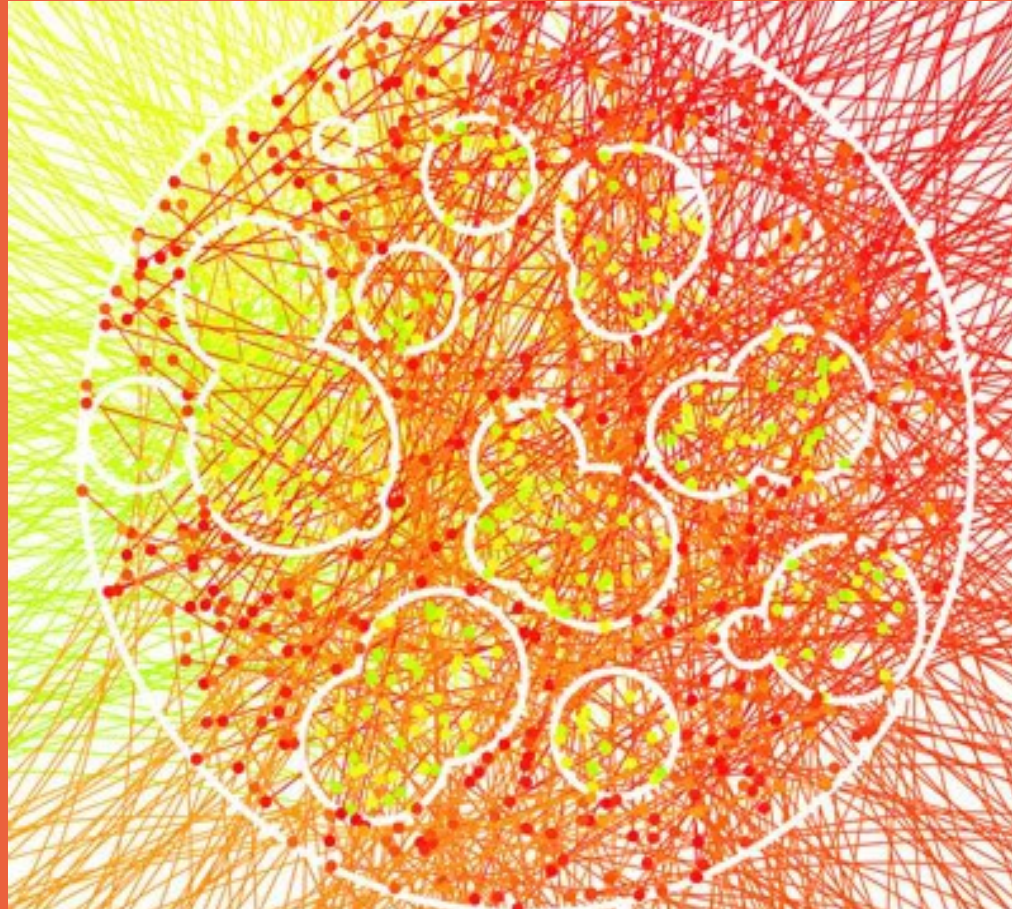


# MONTE CARLO METHODS



**D. Guirado**

**Instituto de Astrofísica de Andalucía – CSIC, Granada, Spain**

**Polarization school, Aussois, 06/06/2013**

# STOCHASTIC ALGORITHMS

**PROBLEM = GET OUTPUT FROM INPUT THROUGH AN ALGORITHM**

Deterministic algorithm:

Input<sub>1</sub> → Output<sub>1</sub>

Stochastic algorithm:

Input<sub>1</sub> + random decision → Output<sub>1</sub>  
→ Output<sub>2</sub>  
→ Output<sub>3</sub>  
...

# MONTE CARLO METHODS

## Definition:

Iterative method including random decisions that gives the correct solution of a problem with a certain probability  $< 1$ .



Named after the district of Monte Carlo (Monaco), European gambling capital.



Monte Carlo




Monte Carlo casino

## Origin:

Study of the diffusion of neutrons in a fusion experiment (Los Álamos Laboratory).

