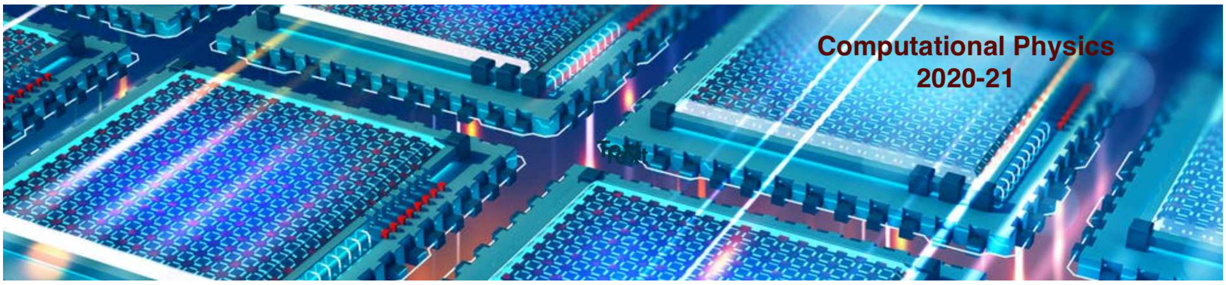
Computational Physics

numerical methods with C++ (and UNIX)

2022-23



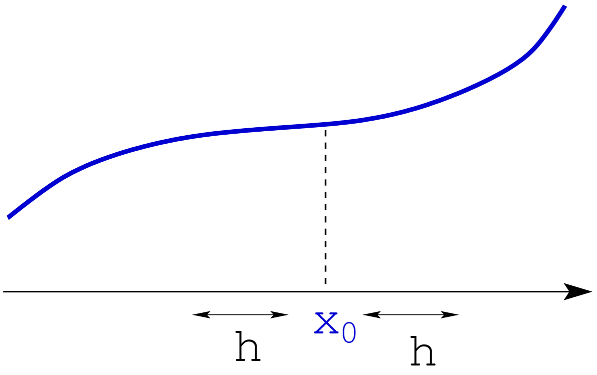
Fernando Barao
Instituto Superior Técnico, Dep. Fisica
email: fernando.barao@tecnico.ulisboa.pt

Computational Physics

Numerical derivatives

Fernando Barao, Phys Department IST (Lisbon)

functions: Taylor expansion



✔ A function can be approximated by a polynomial of order n, near a point x_0,

$$f(x_0+h) = a_0 + a_1 h + a_2 h^2 + \cdots + a_n h^n$$

✔ To determine the polynomial coefficients, we set $h = 0$:

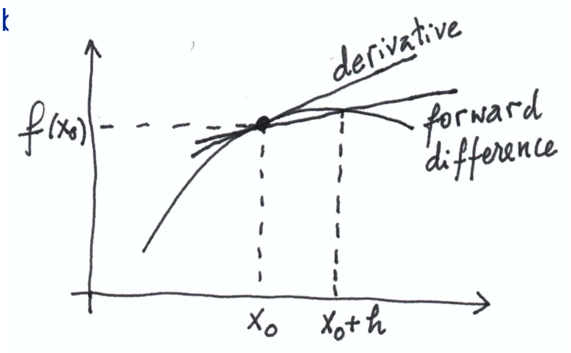
$f(x_0) = a_0$

$f_h'(x_0) = a_1$

$f_h''(x_0) = 2a_2 \;\Rightarrow\; a_2 = \frac{f_h''(x_0)}{2}$

$$\boxed{\begin{aligned} f(x_0+h) &= f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x_0) + \cdots + \frac{f^{(n)}(x_0)}{n!}h^n \\ f(x_0-h) &= f(x_0) - hf'(x_0) + \frac{h^2}{2}f''(x_0) - \cdots + (-1)^n\frac{f^{(n)}(x_0)}{n!}h^n \end{aligned}}$$

Numerical 1st derivative



✔ The differentiation of a function is one of the most b tasks in physics: $\frac{d\vec{v}}{dt} = \frac{\vec{F}}{m}$

✔ **forward difference**

$$f(x_0+h) - f(x_0) = hf'(x_0) + \frac{h^2}{2}f''(x_0) + \cdots - f(x_0)$$

$$\boxed{f'(x_0) \simeq \frac{f(x_0+h)-f(x_0)}{h} + O(h)}$$

✔ **central difference**

$$\begin{aligned} & f(x_0+h) - f(x_0-h) \\ = \;& f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x_0) + \cdots \\ & -\left[f(x_0) - hf'(x_0) + \frac{h^2}{2}f''(x_0) + \cdots\right] \\ = \;& 2hf'(x_0) + \frac{2}{3!}h^3 f^{(3)}(x_0) + \cdots \end{aligned}$$

$$\boxed{f'(x_0) \simeq \frac{f(x_0+h)-f(x_0-h)}{2h} + O(h^2)}$$

the accuracy of the derivative increased by one order!

what h step?

truncation error:

$\delta(\Delta f) = \frac{2}{3!}h^3 f^{(3)} \;\geq \varepsilon_M$

$$\varepsilon_M \sim \begin{cases} 10^{-7} & \text{float} \\ 10^{-15} & \text{double} \end{cases}$$

$\frac{\delta(\Delta f)}{f^{(3)}} = \frac{2}{3!}h^3 \sim \varepsilon_M$

$h^3 \sim 3/2\varepsilon_M$

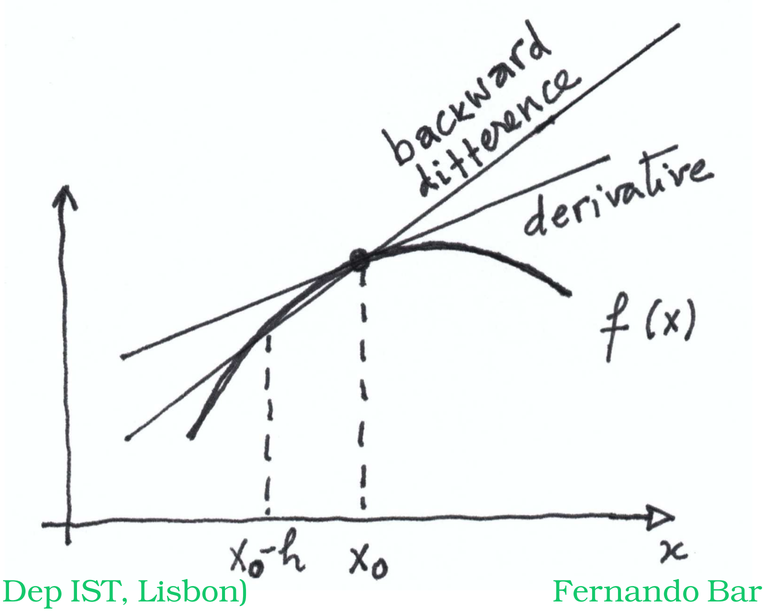
$\boxed{h \sim (10^{-15})^{1/3} \sim 10^{-5}}$

Numerical 1st derivative

- ✔ **backward difference**

$$f(x_0 - h) - f(x_0) = hf^{'}(x_0) + \frac{h^2}{2} f^{''}(x_0) + \cdots - f(x_0)$$

$$\boxed{f^{'}(x_0) \simeq \frac{f(x_0) - f(x_0 - h)}{h} + O(h)}$$



Numerical 1st derivative (cont.)

- ✔ For deriving a higher-order first-derivative expression we can use the function values at $[x_0 - 2h, x_0 - h, x_0 + h, x_0 + 2h]$ for eliminating the next order term (f''').

$$\begin{aligned} f(x_0 \pm h) &= f(x_0) \pm hf'(x_0) + \tfrac{h^2}{2} f''(x_0) \pm \tfrac{h^3}{3!} f'''(x_0) + \cdots \\ f(x_0 \pm 2h) &= f(x_0) \pm 2hf'(x_0) + \tfrac{4h^2}{2} f''(x_0) \pm \tfrac{(2h)^3}{3!} f'''(x_0) + \cdots \end{aligned}$$

- ✔ We can start by eliminating the f'' from combining $f(x_0 \pm h)$ and $f(x_0 \pm 2h)$

$$\begin{aligned} f(x_0 + h) - f(x_0 - h) &= 2hf'(x_0) + \tfrac{2h^3}{3!} f'''(x_0) + O(h^5) \\ f(x_0 + 2h) - f(x_0 - 2h) &= 4hf'(x_0) + \tfrac{16h^3}{3!} f'''(x_0) + O(h^5) \end{aligned}$$

- ✔ Combining these two expressions to eliminate $f'''(x_0)$

$$(-8) \times [f(x_0 + h) - f(x_0 - h)] + [f(x_0 + 2h) - f(x_0 - 2h)] = -12hf'(x_0) + O(h^5)$$

Numerical 1st derivative (cont.)

- The five-point formula for first-derivative:

  $$\boxed{f'(x_0) = \frac{1}{12h}\left[f(x_0-2h) - 8f(x_0-h) + 8f(x_0+h) - f(x_0+2h)\right] + \mathcal{O}(h^4)}$$

  the truncation error goes as $\mathcal{O}(h^4)$

- Numerically the expression above can be improved by decreasing the number of subtractions!

  $$\boxed{f'(x_0) = \frac{1}{12h}\left[\left[f(x_0-2h) + 8f(x_0+h)\right] - \left[8f(x_0-h) + f(x_0+2h)\right]\right]}$$

- Notes
  - supposing we have a set of discrete points $x_i = x_0, x_1, \cdots, x_n$ and their respective function values, the central difference cannot be computed in the extreme abcissa values x_0 and x_n, because we would need the function values on both sides
  - in that case, the backward and forward difference can be used to estimate the derivatives

Forward and backward: higher order

- We can also derive expressions for computing the 1st derivative of $\mathcal{O}(h^2)$ using *forward* and *backward* differences. We will just combine the Taylor series computed at $(x+h)$ and $(x+2h)$ for forward difference and $(x-h)$ and $(x-2h)$ for backward difference.

- For eliminating the next order term (f'') we do for the *forward difference*:

  $$f(x_0+2h) - 4f(x_0+h) = -3f(x_0) - 2hf'(x_0) + \frac{2h^3}{3}f'''(x_0) + \cdots$$

  $$\boxed{f'(x_0) = \frac{-f(x_0+2h) + 4f(x_0+h) - 3f(x_0)}{2h} + \mathcal{O}(h^2)}$$

- For eliminating the next order term (f'') we do for the *backward difference*:

  $$f(x_0-2h) - 4f(x_0-h) = -3f(x_0) + 2hf'(x_0) - \frac{2h^3}{3}f'''(x_0) + \cdots$$

  $$\boxed{f'(x_0) = \frac{f(x_0-2h) - 4f(x_0-h) + 3f(x_0)}{2h} + \mathcal{O}(h^2)}$$

Numerical 2nd derivative

✔ Defining the function expansions at $(x_0 + h)$ and $(x_0 - h)$:

$$f(x_0 \pm h) = f(x_0) \pm h f'(x_0) + \frac{h^2}{2} f''(x_0) \pm \frac{h^3}{3!} f'''(x_0) + \cdots$$

✔ Adding the function expansions, make the odd derivatives disappear:

$$f(x_0 + h) + f(x_0 - h) = 2f(x_0) + h^2 f''(x_0) + \frac{2h^4}{24} f^{(4)}(x_0) + \cdots$$

✔ The three-point formula:

$$\boxed{f''(x_0) = \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} + O(h^2)}$$

✔ The five-pont formula:

$$\boxed{f''(x_0) = \frac{-f(x_0 - 2h) + 16f(x_0 - h) - 30f(x_0) + 16f(x_0 + h) - f(x_0 + 2h)}{12h^2} + O(h^4)}$$

1st derivative: non uniform data points (★)

✔ In case the data grid is composed of uneven intervals of x, the derivative formulas derived before cannot be used
What can we do?

