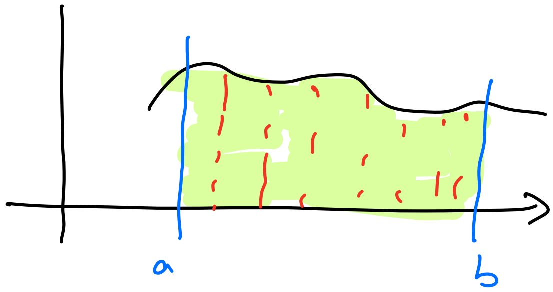
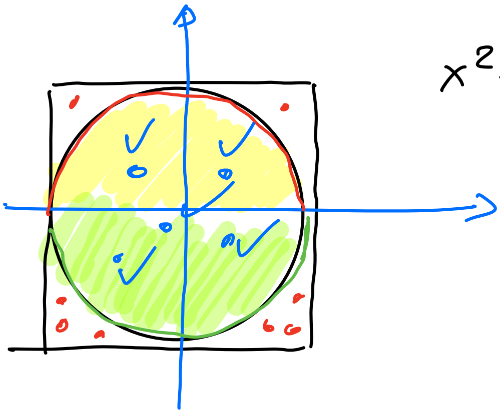
Integrazione numerica

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





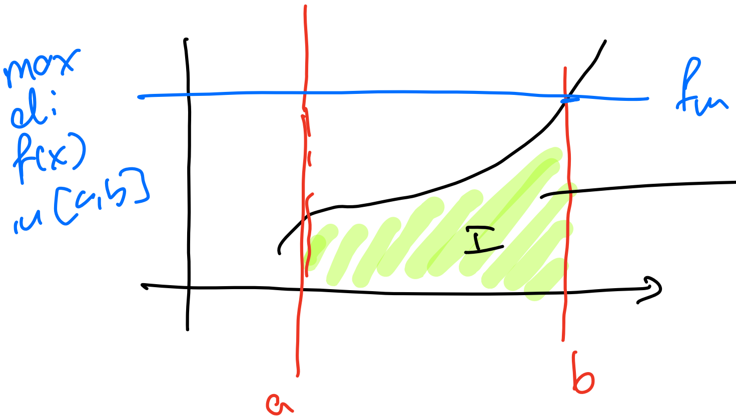
$$x^2+y^2=R^2$$

$$y(x) = \pm\sqrt{R^2-x^2}$$

$$\frac{\text{Area cerchio}}{\text{Area quadrato}} = \frac{\#\text{ punti dentro}}{\#\text{ punti totali generati}}$$

hit & miss

punto $\equiv (x,y)$

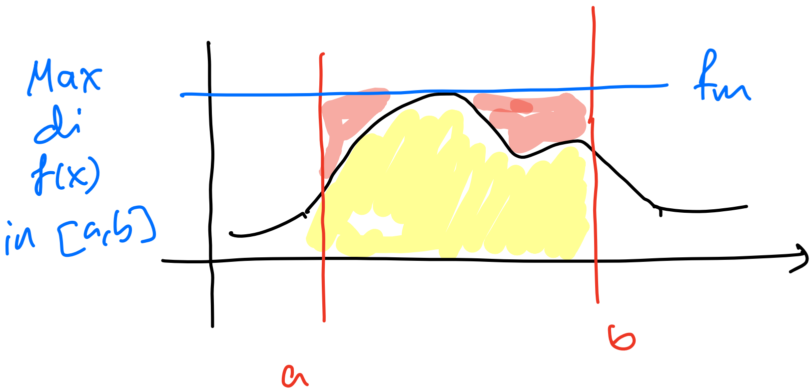


$$\int_a^b f(x)\,dx = I$$

Area Rettangolo:

$$(b-a)\times\underbrace{[\text{Max di } f(x)]}_{fm}$$

$\tanh(x * \lg(x))$



$$\frac{\text{Area sotto curva}}{\text{Area rettangolo}}$$

prob. di colpire sotto $f(x)$

Generare N coppie (x,y)

