Laboratorio di Fisica Computazionale

FI020004-4

Numeri casuali e approcci stocastici

Maria Peressi
Università degli Studi di Trieste - Dipartimento di Fisica
Sede di Miramare (Strada Costiera 11, Trieste)
e-mail: peressi@units.it

(molto materiale è frutto di una lunga collaborazione con il collega prof. G. Pastore che ringrazio!)

- numeri pseudocasuali: proprietà e generazione
- distribuzioni non uniformi (alcuni algoritmi generali; teorema del limite centrale e distribuzione gaussiana [lancio di piu' dadi])
- calcolo di aree/integrali definiti (π)

Simulazioni di fenomeni intrinsecamente casuali:

- cammini casuali
- decadimento radioattivo
- moto browniano

Numeri casuali

Cos’e’ un numero casuale?

3 e’ un numero casuale?

Numeri casuali

Cos'e' un numero casuale?

3 e' un numero casuale?

NO!!! Non si puo' dire nulla di un numero singolo

Numeri casuali

Cos'e' un numero casuale?

3 e' un numero casuale?

NO!!! Non si puo' dire nulla di un numero singolo

(A) 0 1 2 3 4 5 6 7 8... (B) 4 8 6 7 2 0 1 5 3...

La **sequenza** (B) ci appare casuale, (A) no...

Una sequenza di numeri casuali e' una sequenza di numeri che "sembrano" impredicibili, da cui non si riesce ad estrarre alcuna regolarita', pero'...

Giochiamo con un dado

Se lanciamo piu' volte un dado a 6 facce (non truccato!) otteniamo una sequenza di numeri casuali tra 1 e 6 (...)

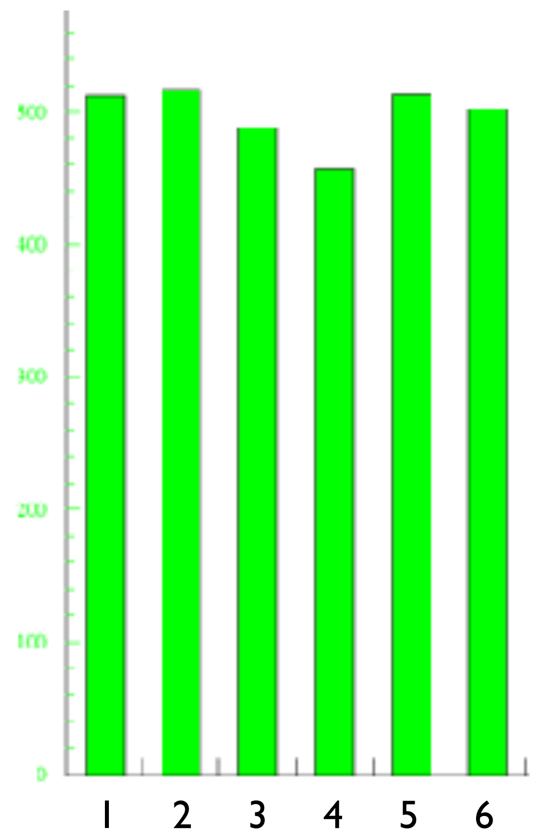
I singoli risultati ci interessano poco, ma possiamo chiederci:

- istogramma delle frequenze ?
- valore medio ?

cioe' **proprieta' statistiche della sequenza**

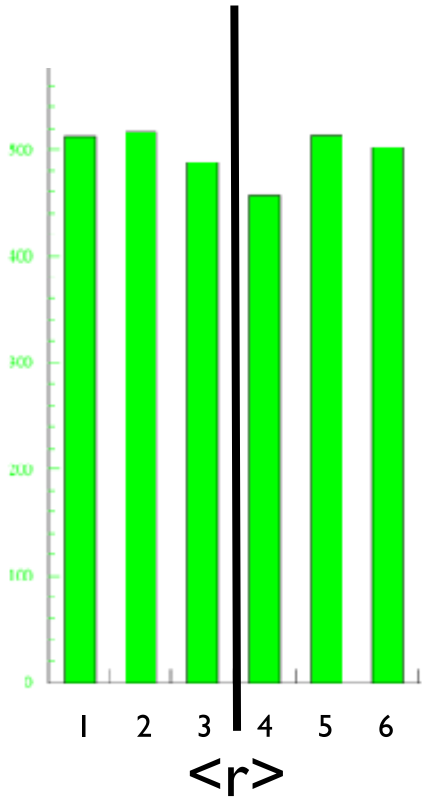
Proprieta' statistiche dei numeri casuali
Istogramma di frequenze

Contiamo quante volte ogni numero e' uscito:



esempio di **distribuzione uniforme**

Proprieta' statistiche dei numeri casuali
Valor medio



dalla distribuzione uniforme
di numeri interi in [1,6],
calcolabile la MEDIA:
$<r> = 3$

Quindi:

Una sequenza di numeri casuali e' una sequenza di numeri che sembrano impredicibili ma che hanno ben definite proprieta' statistiche

Dettagli: come generare numeri casuali

metodi manuali:

- lancio di dadi, di monete...

- estrazione da urne, da tabelle...

Ottime proprieta' di casualita', ma...

lunghi !!! praticamente inutilizzabili!!!

- dispositivi elettronici ad hoc...

NO! altri problemi...

generazione con il computer:

Generazione mediante metodi aritmetici

Generazione di numeri “casuali” (o meglio: “pseudocasuali”) con il computer

Generazione al computer

Un computer (deterministico!) non puo' generare dei numeri rigorosamente casuali...

... ma possiamo scrivere un algoritmo che generi una sequenza di numeri avente le stesse proprietà statistiche di una sequenza di numeri davvero casuali, con distribuzione uniforme e non correlazione tra i numeri generati: NUMERI PSEUDOCASUALI

In pratica: stesso valore di una sequenza realmente casuale!
Vantaggi: rapidita', possibilita' di ricreare la stessa sequenza fornendo all'algoritmo lo stesso **valore iniziale ("seme")**

un metodo possibile

('middle square', John Von Neumann, 1946)

Per generare una sequenza di numeri con 10 cifre:

si parte da un numero, si eleva al quadrato e si prendono le 10 cifre centrali , etc. etc..

Esempio: 5772156649**2 = 33317792380594909291

queste cifre centrali sono il prossimo numero

NB: la sequenza non e' veramente casuale, perche' ogni numero e' determinato dal precedente. Ma 'sembra' casuale!

