Monte Carlo Techniques

Professor Stephen Sekula

Guest Lecture – PHY 4321/7305

Nov. 1, 2013

SMU. | Dedman College of Humanities & Sciences

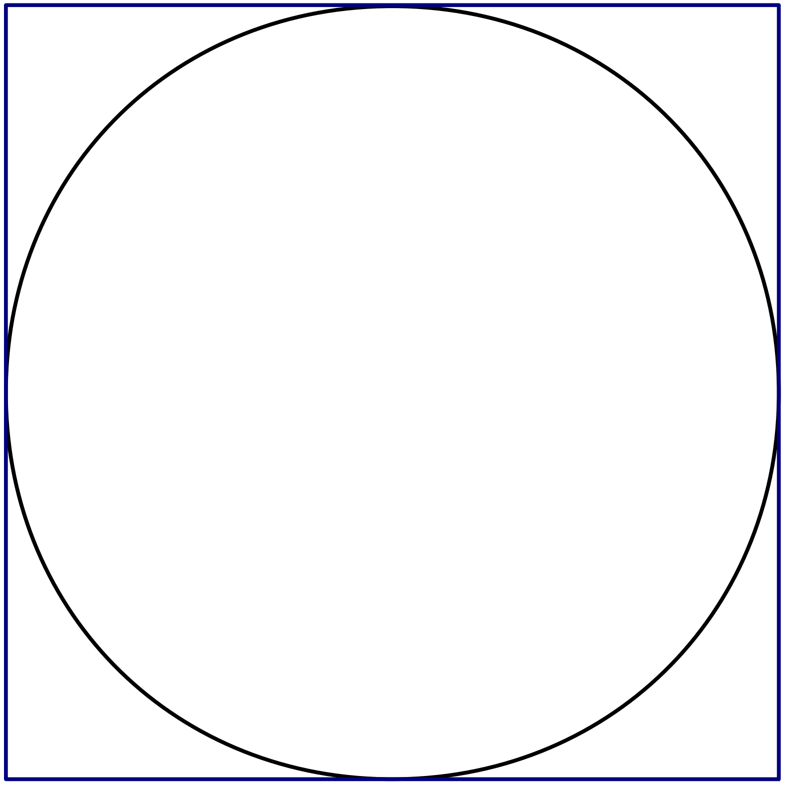
What are "Monte Carlo Techniques"?

- Computational algorithms that rely on repeated random sampling in order to obtain numerical results
- Basically, you run a simulation over and over again to calculate the underlying probabilities that lead to the outcomes
- Like playing a casino game over and over again and recording all the game outcomes to determine the underlying rules of the game
- Monte Carlo is a city famous for its gambling – hence the name of this class of techniques

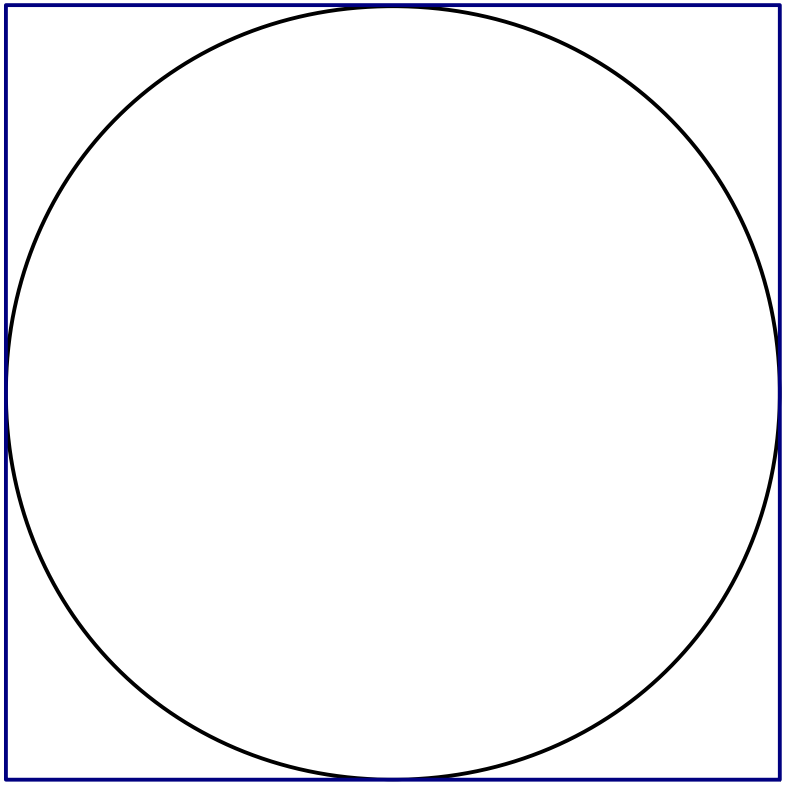
A Simple Physical Example

- Let's illustrate this class of techniques with a simple physical example: numerical computation of π
- π: the ratio of the circumference of a circle to its diameter.
- It's difficult to whip out a measuring tape or ruler and accurately measure the circumference of an arbitrary circle.
- The Monte Carlo method avoids this problem entirely

Begin by drawing a square, inscribed into which is a circle. The properties of the square are much easier to measure.

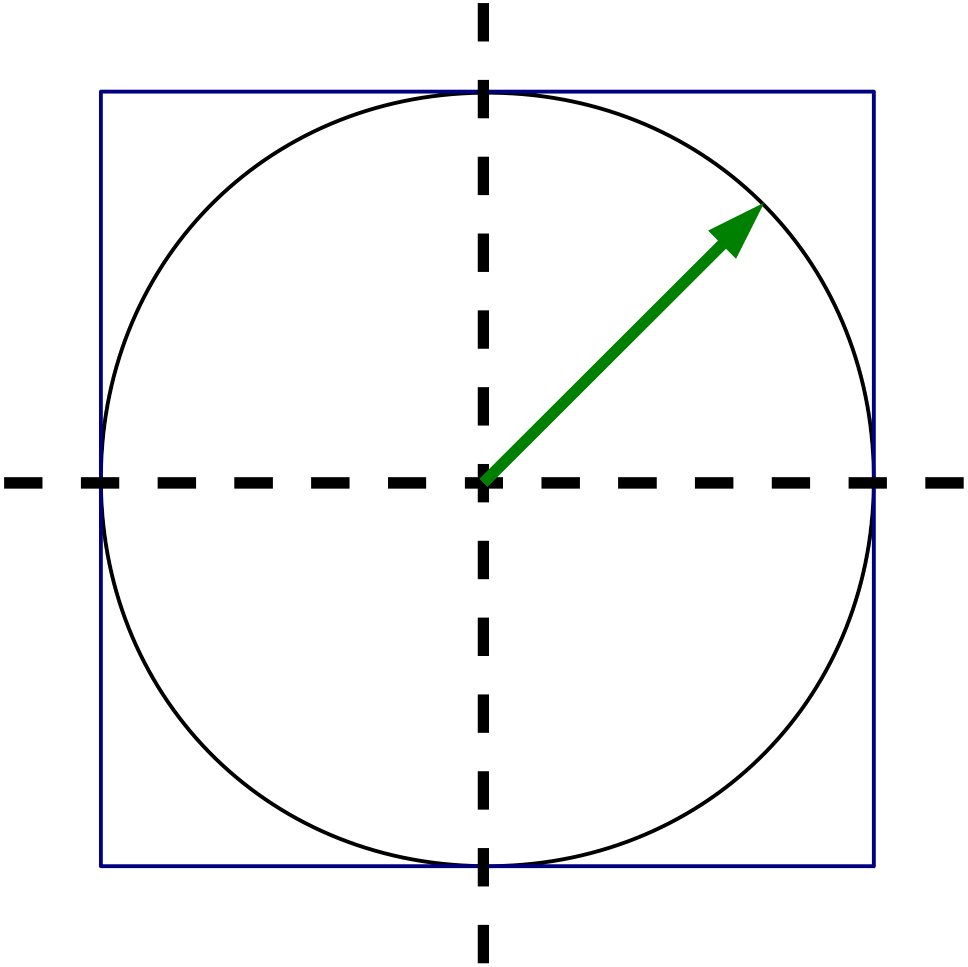


What do we know?