$x \in [a,b] \qquad y \in [0, fm]$

$$I = \frac{\#\text{ punti sotto curva}}{N}\cdot[(b-a) * fm]$$

#punti sotto:

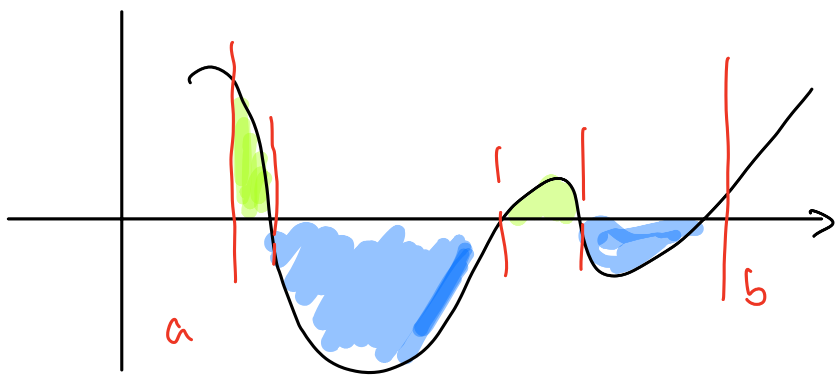
```
x = a + (b-a)*drand48()
y = fun*drand48()
```

Condizione sotto:

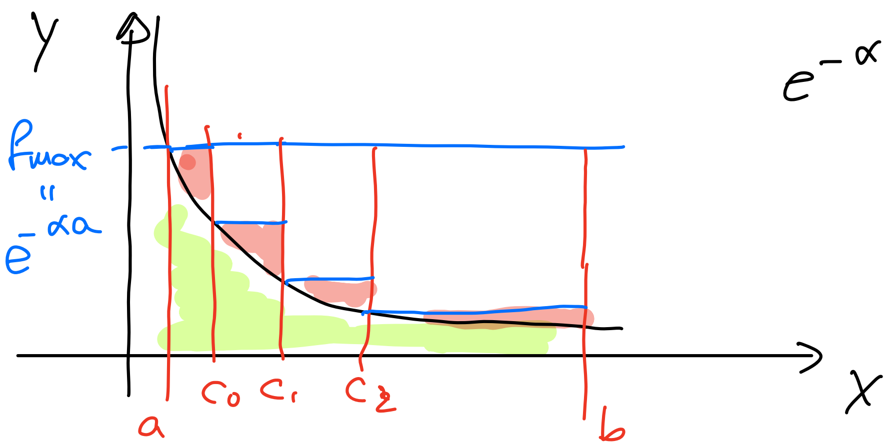
```
if ( y < f(x)) Ndentro++;
```

```
int N = 1000, Ndentro = 0;

for(i=0 ; i<N; i++) {
    x = a + (b-a)*drand48()
    y = fun*drand48()
    if ( y <= f(x)) Ndentro++;
}
```

$$\delta = I_{vero} - I_{MC} \propto \frac{1}{\sqrt{N}}$$



$$I = \int_a^b f(x)dx$$



$$e^{-\alpha x}$$

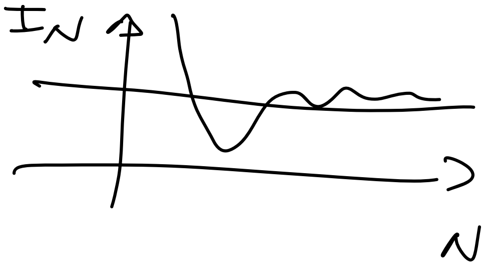
$$f_{max} = e^{-\alpha a}$$

$$f(b) < f(a)$$

Metodo hit&miss in sotto intervalli

$$I = \int_a^b f(x)\,dx = \int_a^{c_0} f(x)\,dx + \int_{c_0}^{c_1} + \int_{c_1}^{c_2} + \int_{c_2}^{b} f(x)\,dx$$