Possibili problemi:
altro esempio ,"brutto":
6100**2 = 37210000
2100**2 = 4410000
4100**2 = 16810000
8100**2 = 65610000

un altro metodo: LCM
(linear congruential method)

$$x_{n+1} = \text{resto intero di } \left(\frac{a \cdot x_n + c}{m} \right) \quad (\text{"modulo"})$$

x_0 : valore iniziale ("seme" o "seed")

a, c, m : opportunamente scelti

$a, c \geq 0, \quad m > x_0, c, a$

Esempio "brutto":

$a=c=x_0=7$, $m=10$: sequenza: {7, 6, 9, 0, 7, 6, 9, 0, ...}

LCM: con operatore “modulo” in C++

esempio:

```
#include <iostream>
using namespace std;

int main()
{
    int M = 8;
    int a = 5;
    int c = 3;
    int X = 1;
    int i;
    for(i=0; i<8; i++)
    {
        X = (a * X + c) % M;
        cout << X << “ “;
    }
    return 0;
}
```

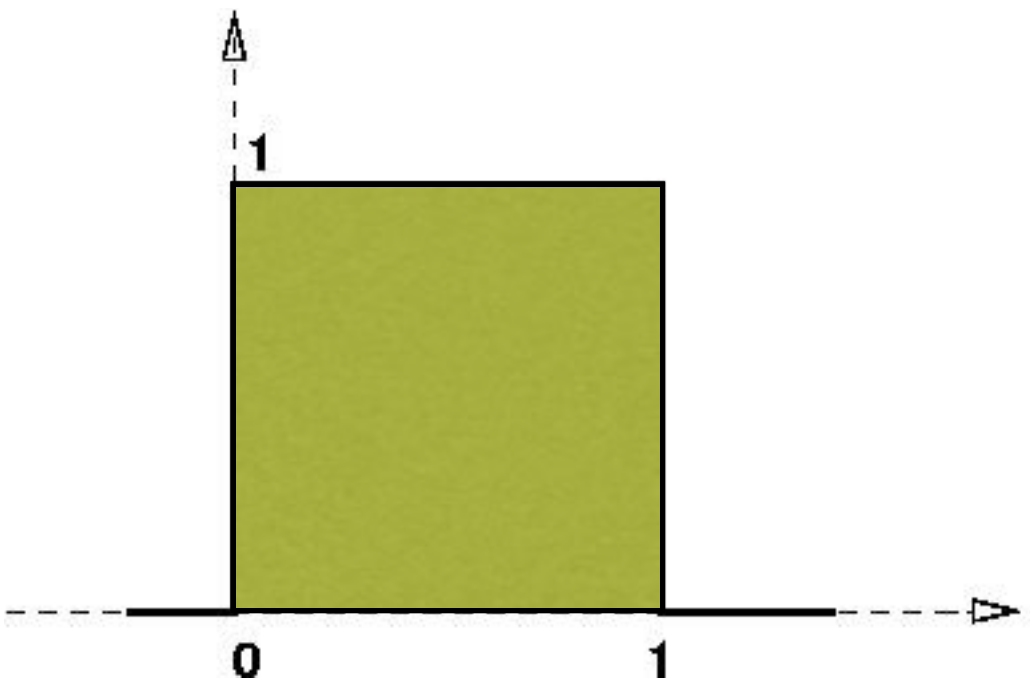
LCM: con operatore “modulo” in Fortran90

esempio:

```
module lcm

contains

  subroutine lcm_seq(a, c, m, seed, seq)
    !! Generate a sequence of pseudo-random numbers using the LCM algorith
    !! Input: the parameters a, c, m and the seed for the algorithm.
    !! Output: an array holding the first m + 1 numbers generated.
    implicit none
    integer, intent(in) :: a, c, m, seed
    integer, dimension(0:m), intent(out) :: seq
    integer :: i
    seq(0) = seed
    do i = 1, m
       seq(i) = mod((a*seq(i-1) + c), m)
    end do
  end subroutine lcm_seq

end module lcm
```

Generazione al computer

Utilizzo di funzioni/subroutine intrinseche o comunque già disponibili (migliori !) che generano numeri casuali tra [0,1] con distribuzione uniforme



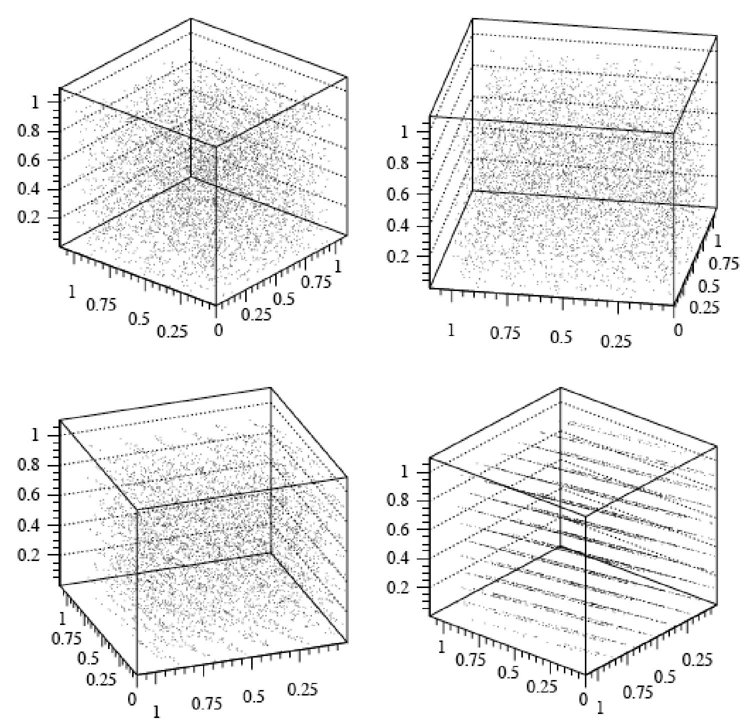
Generazione al computer

Utilizzo di funzioni/subroutine intrinseche o comunque già disponibili (migliori !) che generano numeri casuali tra [0,1] con distribuzione uniforme

