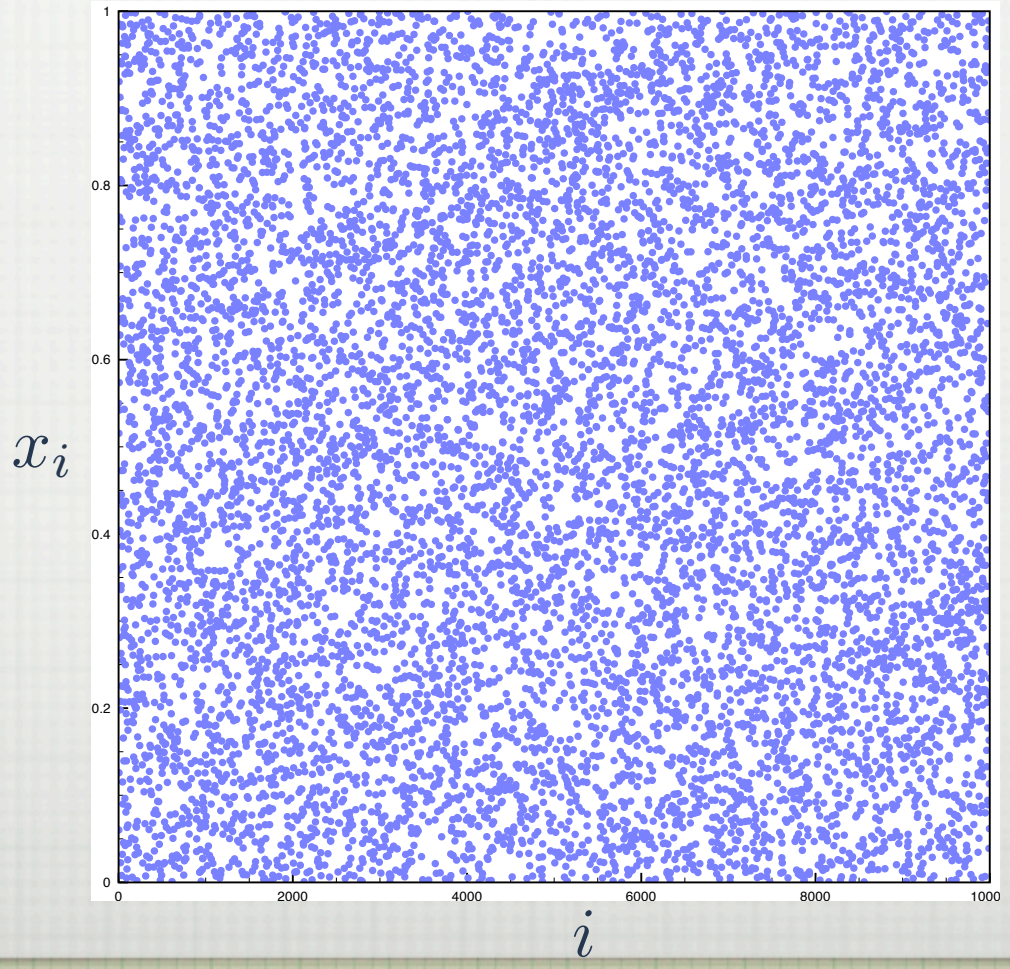
Part 2: Go!

Be patient! It may take a little while to generate your sequence...