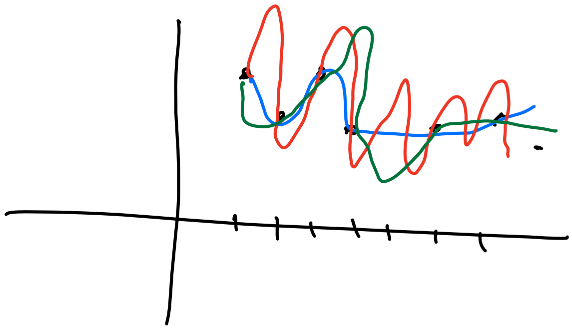
N	I_N



Metodo Monte Carlo

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

$$= \int_a^b \frac{f(x)}{p(x)}\, p(x)\,dx$$



Con $\int_a^b p(x)\,dx = 1$

$p(x)$: distribuzione di probabilità conosciuta nell'intervallo $[a,b]$

$$I = \int_a^b s(x)\, p(x)\,dx \qquad s(x) =: \frac{f(x)}{p(x)}$$

↳ $\langle s(x) \rangle$ in $[a,b)$ con distribuzione $p(x)$

voto	CFU	
20	6	LAB Calcolo
30	12	Meccanica

Media aritmetica : $\frac{20+20}{2}$

Media pesata : $\frac{20 * 6 + 20 * 12}{6 + 12} = \frac{120+240}{18}$

$= 20$

(20, 30) : Med. aritmetica : 25

media pesata : $\frac{20 * 6 + 30 \times 12}{18} = \frac{480}{18}$

$= \int s(x)\, p(x)\, dx \equiv \sum_i s(x_i)\, p(x_i)$

x_i distribuiti secondo $p(x)$

$= \frac{\sum_i \text{voto}_i * \text{CFU}_i}{\sum_i \text{CFU}_i} = 26,\overline{6}$

$$I = \int_a^b f(x)\,dx = \int_a^b \frac{f(x)}{p(x)}\, p(x)\, dx = \int_a^b s(x)\, p(x)\, dx$$

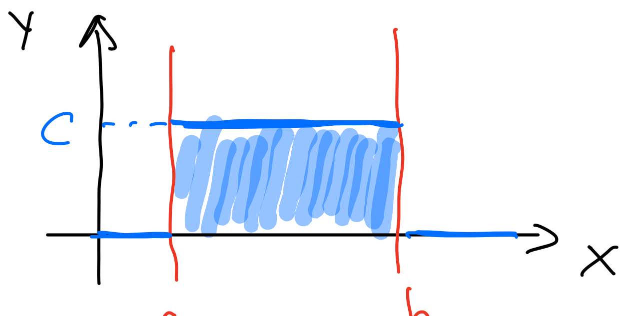
$= \langle s(x) \rangle = \frac{1}{N} \sum_i s(x_i)$

$x_i = \{x_1, \dots, x_N\}$

N valori distribuiti secondo $p(x)$

$= \frac{1}{N} \sum_i \frac{f(x_i)}{p(x_i)}$

$p(x)$:



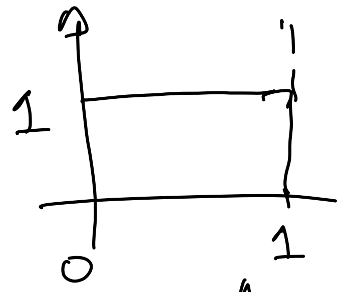
$p(x)$ = costante
distrib. uniforme
in $[a,b]$

$$p(x) \begin{cases} \frac{1}{b-a} & x \in [a,b] \\ \emptyset & \text{fuori} \end{cases}$$