Results from Randu: 3D distribution

https://en.wikipedia.org/wiki/RANDU

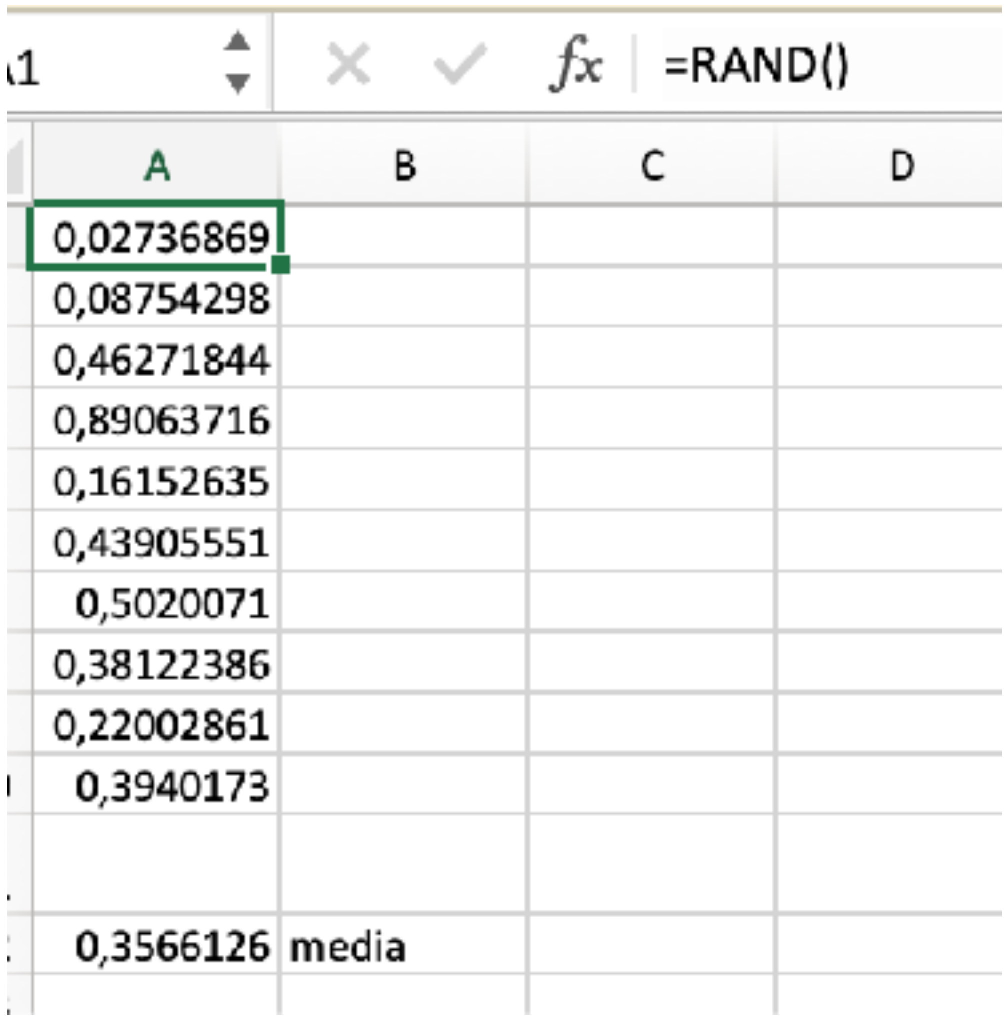
ma oltre all'**uniformità** è importante anche la **non correlazione** !



IBM 1960'

Problem seen when observed at the right angle!

Generazione numeri casuali su foglio excel



=RAND()

A	B	C	D
0,02736869			
0,08754298			
0,46271844			
0,89063716			
0,16152635			
0,43905551			
0,5020071			
0,38122386			
0,22002861			
0,3940173			
0,3566126	media		

Giochiamo con un dado elettronico

facciamo generare al computer
un numero casuale tra 1 e 6

- istogramma delle frequenze ?
- valore medio ?

Gas.zip =>`VariabileCasuale.java`

Giochiamo con due dadi..

lancio simultaneo, e consideriamo la sequenza dei risultati ottenuti dalla somma dei 2 numeri

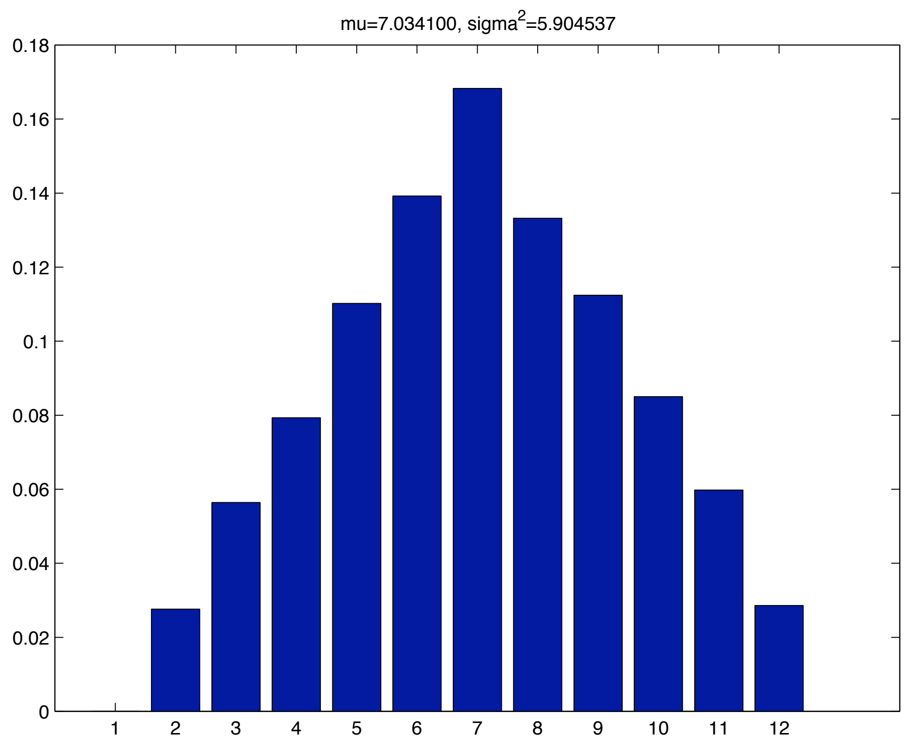
- istogramma delle frequenze: valore piu' probabile?
- valore medio?

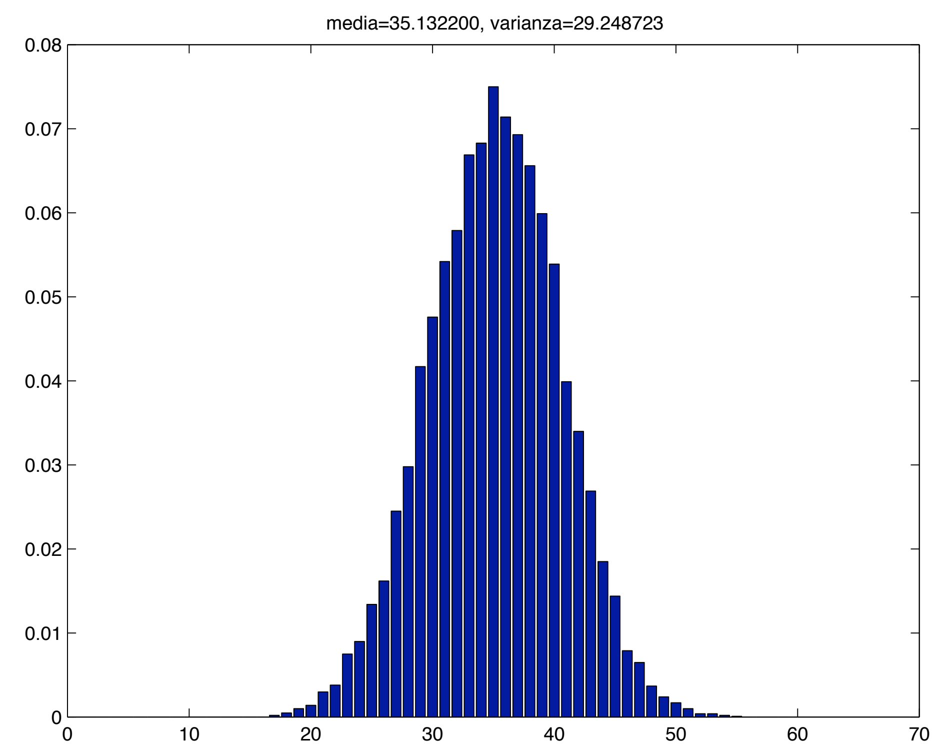
... e con piu' dadi ...