Get Sequence Reset Form Switch to Advanced Mode

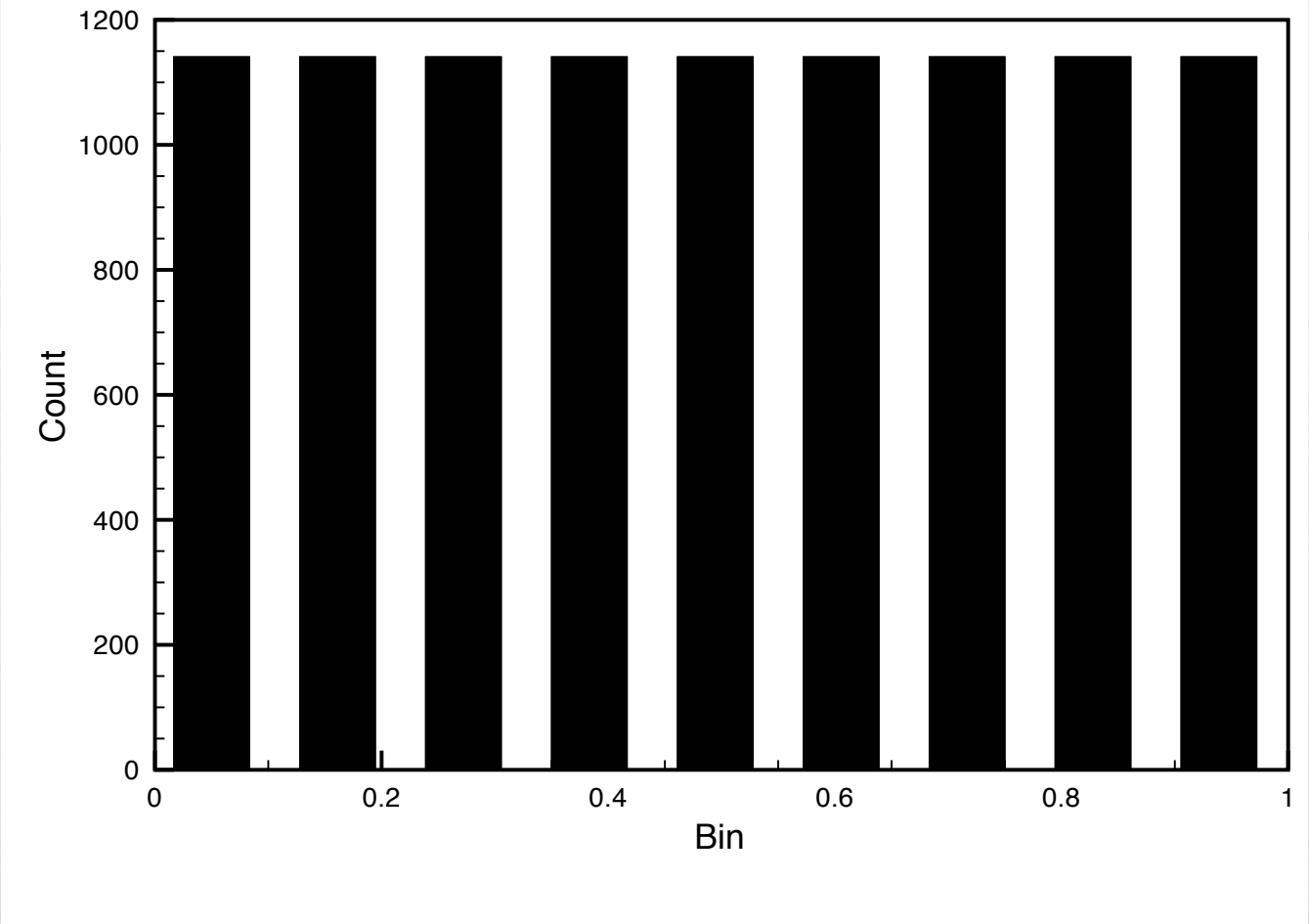
Note: A randomized sequence does not contain duplicates (the numbers are like raffle tickets drawn from a hat). There is also the Integer Generator which generates the numbers independently of each other (like rolls of a die) and where each number can occur more than once.