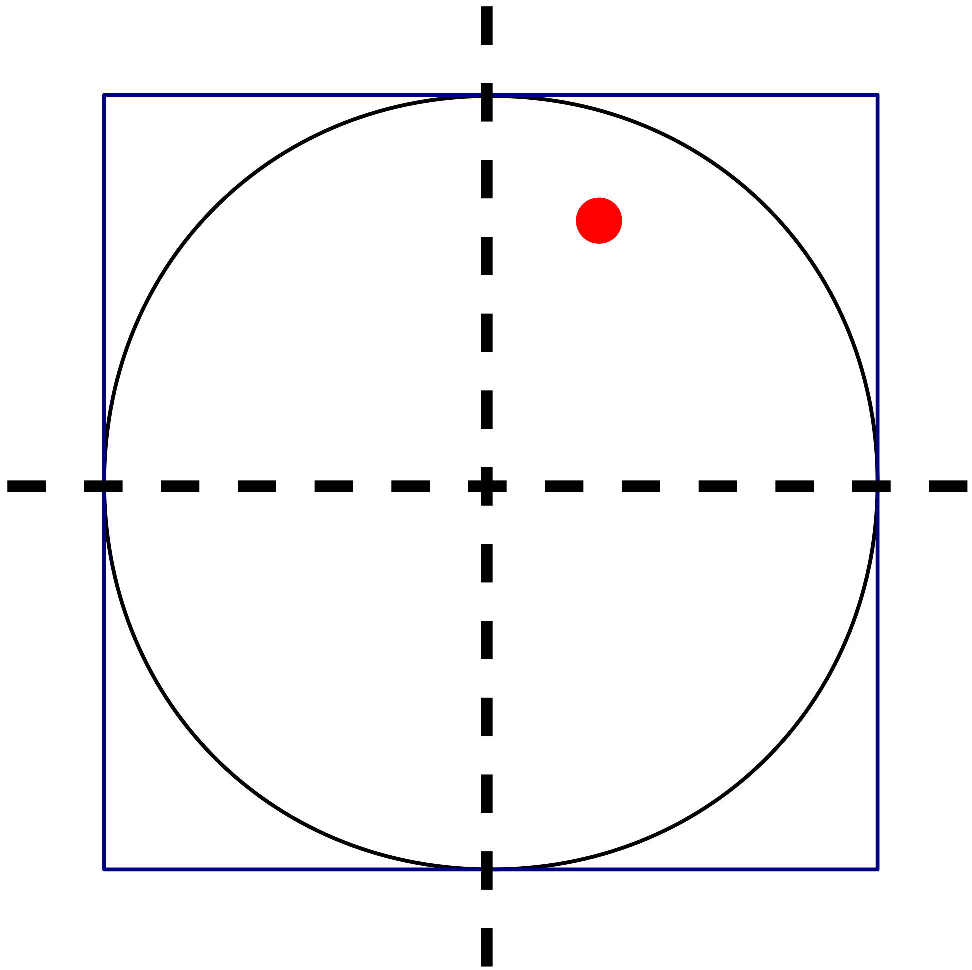


We know the relationship between the radius of a circle and the x and y coordinate of a point on the radius:

$$r=\sqrt{x^2+y^2}$$



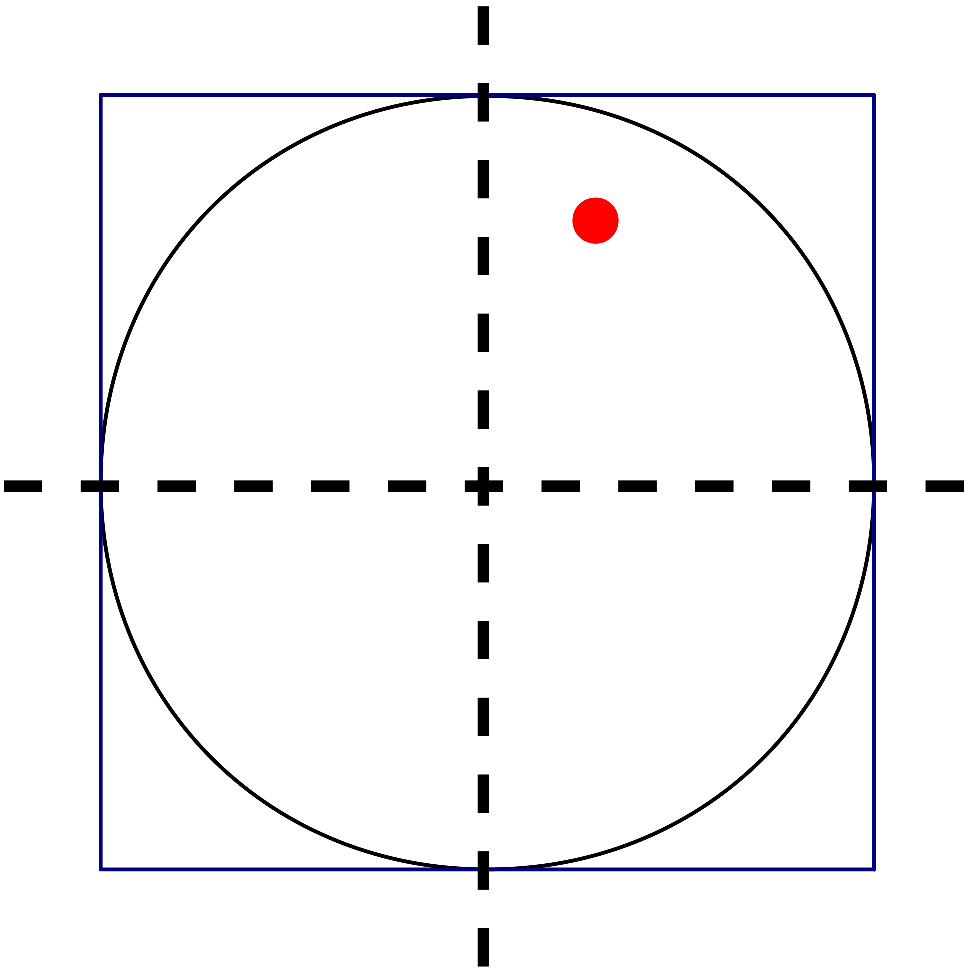
Let us imagine that we have a way of randomly throwing a dot into the square (imagine a game of darts being played, with the square as the board...)



Knowns:

$$r = \sqrt{x^2 + y^2}$$

There is a probability that a uniformly, randomly thrown dot will land in the circle, and a probability that it will land out of the circle. What are those probabilities?

