Computational modeling

Lecture 5 : *Monte Carlo* Integration

Physics:

① Integration
② The *Monte Carlo* method

Programming:

① Subroutine
② Differences Subroutine/Function
③ The Fortran random number subroutine
④ Monte Carlo code explained

Instructor : Cedric Weber
Course : 4CCP1000

Schedule

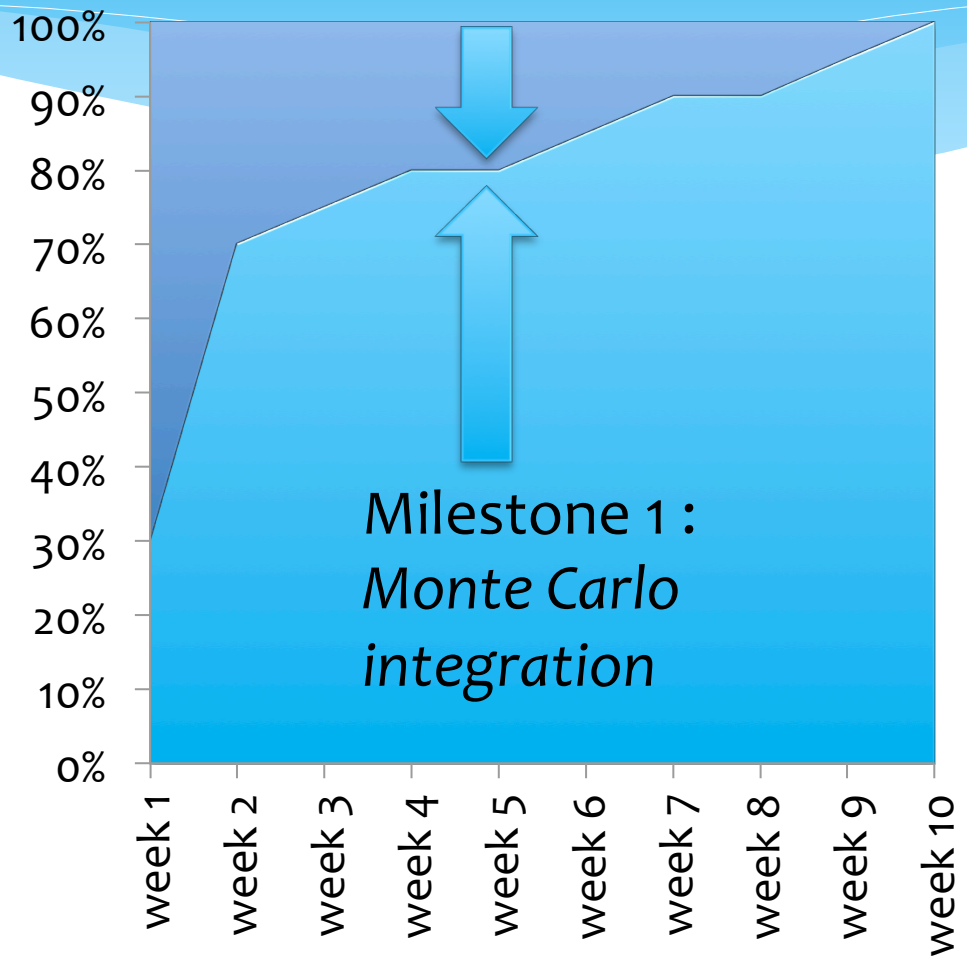
Class/Week	Chapter	Topic	Milestones
1	Monte Carlo	UNIX system / Fortran	
2	Monte Carlo	Fibonacci sequence	
3	Monte Carlo	Random variables	
4	Monte Carlo	Central Limit Theorem	
5	Monte Carlo	"Monte Carlo" integration	Milestone 1
6	Differential equations	The Pendulum	
7	Differential equations	A Quantum Particle in a box	
8	Differential equations	The Tacoma bridge	Milestone 2
9	Linear Algebra	System of equations	
10	Linear Algebra	Matrix operations	Milestone 3

What did you learn last time?

1. Averaging quantities with a ***do loop*** ☒
2. Defining your own functions ☒
3. Using a random generator to produce random values ☒
4. Using ***arrays*** in combination of ***do loops*** ☒

Where are we going?

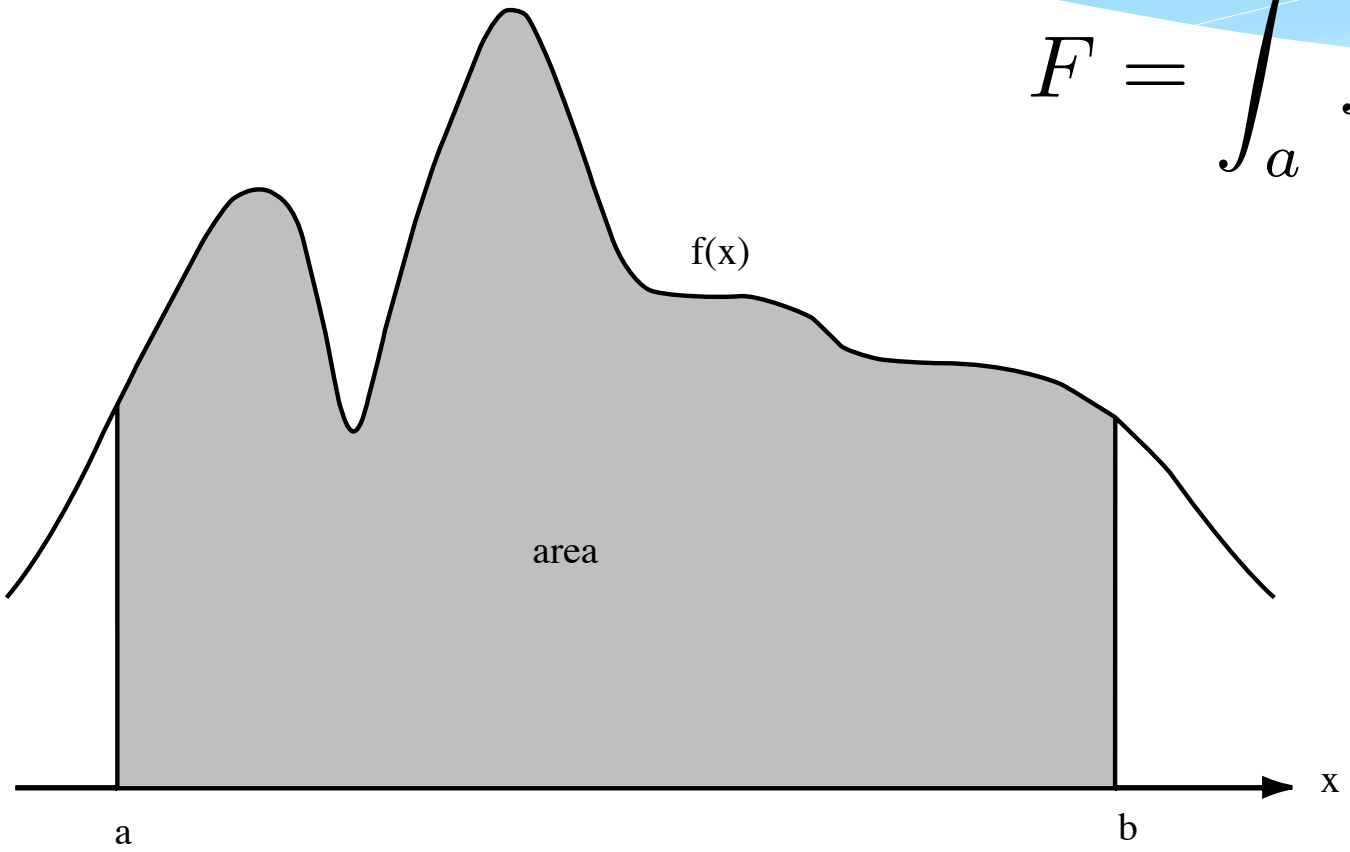
* **Lecture 1:**
  * Introduction
* **Lecture 2:**
  * Fibonacci sequence / **do loop**
* **Lecture 3:**
  * Random variables / **functions**
* **Lecture 4:**
  * *multiple events statistics* / ***arrays***
* **Lecture 5:**
  * *Monte Carlo method*