☞ Eventually interpolate the data points in order to have an uniform distribution of data points

☞ We can approximate the derivative of $f(x)$ by the derivative of an interpolant

☞ We can derive finite difference approximations for unevenly spaced data

1st derivative: three-point formulas (★)

✔ We can develop the function $f(x)$ at the points $x_i \pm h_{i\pm i}$ where $h_{i\pm i}$ is different for the left and right points

✔ Let's combine $f(x_i + h_{i+1})$ and $f(x_i - h_{i-1})$ to eliminate the second order term ($f''(x_i)$)

$$f(x_i + h_{i+1}) \equiv f(x_{i+1}) = f(x_i) + h_{i+1} f'(x_i) + \frac{h_{i+1}^2}{2} f''(x_i) + \mathcal{O}(h^3) \qquad (1)$$

$$f(x_i - h_{i-1}) \equiv f(x_{i-1}) = f(x_i) - h_{i-1} f'(x_i) + \frac{h_{i-1}^2}{2} f''(x_i) + \mathcal{O}(h^3) \qquad (2)$$

Multiplying (1) by (h_{i-1}^2) and (2) by $(-h_{i+1}^2)$ and adding the eqs we get rid of the second derivative:

$$\boxed{f_i' = \frac{h_{i-1}^2 f_{i+1} + (h_i^2 - h_{i-1}^2) f_i - h_i^2 f_{i-1}}{h_i h_{i-1}(h_{i-1} + h_i)} + \mathcal{O}(h^2)}$$

1st derivative: by interpolation (★)

✔ The cubic spline interpolant segment by segment can be used to get the function derivative at any point

$$f_{i,i+1}(x) = \frac{K_i}{6}\left[\frac{(x-x_{i+1})^3}{x_i-x_{i+1}} - (x-x_{i+1})(x_i-x_{i+1})\right] - \frac{K_{i+1}}{6}\left[\frac{(x-x_i)^3}{x_i-x_{i+1}} - (x-x_i)(x_i-x_{i+1})\right] + \frac{y_i(x-x_{i+1}) - y_{i+1}(x-x_i)}{x_i-x_{i+1}}$$

☞ where $K_{i,i+1}$ are the second derivative values

✔ The 1st derivative of the function for x belonging to the segment is:

$$\boxed{f'_{i,i+1}(x) = \frac{K_i}{6}\left[3\frac{(x-x_{i+1})^2}{x_i-x_{i+1}} - (x_i-x_{i+1})\right] - \frac{K_{i+1}}{6}\left[3\frac{(x-x_i)^2}{x_i-x_{i+1}} - (x_i-x_{i+1})\right] + \frac{y_i-y_{i+1}}{x_i-x_{i+1}}}$$

✔ The 2nd derivative of the function for x belonging to the segment is:

$$\boxed{f''_{i,i+1}(x) = K_i\left(\frac{x-x_{i+1}}{x_i-x_{i+1}}\right) - K_{i+1}\left(\frac{x-x_i}{x_i-x_{i+1}}\right)}$$

Computational Physics

Roots of equations

Fernando Barao, Phys Department IST (Lisbon)

Function's roots: bisection method

✔ To find the function's roots, we look for the solution of equation: $f(x) = 0$

✔ We start with an interval $[\mathbf{a_0}, \mathbf{b_0}]$ where there is at least one root: $\boxed{f(a_0) \cdot f(b_0) < 0}$

✔ We sub-divide the interval:

$$\left[a_1 = a_0, b_1 = \frac{a_0+b_0}{2}\right]$$
$$\left[a_1 = \frac{a_0+b_0}{2}, b_1 = b_0\right]$$

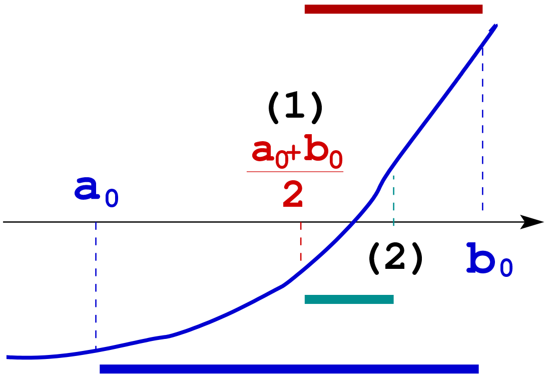
and we choose the interval having the zero crossing:

$f(a_1 = a_0) \cdot f(b_1 = \frac{a_0+b_0}{2}) < 0$

$f(a_1 = \frac{a_0+b_0}{2}) \cdot f(b_1 = b_0) < 0$

and so on:

$[a_0, b_0] \rightarrow [a_1, b_1] \rightarrow \cdots \rightarrow [a_n, b_n]$



✔ Convergence criterium (root value: x_n)

repeat until $b_n - a_n < \varepsilon \cdot x_n$ or/and $f(x_n) < \varepsilon$

Function's roots: bisection algorithm

bisection

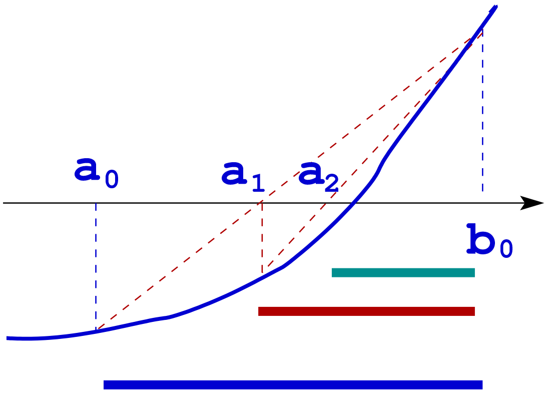
```
// argumentss:
// f: function
// [a,b]: range to search
// eps: convergence tolerance

double ZeroFunctor::Bisection(TF1* f, double a, double b, double eps) {
  double fa = f->Eval(a);
  double fb = f->Eval(b);
  if( fa*fb > 0.0) return 999; // no root
  double xl = a;
  double xr = b;
  while (fabs(xr - xl) >= eps * xr) { // convergence
    double fl = f->Eval(xl);
    double fr = f->Eval(xr);
    x0 = (xr + xl)/2.0;
    double f0 = f->Eval(x0);
    if((fl * f0 ) <= 0.0 ) xr = x0;
    else xl = x0;
  }
  return x0;
}
```

Roots: regula falsi

- ✔ We start with an interval $[\mathbf{a_0}, \mathbf{b_0}]$ where there is at least one root:

$$f(a_0)\ f(b_0) < 0$$

- ✔ With respect to bisection, the new point is provided by the zero of the the linear polynomial passing through the two points a_0 and b_0



$$\begin{aligned} f(x) &= f(a_0) + (x - a_0)\frac{f(b_0)-f(a_0)}{b_0-a_0} = 0 \\ a_1 &= a_0 - f(a_0)\frac{b_0-a_0}{f(b_0)-f(a_0)} \end{aligned}$$

Choose the interval having the zero crossing:
$f(a_1)\ f(b_0) < 0$

- ✔ Repeat until $\mathbf{b_n} - \mathbf{a_n} <$ ” $\cdot\ \mathbf{x_n}$

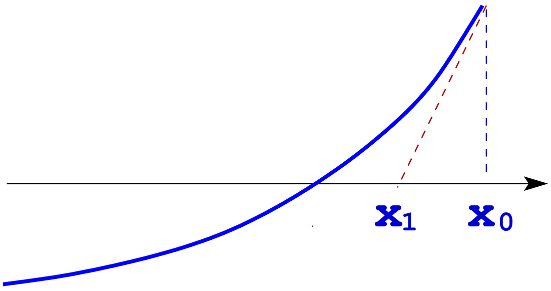
Function’s roots: Newton-Raphson

- we start from a point x_0 near the zero of the function

  the convergence of the method depend how far we are from the zero

- we can approximate the function $f(x)$ at x_0 by a 1st order polynomial and its root will be the first iteration:

  $$f(x_0) + (x_1 - x_0)f'(x_0) = 0$$



- the iterated values:

  $$x^{(i+1)} = x^{(i)} - \frac{f(x^{(i)})}{f'(x^{(i)})}$$

  the analytic derivative of the function is needed

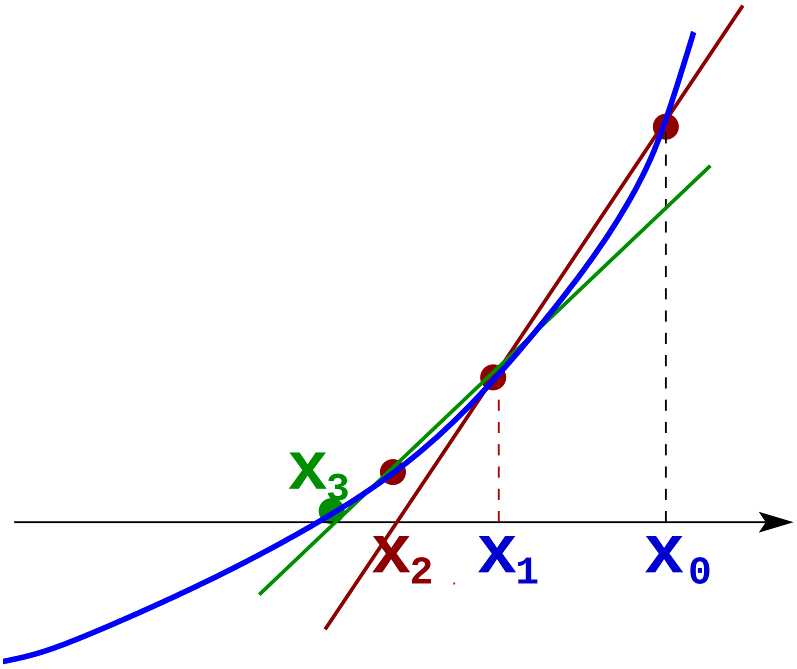
Function’s roots: secant method

- We replace the derivative of the newton-raphson method by a numeric derivative given by:

  $$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

- The iterated values:

  $$x_{i+1} = x_i - f(x_i)\frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})}$$



Systems of non-linear equations: roots

- ✔ Newton-Raphson method can be generalized to systems of non-linear equations

$$\boxed{f_i(x_1, x_2, \cdots, x_n) = 0 \quad \text{with: } i = 1, 2, 3, \cdots, n}$$

We look for solutions of equations: $\vec{f}(\vec{x}) = 0$

- ✔ In the vicinity of the exact solution $(\vec{x})$, each of the functions f_i can be expanded in a Taylor series,

$$\boxed{f_i(\vec{x} + \Delta\vec{x}) = f_i(\vec{x}) + \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j} \Delta x_j + \cdots}$$

where $\frac{\partial f_i}{\partial x_j}$ represent the matrix elements of the Jacobian of the system of fucntions $f_i(\vec{x})$,

$$J(\vec{x}) = \begin{bmatrix} \frac{\partial f_1(\vec{x})}{\partial x_1} & \frac{\partial f_1(\vec{x})}{\partial x_2} & \ldots & \frac{\partial f_1(\vec{x})}{\partial x_n} \\ \frac{\partial f_2(\vec{x})}{\partial x_1} & \frac{\partial f_2(\vec{x})}{\partial x_2} & \ldots & \frac{\partial f_2(\vec{x})}{\partial x_n} \\ \ldots & \ldots & \ldots & \ldots \\ \frac{\partial f_n(\vec{x})}{\partial x_1} & \frac{\partial f_n(\vec{x})}{\partial x_2} & \ldots & \frac{\partial f_n(\vec{x})}{\partial x_n} \end{bmatrix}$$

$$\boxed{\vec{f}(\vec{x} + \Delta\vec{x}) = f(\vec{x}) + J(\vec{x}) \cdot \Delta\vec{x} + \cdots}$$

Systems of non-linear equations: roots

- ✔ Requiring that the corrections $(\Delta\vec{x})$ make the iteration evolve towards the solution,

$$\boxed{(\vec{x} + \Delta\vec{x}) \simeq 0}$$

which implies solving the following system of linear equations,

$$\boxed{(J(\vec{x}) \cdot \Delta\vec{x} = -\vec{f}(\vec{x})}$$

- ✔ The iteration provides in general only an improved approximation of the system of equations roots.