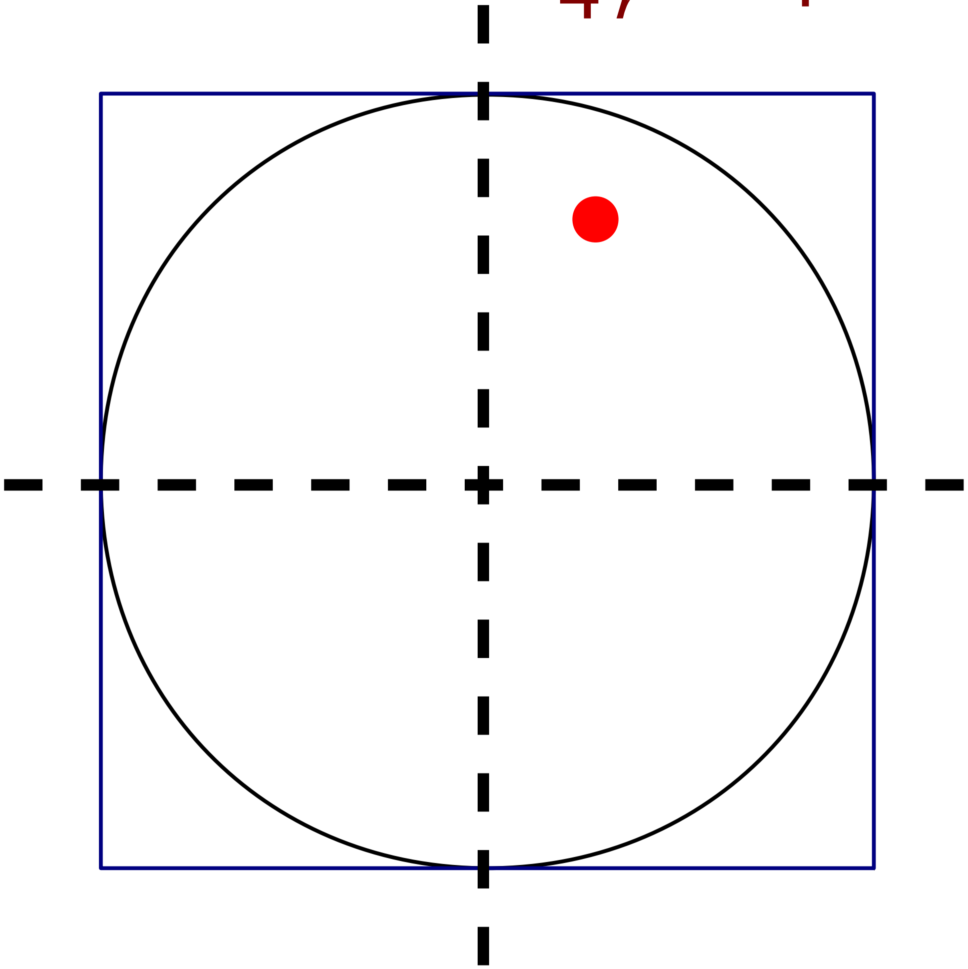


Knowns:

$$r = \sqrt{x^2 + y^2}$$

Probability of landing in the circle is merely given by the ratio of the areas of the two objects:

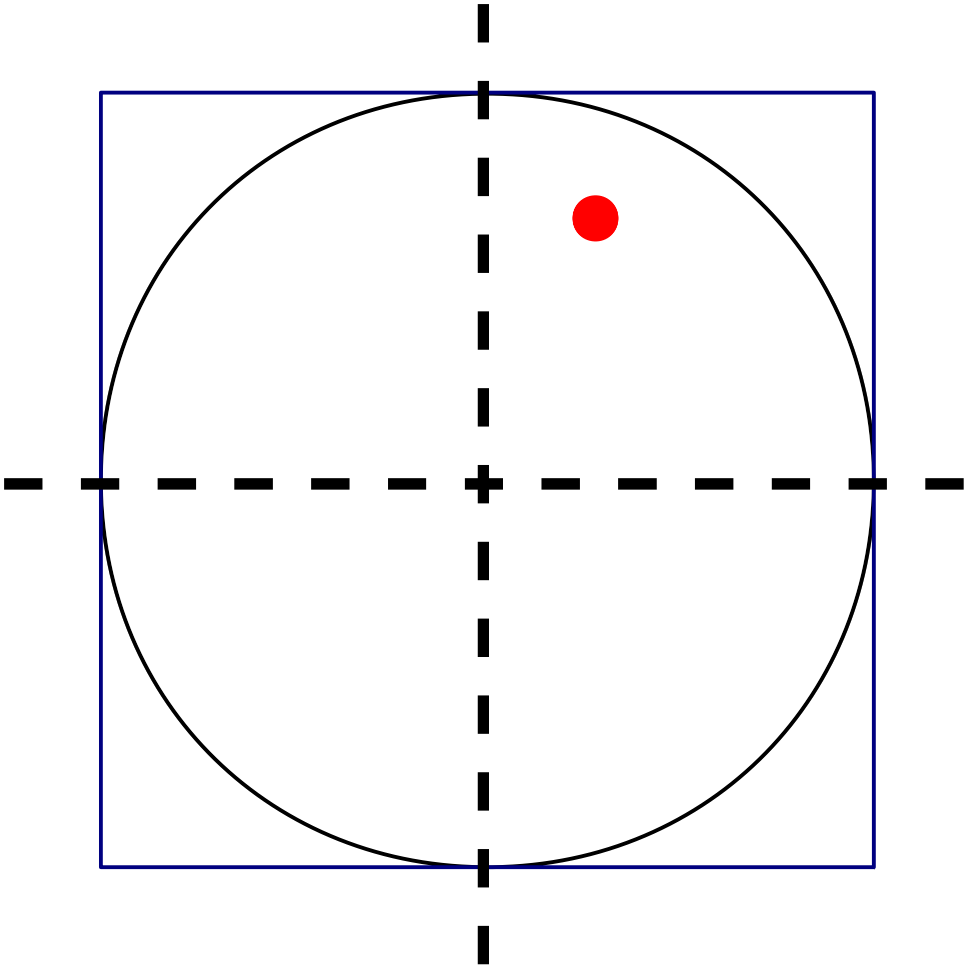
$$P(\mathrm{in}|\mathrm{dot}) = \frac{\pi r^2}{4 r^2} = \frac{\pi}{4}$$



Knowns:

$$r = \sqrt{x^2 + y^2}$$

That's nice – but we're missing a piece . . . just what is that probability on the left side? How can we determine it?



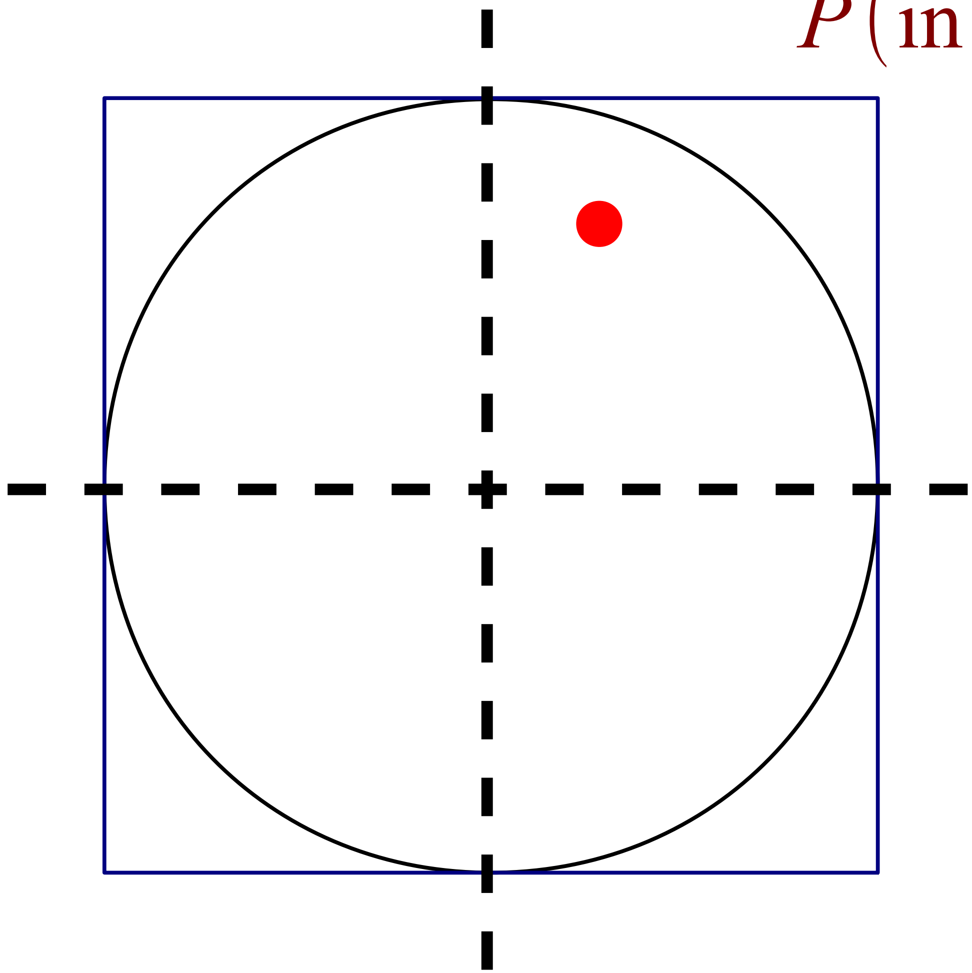
Knowns:

$$r = \sqrt{x^2 + y^2}$$

$$P(\text{in}|\text{dot}) = \frac{\pi}{4}$$

ANSWER: “numerically” - by throwing dots uniformly in the square and counting the number that land inside the circle, divided by the number that we have thrown in total:

$$P(\mathrm{in}|\mathrm{dot}) = \frac{N_{\mathrm{in}}}{N_{\mathrm{total}}}$$

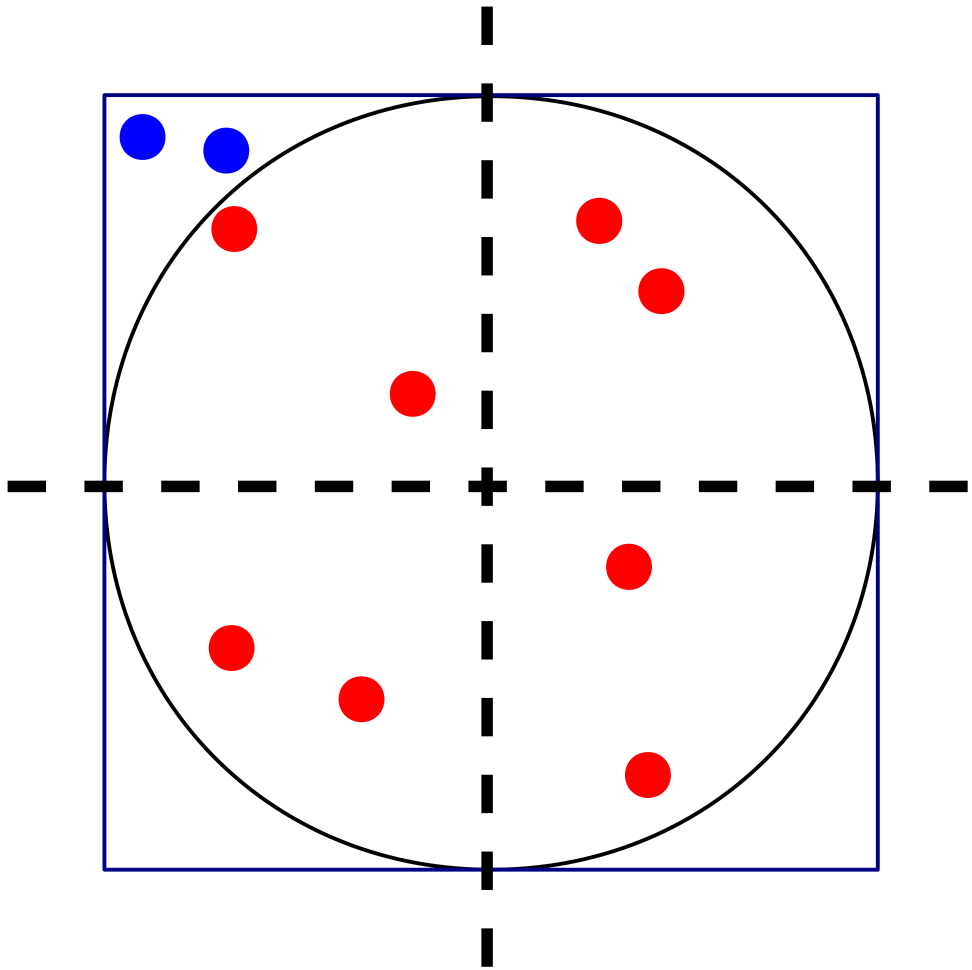


Knowns:

$$r = \sqrt{x^2 + y^2}$$

$$P(\mathrm{in}|\mathrm{dot}) = \frac{\pi}{4}$$

ANSWER: “numerically” - by throwing dots uniformly in the square and counting the number that land inside the circle, divided by the number that we have thrown in total:



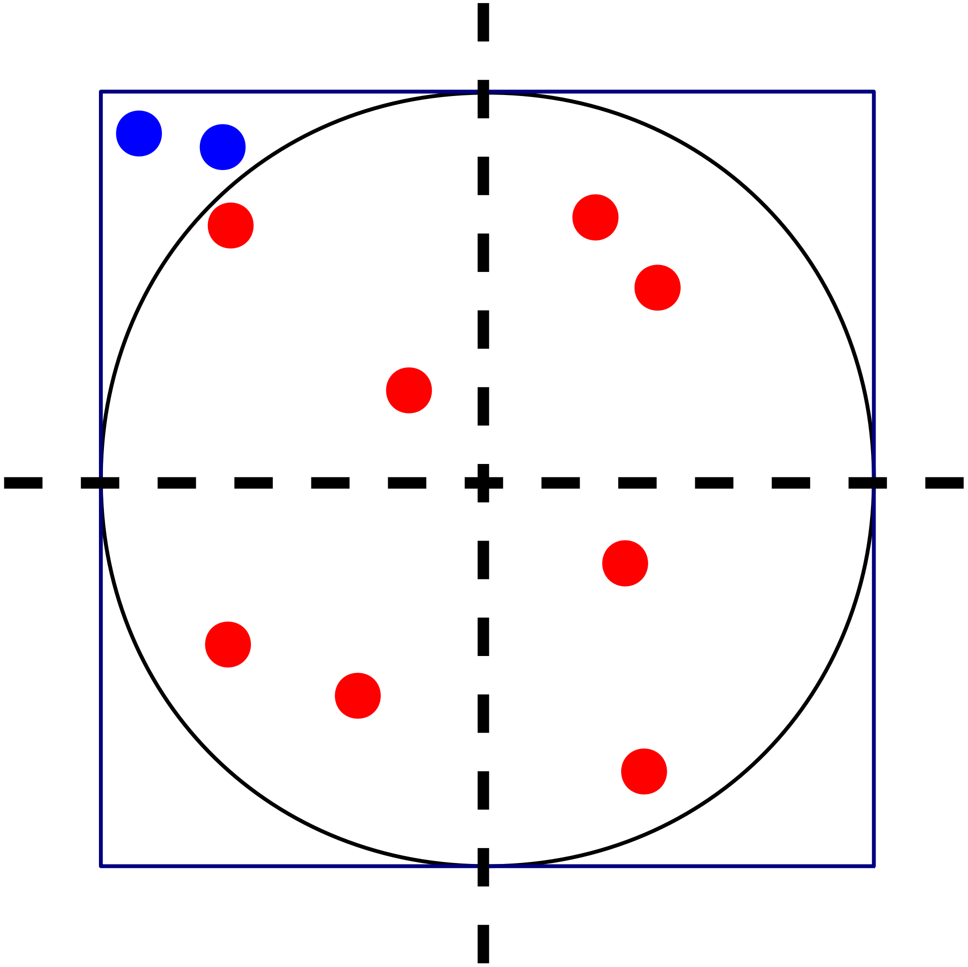
Knowns:

$$r = \sqrt{x^2 + y^2}$$

$$\frac{N_{\text{in}}}{N_{\text{total}}} = \frac{\pi}{4}$$

π is then simply determined numerically via:

$$\pi = 4\frac{N_{\mathrm{in}}}{N_{\mathrm{total}}}$$



Knowns:

$$r = \sqrt{x^2 + y^2}$$

$$\frac{N_{\mathrm{in}}}{N_{\mathrm{total}}} = \frac{\pi}{4}$$

The Pieces

- Random numbers
  - needed to "throw dots" at the board
- Uniformity of coverage
  - we want to pepper the board using uniform random numbers, to avoid creating artificial pileups that create new underlying probabilities
- Code/Programming
  - You can do this manually with a square, an inscribed circle, coordinate axes, and a many-sided die.
  - But that limits your time and precision – computers are faster for such repetitive tasks