Part I : Introduction to Integration

Integration

$$F = \int_a^b f(x)\,dx.$$



Method 1 : Discretization

* Discretize the horizontal axis, approximate the surface enclosed by the function f(x) by a set of rectangles

* Example: **f(x) = cos(x)**

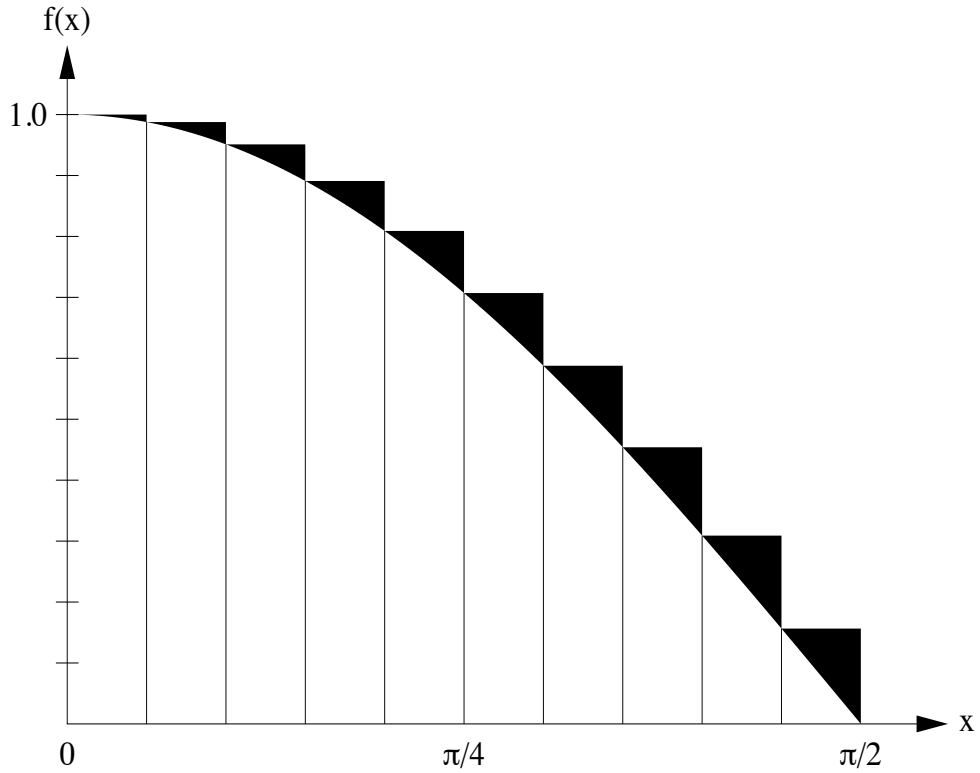
* width of a rectangle:

* Discretization x: $$\Delta x = \frac{b-a}{n}$$

* Area=sum of rectangles:

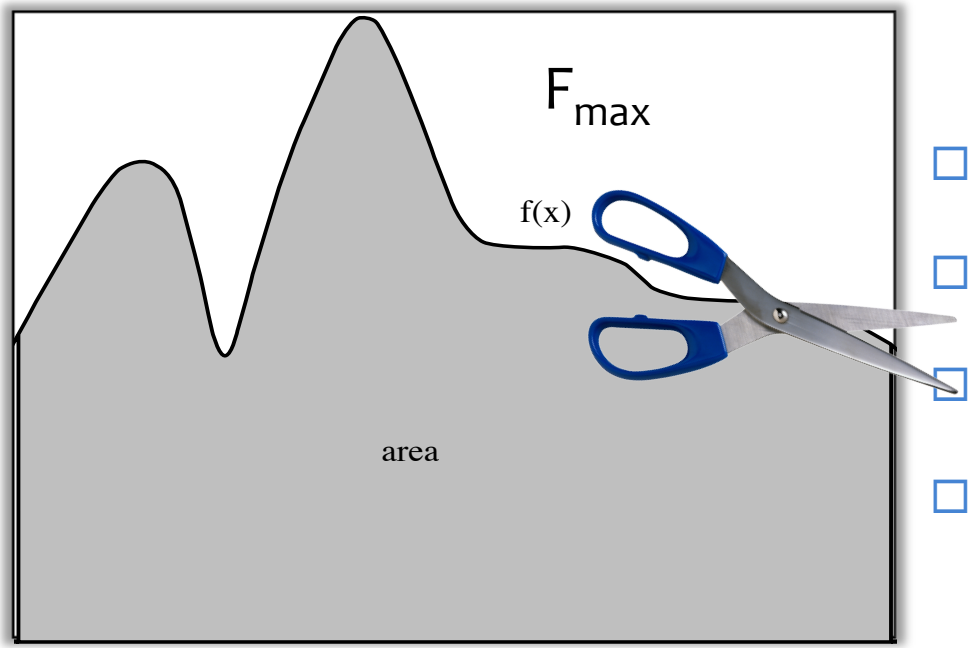
$$x_n = x_0 + n\Delta x$$

$$F_n = \sum_{i=0}^{n-1} f(x_i)\Delta x$$



Method 2 : cutting and weighting

- Plot the function f(x) on a piece of paper, from **a** to **b**, and from **0** to $\mathbf{F_{max}}$
- Weight the full sheet of paper, we define this as **Ws**
- Plot the function on a sheet of paper
- Cut the sheet in paper along the function
- Weight the sheet of paper contained between the horizontal axis and the function f(x), we define this as **Wf**



- ρ : paper weight density
- **Ws** = (b-a) x F_{max} x ρ
- **Wf** = integral x ρ
- Integral = **Wf**/**Ws** x (b-a) x F_{max}

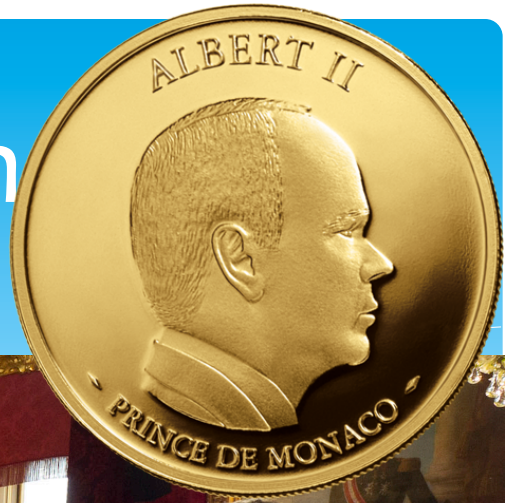
Part II : Introduction to Monte Carlo

Mor

Sweden
Faroe Islands
Finland
Norway
Aland
Estonia
Latvia
Denmark
Lithuania
Belarus
Isle of Man
Ireland
United Kingdom
Poland
Netherlands
Ukraine
Germany
Belgium
Guernsey
Jersey
Luxembourg
Czech Republic
Slovakia
Moldova
Austria
Hungary
Romania
France
Switzerland
Liechtenstein
Slovenia
Croatia
Serbia
Bosnia and Herzegovina
Bulgaria
San Marino
Monaco
Montenegro
Italy
Macedonia (FYROM)
Andorra
Holy See (Vatican)
Albania
Portugal
Spain
Greece
Cyprus
Gibraltar
Mälta
Morocco
Tunisia

Monte Carlo … about summer vacation?