An iterative approach is needed:

$$\boxed{J\left[\vec{x}^{(k)}\right] \cdot \Delta\vec{x}^{(k)} = -\vec{f}\left[\vec{x}^{(k)}\right]} \quad \rightarrow \quad \boxed{\vec{x}^{(k+1)} = \vec{x}^{(k)} + \Delta\vec{x}^{(k)}}$$

- ✔ Convergence of solution:

$\boxed{|\Delta x_i^{(k+1)}| \leq \varepsilon \cdot x_i^{(k+1)}|}$ for $i = 1, 2, \cdots, n$

and complementarly,

$\boxed{|\Delta f_i\left[\vec{x}^{(k+1)}\right]| \leq \varepsilon|}$ for $i = 1, 2, \cdots, n$

Systems of non-linear equations: roots

✔ Jacobian

To approximate the partial derivatives of the functions f_i with respect to the variables x_j, we will use numerical derivatives,

$$f_i(\cdots, x_j + h, \cdots) = f_i(\cdots, x_j, \cdots) + h \cdot \left.\frac{\partial f_i}{\partial x_j}\right|_{\vec{x}} + \frac{h^2}{2} \cdot \left.\frac{\partial^2 f_i}{\partial x_j^2}\right|_{\vec{x}} + \cdots$$

$$f_i(\cdots, x_j - h, \cdots) = f_i(\cdots, x_j, \cdots) - h \cdot \left.\frac{\partial f_i}{\partial x_j}\right|_{\vec{x}} + \frac{h^2}{2} \cdot \left.\frac{\partial^2 f_i}{\partial x_j^2}\right|_{\vec{x}} + \cdots$$

The Jacobian matrix elements will be given by,

$$\boxed{J_{ij} \equiv \frac{\partial f_i}{\partial x_j} \simeq \frac{1}{2h}\left[f_i(\cdots, x_j + h, \cdots) - f_i(\cdots, x_j - h, \cdots)\right]}$$

Computational Physics

Numerical integration

Fernando Barao, Phys Department IST (Lisbon)

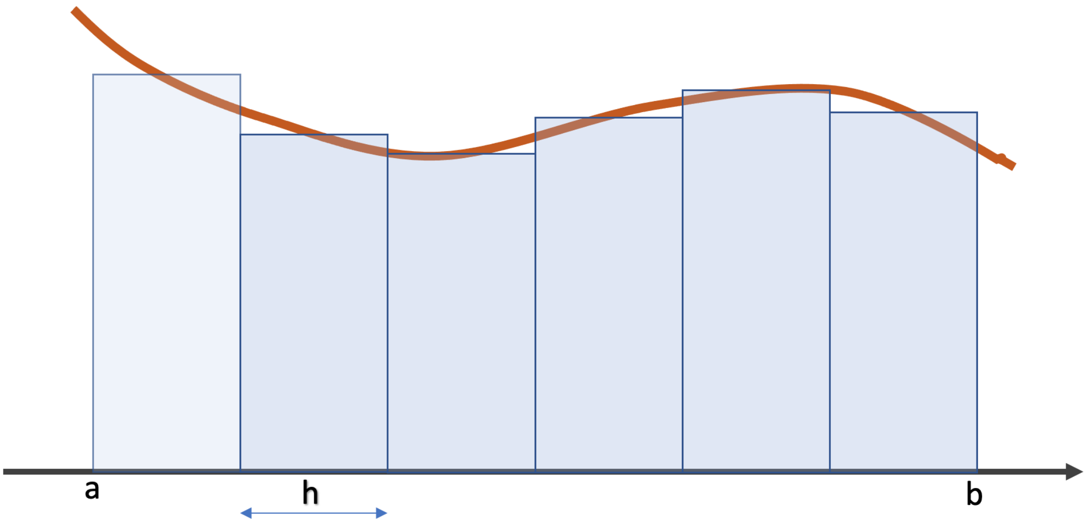
Numerical integration: mid-point rule

$$\boxed{F = \int_a^b f(x)dx \rightarrow \sum_{i=0}^{n} h\, f(\bar{x}_i) \qquad \text{erro: } \varepsilon = \frac{M_{(2)}\,(b-a)^3}{24n^2}}$$

n: number of subintervals between [a,b]

$M_{(2)}$: maximum of 2nd derivative ($f''(x)$) in [a,b]

$\boxed{\bar{x}_i = \frac{x_i + x_{i+1}}{2}}$,
mean point of interval



Numerical integration: mid-point rule error

- ✔ Integral range $[a, b]$ divided in n intervals of width, $h = \frac{b-a}{2}$
- ✔ Let's compute the error associated to the integral of every slice

$$\Delta F = \int_{x_i}^{x_{i+1}} f(x)\,dx - h \cdot f(\bar{x}_i) = \int_{x_i}^{x_{i+1}} [f(x) \; - f(\bar{x}_i)]\; dx$$

Making Taylor expansion,

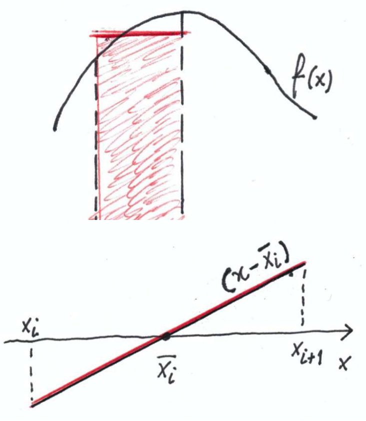
$$f(x) \sim f(\bar{x}_i) + (x - \bar{x}_i) \cdot f'(\bar{x}_i) + \tfrac{1}{2} \cdot (x - \bar{x}_i)^2 \cdot f$$

$$= \int_{x_i}^{x_{i+1}} \left[(x - \bar{x}_i)\, f'(x_i) + \tfrac{1}{2} \cdot (x - \bar{x}_i)^2 \cdot f''(\bar{x}_i)\right] d.$$

$$= \int_{x_i}^{x_{i+1}} \tfrac{1}{2} \cdot (x - \bar{x}_i)^2 \cdot f''(\bar{x}_i)\, dx = \tfrac{1}{6}\, f''(\bar{x}_i) \cdot (x -$$

$$= \tfrac{1}{6}\, f''(\bar{x}_i) \cdot \left[(x_{i+1} - \bar{x}_i)^3 - (x_i - \bar{x}_i)^3\right]$$

$$= \tfrac{1}{6}\, f''(\bar{x}_i) \cdot \tfrac{h^3}{4} = f''(\bar{x}_i) \cdot \tfrac{h^3}{24}$$



- ✔ Taking the sum of the n intervals,

$$\Delta F^{tot} = \textstyle\sum_{i=1}^{n} f''(\bar{x}_i) \cdot \frac{h^3}{24} = \frac{h^3}{24} \sum_{i=1}^{n} f''(\bar{x}_i)$$

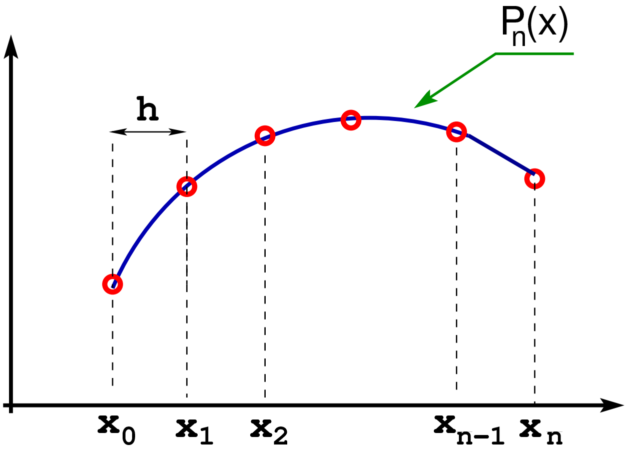
taking the mean value:,$\sum_{i=1}^{n} f''(\bar{x}_i) = n \cdot \bar{f}''(\bar{x}_i)$

$$= \tfrac{h^2}{24} \cdot \tfrac{(b-a)}{n} \cdot n\, \bar{f}''(\bar{x}_i) = \tfrac{h^2}{24} \cdot (b-a) \cdot \left|\bar{f}''(\bar{x}_i)\right|$$

Numerical integration: Newton-Cotes

- ✔ Numerical integration consists in replacing the **integral** continous operator by a **sum** of weighted (w_i) function values ($f(x_i)$),

$$F = \boxed{\int_a^b f(x)dx \rightarrow \sum_{i=0}^{n} w_i f(x_i)}$$



- ✔ Newton-Cotes formulas are based on local interpolation and they are characterized by equally spaced abcissas
- ✔ They correspond to the **trapezoidal** and **Simpson** methods
- ✔ This approach is generally used when the function can be easily computed at equal intervals

Numerical integration: Newton-Cotes (cont.)

- ✔ Divide the range of integration $[\mathbf{a}, \mathbf{b}]$ into $\mathbf{n}$ intervals of width $\mathbf{h} = (\mathbf{b} - \mathbf{a})/\mathbf{n}$ with the nodes of intervals defined by $\mathbf{x_o}, \mathbf{x_1}, \cdots, \mathbf{x_n}$
- ✔ Next, we approximate the function $\mathbf{f(x)}$ by a polynomial of degree $\mathbf{n}$ passing through all the nodes. Using the Lagrange form,

$$P_n(x) = \sum_{i=0}^{n} f(x_i)\ \ell_i(x)$$

The linear polynomial ($n = 1$):

$$P(x) = P_0 + P_1 \cdot x = f(x_1) \cdot \frac{x - x_2}{x_1 - x_2} + f(x_2) \cdot \frac{x - x_1}{x_2 - x_1}$$

- ✔ The integral can therefore be expressed as:

$$F \boxed{= \int_a^b f(x)dx = \sum_{i=0}^{n} \left[f(x_i) \int_a^b \ell_i(x)dx \right] \quad \rightarrow \quad \sum_{i=0}^{n} w_i f(x_i) \qquad (i = 0, 1, \cdots, n)}$$

with $\quad w_i = \int_a^b \ell_i(x)dx \quad$ where ℓ_i are the Lagrange cardinal functions