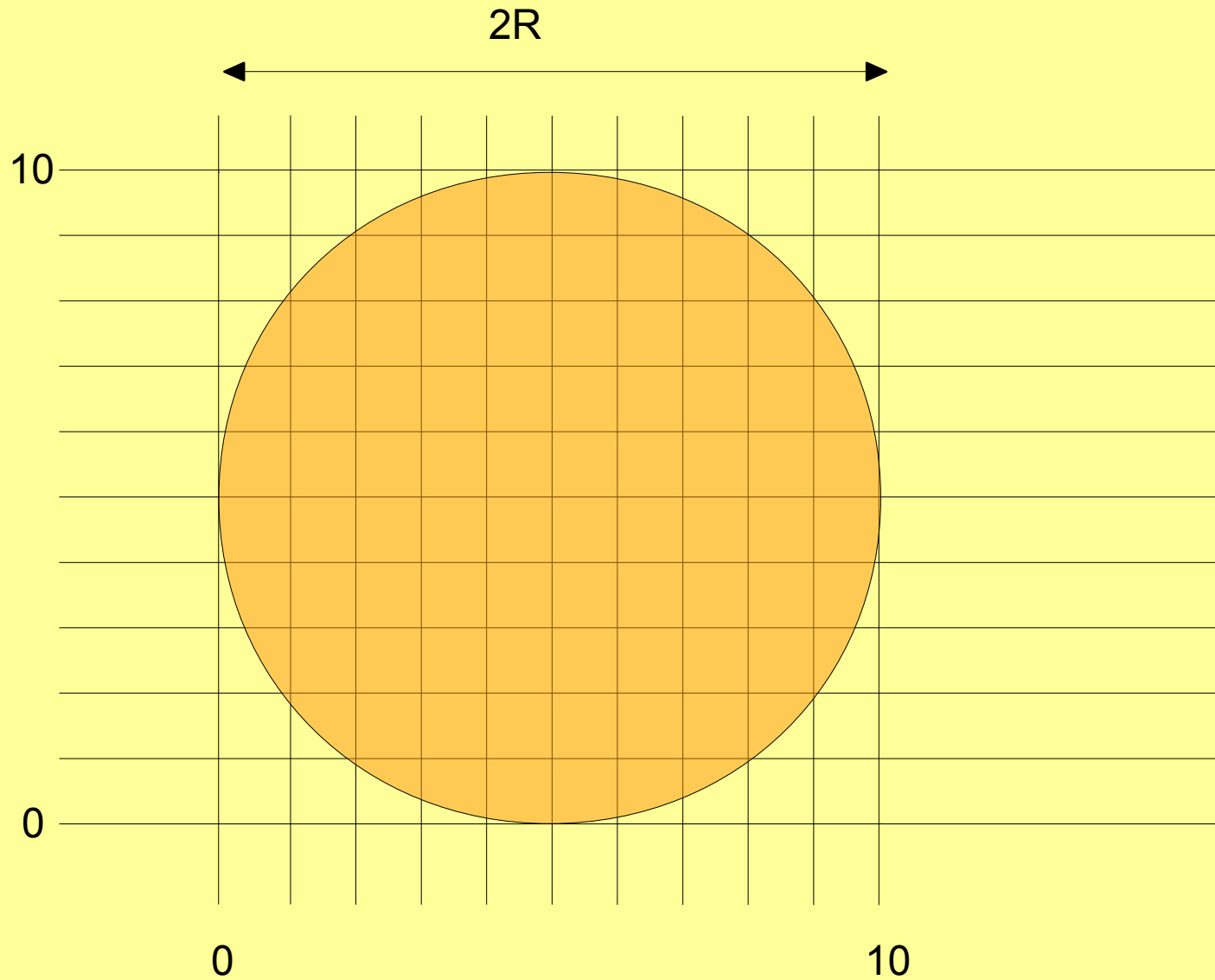


# TYPICAL MONTE CARLO METHOD

Input parameters +  $\xi_i$    $x_i$

$$X = \frac{1}{N} \sum_{i=1}^N x_i$$

# PRACTICE #1: VALUE OF $\pi$



$$\left\{ \begin{array}{l} Area = \pi R^2 \\ Area = 4R^2 \frac{N(\text{darts inside})}{N(\text{darts thrown})} \end{array} \right\} \Rightarrow \pi = 4 \frac{N(\text{darts inside})}{N(\text{darts thrown})} = 4 \frac{1}{N} \sum_{i=1}^N x_i$$