FIRST TEST: VISUALIZING DISTRIBUTION

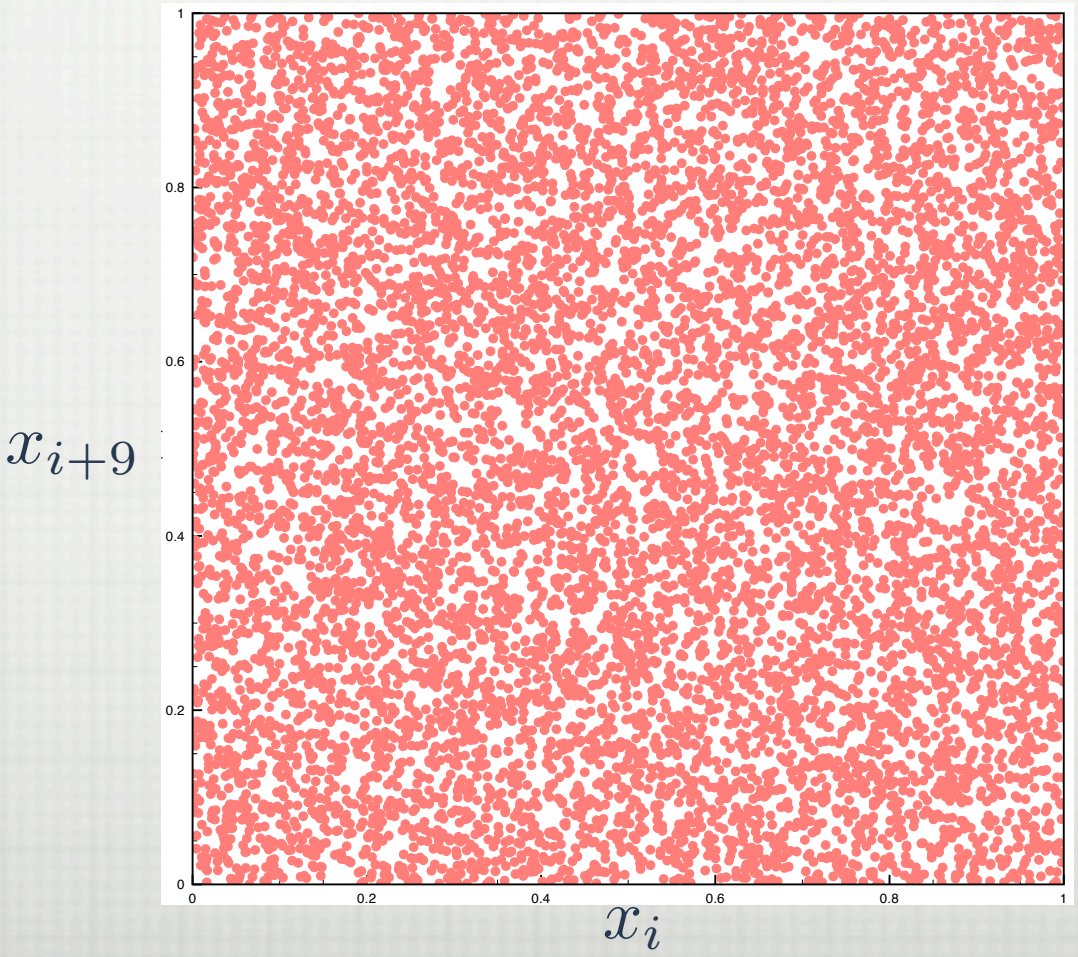


RANDOM.ORG

TEST: UNIFORMITY



THIRD TEST: VISUALIZING AUTOCORRELATION



RANDOM.ORG

NUMERICAL TESTS: MOMENTS

K	NUMERICAL	THEORETICAL
0	1.00000000	1.0000000
1	0.49999999	0.5000000
2	0.33335000	0.3333333
3	0.25002500	0.2500000
4	0.20003000	0.2000000
5	0.16670000	0.1666667
6	0.14289286	0.1428571
7	0.12503750	0.1250000
8	0.11115000	0.11111111
9	0.10004000	0.1000000