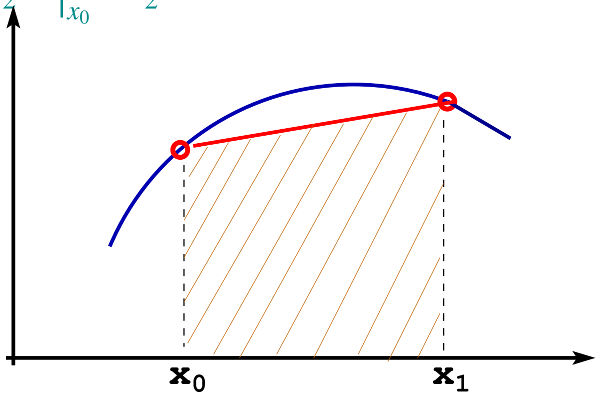
Trapezoidal rule

- ✔ The trapezoidal rule results from $\mathbf{n = 1}$, i.e., from defining a linear polynomial passing in two points $\mathbf{x_0, x_1}$ separated of a distance $\mathbf{h}$,

$$F = \int_{x_0}^{x_1} f(x)dx \simeq \sum_{i=0}^{1} f(x_i) \int_{x_0}^{x_1} \prod_{\substack{j=0 \\ (j\neq i)}}^{1} \frac{x - x_j}{x_i - x_j}$$

$$\begin{aligned}
w_0 &= \int_{x_0}^{x_1} \ell_0(x)dx = \int_{x_0}^{x_1} \frac{x-x_1}{x_0-x_1}dx = -\frac{1}{h}\ \frac{(x-x_1)^2}{2}\Big|_{x_0}^{x_1} = \frac{h}{2} \\
w_1 &= \int_{x_0}^{x_1} \ell_1(x)dx = \int_{x_0}^{x_1} \frac{x-x_0}{x_1-x_0}dx = -\frac{1}{h}\ \frac{(x-x_0)^2}{2}\Big|_{x_0}^{x_1} = \frac{h}{2} \\
F &= \frac{h}{2}f(x_0) + \frac{h}{2}f(x_1) \\
&= \frac{h}{2}\left[f(x_0) + f(x_1)\right]
\end{aligned}$$

$$\boxed{F = \frac{h}{2}\left[f(x_i) + f(x_{i+1})\right]}$$



Trapezoidal rule error

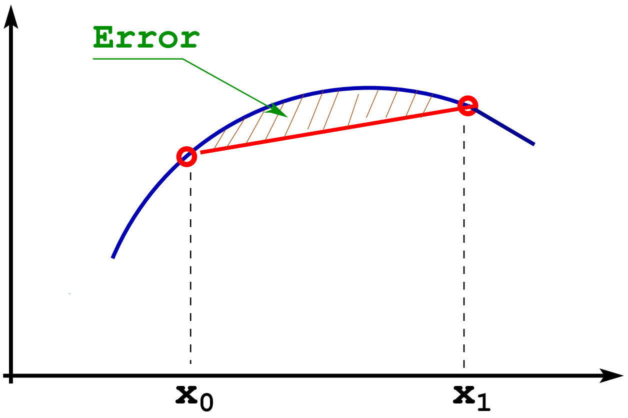
- ✔ The error from integrating the function with the **trapezoidal rule** is due to the approximation of the function

$$\Delta F = \int f(x)dx - \int P_n(x)dx$$

- ✔ For a given slice $[x_i, x_{i+1}]$, the truncation error associated to the linear approximation is $f(x) - P_1(x) = \frac{(x-x_i)(x-x_{i+1})}{(n+1)!}f''(\chi)$ (χ, lies in $[x_i, x_{i+1}]$)

- ✔ The slice trapezoidal error:

$$\begin{aligned}
\Delta F_i &= \int_{x_i}^{x_{i+1}} \frac{(x-x_i)(x-x_{i+1})}{2!} f''(\chi)dx \\
&= \frac{f''(\chi)}{2} \int_{x_i}^{x_{i+1}} (x-x_i)(x-x_{i+1})dx \\
&\simeq -\frac{h^3}{12}f''\left(\frac{(x_i+x_{i+1})}{2}\right) = -\frac{h^3}{12}f''_{i+1/2}
\end{aligned}$$



solving the integral:

$$\int_{x_i}^{x_{i+1}} \underbrace{(x - x_i)}_{u}\underbrace{(x - x_{i+1})dx}_{dv} =$$

$$\frac{1}{2}(x - x_i)(x - x_{i+1})^2\Big|_{x_i}^{x_{i+1}} - \frac{1}{2}\int_{x_i}^{x_{i+1}}(x - x_{i+1})^2dx =$$

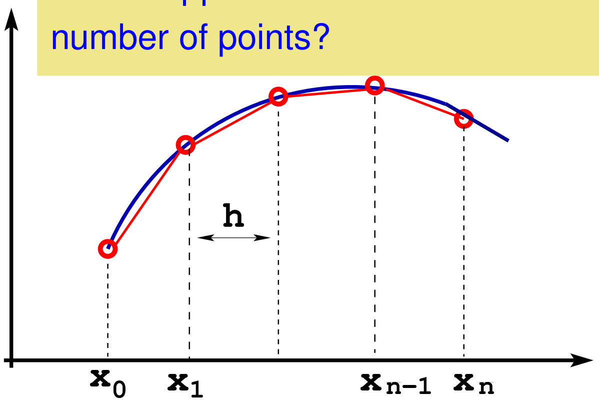
$$-\ \frac{1}{6}(x - x_{i+1})^3\Big|_{x_i}^{x_{i+1}} =$$

$$\frac{1}{6}(x_i - x_{i+1})^3 = -\frac{h^3}{6}$$

Trapezoidal rule (cont.)

- ✔ For extending now the range of integration to the interval [a,b], we divide it in **n** intervals, each with a width **h**

What happens if we double the number of points?



- ✔ For every interval

$F = \frac{h}{2}\left[f(x_i) + f(x_{i+1})\right]$

- ✔ For all the range divided in **n** intervals $(\mathbf{i} = \mathbf{0}, \mathbf{1}, , \cdots , \mathbf{n} - \mathbf{1})$

$$\boxed{F \simeq \frac{h}{2}\sum_{i=0}^{n-1}\left[f(x_i) + f(x_{i+1})\right] = \frac{h}{2}\left[f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)\right]}$$

- ✔ The truncation error $(n = (b-a)/h)$:

$$\boxed{\Delta F = \sum_{i=0}^{n-1}\Delta F_i = -\frac{h^3}{12}\sum_{i=0}^{n-1} f''(\chi) = -\frac{h^3}{12}n < f''(\chi) >= -\frac{h^2(b-a)}{12} < f''(\chi) >}$$

Trapezoidal rule: adaptive scheme

An algorithm for the trapezoidal rule involving the calculation of the integral with an improved accuracy

We will double the number of slices and can keep under control the error of the integral (the difference of the next order with the previous one, provides a good estimation)

Let's use a progressive index $k = 1, 2, 3, \cdots$, that double the number of slices: $n = 2^{k-1} = 1, 2, 4, \cdots$

- ✔ Number of slices = 1 : [a,b] $(k = 1)$

$$F_1 = \frac{b-a}{2}\ \left[f(a) + f(b)\right]$$

- ✔ Number of slices = 2 : [a,b] $(k = 2)$

$$F_2 = \frac{b-a}{4}\left[f(a) + 2f\left(a + \frac{b-a}{2}\right) + f(b)\right] = \frac{1}{2}\ \underbrace{\frac{b-a}{2}\ \left[f(a) + f(b)\right]}_{F_1} + \frac{b-a}{2} f\left(a + \frac{b-a}{2}\right)$$

- ✔ Number of slices = 4 : [a,b] $(k = 3)$

$$\begin{aligned} F_3 &= \frac{b-a}{8}\left[f(a) + 2f\left(a + \frac{b-a}{4}\right) + 2f\left(a + 2\frac{b-a}{4}\right) + 2f\left(a + 3\frac{b-a}{4}\right) + f(b)\right] \\ &= \underbrace{\frac{1}{2}\ \frac{b-a}{4}\left[f(a) + 2f\left(a + \frac{b-a}{2}\right) + f(b)\right]}_{F_2} + \frac{b-a}{4}\left[f\left(a + \frac{b-a}{4}\right) + f\left(a + 3\frac{b-a}{4}\right)\right] \end{aligned}$$

Trapezoidal rule: adaptive scheme

$$\boxed{F_k = \frac{1}{2}\,F_{k-1} + \frac{b-a}{2^{k-1}}\sum_{i=1}^{2^{k-2}} f\left(a + (2i-1)\,\frac{b-a}{2^{k-1}}\right)}$$

Trapezoidal rule: Problem

Calcular o integral

$$\int_0^1 cos(x)dx$$

e uma estimativa do erro, utilizando a regra do trapézio.

Romberg integration

It improves the results of numerical integration using error-correction techniques

uses two estimates of the integral with different precisions, to compute a more accurate approximation

trapezoidal rule

$I = I(h) + E(h)$

I: exact value of integral

$I(h)$: integral evaluation using trapezoidal rule with step size $\frac{b-a}{n}$

$E(h)$: truncature error

$$E(h) \simeq -\frac{b-a}{12}\ h^2\ \bar{f}''$$

How to combine different precision estimations?

$I = I(h_1) + E(h_1) = I(h_2) + E(h_2)$

with $E(h_i) = \mathcal{O}(h_i^2)$

Assuming $\bar{f}''$ constant regardless of step size (h),

$$\frac{E(h_1)}{E(h_2)} \simeq \left(\frac{h_1}{h_2}\right)^2 \ \rightarrow\ E(h_2) = \frac{I(h_2)-I(h_1)}{\left(\frac{h_1}{h_2}\right)^2-1}$$

It can be shown that the error is $\mathcal{O}(h^4)$

$$\boxed{I = I(h_2) + E(h_2) = I(h_2) + \frac{I(h_2)-I(h_1)}{\left(\frac{h_1}{h_2}\right)^2-1}}$$

Romberg integration (cont.)

Defining following indices:

k, level of integration

$k = 1$ trapezoidal rule, $\mathcal{O}(h^2)$

$k = 2$ differences, $\mathcal{O}(h^4)$

$k = 3$ differences, $\mathcal{O}(h^6)$

$\cdots$

j, integral accuracy level related to number of slices (h size)

$j = 1, 2, 3, \cdots$

$$\boxed{I_{j+1,k+1} = I_{j+1,k} + \frac{1}{4^k - 1}\left(I_{j+1,k} - I_{j,k}\right)}$$

For example,

$$I_{2,2} = I_{2,1} + \frac{1}{4-1}\left(I_{2,1} - I_{1,1}\right)$$

$$I_{3,2} = I_{3,1} + \frac{1}{4-1}\left(I_{3,1} - I_{2,1}\right)$$

$$\begin{array}{cc} \begin{array}{cccc} k=1 & k=2 & k=3 & k=n \end{array} & \\ \left(\begin{array}{cccc} I_{1,1} & & & \\ I_{2,1} & I_{2,2} & & \\ I_{3,1} & I_{3,2} & I_{3,3} & \\ \vdots & \vdots & \vdots & \vdots \\ I_{n,1} & I_{n,2} & I_{n,3} & I_{n,n} \end{array}\right) & \begin{array}{c} (j=1) \\ (j=2) \\ (j=3) \\ \\ (j=n) \end{array} \end{array}$$

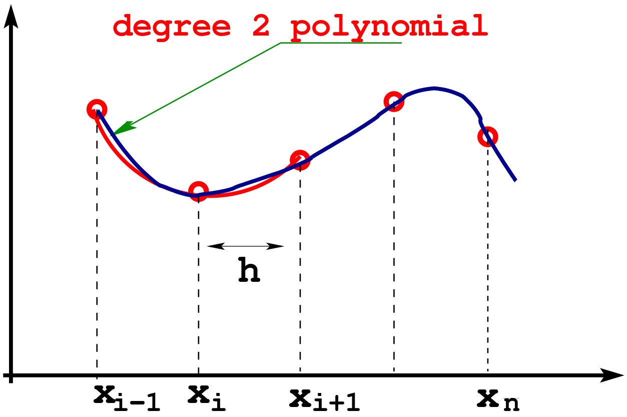
Within same level of integration, integral estimation is improving (check variation)

Next level of integration is better (check)

Optimal value will be $I_{n,n}$

Simpson rule

- ✔ Making $n = 2$ in Newton-Cotes formula is equivalent to use a degree 2 polynomial approximation for describing the function $f(x)$



- ✔ This method requires segments defined by **pairs of slices** in order to have the polynomial defined (adjacent slices)
- ✔ The result is that the **number of slices has to be even**. The integral for a pair of slices made with the three points $[x_{i-1}, x_i, x_{i+1}]$

$$F_i = \int_{x_{i-1}}^{x_{i+1}} f(x)\, dx \simeq \frac{h}{3}\left[f(x_{i-1}) + 4f(x_i) + f(x_{i+1})\right]$$

Simpson rule (cont.)