Monte Carlo is about … Kin

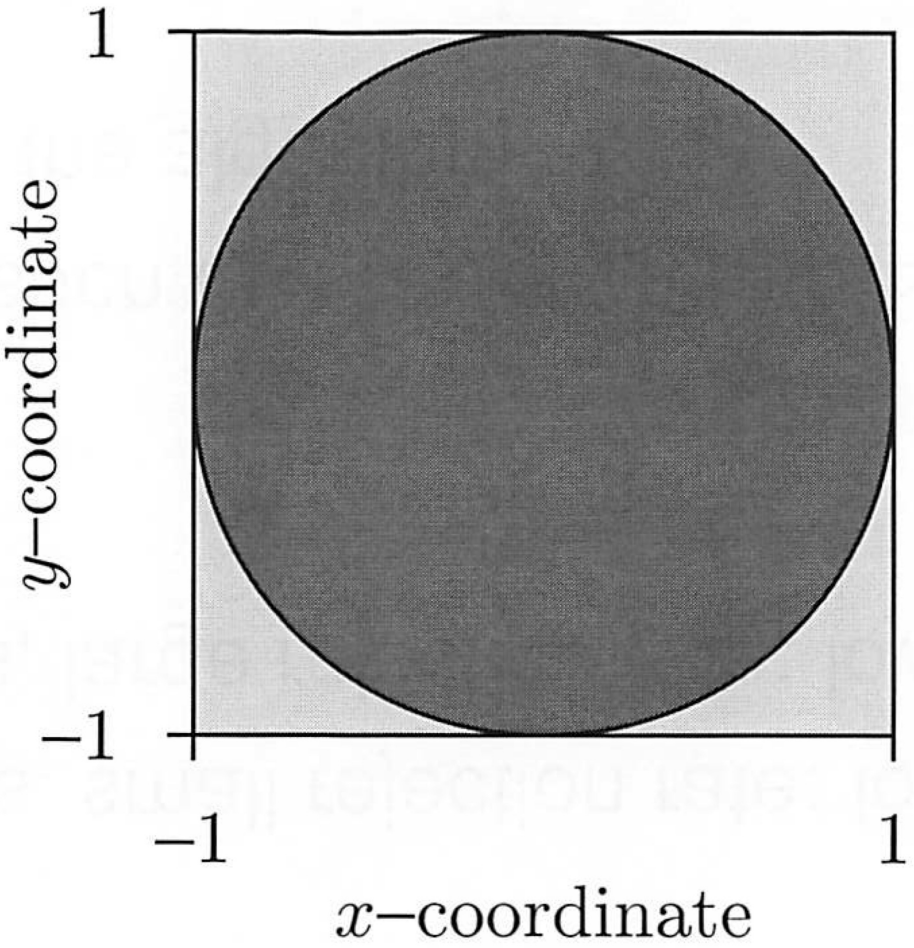


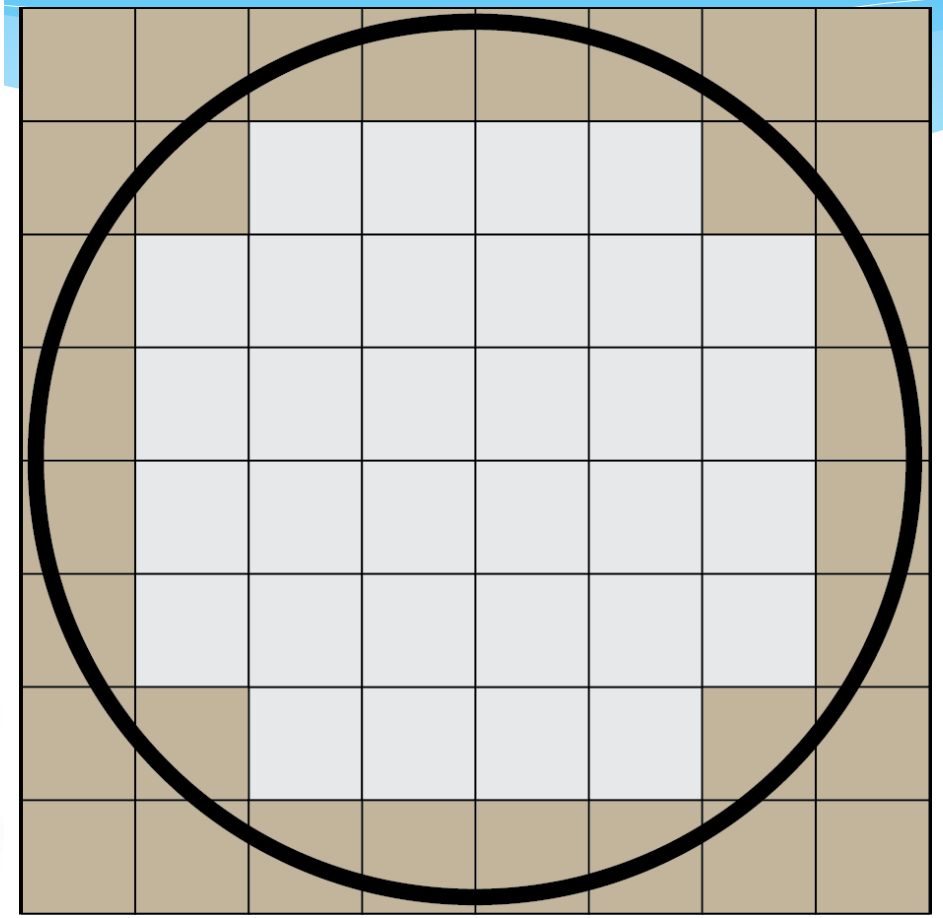
SNACK
top
NAZA

CASINO ROYALE

Monte Carlo : The Casino

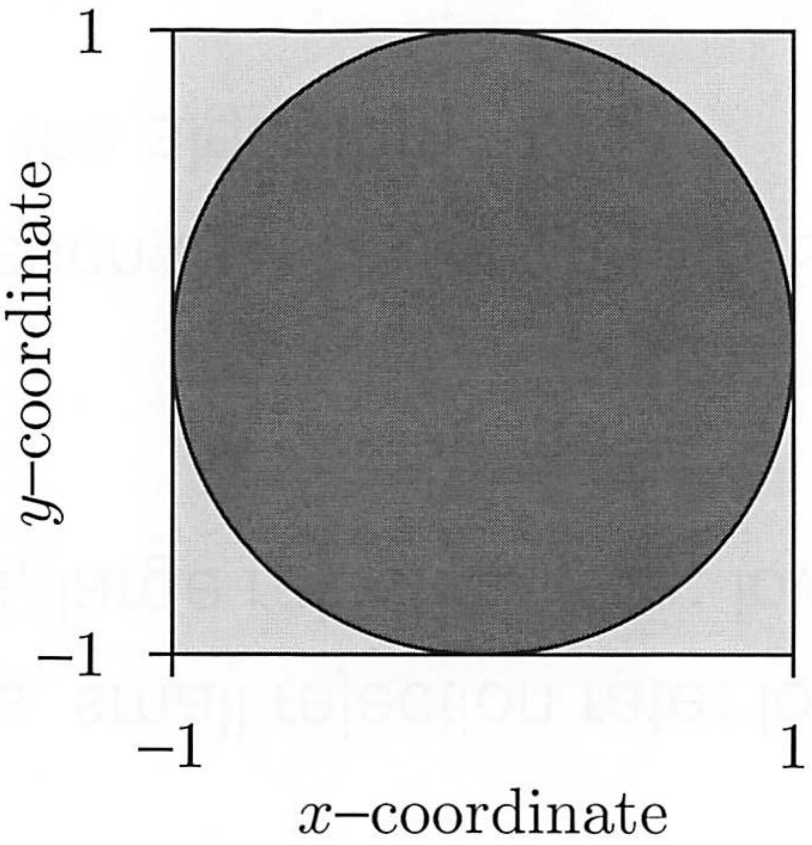
2D Integration : The area of a circle





Area of circle = sum of squares contained within the circle (light gray)

2D Integration : The area of a circle

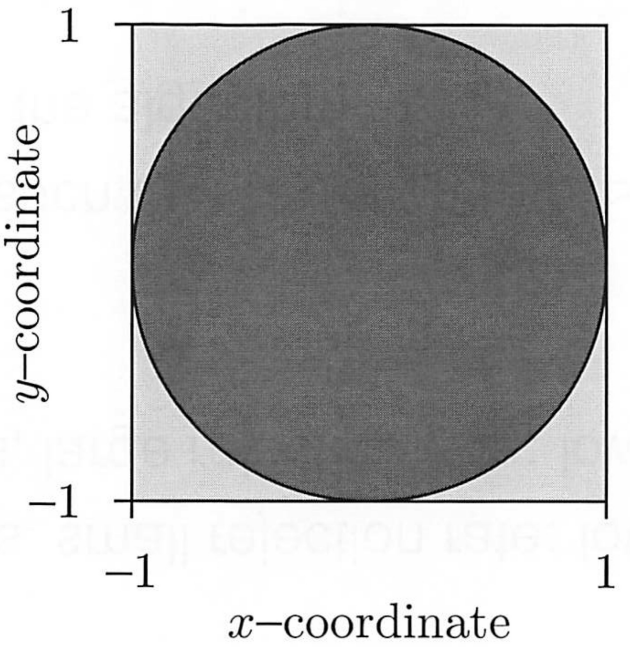


Game : children throw stones in a square, try to hit the circle.