`VariabileCasuale.java`

Giochiamo con due dadi..

risultato degli esperimenti





Dalla figura appare evidente come nel caso di due dadi (sinistra) il punteggio più probabile è 7 (probabilità $6/36 = 1/6$) e quelli meno probabili 2 e 12 (probabilità $1/36$). Inoltre all'aumentare del numero dei dadi (destra) le frequenze tendono ad essere distribuite secondo una campana gaussiana corrispondente ad una distribuzione cosiddetta normale.

un’occhiata a “cosa c’e’ dentro la scatola”...

Numeri random in Java - I

In **Java** si puo’ utilizzare semplicemente la classe **Math**:

```
// valore compreso tra 0 e 1
double x = Math.random();
```

(da’ lo stesso risultato che Random nextDouble() descritto nella prossima diapositiva)

Numeri random in Java - II

oppure la classe **`java.util.Random:`**

```
 import java.util.Random; // da scrivere all'inizio del codice
```

Costruttori Random

```
Random r = new Random();      // per default il seme (seed)
                             // viene preso dal clock della macchina
Random r = new Random(long seed); //qui invece e' fissato per
                             //poter riprodurre una sequenza
```

Metodi Random

```
Tutti i seguenti metodi ritornano un numero random con distribuzione
uniforme, tranne nextGaussian(). Qui x e' un oggetto Random.
type        chiamata            Descrizione
int i =    r.nextInt(int n)    da' un random int tra 0 e n
int i =    r.nextInt()        da' un random int tra 0 e 2**32
long l =   r.nextLong()        da' un random long (full range)
float f =  r.nextFloat()        da' un random float tra 0 e 1
double d =    r.nextDouble()   da' un random double tra 0 e 1
boolean b =   r.nextBoolean()  da' un random boolean (true o false)
double d =    xrnextGaussian() da' un n. random, media 0 e dev. st. 1
```

Esempio uso di Random

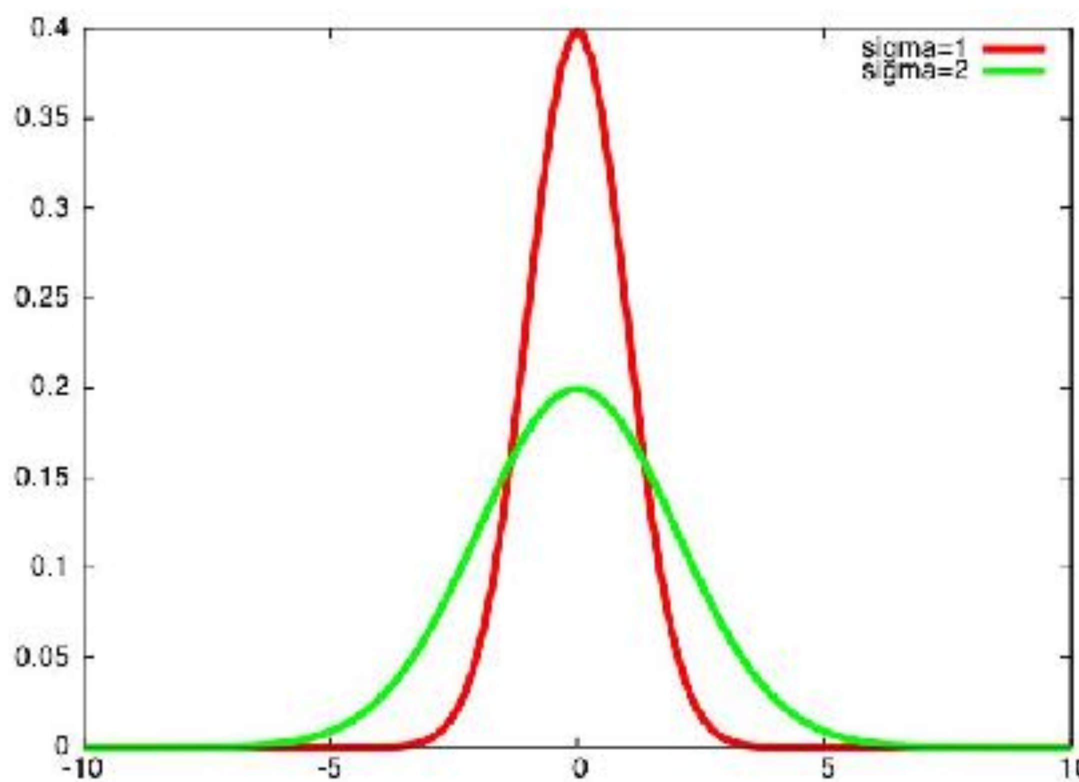
```java
import java.util.Random;
public class Dado  {
 //costruttore che costruisce un  dado
 //  con s facce
    public Dado(int s) {
            facce = s;
            generatore = new Random();
    }

    public int lancia()  {
            return 1 +
            generatore.nextInt(facce);
    }

    private Random generatore;
    private int facce;
}
```

```java
// Questo programma simula 10 lanci del
     dado

public class TestaDado {
    public static void main(String[] args)
     {
     Dado d = new Dado(6);
     final int LANCI = 10;
     for (int i = 1; i <= LANCI; i++)  {
            int n = d.lancia();
            System.out.print(n + " ");
     }
     System.out.println();
    }
 }
```

Distribuzione non uniforme - I

Riprendiamo i risultati dell'esperimento con 2 dadi:
Se consideriamo la somma di N variabili casuali uniformi, l'andamento della distribuzione della variabile "somma" non e' piu' uniforme ma, all' aumentare di N, approssima una distribuzione particolare dal caratteristico profila a forma di campana (distribuzione normale o gaussiana):

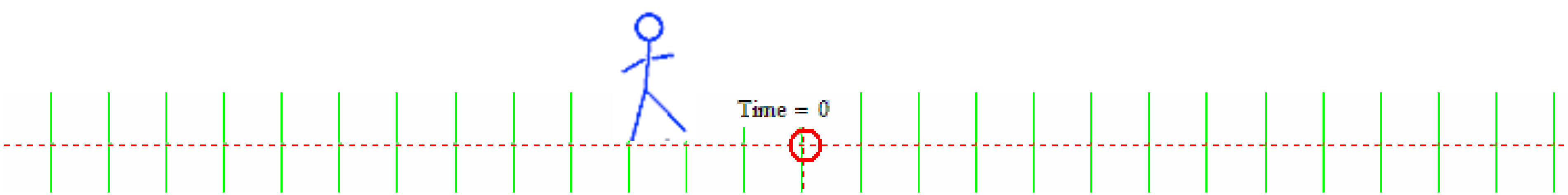
$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{x^2}{2\sigma^2}\right)$$