- ✔ For an integration range $[a, b]$, we divide it in n intervals (even) of width $h = \frac{b-a}{n}$,

$$\begin{aligned} F &= \int_a^b f(x)\, dx \simeq \sum_{i=1,3,5,\cdots}^{n} \left[\int_{x_{i-1}}^{x_{i+1}} f(x)dx\right] \\ &= \frac{h}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\right] \end{aligned}$$

- ✔ Error:

$$\varDelta F = \frac{(b-a)h^4}{180} f^{(4)}(\chi)$$

Simpson rule requires that the number of slices n shall be even. If this is not the case, we can integrate over the $n-1$ slices with Simpson method and integrate the last slice using a degree 2 polynomial built from $[x_{n-2}, x_{n-1}, x_n]$

$$\int_{x_n-h}^{x_n} f(x)dx = \frac{h}{12}\left(-f_{n-2} + 8f_{n-1} + 5f_n\right)$$

Integration errors: step size

Aiming at obtaining an accuracy ε

trapezoidal rule

$$\Delta F = \frac{h^2}{12}(b-a)M_{(2)} = \frac{(b-a)^3}{12}\frac{M_{(2)}}{n^2} < \varepsilon \;\Rightarrow\; \boxed{n^2 > \frac{1}{\varepsilon}\frac{M_{(2)}}{12}(b-a)^3}$$

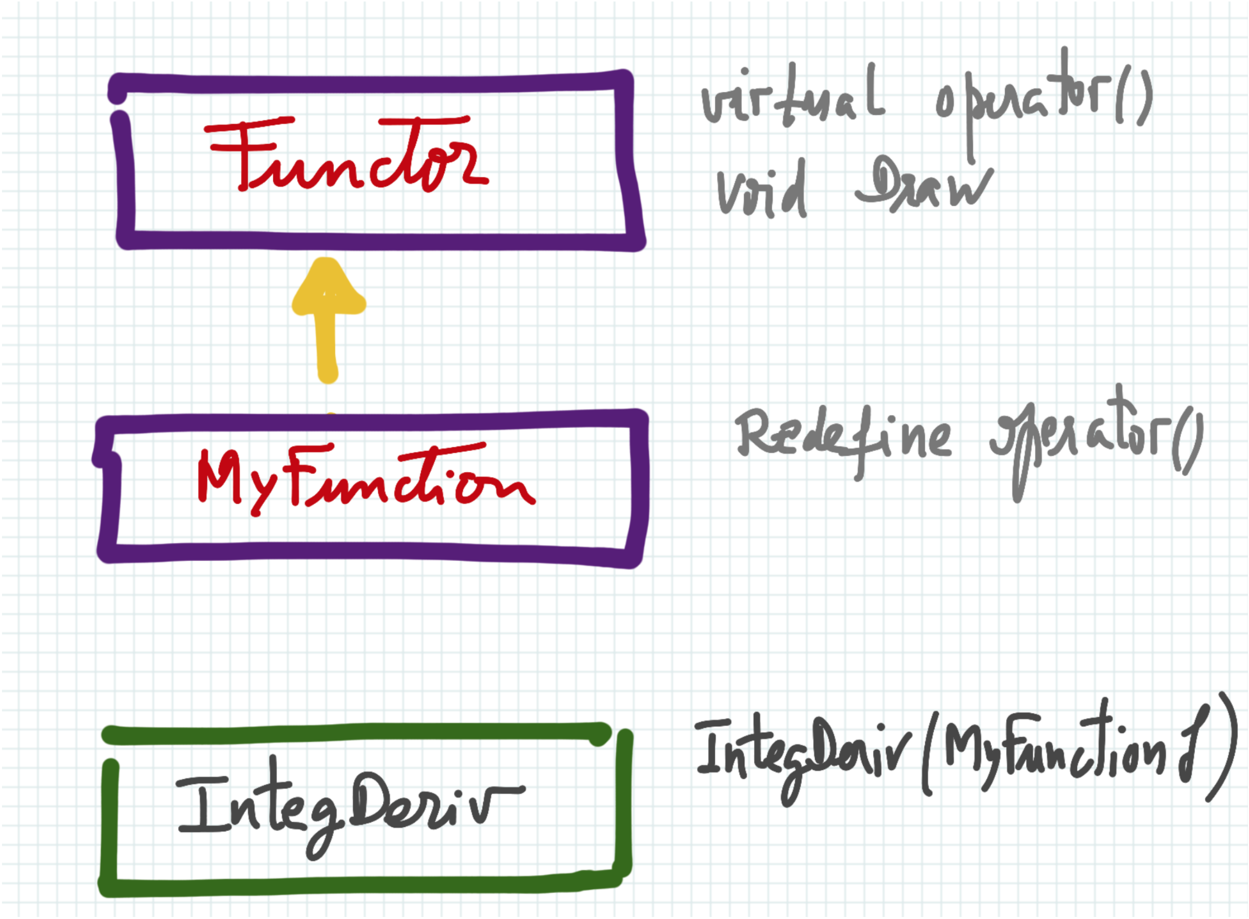
simpson rule

$$\Delta F = \frac{h^4}{180}(b-a)M_{(4)} = \frac{(b-a)^5}{180}\frac{M_{(4)}}{n^4} < \varepsilon \;\Rightarrow\; \boxed{n^4 > \frac{1}{\varepsilon}\frac{M_{(4)}}{180}(b-a)^5}$$

mid-point rule

$$\Delta F = \frac{h^2}{12}(b-a)M_{(2)} = \frac{(b-a)^3}{24}\frac{M_{(2)}}{n^2} < \varepsilon \;\Rightarrow\; \boxed{n^2 > \frac{1}{\varepsilon}\frac{M_{(2)}}{24}(b-a)^3}$$

C++ classes



C++ classes: Functor

Functor

```
#include <string>
#include "TCanvas.h"

class Functor {

 public:
  Functor(std::string s="Functor") : name(s) {;}
  ~Functor() = default;

  virtual double operator()(double x);

  virtual void Draw(double xi, double xf, int num,
                    std::string xtitle="x", std::string ytitle="y");

protected:
  static TCanvas *c;
  std::string name;
};
```

C++ classes: MyFunction

MyFunction.h

```
#include "Functor.h"

class MyFunction : public Functor {
 public:
  MyFunction() : Functor("MyFunction") {;}

  // redefine function
  double operator()(double);
};
```

MyFunction.C

```
#include <cmath>

double MyFunction::operator() (double x) {
  return pow(x,4.)*sqrt(x*x+1);
}
```

C++ classes: IntegDeriv

IntegDeriv

```
#include "MyFunction.h"
class IntegDeriv {

 public:
  IntegDeriv(Functor&);
  ~IntegDeriv() = default;

  // test function
  void Dump(double, double);

  // integration methods
  void TrapezoidalRule(double xi, double xf, double& Integral, double& Err
  void simpsonRule(double xi, double xf, double& Integral, double& Error);

  // derivative methods

private:
  Functor& F;
};
```

Computational Physics

Monte-Carlo methods

Fernando Barao, Phys Department IST (Lisbon)

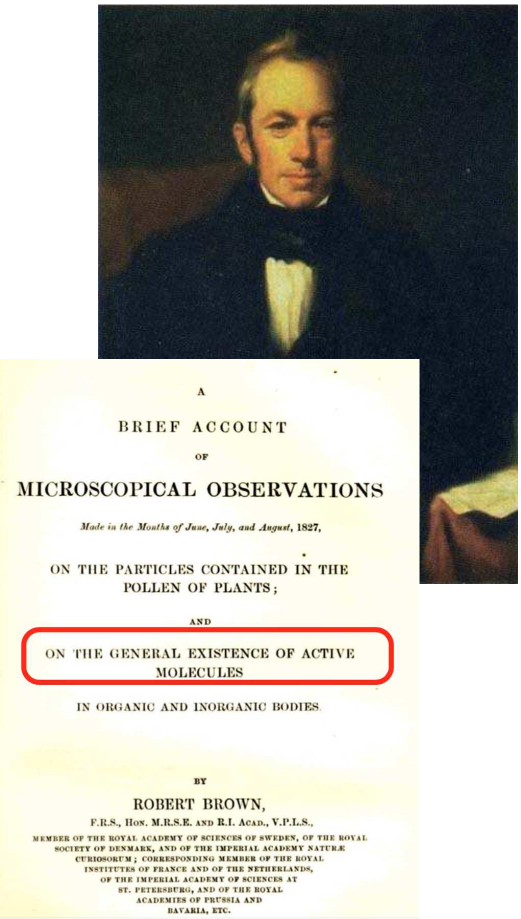
Random Walk and Diffusion

- ✔ Robert Brown a scotish botanist made observations in 1827 of pollen particles in water having random motion

  ☞ **Brownian motion was born!**

- ✔ The atomic nature of matter was fairly widely accepted among scientists at the turn of 20th century, but not universally

  ☞ **NO experimental evidence!**



A

BRIEF ACCOUNT

OF

MICROSCOPICAL OBSERVATIONS

Made in the Months of June, July, and August, 1827,

ON THE PARTICLES CONTAINED IN THE POLLEN OF PLANTS;

AND

ON THE GENERAL EXISTENCE OF ACTIVE MOLECULES

IN ORGANIC AND INORGANIC BODIES.

BY

ROBERT BROWN,

F.R.S., HON. M.R.S.E. AND R.I. ACAD., V.P.L.S.,

MEMBER OF THE ROYAL ACADEMY OF SCIENCES OF SWEDEN, OF THE ROYAL SOCIETY OF DENMARK, AND OF THE IMPERIAL ACADEMY NATURÆ CURIOSORUM; CORRESPONDING MEMBER OF THE ROYAL INSTITUTES OF FRANCE AND OF THE NETHERLANDS, OF THE IMPERIAL ACADEMY OF SCIENCES AT ST. PETERSBURG, AND OF THE ROYAL ACADEMIES OF PRUSSIA AND BAVARIA, ETC.

Random Walk and Diffusion (cont.)

- ✔ 1905 was a great year for Physics! A. Einstein published, "On the Movement of Small Particles Suspended in Stationary Liquids Required by the Molecular-Kinetic Theory of Heat", Annalen der Physik 19, p. 549 (1905)

- ✔ He links the Brownian motion with the atomic nature of matter, providing proof ideas to be explored in laboratory!

  look to the mean square displacement $< \Delta r^2 >$ of particles that can be directly observed

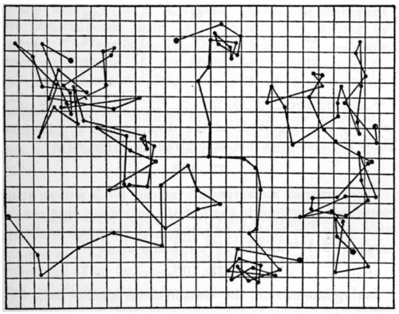
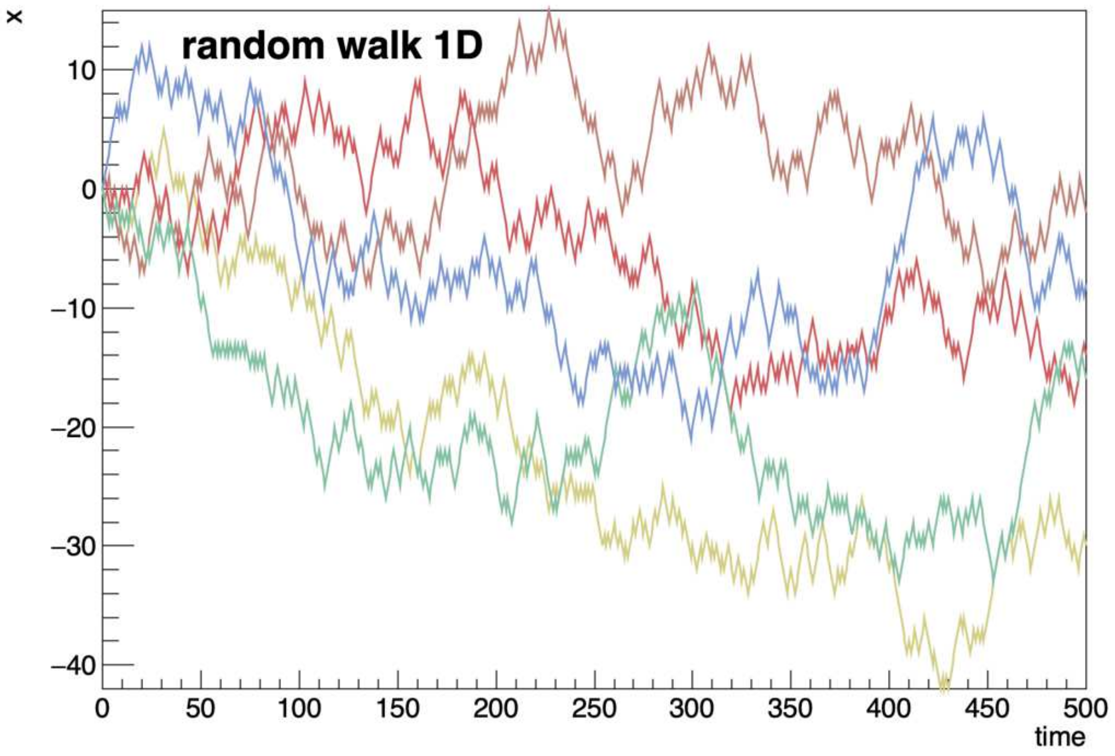


Fig. 1. These tracks of three particles are by J. Perrin.[6] The dots show the particle positions at 30-second intervals, with lines joining successive points. The scale is 1 division equals 0.0003125 cm. The particle radius is 0.52 μm.