$$\int_a^b p(x)\, dx = 1$$

$$\int_a^b c\cdot dx = 1 = c\cdot(b-a) \Rightarrow p(x) = c = \frac{1}{b-a}$$

$[a,b] = [0,1]$



$$P(x>0.5) = \int_{0.5}^{1} p(x)\,dx = \int_{0.5}^{1} dx = 0.5$$

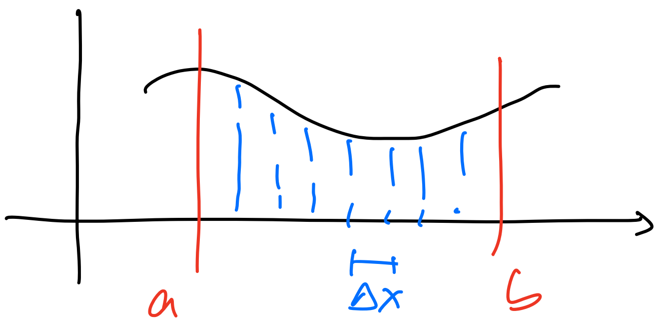
1) estrarre x casuali in $[a,b]$

$$x = a + (b-a) * drand48()$$

$$I = \frac{1}{N}\sum_i \frac{f(x_i)}{p(x_i)} = \frac{1}{N}\sum_i \frac{f(x_i)}{\frac{1}{b-a}} = \frac{(b-a)}{N}\sum_i f(x_i)$$

$$\Rightarrow I = \int_a^b f(x)\,dx = \frac{b-a}{N} * \sum_i f(x_i)$$

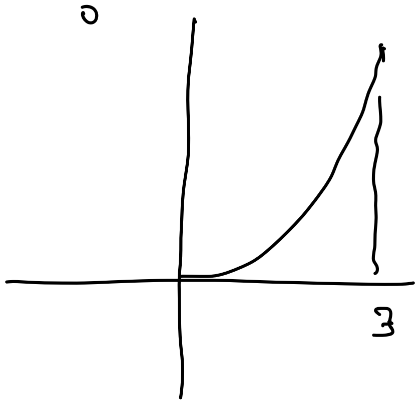
x_i estratti da uniforme in $[a,b]$



$$\Delta x = \frac{b-a}{N}$$

```c
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

double myf(double);

int main() {

  double sum = 0;

  double a=0., b=3;

  int npt;
  for(npt=10; npt<=1e6; npt*=10) {
    sum = 0;
    for(int i=0; i<npt; i++) {
      double x = a + (b-a)*lrand48()/RAND_MAX;
      sum += myf(x);
    }
    sum = (b-a)*sum/npt;

    printf("#punti: %8d \t Integral: %.5f\n", npt, sum);
  }

  return 0;
}

double myf(double x) {
  return x*x;
}
```

$$\int_0^3 x^2 dx = \frac{1}{3} 3^3 = 9$$

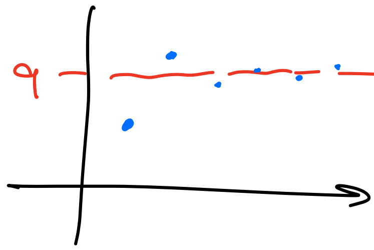


Sum += myf(x) * (b-a)/Npt

```
Mac:material rahatlou$ gcc -o /tmp/app mcint.c
Mac:material rahatlou$ /tmp/app
#punti:       10         Integral: 6.99096
#punti:      100         Integral: 9.49756
#punti:     1000         Integral: 8.83431
#punti:    10000         Integral: 9.00245
#punti:   100000         Integral: 8.98946
#punti:  1000000         Integral: 9.01072
```

Valore vero

$I = 9.00000$

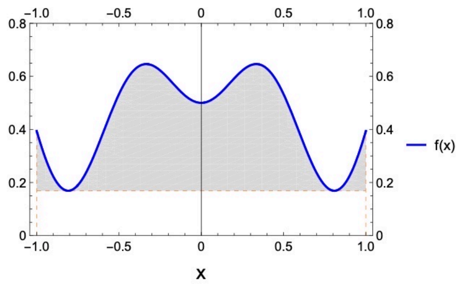


► **Esercizio in `C`:** Si vuole scrivere un programma `hitmiss.c` che utilizzi il metodo *hit & miss* per calcolare l'area sottostante la funzione