$N = N(\text{darts thrown})$   
 $x_i = \begin{cases} 1 & \text{if dart goes inside} \\ 0 & \text{if dart goes out} \end{cases}$

## STOCHASTIC METHODS WITH COMPUTERS? HOW?

Stochastic method = generating random number + performing operation

$$N \rightarrow \infty \Rightarrow t \rightarrow \infty$$

Computers make operations much faster.

Computer are deterministic, no random numbers.

Solution: *pseudo-random numbers*.

## PSEUDO-RANDOM NUMBERS

Finite (but large) lists of numbers generated by an algorithm.

No correlations between the numbers.

Examples:

- *Numerical Recipes*.
- *MKL* (optimized for *iFort*)
- <http://www.psychicscience.org/random.aspx>

*ran2* (Numerical Recipes)

- For any *Fortran* or *C* compiler.
- The authors offer \$1000 to the first person who finds a statistical test that proves that *ran2* fails.

## CENTRAL LIMIT THEOREM

$\rho(x) =$  Any probability density function with average  $\mu$  and variance  $\sigma^2$ .  
 $x_1, x_2, \dots, x_N =$  collection of  $N$  values sampled through  $\rho$ .

Then, the probability density distribution  $\rho_N(X)$ , which elements are

$X_N = \frac{1}{N} \sum_{i=1}^N x_i$  has a average of  $\mu$  and a variance of  $\frac{\sigma^2}{N}$ , where

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2.$$

## LAW OF LARGE NUMBERS

$N \rightarrow \infty \Rightarrow \rho_N \rightarrow \text{gaussian.}$

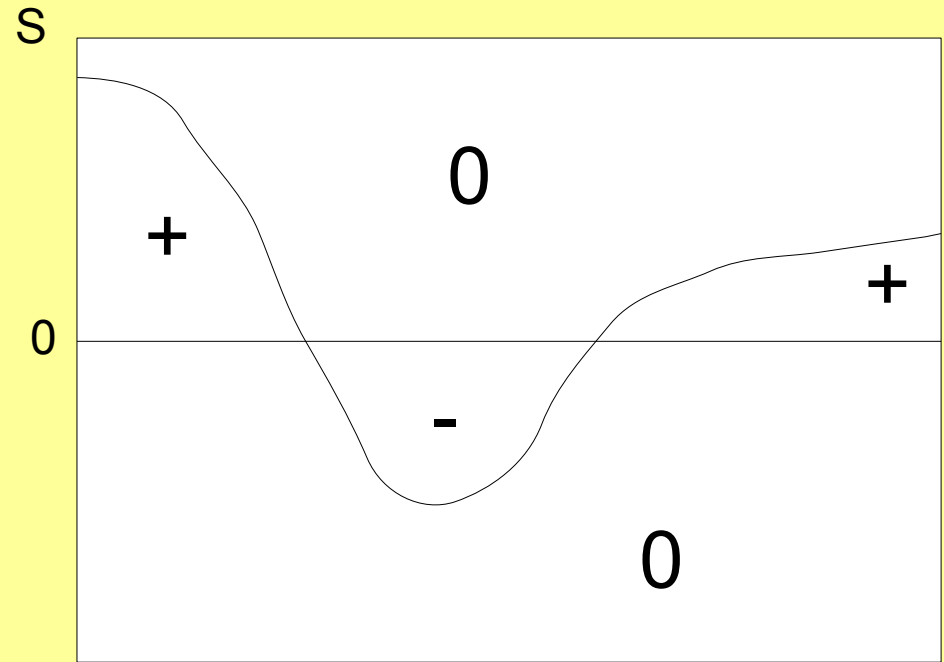
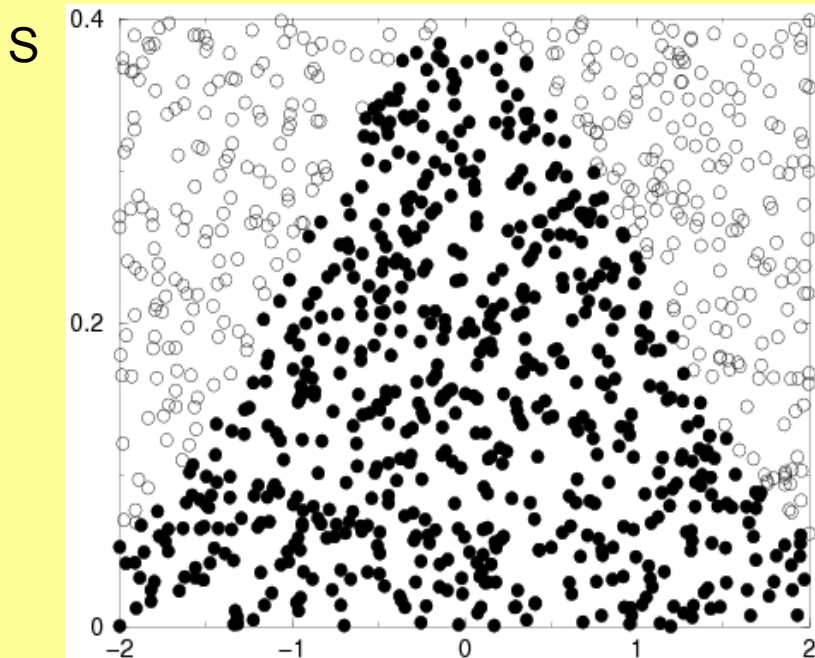
## HOW TO CALCULATE THE ERROR IN A MONTE CARLO METHOD