Computational Examples

- I will demonstrate the underlying computation framework principles using OCTAVE, a free clone of MATLAB
- Why? I have a web interface running on my own server that lets us ALL follow along and write code today!
- At the end of this, you will have a program you can take with you and adapt into ANY language.
- If you've never seriously written code before, today is your “lucky” day

Basics of Coding

- Numbers – all programming languages can minimally handle numbers: integers, decimals
- Variables – placeholders for numbers, whose values can be set at any time by the programmer
- Functions – any time you have to repeatedly perform an action, write a function. A “function” is just like in math – it represents a complicated set of actions on variables
- Code – an assembly of variables and functions whose goal is determined by the programmer. “Task-oriented mathematics”
- Coding is the poetry of mathematics – it takes the basic rules of mathematics and does something awesome with them.

Commands | Plots | Files | Functions | Account

Input your commands here

```
Ntotal = 100
```

Submit to Octave | Clear

Output:

```
GNU Octave, version 3.2.4
Copyright (C) 2009 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type `warranty'.

Octave was configured for "i686-pc-linux-gnu".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html

Report bugs to <bug@octave.org> (but first, please read
http://www.octave.org/bugs.html to learn how to write a helpful report).

For information about changes from previous versions, type `news'.

Ntotal =  100
```

You type it, it does it.

Errors:

```
warning: X11 DISPLAY environment variable not set
```

Commands | Plots | Files | Functions | Account

Input your commands here

```
Ntotal = 100
Nin = 55
Nin/Ntotal
```

Submit to Octave | Clear

Output:

```
GNU Octave, version 3.2.4
Copyright (C) 2009 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type `warranty'.

Octave was configured for "i686-pc-linux-gnu".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html

Report bugs to <bug@octave.org> (but first, please read
http://www.octave.org/bugs.html to learn how to write a helpful report).

For information about changes from previous versions, type `news'.

Ntotal =  100
Nin =  55
ans =  0.55000
```

Uniform Random Numbers

- Computers can generate (pseudo)random numbers using various algorithms
  - this is a whole lecture in and of itself – if you're interested in pseudo-random numbers, etc. go do some independent reading
- We will utilize the “rand” function in OCTAVE to obtain our uniform random numbers

Commands | Plots | Files | Functions | Account

Input your commands here

```
x=rand
x=rand
x=rand
x=rand
```

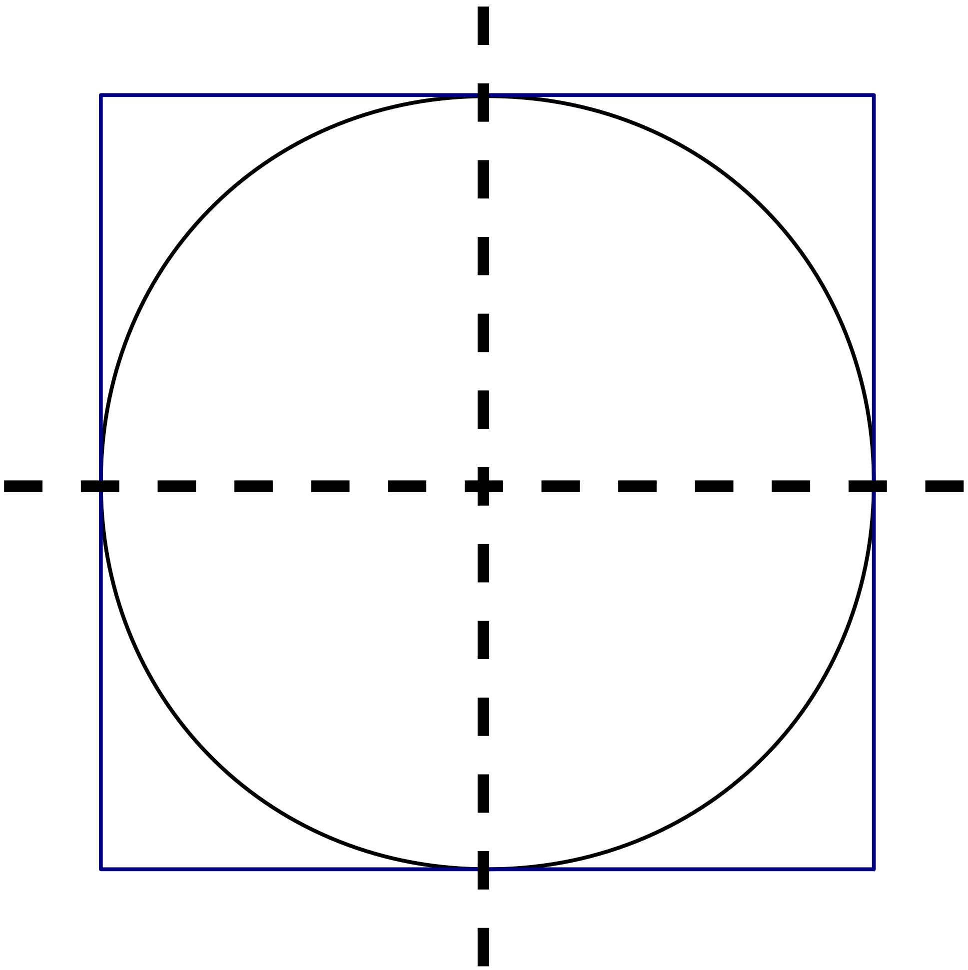
Submit to Octave | Clear

Output:

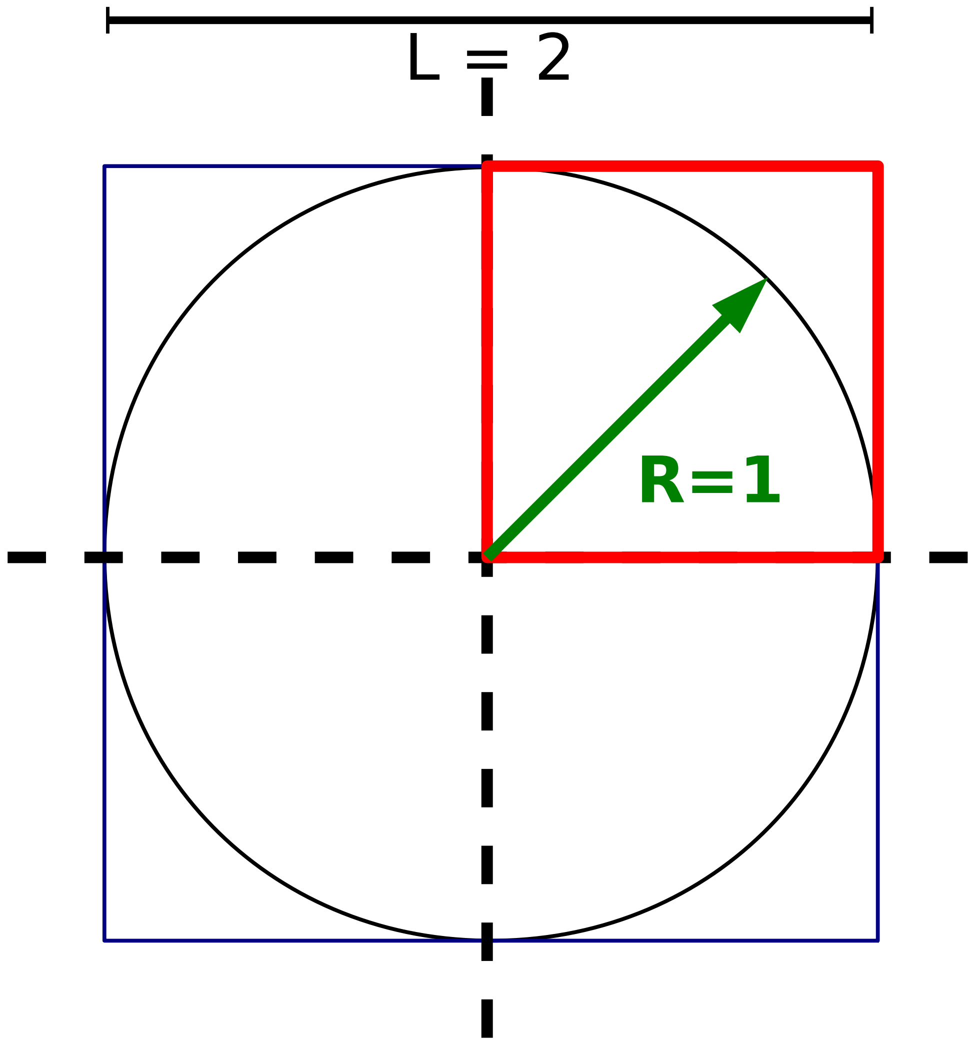
```
GNU Octave, version 3.2.4
Copyright (C) 2009 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type `warranty'.

Octave was configured for "i686-pc-linux-gnu".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html

Report bugs to <bug@octave.org> (but first, please read
http://www.octave.org/bugs.html to learn how to write a helpful report).

For information about changes from previous versions, type `news'.

x =  0.54147
x =  0.021251
x =  0.27853
x =  0.94379
```