Reference: W. Krauth "*Introduction to Monte Carlo algorithms*", Springer 1998

Monte Carlo algorithm

* Randomly throw stones within the squares
* We count the number of hits within the circle



  * Hits within the circle: **Ncircle**
  * Total number of trials: **Ntot**

* We define probabilities associated with the number of hits (as defined by Bernoulli in the last lecture, e.g. with coin flips).
  * Probability to hit the square : **$P_{square} = 1$**
  * Probability to hit the circle : **P_{circle} ?**

* P_{circle} = …. [Fill in] …
* If the hits are uniformly distributed within the square, we also have that :
* P_{circle} / P_{square} = Area of circle / Area of square = … [fill the blank]…
* Question: can you use this algorithm to extract π ?

Monte Carlo : pseudo-code

1. Program montecarlo
2. Nhits initialized to zero
3. Do i=1,Ntot
   X a random number in [-1,1]
   Y a random number in [-1,1]
   If($x^2+y^2<1$) Nhits=Nhits+1
4. End do
5. Output: Nhits
6. **Probability to hit the circle** : Nhits/Ntot
7. **Area of circle** : Nhits/Ntot x Area of square = Nhits/Ntot x 4
8. Compare with exact result, area of the circle = π
9. Output π

Five trials with $N = 4000$

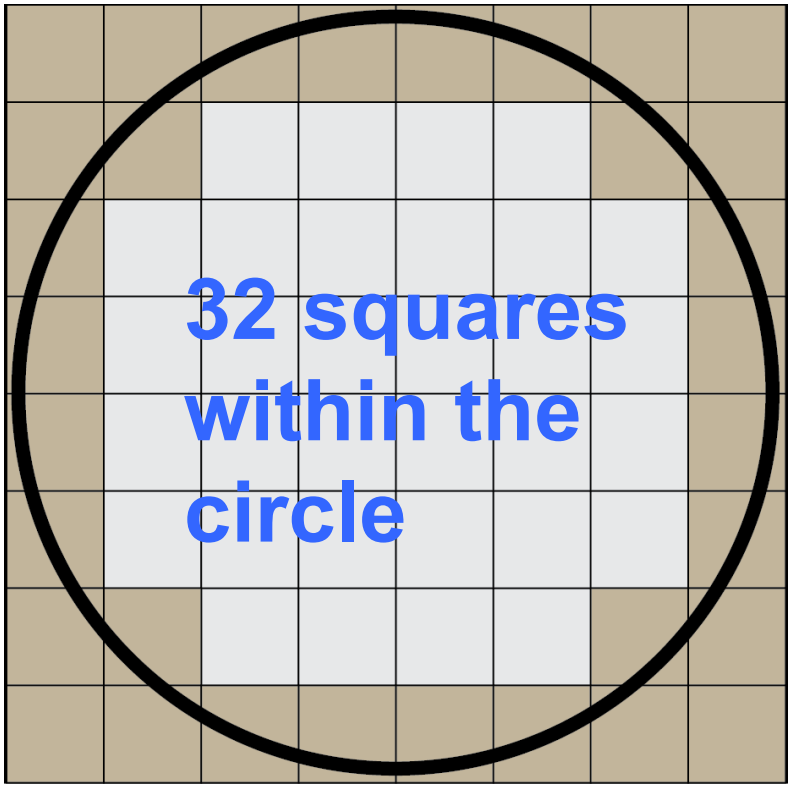
run	N_{hits}	estimation
1	3156	3.156
2	3129	3.129
3	3154	3.154
4	3134	3.134
5	3148	3.148

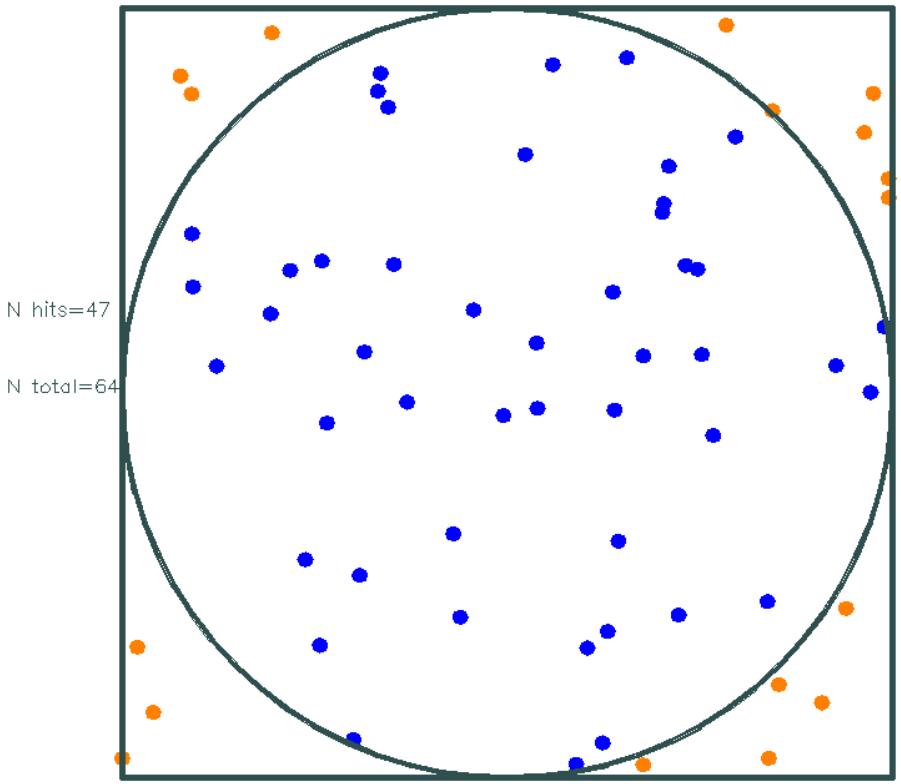
Reference: W. Krauth "Introduction to Monte Carlo algorithms", Springer 1998, cond-mat/961218