Random Walk and Diffusion (cont.)

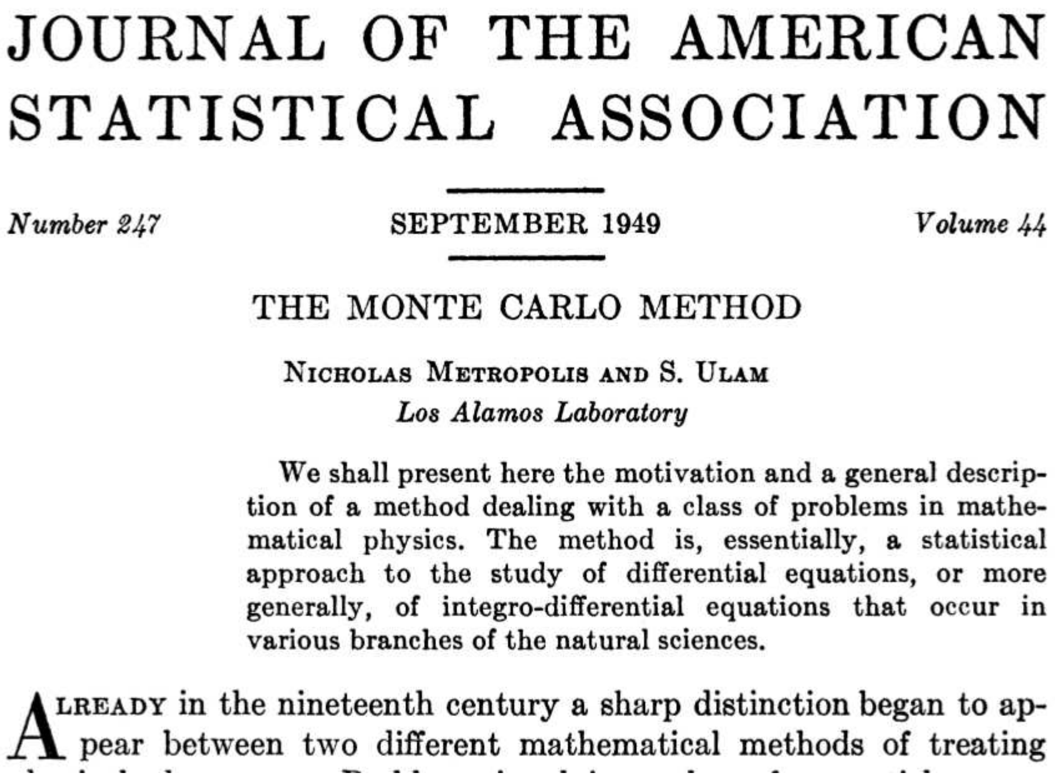
Homework proposed this week: study the random walk of N particles leaving at $t = 0$ from $x = 0$, at one-dimension



Monte-Carlo methods

- ✔ any method using random variables for a numerical calculation
  - ☞ we ask for a statistical answer!
- ✔ founding article:
  "The monte carlo method", N. Metropolis, S. Ulam (1949)
- ✔ applications: physics, engineering, finance, ...
- ✔ aims of the method:
  - ► generate samples of random variables ($\vec{X}$) according to a density probability distribution $p(\vec{X})$
  - ► estimate expectation values ($<>$) of variables or functions

Monte-Carlo methods



JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION

Number 247 SEPTEMBER 1949 *Volume 44*

THE MONTE CARLO METHOD

NICHOLAS METROPOLIS AND S. ULAM
Los Alamos Laboratory

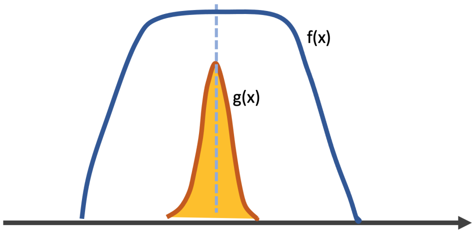
We shall present here the motivation and a general description of a method dealing with a class of problems in mathematical physics. The method is, essentially, a statistical approach to the study of differential equations, or more generally, of integro-differential equations that occur in various branches of the natural sciences.

ALREADY in the nineteenth century a sharp distinction began to appear between two different mathematical methods of treating

Statistical concepts

✔ the expected value of a variable X sampled N times $(X_1, X_2, \cdots, X_N)$

$$E(X) = \langle X \rangle = \frac{1}{N}\sum_{i=1}^{N} X_i$$



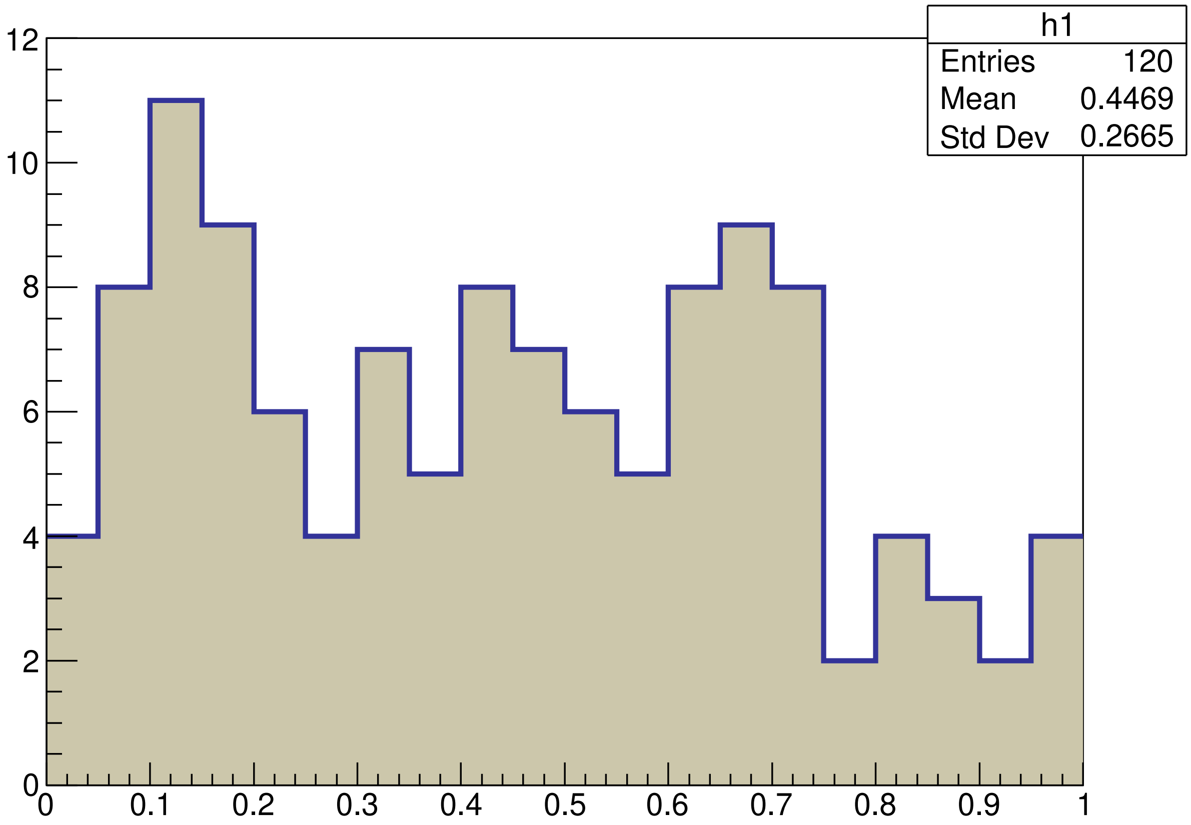
✔ the variance of the sample:

$$Var(X) \equiv \sigma_X^2 \simeq \frac{1}{N-1}\sum_{i=1}^{N-1}(X_i - \langle X \rangle)^2 = \left\langle (X - \langle X \rangle)^2 \right\rangle = \left\langle X^2 \right\rangle - \langle X \rangle^2$$

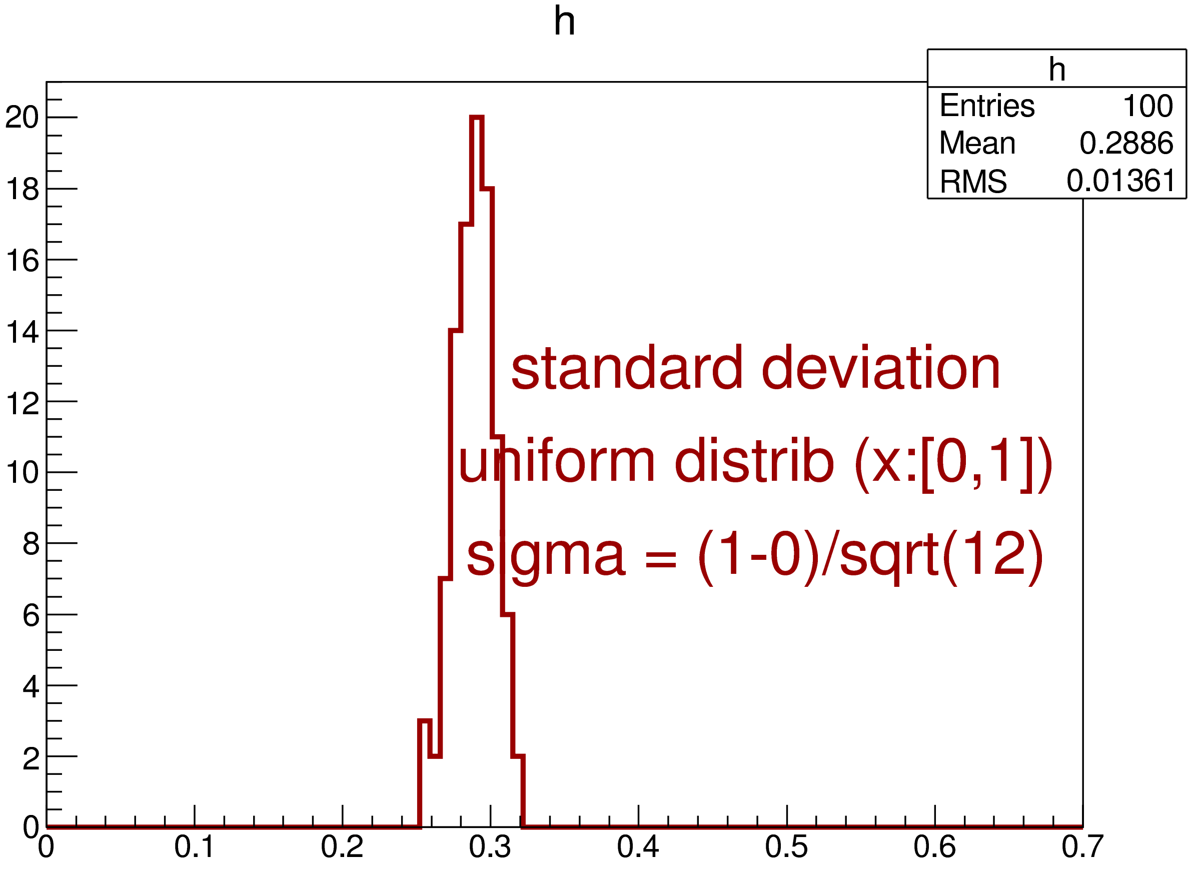
✔ the standard deviation of the sample:

$$\sigma_X \simeq \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(X_i - \langle X \rangle)^2}$$

uniform distribution



uniform dist: sigma



standard deviation

uniform distrib (x:[0,1])

sigma = (1-0)/sqrt(12)