Distribuzione non uniforme - II

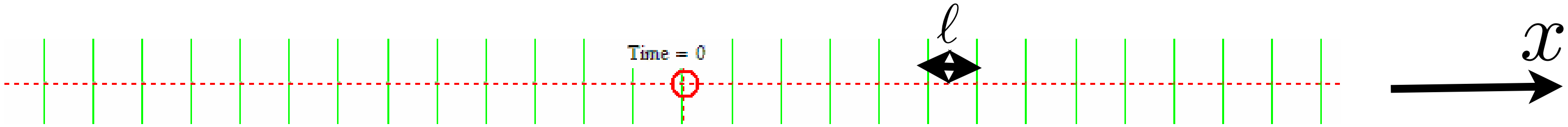
Che significato ha la somma di N variabili casuali uniformi ?

puo' rappresentare ad esempio la posizione finale di un 'camminatore' che fa N passi a caso, a destra o a sinistra con equiprobabilita' ('random walk');
simulazione ad esempio considerando i passi tutti di lunghezza uguale e generando una variabile intera casuale 0 o 1



$$p_{\rightarrow} = p_{\leftarrow} = 50\%$$

random walk 1D



Camminatore che puo' andare a dx o sin ad ogni passo:

N : numero di passi

ℓ : lunghezza del passo

($s_i = \pm\ell$ spostamento relativo al passo i)

x_N : spostamento dal punto di partenza dopo N passi

($x_N = \sum_{i=1}^{N} s_i, \quad x_N \in [-N\ell, +N\ell]$)

$p_\rightarrow, \; p_\leftarrow$: probabilità di spostamento a dx o sin

Cosa possiamo calcolare? **Medie sui camminatori**:

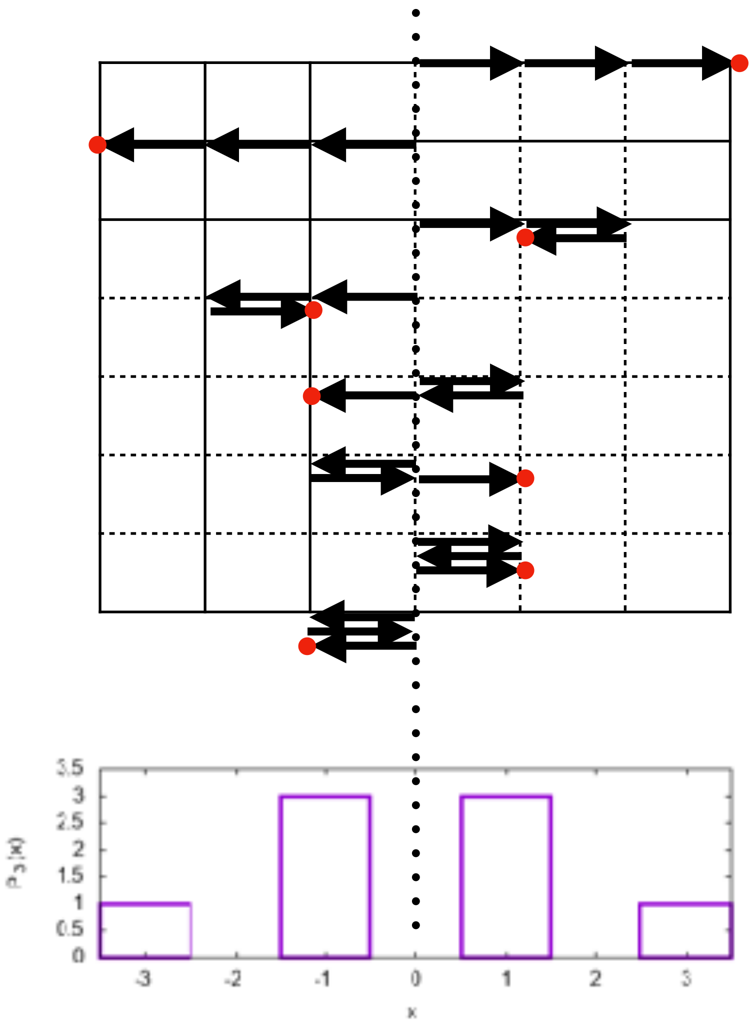
$\langle x_N \rangle$: spostamento medio

$\langle x_N^2 \rangle$: spostamento quadratico medio

$P_N(x)$: probabilità che dopo N passi un camminatore arrivi nella posizione finale x

random walk 1D

N = 3 => 8 possibili differenti cammini



$$\Rightarrow P_3(0) = P_3(\pm 2) = 0; \quad P_3(\pm 1) = 3,\ P_3(\pm 3) = 1$$

random walk 1D

E’ facile calcolare analiticamente la probabilita’ per un ‘random walk’ di n passi di finire in un punto x: si tratta di un triangolo di Pascal con degli zeri !

$\downarrow$ *posizione* x

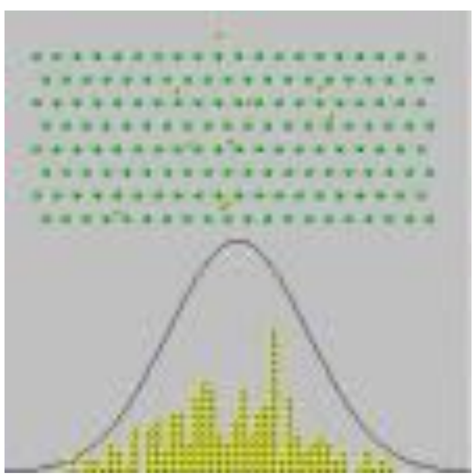
numero di passi $\rightarrow$

$n \setminus x$	-5	-4	-3	-2	-1	0	1	2	3	4	5
0						1					
1					$\frac{1}{2}$	0	$\frac{1}{2}$				
2				$\frac{1}{4}$	0	$\frac{2}{4}$	0	$\frac{1}{4}$			
3			$\frac{1}{8}$	0	$\frac{3}{8}$	0	$\frac{3}{8}$	0	$\frac{1}{8}$		
4		$\frac{1}{16}$	0	$\frac{4}{16}$	0	$\frac{6}{16}$	0	$\frac{4}{16}$	0	$\frac{1}{16}$	
5	$\frac{1}{32}$	0	$\frac{5}{32}$	0	$\frac{10}{32}$	0	$\frac{10}{32}$	0	$\frac{5}{32}$	0	$\frac{1}{32}$

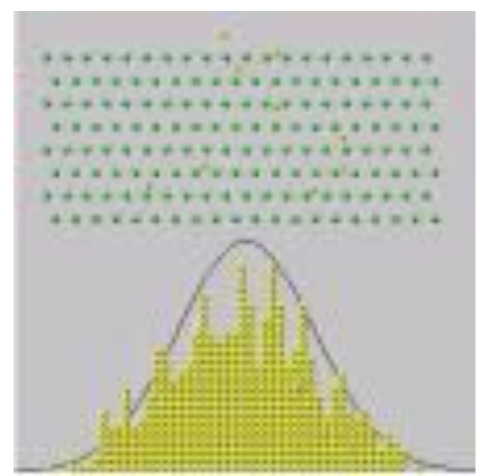
La tavola di Galton

distribuzione di probabilita'
della posizione finale di un 'random walk' 1D

Biglie che partono da una stessa posizione centrale e cadono deviate da chiodi disposti secondo uno schema triangolare. Ogni volta che una sferetta colpisce un chiodo ha il 50% di probabilità di cadere a sinistra e il 50% a destra.
Le palline si accumulano nelle scanalature collocate alla base della struttura, formando delle pile con una certa distribuzione.