Periodic tiling / Random tiling

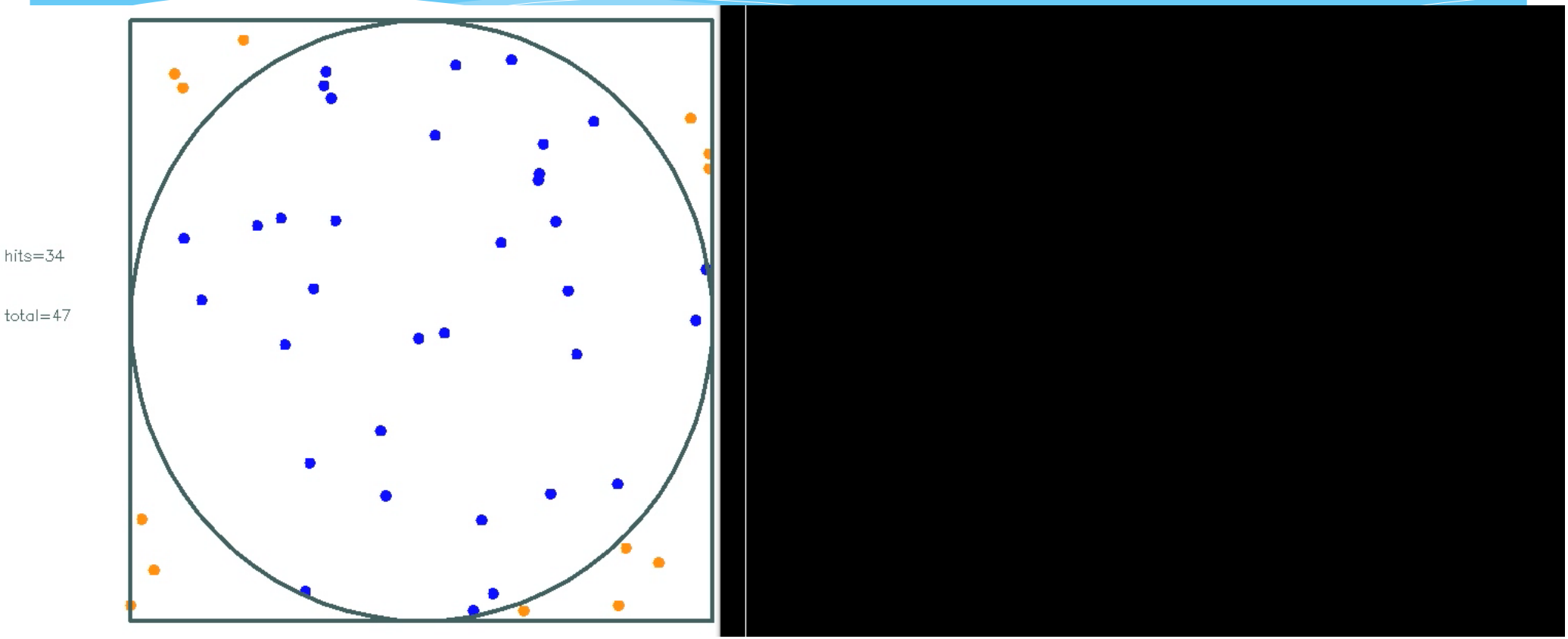
N squares in total = 64 (8x8 grid)
N squares contained in circles=32
Area circle / Area total = **0.5**
Area total = 4
➔ **Estimated Area circle=2**
Exact Area circle=3.1428

N hits = 47
N total = 64
Area circle / Area total = N_{hits}/N_{total}= **0.734**
Area total = 4
➔ **Estimated Area circle=2.93**
Exact Area circle=3.1428





Simulation - Movie



Conclusion: Monte Carlo is better when N_{total} is small, periodic mesh is better when N_{total} is large

Why Monte Carlo?

"Monte Carlo is an extremely bad method; it should be used only when all alternative methods are worse." Alan Sokal

Monte Carlo methods in statistical mechanics, 1996

The **error** is only shrinking as $1/\sqrt{N}$

Other simple methods for integrations in two dimensions: $1/N^{3/2}$

In more than two dimensions (2D), e.g. in three dimensions (3D) (volume of a sphere,…):

$1/N^{3/D}$

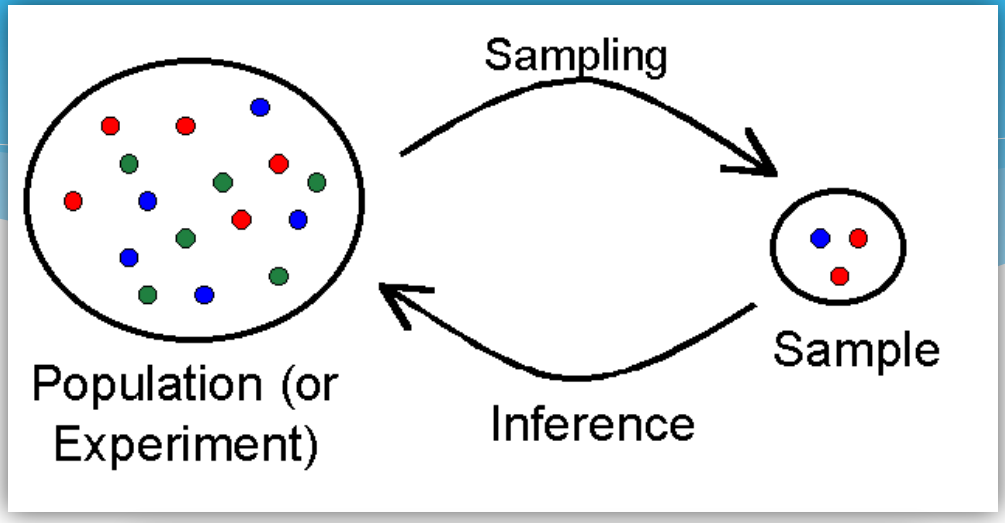
Which method is better for **D=10** ? Answer = …[FILL IN] ….