Random numbers: generators

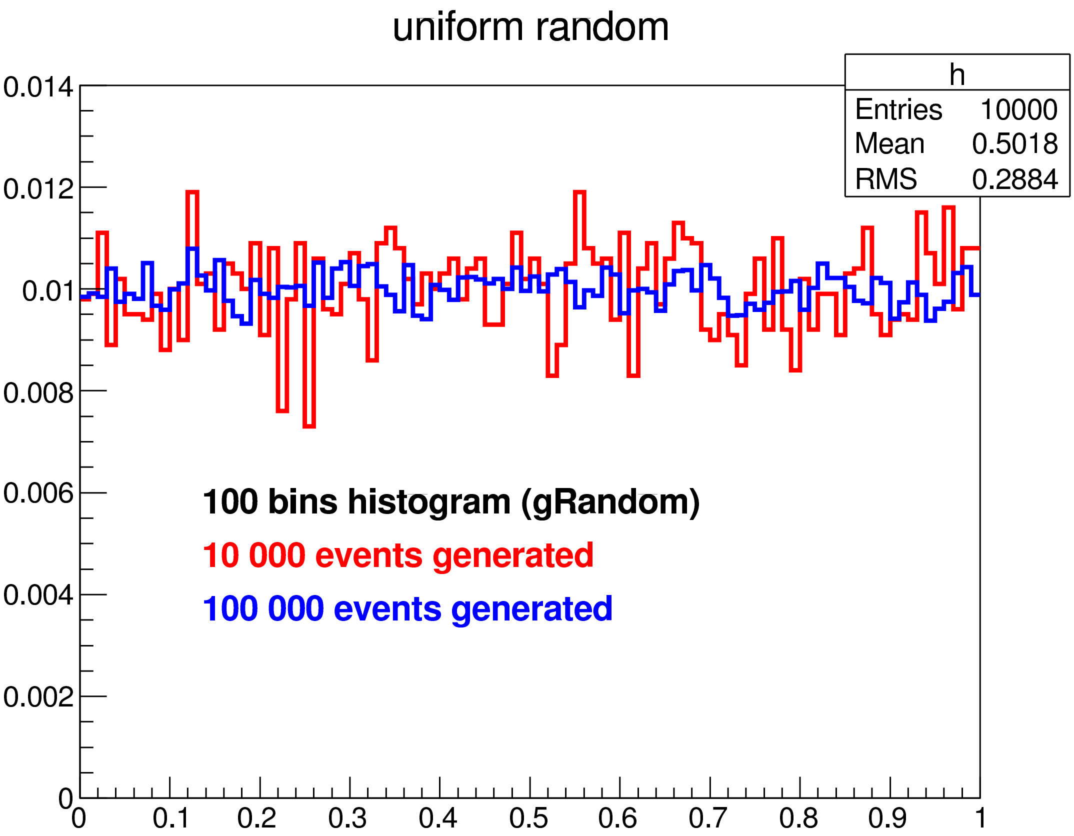
- ✔ the most common uniform random number generators are based on Linear Congruential relations

$$N_i = (aN_{i-1} + c)\,\%m$$

  - ☞ For example, with the parameters: a=6, c=7, m=5
    and a seed: $N_0 = 2$
    we get a **period 5** generator: 4, 1, 3, 0, 2, 4, 1, 3, 0, 2, ...

- ✔ a good uniform random number generator:
  - ☞ produces a uniform distribution in the all the generation range
  - ☞ shows no correlations between random numbers
  - ☞ the period of sequence repetition is as large as possible
  - ☞ the generation algorithm shall be fast

Random numbers



PDFs - prob density distributions

- ✔ PDFs: the probability density function $p(X)$ give us the probability of an event (a value X_i in this case) to occur

$$\int_{-\infty}^{+\infty} p(X)\, dX = 1$$

- ✔ for a discrete variable X, its expectation value is given by:

$$\langle X\rangle = \frac{1}{N}\sum_{i=1}^{N} p(X_i)\; X_i$$

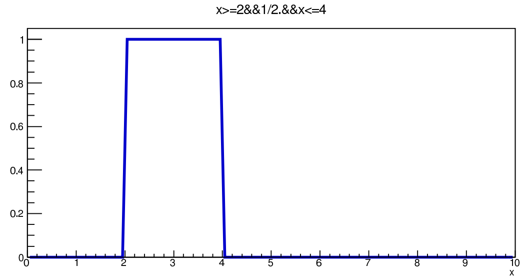
- ✔ for a continuous variable X or function $f(X)$, the expectation value is given by:

$$\langle X\rangle = \int_{-\infty}^{+\infty} p(X)\; X\; dX$$
$$\langle f\rangle = \int_{-\infty}^{+\infty} p(X)\; f(X)\; dX$$

Important PDFs

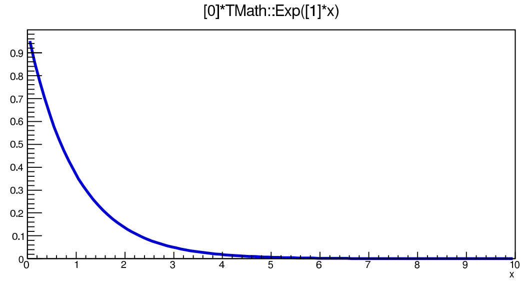
- ✔ uniform distribution: $X[a, b]$

$$p(X) = \frac{1}{b-a}H(X-a)H(b-X)$$



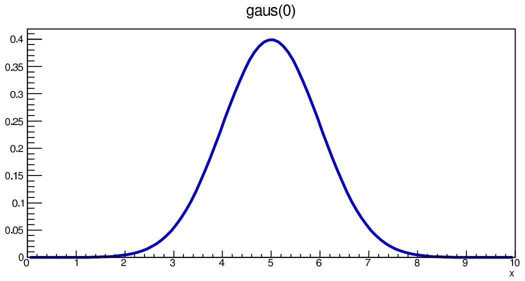
- ✔ exponential distribution: $X[0, \infty]$

$$p(X) = \alpha e^{-\alpha X}$$



- ✔ normal distribution: $X[-\infty, +\infty]$

$$p(X) = \frac{1}{\sqrt{2\pi}\sigma}\; e^{-\frac{1}{2}\left(\frac{X-\mu}{\sigma}\right)^2}$$



Simulation

- ✔ **Simulation** is very important for understanding real situations or for modelling the behaviour of a system

  it is largely used on particle and astroparticle physics for designing the instruments used for particles detection

- ✔ the various real conditions the system has can be introduced easily in a simulated process

- ✔ Suppose you had to design a detector system for detecting photons coming from Compton scattering on a material?

  I assume my gamma source emits a beam very colimated along an axis (x for instance) and in between I have a block of material where Compton is going to happen...

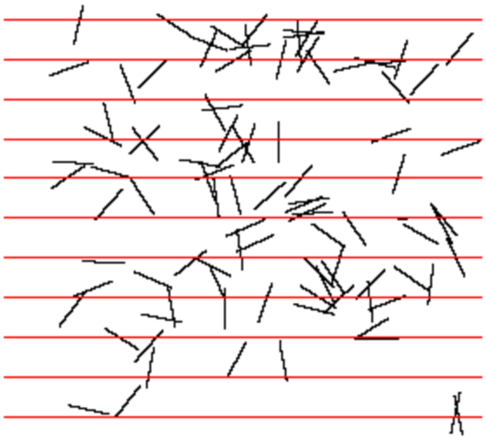
  What we need to know?

Buffon's needle problem

Buffon's needle problem is a question first posed in the 18th century by Georges-Louis Leclerc where simulation can help us a lot!

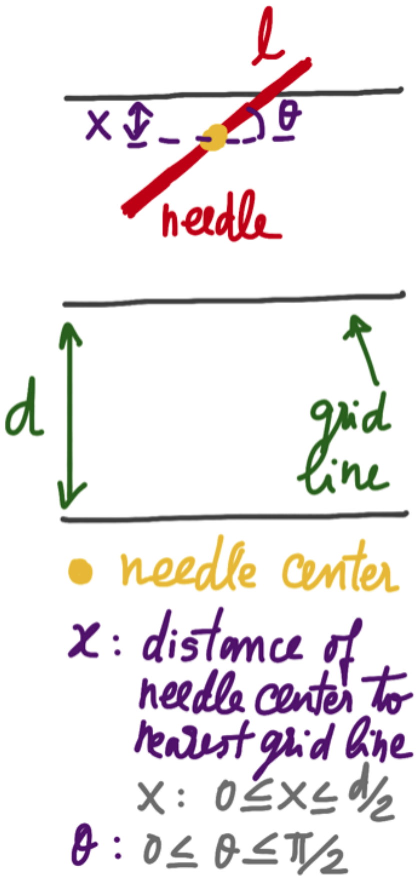
A needle of length ℓ is thrown randomly onto a grid of parallel lines, separated by a distance d, with $d > \ell$

What is the probability that a needle intersects a line?