$$f(x) = \sin(3x)\cos(3x)\tanh(x) + 0.5 \tag{1}$$

nell'intervallo $[-1, 1]$. Siano $y_{\min}$ e $y_{\max}$ rispettivamente i valori minimo e massimo che tale funzione assume nell'intervallo considerato. Si scriva un programma per calcolare, l'area A della superficie compresa tra la funzione $f(x)$ e la retta $y = y_{\min}$ nell'intervallo $[-1, 1]$.
Per calcolare l'area A, bisogna generare casualmente in modo uniforme N_p punti nel rettangolo individuato dai punti $(-1, y_{\min})$, $(-1, y_{\max})$, $(1, y_{\max})$ e $(1, y_{\min})$ e contare il numero di volte N_h che tali punti si trovano nell'area grigia mostrata in figura. Una stima dell'area A sarà fornita dalla seguente formula:

$$A = 2(y_{\max} - y_{\min})N_h/N_p \tag{2}$$



Suggerimento: in C la funzione per calcolare $\tanh(x)$ è `tanh()`, definita in `math.h`.
Il programma dovrà:

1. Definire tramite una direttiva del precompilatore il numero di punti $N_f = 10000$ per i quali calcolare la funzione $f(x)$ e il numero $N_p = 100000$ (10^5) di punti da generare.

2. Identificare numericamente il valore massimo e minimo della funzione nell'intervallo nel modo seguente:

   - Calcolare il valore della funzione $f(x)$ in N_f punti nell'intervallo $[-1, 1]$ e memorizzare questi valori in array bidimensionale chiamato `valori` e costituito da N_f righe e 2 colonne, dove nella prima colonna verrano memorizzati gli N_f valori di x generati e nella seconda colonna i corrispondenti valori della funzione $f(x)$.
   - Utilizzando i dati contenuti nell'array `valori`, stimare il valore minimo $y_{\min}$ e massimo $y_{\max}$ assunto dalla funzione $f(x)$ nell'intervallo $[-1, 1]$.

3. Calcolare l'area A con il metodo *hit & miss*, ovvero:

   - Generare a caso N_p di punti all'interno del rettangolo suddetto.
   - Contare il numero di volte N_h in cui tali punti cadono all'interno dell'area grigia mostrata in figura.
   - Calcolare l'area A tramite l'eq. (2).

4. Stampare su schermo il valore dell'area A così ottenuto con 5 cifre dopo la virgola, per $N_p = 10^5$.

Nello scrivere il programma si richiede che vengano implementate le seguenti funzioni:

- `func(…)` che restituisce il valore della funzione $f(x)$ calcolata per il valore passato come argomento

- `riempi(…)`, che richiede come argomento l'array `valori` e memorizza in tale array N_f valori della funzione $f(x)$ nell'intervallo $[-1, 1]$ come spiegato in precedenza.

- `maxMin(…)` che richiede come argomento l'array `valori` e determina il valore massimo e minimo della funzione $f(x)$ utilizzando tale array. La funzione deve utilizzare due puntatori passati come argomenti per restituire i valori del massimo e del minimo.

- `integrale(…)`, che calcola l'area A con il metodo *hit &miss* illustrato in precedenza. Tale funzione prende come argomenti i valori $y_{\min}$ e $y_{\max}$ calcolati in precedenza con la funzione `maxMin()`.

Una volta verificato il corretto funzionamento del programma, modificarlo come segue: invece di fissare il valore di `Np` tramite il pre-compilatore, definire una variabile in modo che il programma calcoli l'integrale per valori `Np` pari a 2^{10}, 2^{11}, ...2^{20}, e scriva in un file di nome `integrale.dat` il numero di punti generati e il corrispondente valore dell'area A ottenuto. Quest'ultimo va scritto con quattro cifre dopo la virgola. Ci devono quindi essere due valori per ciascuna riga del file.