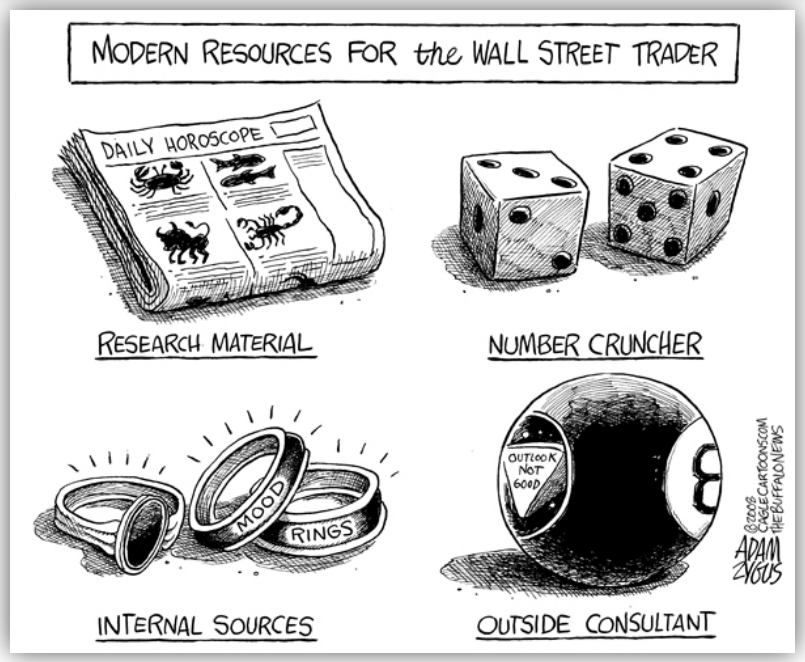
How does Monte Carlo work

- From a large ensemble we picked up randomly a small selection out of it

- We did this in such a way that the chosen **sample is representative of the full population**

- For the volume of a circle : we chose uniformly points belonging to the square, **uniformly chosen points are representative of all the points** within the square





Part III : How are we going to do all of this?

Monte Carlo code

1. Subroutine

2. Fortran internal random subroutine

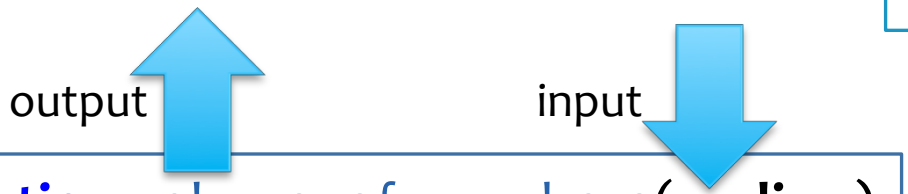
3. Monte Carlo code

Subroutines : definition

1. **Subroutines in Fortran are very similar to functions**
2. **Only difference: they do not return any values**
3. **Subroutines have inputs and outputs instead**

```
subroutine volume_of_a_sphere( volume, radius)
implicit none
real(8)  :: volume, radius
                          volume = radius**3.0
end subroutine
```

***Subroutine*: radius in the input, volume is the output, both are arguments**




```
Function volume_of_a_sphere( radius )
real(8) :: radius, volume_of_a_sphere
volume_of_a_sphere =  radius**3.0
End function
```

***Function* : radius the input (argument), the output value if given by the function itself volume_of_a_sphere**

Functions versus Subroutine : example

FUNCTION

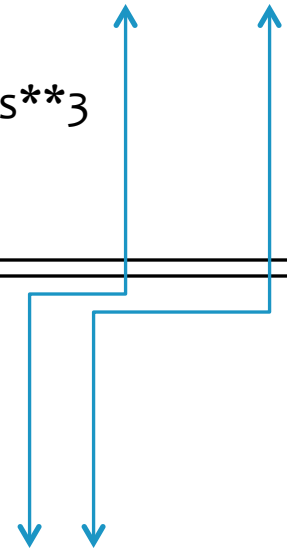
```
1.  Module mylibrary
2.  Contains
3.   Function volume_of_a_sphere(radius)
4.   real(8) :: radius, volume_of_a_sphere
5.   volume_of_a_sphere=radius**3
6.   End function
7.  End module
```

```
8.   program test
9.  Use mylibrary
10. Implicit none
11.  real(8) :: a
12.  a=volume_of_a_sphere(2.0)
13.  end program
```

SUBROUTINE

```
1.  Module mylibrary
2.  Contains
3.   Subroutine volume_of_a_sphere
     (volume,radius)
4.   real(8) :: radius, volume
5.   volume_of_a_sphere=radius**3
6.   End function
7.  End module
```

```
8.   program test
9.  Use mylibrary
10. Implicit none
11.  Real(8) :: a
12.  call volume_of_a_sphere(a,2.0)
13.  end program
```



1. Subroutine

2. Fortran internal random subroutine

3. Monte Carlo code

The random number Fortran subroutine

* In Fortran, the internal subroutine (by internal we mean it is part of the compiler) ***random_number(r)*** provides a new random number each time it is used

* To use a subroutine, you need to use the prefix ***call***

* Each time the subroutine ***random_number (r)*** is executed, r is replaced with a new random value between 0 and 1

```
1.  program testrandom
2.  Implicit none
3.  Real(8) :: r

4.  call random_number ( r )
5.  write(*,*) 'this is my first number ', r

6.  call random_number ( r )
7.  write(*,*) 'this is my second number ', r

8.  end program
```

Random number : Fortran integrated **subroutine**

```
1.  program testran
2.  implicit none
3.    Real(8)  ::  r

4.    Write(*,*) " my random number is
      = ", r
5.    call random_number( r )

6.    ! each time random_number is called,
      it returns a different value in the
      variable r, this is the random number

7.  end program
```

What is wrong in this code ?