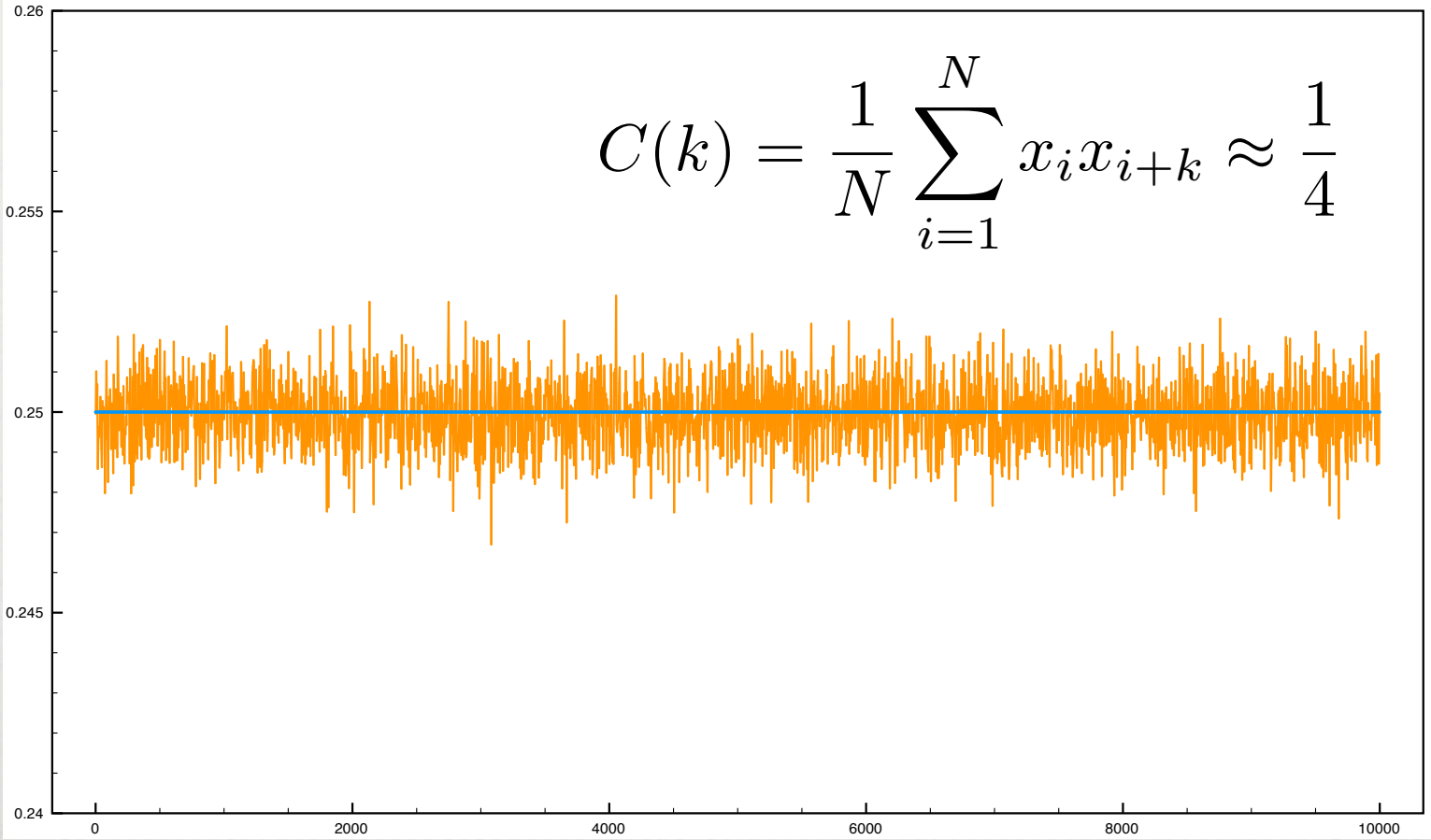
$$\frac{1}{N}\sum_{i=1}^{N} x_i^k \approx \frac{1}{k+1}$$

```
\frac{1}{N}\sum_{i=1}^{N} x_i^k\approx \frac{1}{k+1}
```

RANDOM.ORG

NUMERICAL TEST: AUTOCORRELATIONS

$$C(k) = \frac{1}{N}\sum_{i=1}^{N} x_i x_{i+k} \approx \frac{1}{4}$$



$C(k)$

k

RANDOM.ORG

SUMMARY

- Several computational methods for random number generation exist, but often fall short of the goal of true randomness — though they may meet, with varying success, some of the statistical tests for randomness intended to measure how unpredictable their results are (that is, to what degree their patterns are discernible)

  - Statistical test for randomness:

    - UNIFORMITY

    - CORRELATION

YOUR TURN

- Write a c++ code that generates a distribution from 0 to 1
  - Plot bins, distribution, etc
  - Compute the various tests listed here (moment, correlation)
- Do the same for a distribution you get from random.org

```
    #INCLUDE <TIME.H>

DOUBLE RANDDOUBLE(DOUBLE LOW, DOUBLE HIGH)
{
    DOUBLE TEMP;
    /* CALCULATE THE RANDOM NUMBER & RETURN IT */
    TEMP = ((DOUBLE) RAND() / (STATIC_CAST<DOUBLE>(RAND_MAX) + 1.0))* (HIGH - LOW) + LOW;
    RETURN TEMP;
}

IN YOUR CODE, YOU CAN USE THESE TWO FUNCTIONS; THE FIRST ONE IS FOR INITIALIZING THE SEED,
THE SECOND ONE IS TO GET A NUMBER BETWEEN 0.0 AND 1.0

SRAND( (UNSIGNED)TIME(0) );
RANDDOUBLE(0.0,1.0);
```