poche



tante

In questa simulazione le pile nelle scanalature sono la somma di N=8 variabili casuali, ma se si aggiungono più file di chiodi (cioè se si aumenta N, il numero di variabili casuali), l'andamento della distribuzione approssima quello della distribuzione normale.

random walk 1D: simulazioni - Fortran90

Algoritmo di base:

(1 run= 1 particella= 1 camminatore)

ix = posizione del camminatore

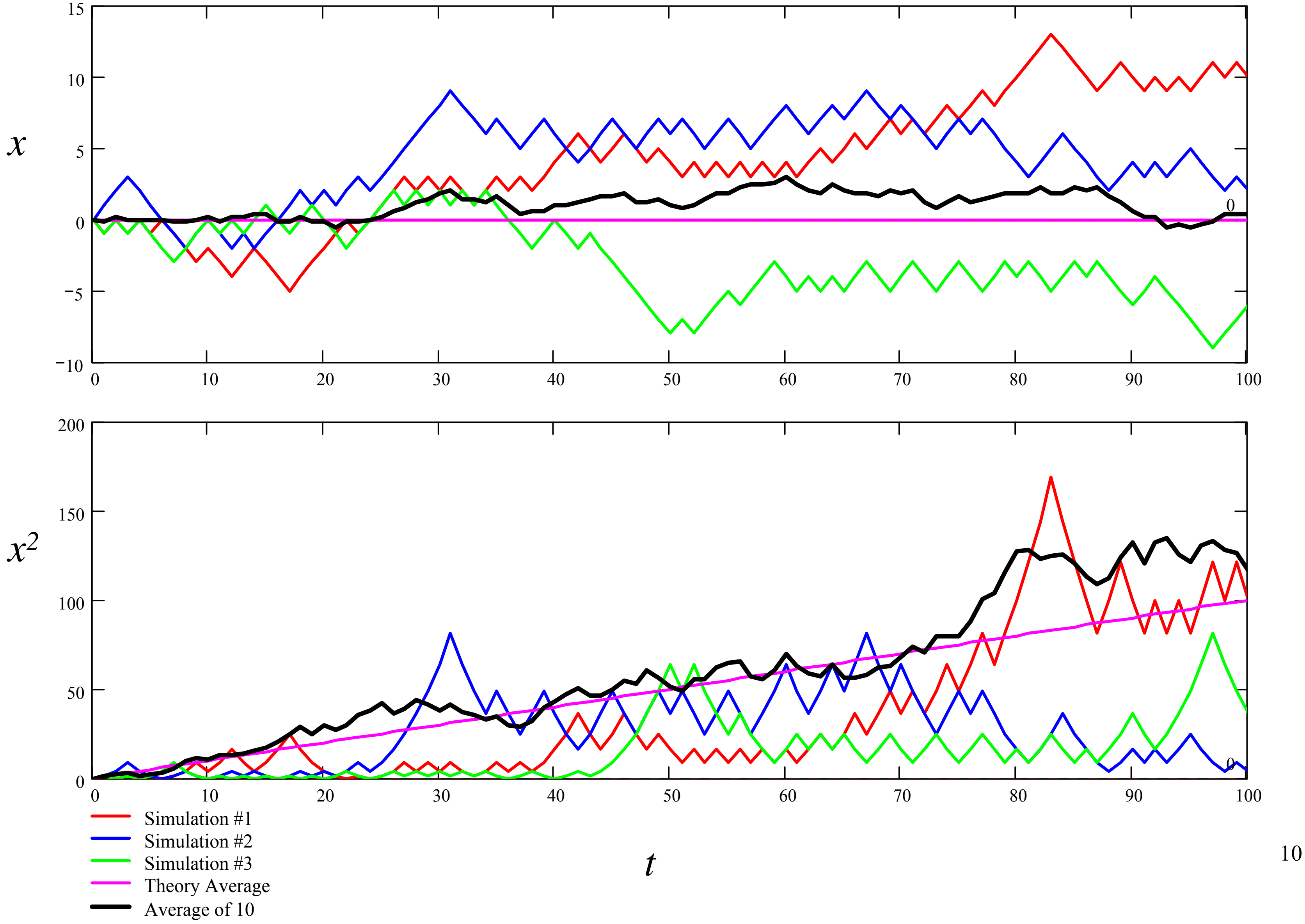
x_N, x2_N = quantità cumulative

rnd(N) = sequenza di N numeri random

```
do irun = 1, nruns
   ix = 0 ! initial position of each run
   call random_number(rnd) ! get a sequence of random numbers
   do istep = 1, N
      if (rnd(istep) < 0.5) then ! random move
         ix = ix - 1 ! left
      else
         ix = ix + 1 ! right
      end if
      x_N (istep) = x_N (istep) + ix
      x2_N(istep) = x2_N(istep) + ix**2
   end do
   P_N(ix) = P_N(ix) + 1 ! accumulate (only for istep = N)
end do
```

But we can monitor what happens for each intermediate step by using arrays x_N() and x2_N() and including the calculation inside the loop on the steps

random walk 1D: risultati



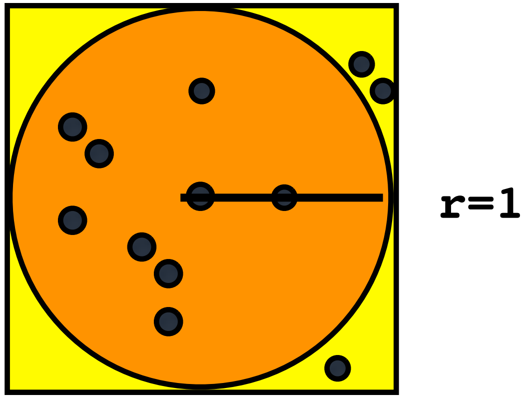
Possiamo fare della matematica o della fisica con i numeri casuali?

calcolo di aree e integrali definiti

Bersagli, freccette, aree e numeri casuali

I numeri casuali servono :