- The subroutine **Random_number** is **readily available in Fortran** (part of the programming language, as sin,cos,exp)
- It takes "**r**" as an argument, and replaces "**r**" with a new value, a random value between [0,1], e.g. r=0.523123
- A subroutine has one or more arguments
- To execute a subroutine, notice the syntax :
  - Call *name_of_subroutine*(*argument*)

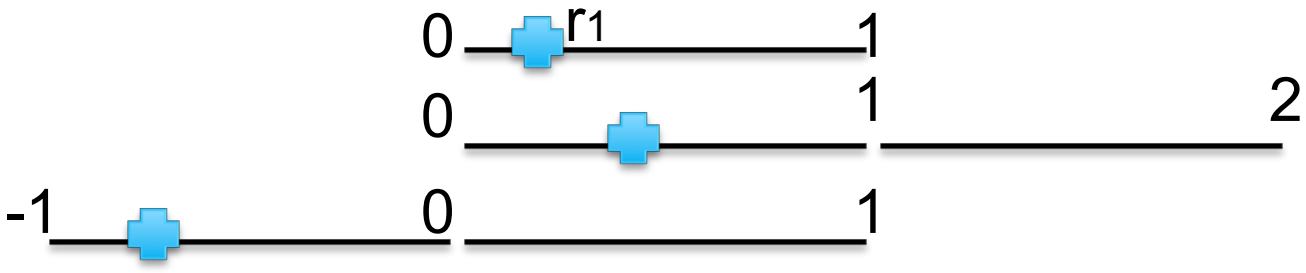
Picking randomly a point in the square

Timeline

```
1.   program test
2.   Implicit none
3.   real(8)        :: r,s
4.   real(8)        :: x, y

5.    call random_number( r )
6.    call random_number( s )
7.
8.     x  =  [Fill in.. ]  *  r   -  [ Fill in ..]
9.     y  =  [Fill in.. ]  *  s   -  [ Fill in ..]
10.  end program
```

- Variable r is spanning which interval?
- How can I obtain x spanning the invertal [-1,1]?



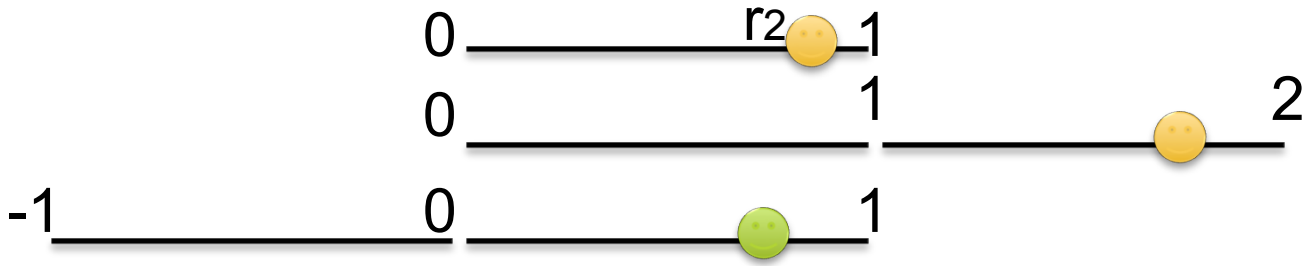
Picking randomly a point in the square

Timeline

```
1.   program test
2.   Implicit none
3.   real(8)        :: r,s
4.   real(8)        :: x, y

5.    call random_number( r )
6.    call random_number( s )
7.
8.     x  =  [Fill in.. ]  *  r   -  [ Fill in ..]
9.     y  =  [Fill in.. ]  *  s   -  [ Fill in ..]
10.  end program
```

- Variable r is spanning which interval?
- How can I obtain x spanning the invertal [-1,1]?



Your turn...

Timeline

```
1.    program test
2.    ……fill in……
3.    real(8)        :: r,s
4.    real(8)        :: x,y
5.    integer(4)     :: i

6.    do i=1,N

7.      call random_number( r )
8.      call random_number( s )
9.      x=…..???.....
10.     y=…. ??? .....
11.     write(*,*) '(x,y)=',x,y

12.   end do

13.   end program
```

- Variable r is spanning which interval?
- How can I obtain x spanning the invertal [-1,1]?
- How can I make sure that I defined all my variables?
- What is wrong with my code? It does not do anything?
- How many output lines in the terminal do I obtain from this program

1. Subroutine

2. Fortran internal random subroutine

3. Monte Carlo code

Timeline

```
1.    program montecarlo
2.    …….fill in…….
3.    real(…fill in ...)          :: r,s
4.    real(8)                     :: x,y
5.    integer(4)                  :: i

6.    Nhit=0
7.    Ntotal=0

8.    do i=1,N
9.      call random_number( r )
10.     call random_number( s )
11.     x=…..???.....
12.     y=…. ??? .....

13.     if( x**2 + y**2 < 1)then
14.       …..
15.     else
16.       …..
17.     endif
18.     write(*,*) ‘(x,y)=‘,x,y
19.   end do

20.   end program
```

Your turn...

- How can I keep track of how many points fall within the circle?
- Nhit and Ntotal, why are those set to zero above the “do loop”?
- What did I forget to do with my variables Nhit and Ntotal?
- What is this condition?

Timeline

```
1.  program montecarlo
2.  ……fill in……
3.  real(… fill in …)          :: r,s
4.  Real(8)                    :: x,y
5.  integer(4)                 :: i
6.  Nhit=0
7.  Ntotal=0
8.  do i=1,N
9.    call random_number( r )
10.   call random_number( s )
11.   x=……???.....
12.   y=…. ??? .....
13.   if( x**2 + y**2 < 1)then
14.     …..
15.   else
16.     …..
17.   endif
18.   write(*,*) '(x,y)=',x,y
19. end do
20. Write(*,*) 'Total number of hits: ', Nhit
21. Write(*,*) 'Area of circle is : ', …..fill……
22. end program
```

Your turn.

- Where should I add a command "write" to show the progression during the do loop, and show what is the area of the circle obtained from the "i" first random trials?

Timeline

```
program montecarlo
 ……..fill in……. .
 real(… fill in …)          :: r,s
 Real(8)                    :: x,y
 Integer(4)                 :: i
 Nhit=0
 Ntotal=0
 do i=1,N
  call random_number( r )
  call random_number( s )
  x=… ..???.......
  y=… . ??? …..
  if( x**2 + y**2 < 1)then
   …..
  else
   …..
  endif
  write(*,*) ‘(x,y)=‘,x,y
  write(*,*) Nhit/Ntotal
 end do
 Write(*,*) ‘Total number of hits: ‘, Nhit
 Write(*,*) ‘Area of circle is : ‘, …..fill…..
 end program
```

Your turn.

- Is this correct?
- How can you modify this line such that I write the output into a file fort.100 and plot it after the end of my calculations?
- Note: once you correct this line, you will obtain an output file that you can plot with the command:
  - **xmgrace** fort.100

Problems to be solved

* **To make your life simpler:** we provide you with a code to fill in, you can obtain it from Keats (montecarlo_template.f90)

* **Exercice 1**: surface of a circle

* **Exercice 2**: volume of a sphere