$$(N \rightarrow \infty)$$

- Fit all calculated  $x_i$  to a gaussian distribution and obtain  $\sigma$ . Let us call the correct solution of the problem  $\mu$ . Then:
- $X \in \left[ \mu - \frac{\sigma}{\sqrt{N}}, \mu + \frac{\sigma}{\sqrt{N}} \right]$  with  $p = 0.682$ .
- $X \in \left[ \mu - 2 \frac{\sigma}{\sqrt{N}}, \mu + 2 \frac{\sigma}{\sqrt{N}} \right]$  with  $p = 0.954$ .
- $X \in \left[ \mu - 3 \frac{\sigma}{\sqrt{N}}, \mu + 3 \frac{\sigma}{\sqrt{N}} \right]$  with  $p = 0.998$ .
- If the accuracy is not enough, increase  $N$  (the error decreases as  $\sim \frac{1}{\sqrt{N}}$ ).

RESULTS FROM MONTE CARLO SIMULATIONS HAVE ERRORS!

## PRACTICE #2: INTEGRATION



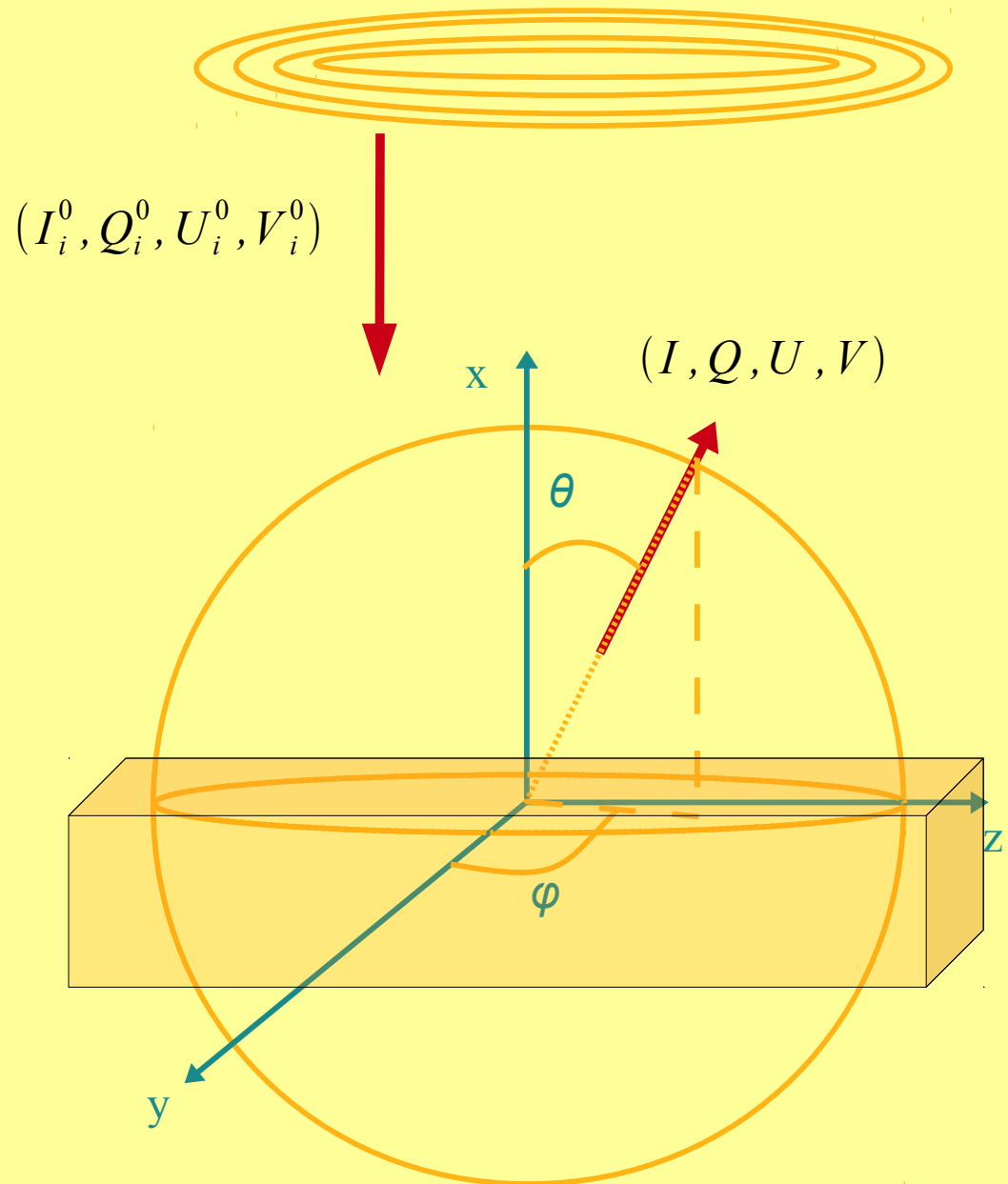
$$\text{Integral} = S \frac{N(\text{darts inside positive}) - N(\text{darts inside negative})}{N(\text{darts thrown})} = S \frac{1}{N} \sum_{i=1}^N x_i$$