"rand" generates a uniform random decimal number between 0 and 1 (inclusive)

Designing our “game board”



Designing our “game board”



We don't need the whole game board – we can just use one-quarter of it. This keeps the program simple!

Alternative: you can rescale the output of “rand” to generate random numbers between -1 and 1

$$\frac{(1/4)\pi r^2}{(1/4)4r^2} = \frac{\pi}{4} = P(\mathrm{in}|\mathrm{dot})$$

Commands | Plots | Files | Functions | Account

Input your commands here

```
x=rand
y=rand
r=sqrt(x^2+y^2)
```

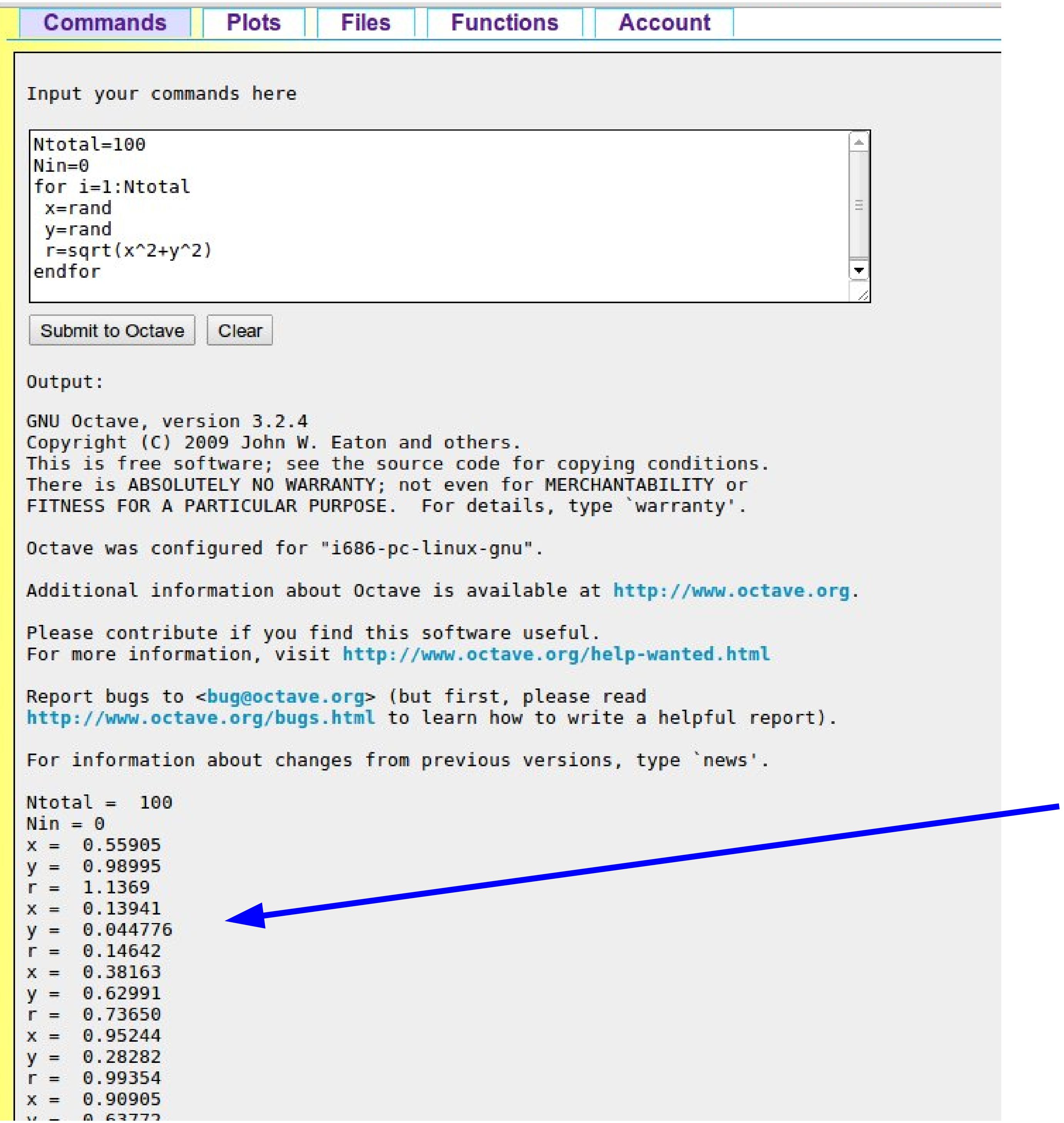
Submit to Octave | Clear

```
Output:

GNU Octave, version 3.2.4
Copyright (C) 2009 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type `warranty'.

Octave was configured for "i686-pc-linux-gnu".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html

Report bugs to <bug@octave.org> (but first, please read
http://www.octave.org/bugs.html to learn how to write a helpful report).

For information about changes from previous versions, type `news'.

x =  0.52718
y =  0.48970
r =  0.71953
```

Repetition

- You don't want to manually type 100 (or more) computations of your dot throwing
- You need a loop!
- A “loop” is a small structure that automatically repeats your computation an arbitrary number of times
- In OCTAVE:

```
Ntotal=100
for i=1:Ntotal
 x=rand
 y=rand
endfor
```



"Loops" are powerful – they are a major workhorse of any repetitive task coded up in a programming language.

PRO TIP:

If you don't want all that annoying output to clutter your window, end your lines with a semi-colon, ; (output slows down code!)

Final Piece

- So we have generated a dot by generating its x and y coordinates throwing uniform random numbers...
- How do we determine if it's “in” or “out” of the circle?
- ANSWER:
  - if $r = \sqrt{(x^2+y^2)} < R$, it's in the circle; otherwise, it is out of the circle!