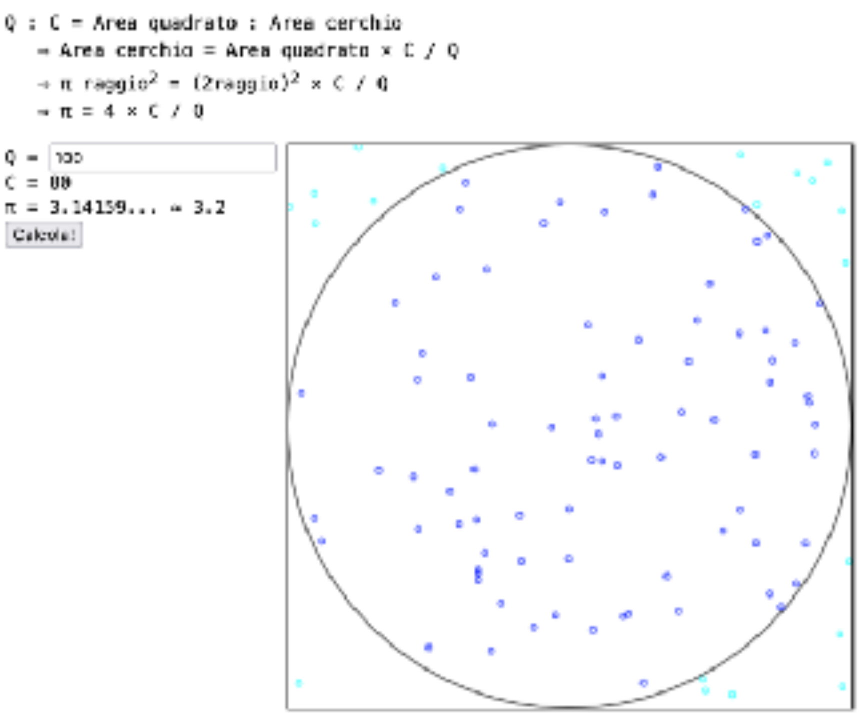
- per calcolare aree... (e il π...) quindi problemi di cui vorremmo soluzione esatta!

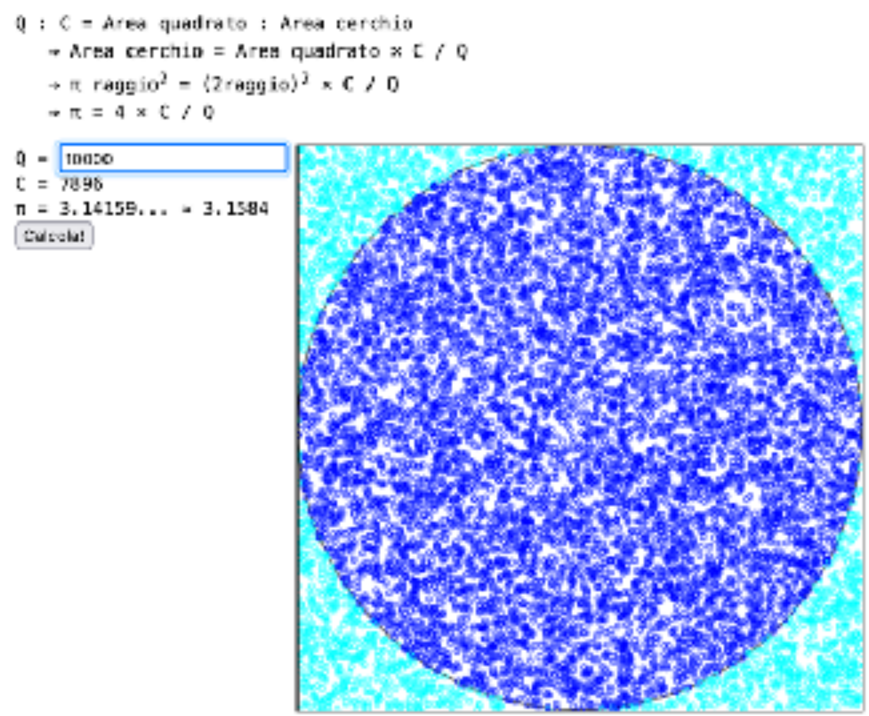


$$\pi = \frac{\text{Area cerchio}}{\text{Area quadrato/4}} = \frac{\text{4*\# punti nel cerchio}}{\text{\# punti nel quadrato}}$$

- (vedi Pi.zip => **`PiMonteCarlo.java`**)

Calcolo di π: risorse on line





http://www.caressa.it/prisma/pioggianelpigreco.html

decadimento radioattivo

$N(t)$ atomi presenti al tempo t
λ probabilità per ogni atomo di decadere in Δt
$\Delta N(t)$ atomi che decadono tra t e $t + \Delta t$

$$\Delta N(t) = -\lambda N(t) \Delta t \quad \Rightarrow \quad N(t) = N(t=0) e^{-\lambda t}$$

Scopo di un possibile esercizio:

- fissare λ (<1 poiché è una probabilità)
- eseguire la simulazione stocastica
- controllare se N(t) ha il comportamento atteso,

calcolando la costante di decadimento da un fit dei dati numerici ottenuti e vedere se corrisponde a λ

decadimento radioattivo

$N(t)$ atomi presenti al tempo t

$\boxed{\lambda}$ probabilità per ogni atomo di decadere in $\boxed{\Delta t}$

$\Delta N(t)$ atomi che decadono tra t e $t + \Delta t$

$$\Delta N(t) = -\lambda N(t) \Delta t \quad \Rightarrow \quad N(t) = N(t=0) e^{-\lambda t}$$

Scopo di un possibile esercizio:

- fissare λ

Nella nostra simulazione: $1 \text{ passo} \quad \Leftrightarrow \quad \Delta t$

(l'istante di tempo è implicitamente fissato in una simulazione Monte Carlo!

in qualche modo, decidiamo la discretizzazione fissando λ)

decadimento radioattivo

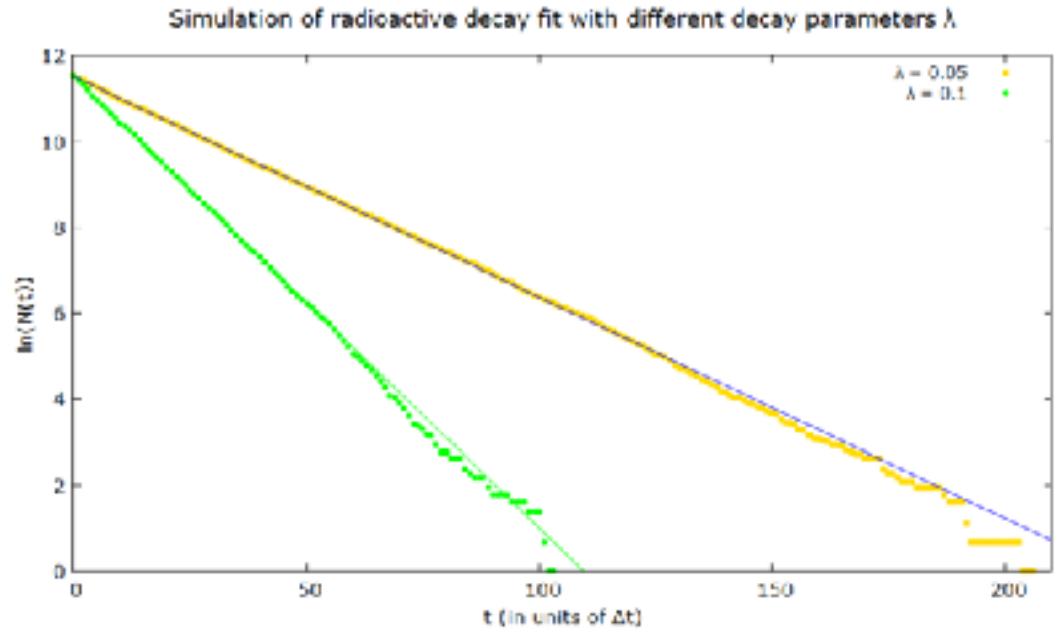


Figura 5: Fit della distribuzione di $N(t)$ in unità di Δt per diversi valori del parametro di decadimento λ.