$$N = N(\text{darts thrown})$$

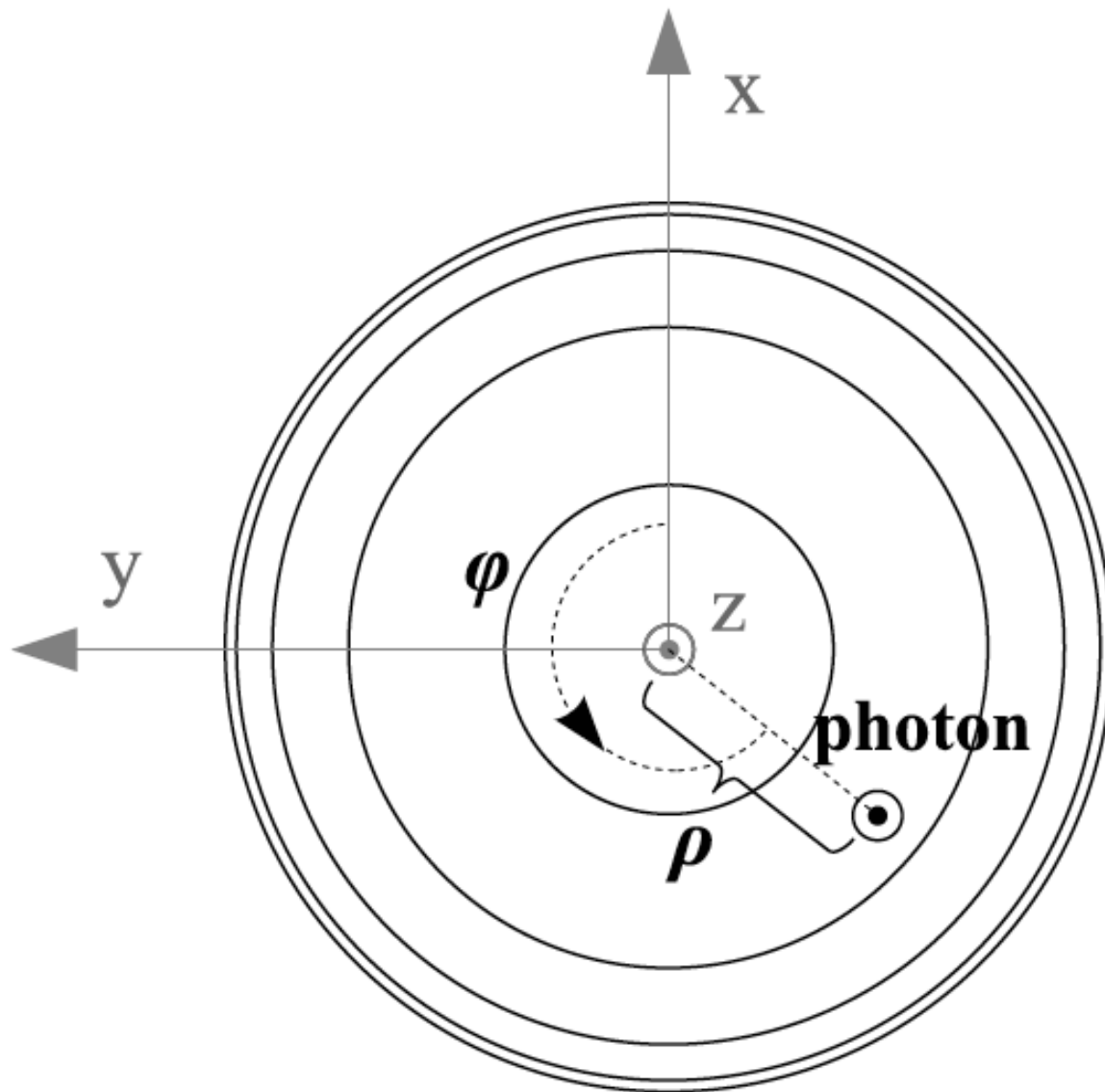
$$x_i = \begin{cases} 0 & \text{if dart goes outside} \\ 1 & \text{if dart goes inside positive} \\ -1 & \text{if dart goes inside negative} \end{cases}$$