Commands | Plots | Files | Functions | Account

Input your commands here

```
Ntotal=100;
Nin=0;
R=1;
for i=1:Ntotal
 x=rand;
 y=rand;
 r=sqrt(x^2+y^2);
 if (r<R)
   Nin = Nin + 1;
 endif
endfor

my_pi = 4*Nin/Ntotal
```

Submit to Octave | Clear

Output:

```
GNU Octave, version 3.2.4
Copyright (C) 2009 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type `warranty'.

Octave was configured for "i686-pc-linux-gnu".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html

Report bugs to <bug@octave.org> (but first, please read
http://www.octave.org/bugs.html to learn how to write a helpful report).

For information about changes from previous versions, type `news'.

my_pi =  3.2000
```

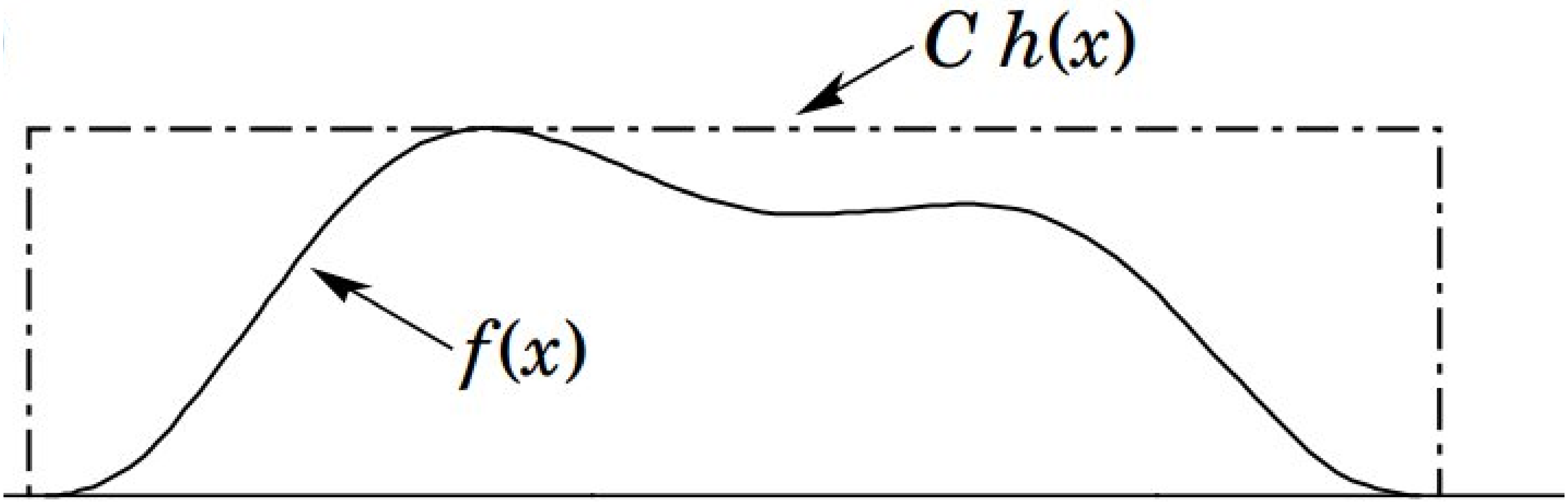
A working program.

You can increase Ntotal to get increased precision!

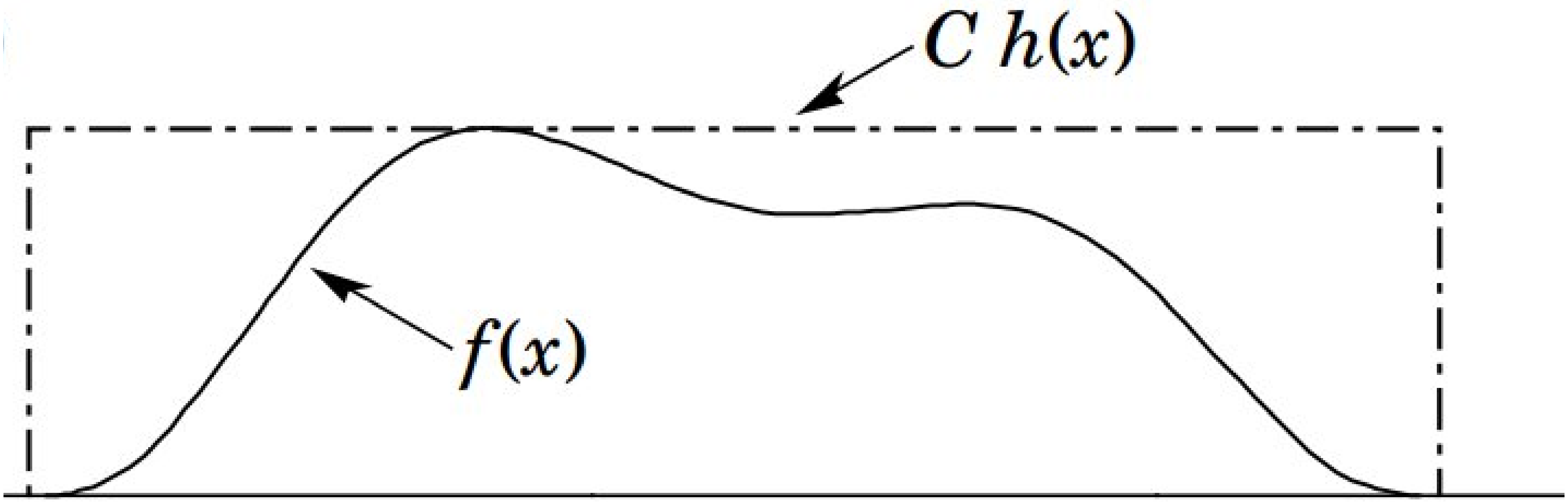
The uncertainty on π goes as 1/√Ntotal, so for Ntotal=100 you expect a 10% uncertainty; for Ntotal=1e6, you expect a 0.1% uncertainty.

Why is this powerful?

- You've just learned how to compute an integral NUMERICALLY.
- You can apply this technique to any function whose integral (area) you wish to determine
- For instance, consider the next slide.



- Given an arbitrary function, f(x), you can determine its integral numerically using the "Accept/Reject Method"
- **First**, find the maximum value of the function (e.g. either analytically, if you like, or by calculating the value of f(x) over steps in x to find the max. value, which I denote F(x))
- **Second**, enclose the function in a box, h(x), whose height is F(x) and whose length encloses as much of f(x) as is possible.
- **Third**, compute the area of the box (easy!)
- **Fourth**, throw points in the box using uniform random numbers. Throw a value for x, denotes x'. Throw a value for y, denoted y'. If y' < f(x'), it's a hit! If not, it's a miss!