► **Esercizio in Python:** Creare uno script `python` chiamato `area.py` per leggere i dati dal file `integrale.dat`, e riporti su un grafico l'andamento dell'area A in funzione del numero di punti. Il grafico dovrà riportare una legenda ed opportuni label per gli assi e deve essere salvato con il nome di `area.png`

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

// pt: 2 uso direttive pre-compilatore
#define NPMAX 10000
#define XMIN -1.
#define XMAX 1.

// bonus 2
double uniforme(double, double);

// pt: 2 interfaccia
double func(double);
// pt: 2 interfaccia
void riempi(double [NPMAX][2], int);
// pt: 2 interfaccia
void maxMin(double [NPMAX][2], int, double*, double*);
// pt: 2 interfaccia
double integrale(double, double, int);
```

```c
int main() {
    // pt: 1 chiamata a stand48() nella main()
    srand48( time(0) );

    // pt: 2 corrette variabili e dimensione array
    int Np;
    double valori[NPMAX][2] = {0};
    double area;
    double ymin, ymax;
    int j;

    // pt: 1 apertura e controllo file
    FILE* pf;
    pf = fopen("integrale.dat", "w");
    if(!pf) {
        printf("problma con apertura file... exit\n");
        exit(-1);
    }

    // riempi array con NMAX valori
    riempi(valori, NPMAX);

    // trova min e max di f(x)
    maxMin(valori, NPMAX, &ymax, &ymin);

    //pt: 2 corretto ciclo
    // calcolo integrale con numero crescente di punti
    for(j=10; j<=20; j++) {
        Np = pow(2,j);
        area = integrale(ymin, ymax, Np);
        // pt: 1 scrittura info su file
        printf("Np: 2^%-2d (%6d) \t area: %.5f\n", j, Np, area);
        fprintf(pf, "%7d \t %.5f\n", Np, area);
    }

    fclose(pf);

    return 0;
}
```

```c
double uniforme(double a, double b) {
    return a+(b-a)*lrand48()/RAND_MAX;
}
// pt: 1 corretta funzione
double func(double x) {
    return sin(3*x)*cos(3*x)*tanh(x)+0.5;
}

// pt: 3 corretto riempimento array
void riempi(double mat[NPMAX][2], int  np) {
    int i;
    for(i=0; i<np; i++) {
        mat[i][0] = uniforme(XMIN, XMAX);
        mat[i][1] = func(mat[i][0]);
    }

}
```

*(*(mat+i)) = un.f(~)

*(*(mat+i)+1) = func(mat[i][0])

x = uniforme(XMIN, XMAX)

mat[i][0] = x_i

mat[i][1] = func (x)

```c
// pt: 4 corretta determinazione min e max
void maxMin(double mat[NPMAX][2], int n, double* max, double* min) {
    *max = -1;
    *min = 1000.;
    int i;
    // ciclo sui valori di f(x)
    for(i=0; i<n; i++ ) {
        // massimo
        if(mat[i][1]>*max) *max = mat[i][1];
        // minimo
        if(mat[i][1]<*min) {
                *min = mat[i][1];
        }
    }
}

// pt: 3 calcolo integrale hit & miss
double integrale(double ymin, double ymax, int Np) {
    int Nh = 0;
    int i;
    for (i=0; i<Np; i++) {
        double x = uniforme(XMIN,XMAX);
        double y = uniforme(ymin, ymax);
        if( y < func(x) ) Nh++;
    }
    double area  = 2.*(ymax-ymin)*Nh/Np;
    return area;

}
```

```python
import numpy as np
import matplotlib.pyplot as plt
import math as m

plt.title("Area con metodo hit & miss in funzione del numero di punti")

Np, area = np.loadtxt("integrale.dat", unpack = True)

plt.plot(Np, area, '.-', color='blue', label='area')

plt.xlim(min(Np)*0.8, max(Np)*1.2)
plt.ylim(min(area)*0.9, max(area)*1.1)

plt.xlabel('Numero di punti')
plt.xscale('log')
plt.ylabel('Area')

plt.grid()
plt.legend(loc='upper right')
#plt.savefig('area.png')

plt.show()
```