# PRACTICE #3: LINEAR POLARIZATION IN SCATTERING

$$(I, Q, U, V) = \frac{1}{N} \sum_{i=1}^N (I_i^0, Q_i^0, U_i^0, V_i^0)$$

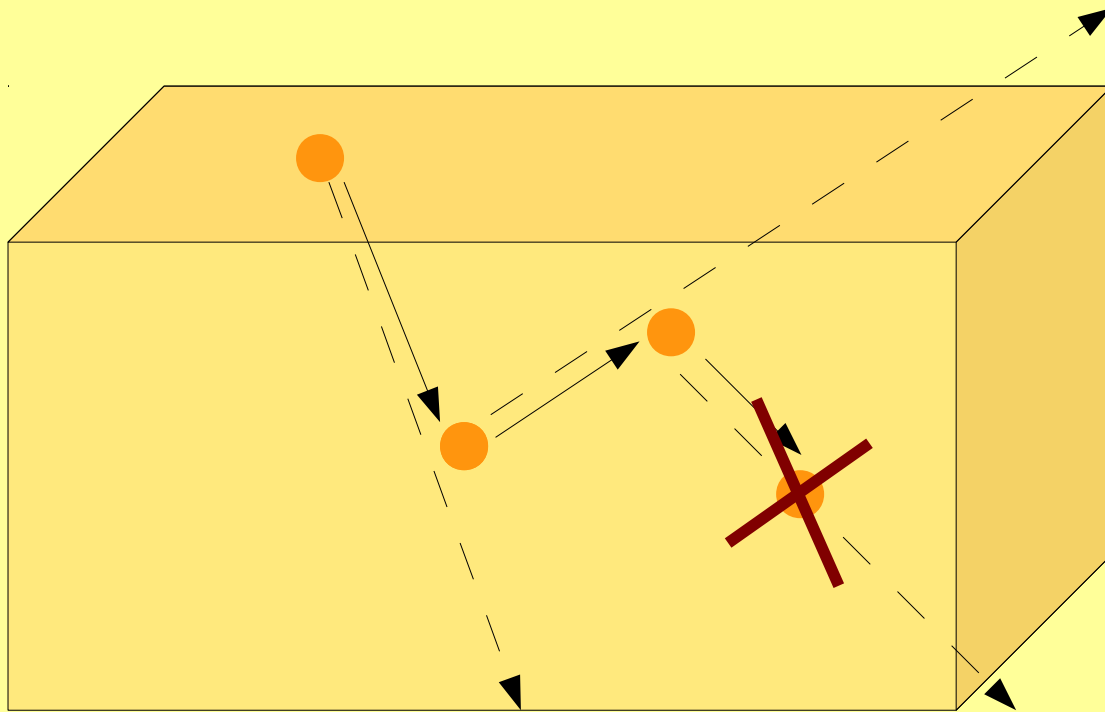


## PRACTICE #3: LINEAR POLARIZATION IN SCATTERING



$$\varphi \in [0, 2\pi]$$
$$\rho = \rho_1 + \sqrt{\xi(\rho_2^2 - \rho_1^2)}$$

## PRACTICE #3: LINEAR POLARIZATION IN SCATTERING



### Particles:

- Any size distribution.
- Any refractive index (even optically activity).
- Any geometry.
- Any orientation (aligned or not).

Any optical thickness of the coma (single or multiple scattering).

Inhomogeneous coma.

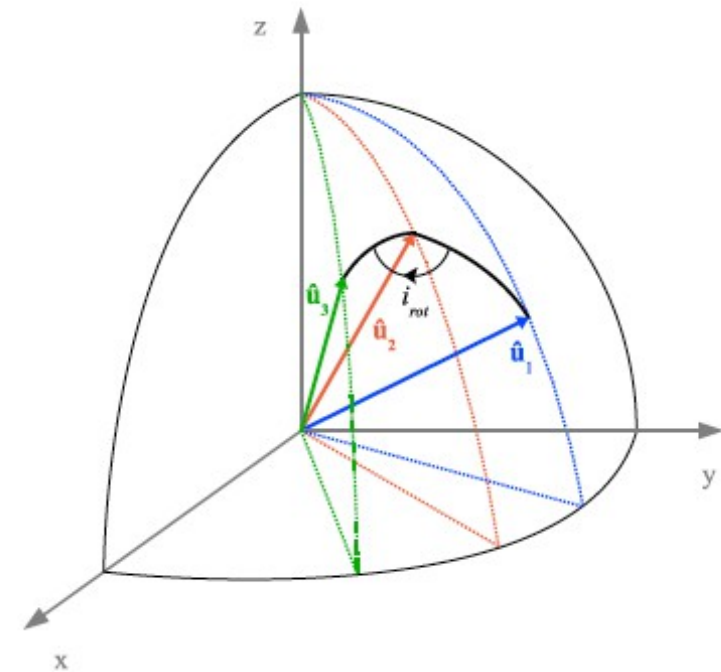
Packet of photons ( $W$ ) {  
- Improves statistics.  
- Reduces computation time.