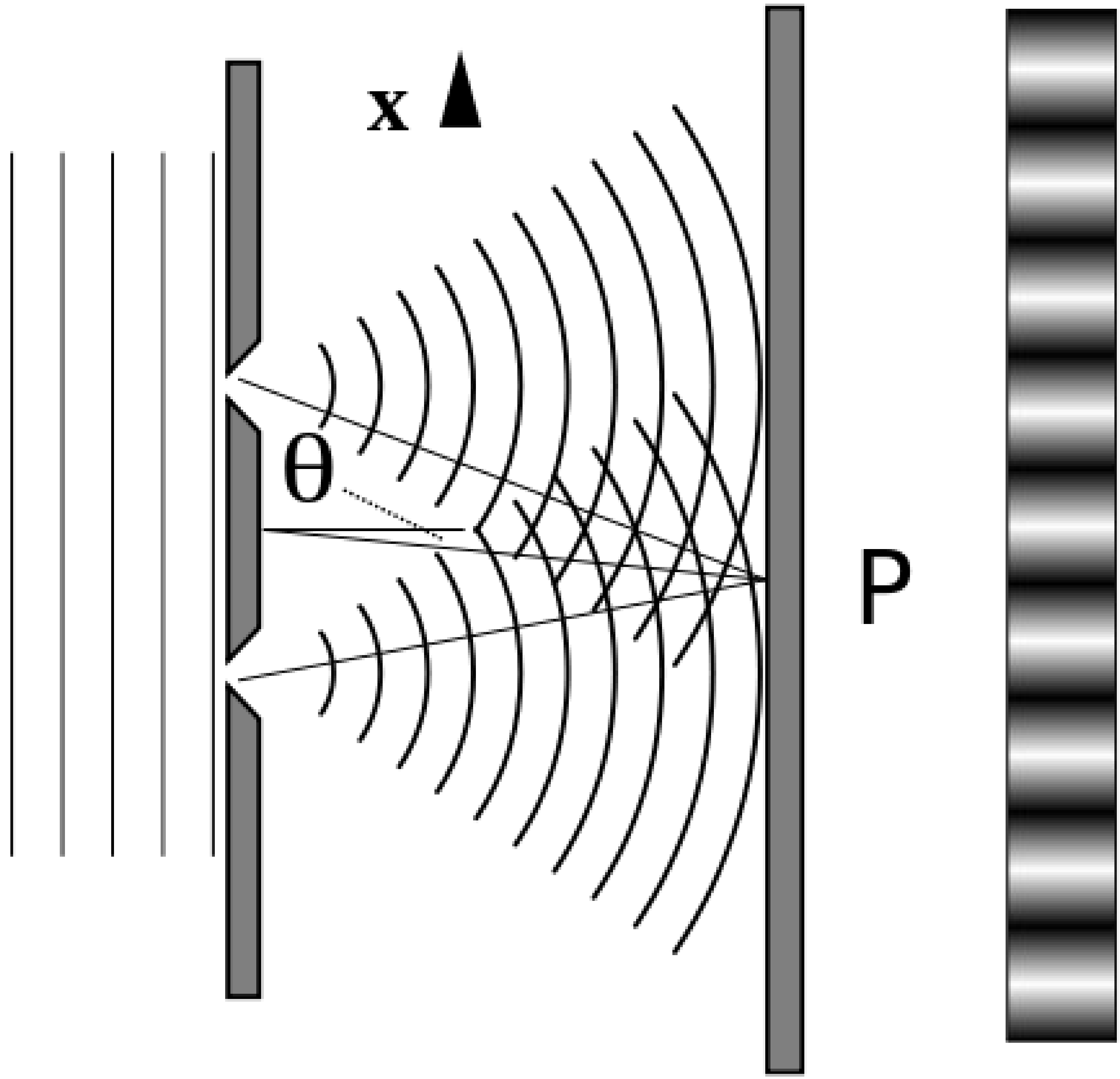
$$\frac{N_{\text{hits}}}{N_{\text{total}}} = \frac{I\left(f\left(x\right)\right)}{A\left(h\left(x\right)\right)}$$

This, in the real world, is how physicists, engineers, statisticians, mathematicians, etc. compute integrals of arbitrary functions.

Learn it. Love it. It will save you.

Generating Simulations

- The Monte Carlo technique, given a function that represents the probability of an outcome, can be used to generate “simulated data”
- Simulated data is useful in designing an experiment, or even “running” an experiment over and over to see all possible outcomes


x
θ
P

Young's Double-Slit Experiment Simulation

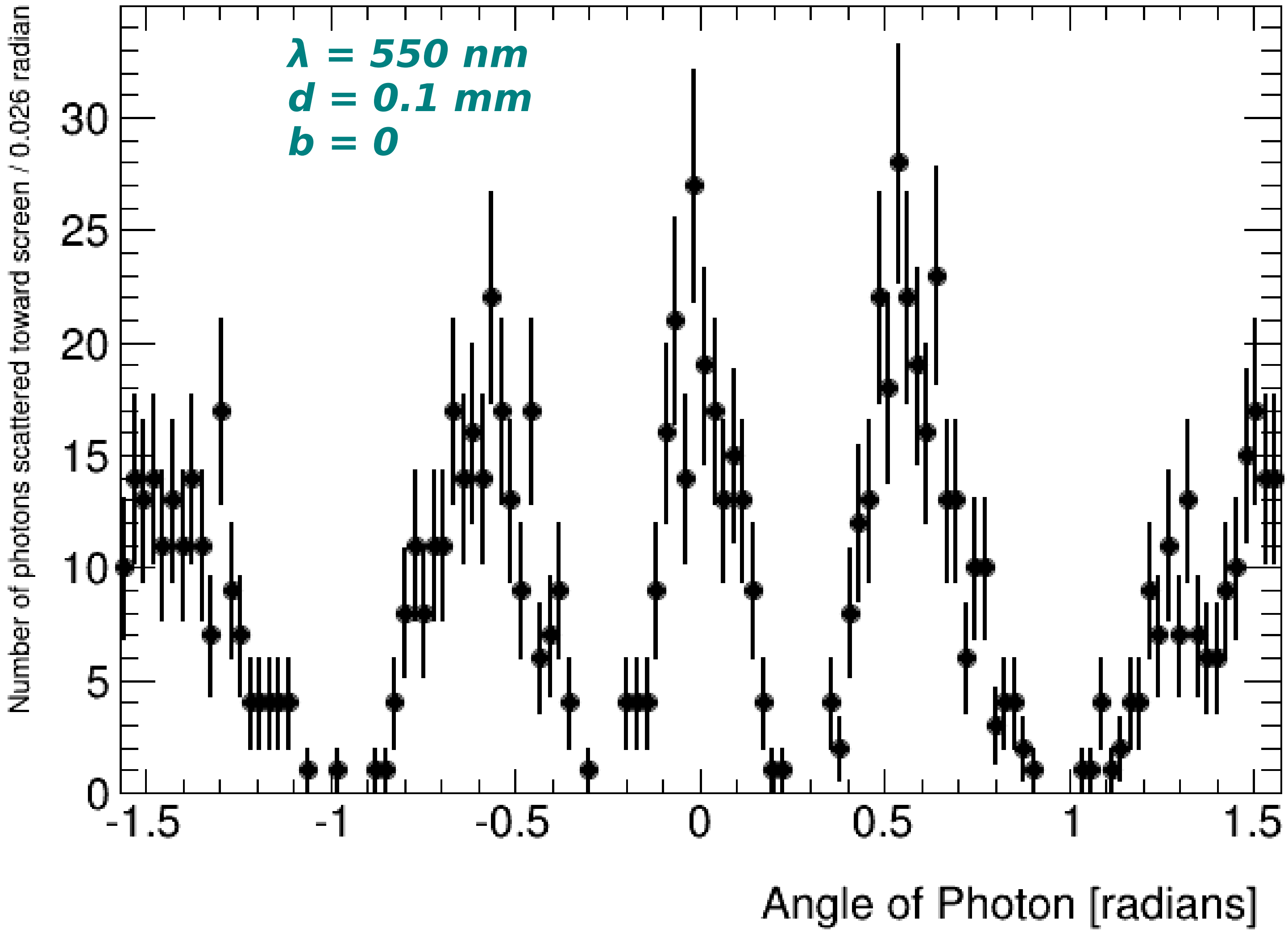
- Consider slits of width, b, separated by a distance, d.
- Given the function that describes the probability of finding a photon at a given angle:

$$I(\theta) \propto \cos^2\left[\frac{\pi d \sin(\theta)}{\lambda}\right] \mathrm{sinc}^2\left[\frac{\pi b \sin(\theta)}{\lambda}\right]$$

$$\mathrm{sinc(x)} = \begin{cases} \sin(x)/x \quad (x \neq 0) \\ 1 \quad (x = 0) \end{cases}$$

Next Steps

- Need the max. value of $I(\theta)$
  - occurs at $\theta = 0$
- Use that to compute the height of the box; the width of the box is π (ranging from $-\pi/2$ to $+\pi/2$)
- “Throw” random points in the box until you get 1000 “accepts”
- Now you have a “simulated data” sample of 1000 photons scattered in the two-slit experiment.



1000 simulated photons scattered through a double-slit experiment. This was done in C++ using the free ROOT High-Energy Physics data analysis framework, so I could easily generate a *histogram* – a binned data sample.

Resources

- Octave: http://www.gnu.org/software/octave/
- Python: http://www.python.org/
- Mathematica: http://www.wolfram.com/mathematica/
- Monte Carlo Techniques: http://en.wikipedia.org/wiki/Monte_Carlo_method