λ aspettato $(\cdot 10^{-2})$	λ stimato $(\cdot 10^{-2})$	$N(0)$ stimato $(\cdot 10^{3})$	σ_λ
5	5.14 ± 0.04	100 ± 2	0.8 %
10	10.5 ± 0.2	100 ± 3	1.9 %
15	16.1 ± 0.3	99 ± 2	1.9 %
30	35.8 ± 0.7	101 ± 3	2.0 %

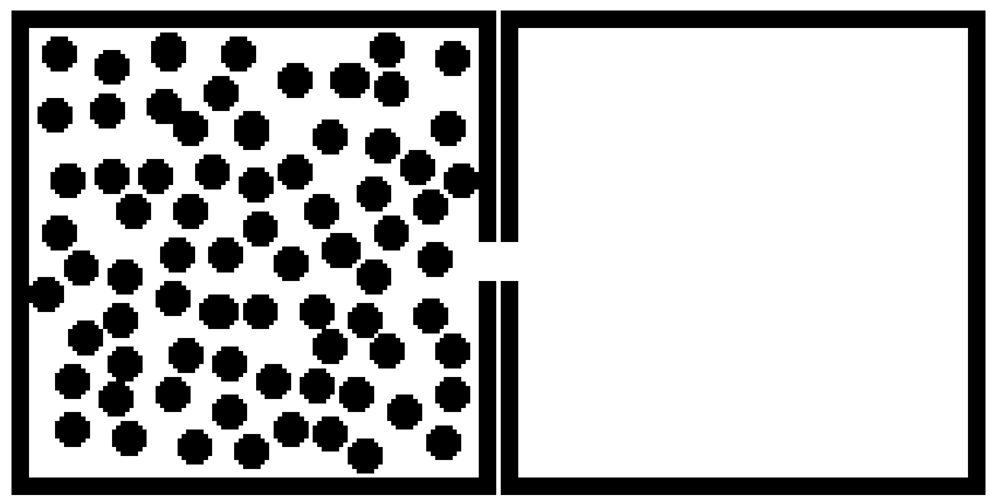
Tabella 3: Stime dei parametri λ e $N(0)$ dal fit della distribuzione di $N(t)$ in scala semilogaritmica.

Scatole, palline, diffusione e numeri casuali

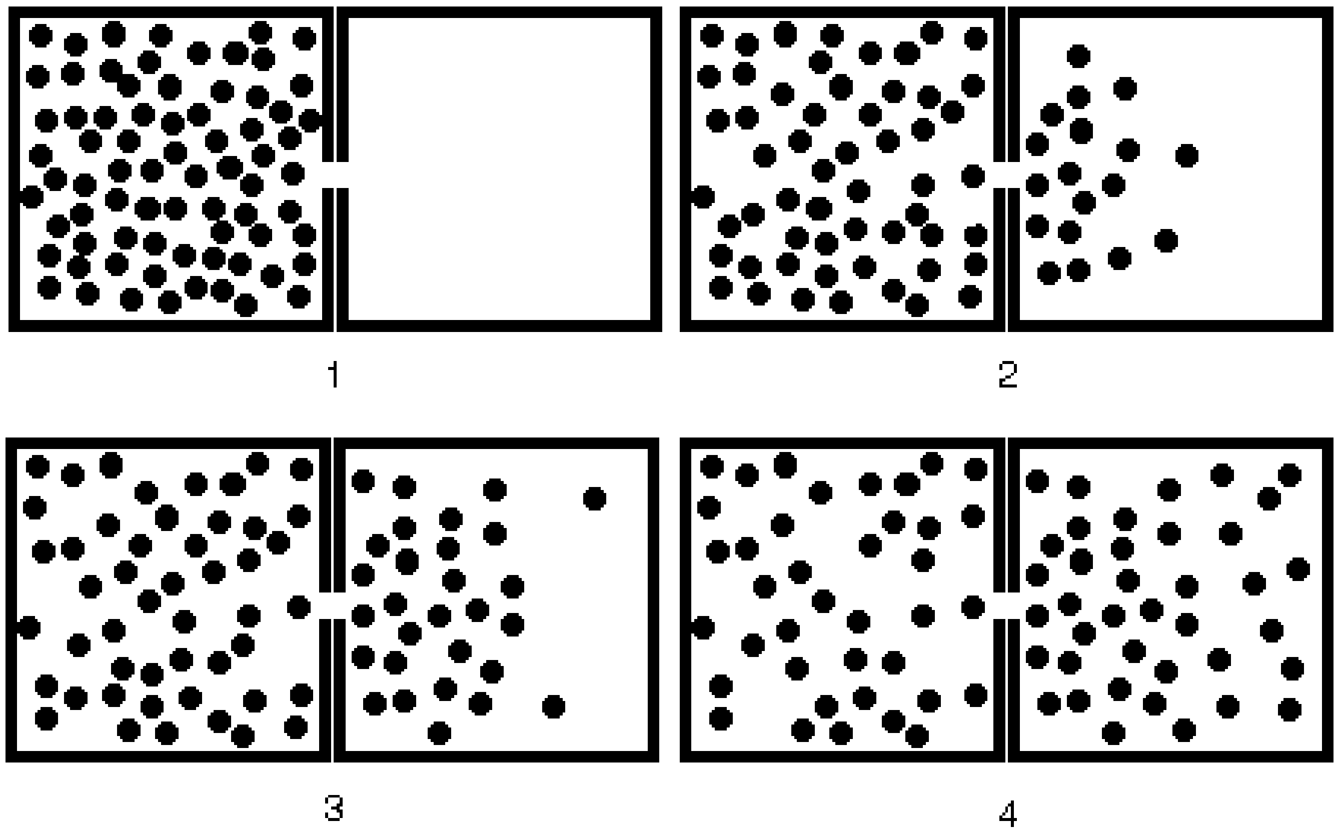
I numeri casuali servono :

- per simulare processi casuali (approccio statistico)

Gas in un recipiente diviso in due parti da una parete con sportellino apribile che mette in comunicazione le due zone del contenitore. Il gas inizialmente e' concentrato tutto in una delle due parti.



DOMANDA interessante: come cambia nel tempo il numero di particelle nelle due parti?
DOMANDA poco interessante : qual e' con precisione la traiettoria di ogni particella?



Una volta aperto lo sportellino, il gas si espande gradualmente nell'altra parte fino a che non e' stata raggiunta una distribuzione uniforme del gas nell'intero contenitore.

distribuzione uniforme nelle due parti della scatola

osserviamo che il sistema raggiunge questo stato ma le fluttuazioni attorno a tale stato non diminuiscono nel tempo. Per diminuire tali fluttuazioni possiamo aumentare il numero di palline in ciascun urna oppure eseguire diverse simulazioni e calcolare delle medie dei risultati ottenuti.