$W=1$  at the beginning  
Ends when  $W < W_{min}$

## PRACTICE #3: LINEAR POLARIZATION IN SCATTERING

- $\varphi$  uniformly randomly obtained.
- Choosing the scattering direction:  
 $p(\theta) \propto [F_{11}I_0 + F_{12}Q_0 + F_{13}U_0 + F_{14}V_0] \sin \theta$ .
- Change of the scattering plane.

$$\begin{pmatrix} I \\ Q \\ U \\ V \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 2i_{rot} & \sin 2i_{rot} & 0 \\ 0 & -\sin 2i_{rot} & \cos 2i_{rot} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} I_0 \\ Q_0 \\ U_0 \\ V_0 \end{pmatrix}$$



- Multiplying by the scattering matrix.

## SOME COMPUTATIONAL CONSIDERATIONS

\$~top

```
example — TOP — 80x24
TOP
Processes: 109 total, 3 running, 2 stuck, 104 sleeping, 639 threads 12:44:46
Load Avg: 2.44, 2.46, 2.16 CPU usage: 3.3% user, 4.67% sys, 92.28% idle
SharedLibs: 420K resident, 0B data, 0B linkedit.
MemRegions: 36453 total, 1013M resident, 61M private, 958M shared.
PhysMem: 1337M wired, 2671M active, 272M inactive, 4279M used, 3911M free.
VM: 233G vsize, 1091M framework vsize, 1038261(0) pageins, 61409(0) pageouts.
Networks: packets: 2032517/2135M in, 1555298/199M out.
Disks: 2611533/587G read, 2847308/732G written.

PID      COMMAND      %CPU  TIME    #TH  #WQ  #POR  #MREG  RPRVT  RSHRD  RSIZE
42594    TOP          5.9   00:00.46 1/1  0    28   29    1248K  216K   1956K
42585    cupsd        0.0   00:00.04 3    1    47   61    2884K  220K   4568K
42574    cookied      0.0   00:00.01 2    1    42   54    572K   224K   1320K
42573    xpchelper    0.0   00:00.01 2    2    38   55    1056K  220K   4672K
42570    quicklookd  0.0   00:00.60 6    1    99   177   18M    11M    32M
42493    mdworker     0.0   00:00.07 3    1    53   77    1696K  28M    7272K
42432    bash         0.0   00:00.03 1    0    20   24    460K   216K   1240K
42431    login        0.0   00:00.03 2    1    33   63    1000K  216K   2268K
42429    Terminal    0.2   00:01.29 5    1    120  208+  5992K+ 28M    19M+
41277    distnoted   0.0   00:02.61 4    3    40   58    2516K- 240K   3216K-
41273    launchd     0.0   00:00.18 2    0    56   49    856K   412K   1084K
41261    ARDAgent    0.0   00:00.06 4    1    72   90    1452K  25M    4176K
40796    usbmuxd     0.0   00:00.05 3    1    43   62    1000K  220K   2604K
40068    TextEdit    0.0   00:01.37 3    2    130  187   4508K  21M    17M
```

# PARALLELIZE YOUR CODE

Classical method: MPI. New easier alternative: *OpenMP* (Fortran and C).

How it works:

```
PROGRAM OMP_SUM2
INTEGER NMAX
PARAMETER(NMAX=20000)
INTEGER I
REAL A(NMAX),C(NMAX)
DO I = 1,NMAX
  A(I) = I * 1.0
  C(I)=1.
ENDDO
C$OMP PARALLEL shared(A,B,C,NMAX) private(I,J)
C$OMP DO
  DO I = 1,NMAX
    DO J=1,I
      C(I)= C(I)+A(J)
    END DO
  ENDDO
C$OMP END DO
C$OMP END PARALLEL
WRITE(*,*)C(NMAX)
END
```

Compile: `$~gfortran omp_sum2.f -o omp_sum2`



Sequential omp\_sum2

Compile: `$~gfortran -fopenmp omp_sum2.f -o omp_sum2`



Parallel omp\_sum2

Execute: `$~time ./omp_sum2`

**Download**

**[http://www.iaa.es/~dani/swap/monte\\_carlo.zip](http://www.iaa.es/~dani/swap/monte_carlo.zip)**