Buffon's needle modelling

- ✔ Our phase-space is made of variables x and θ which are defined in the ranges, $x : [0, d]$ and $\theta : [0, \pi/2]$
  their phase-space make a rectangle of sides d and $\pi/2$
- ✔ A needle crossing a line has to fullfill the condition,
  $x < \frac{\ell}{2} \sin\theta$
  where x is the needle center distance wrt nearest grid line
- ✔ The probability of crossing a line is calculated as the ratio between the two areas:
  1) the area of the condition function: $A_1 = \int_0^{\pi/2} \frac{\ell}{2} \sin\theta d\theta$
  2) the area of the rectangle: $A_2 = \pi/2 \; d/2$
  $p = \frac{\int_0^{\pi/2} \ell \sin\theta \, d\theta}{\pi/2 \; d} = \frac{2\ell}{\pi d}$
- ✔ A more elaborated reasoning can be made based on joint probability density $p(x, \theta)$ which corresponds to the probability of having a given (x, θ) pair of values.
  The fact that these variables are independent allow us to write: $p(x, \theta) = p(x) \; p(\theta)$

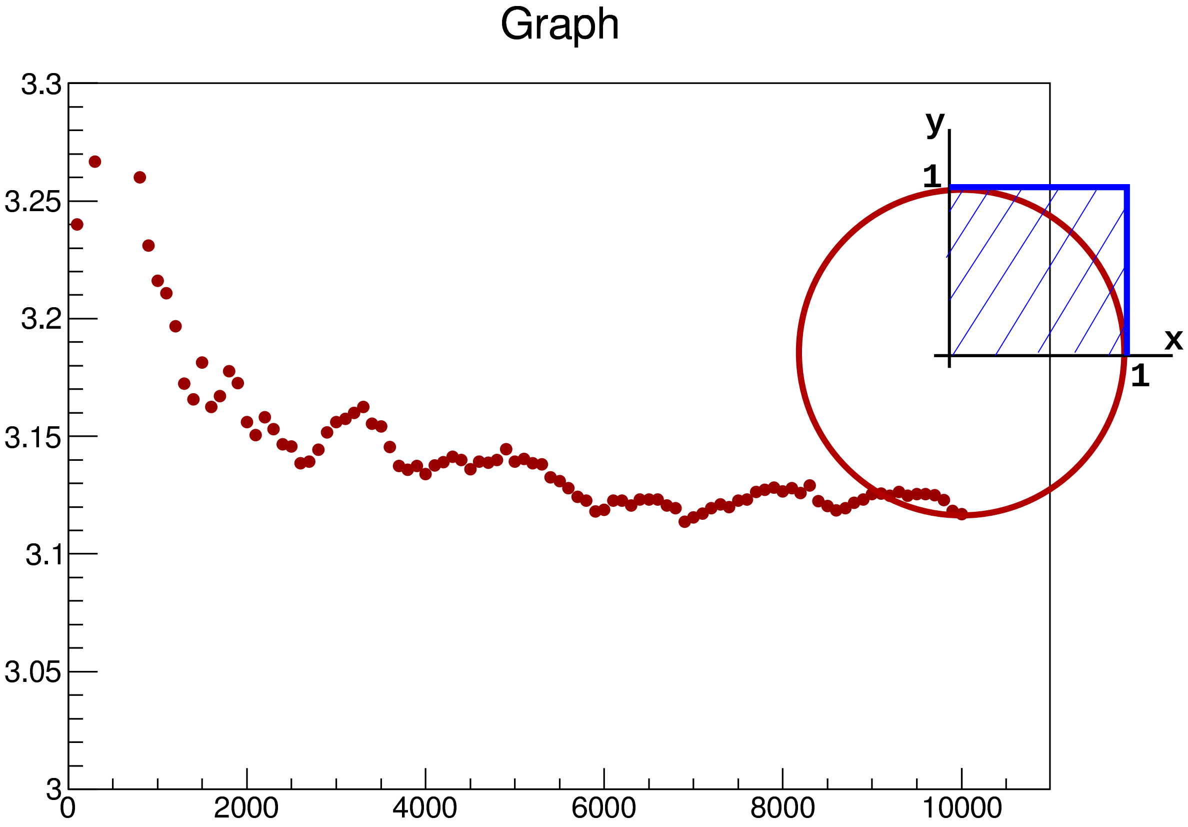


Buffon's needle modelling

Making a simulation experiment:

- ✔ throw a needle: generate a random angle θ and a random needle center x
  $x : [0, d/2]$ and $\theta : [0, \pi/2]$
- ✔ a needle do cross a line if $x < \frac{\ell}{2} \sin(\theta)$
- ✔ count how many times a needle do cross a line and compute probability,
  $P_{crossing} = \frac{N^{events}_{crossing}}{N^{events}_{total}}$
- ✔ to be explored: π calculation !
  it comes from the fact that the probability of crossing a line depends on π

Pi evaluation



Generating random variables

- The common problem is to have a variable x distributed according a given distribution function $p(x)$
  we are going to make a variable change such that the number of events (randoms) generated is independent of the used variable

  $$dN = p(x)dx = p(y)dy$$

- Suppose that y is a random variable distributed in $[0, 1]$ and x starts at x_0

  $$\boxed{p(y) = 1 \quad \Rightarrow \quad \int_0^y dy' = \int_{x_0}^x p(x')dx' \quad \Rightarrow \quad y = \int_{x_0}^x p(x')dx'}$$

- For generating a random variable x in a interval $[x_0, x_1]$ we just make sure that:

  $$\int_{x_0}^{x_1} p(x)dx = 1$$

  and we invert the relation above (boxed) giving us $x(y)$.
  Provided a random y in $[0, 1]$ we generate a random x in $[x_0, x_1]$.

uniform [0,1] → uniform [a,b]

✔ Let's transform a variable
y uniformly distributed in $[0,1]$
into a variable
x uniformly distributed in $[a,b]$

✔ Normalization of $p(x)$:

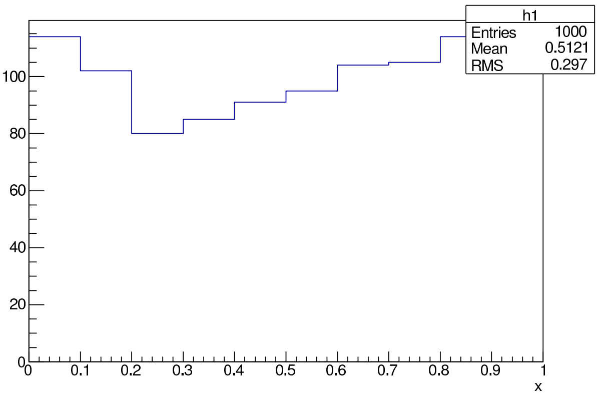
$$p(x) = \frac{1}{b-a}$$

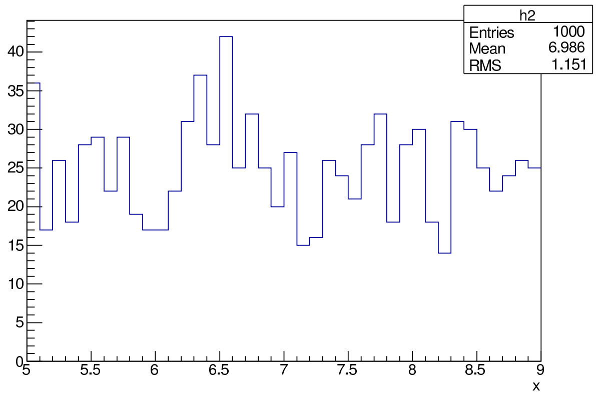
✔ Transformation:

$$y = \int_a^x \frac{1}{b-a} dx' = \frac{x-a}{b-a}$$

$$\Rightarrow \quad x - a = (b-a)y$$

$$\boxed{x = a + (b-a)\,y}$$





uniform [0,1] → exponential $[0,\infty]$

✔ Let's transform a variable
☞ y uniformly distributed in $[0,1]$
into a variable
☞ x distributed according to
$\boxed{p(x) \propto e^{-x}}$ in $[0,\infty]$

✔ Normalization of $p(x)$:

$$k \int_0^{+\infty} e^{-x}\, dx = 1$$

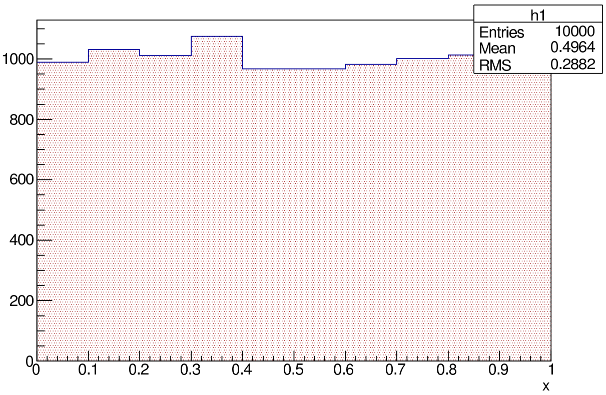
$$\Rightarrow k\left[-e^{-x}\right]_0^\infty = 1 \Rightarrow \quad k = 1$$

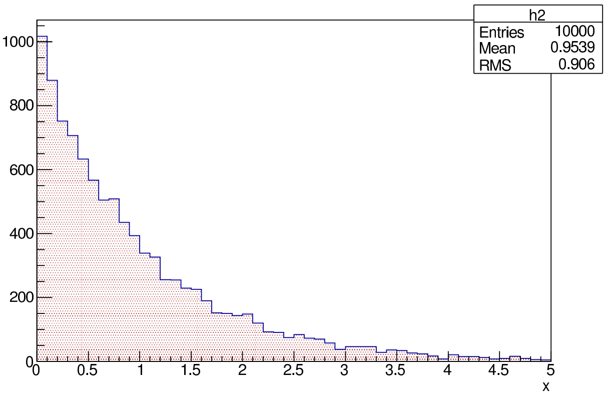
$$\boxed{p(x) = e^{-x}}$$

✔ Transformation:

$$y = \int_0^x e^{-x'} dx' = \left[-e^{-x'}\right]_0^x = 1 - e^{-x}$$

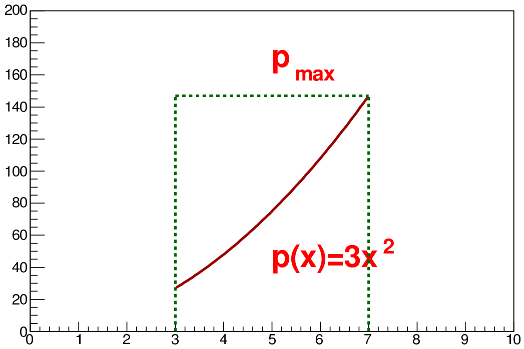
$$\boxed{x = -\ln(1-y)}$$

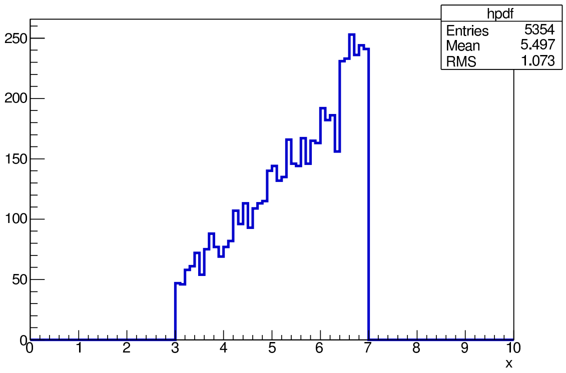




randoms gen: acceptance-rejection

- ✔ For applying the method before said we need to integrate the pdf and invert it
- ✔ In case it is not possible we can use the more generic but less efficient method of acceptance-rejection
- ✔ For generating a x variable in the interval $[a, b]$ distributed according to a pdf $p(x)$ we do the following:
  - ☞ generate a uniform random x_R between $[a, b]$
  - ☞ compute $p(x_R)$ and the ratio $\frac{p(x_R)}{p_{max}}$
  - ☞ generate a second random u_R from $U(0, 1)$
  - ☞ if $u_R \leq \frac{p(x_R)}{p_{max}}$ accept the variable x_R





Monte Carlo integration

- ✔ we want to evaluate the following integral:

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

- ✔ remember that the expectation value of the function $f(x)$ for x distributed according to a PDF $p(x)$

$$\langle f \rangle = \int_a^b f(x)\, p(x)\, dx \quad \text{with:} \int_a^b p(x)\, dx = 1$$

- ✔ choosing x to be uniformly distributed in the interval $[a, b]$, one has:

$$\begin{aligned} p(x) &= \tfrac{1}{b-a} \\ \langle f \rangle &= \int_a^b f(x)\, p(x)\, dx = \tfrac{1}{b-a} \int_a^b f(x)\, dx \end{aligned}$$

MC integration

$$\begin{aligned} F &= \int_a^b f(x)\, dx \\ &= (b-a)\langle f \rangle \\ &= \tfrac{(b-a)}{N} \sum_{i=1}^N f(x_i) \end{aligned}$$

x_i is a random variable uniformly distributed in the interval $[a, b]$

error estimation

$$\sigma_F = (b-a)\sigma_{\langle f \rangle}$$

$$\sigma_f^2 = \left\langle f^2 \right\rangle - \langle f \rangle^2$$

$$\sigma_{\langle f \rangle}^2 = \frac{\sigma_f^2}{N}$$

$$\boxed{\sigma_F = (b-a)\frac{\sigma_f}{\sqrt{N}} = \frac{(b-a)}{\sqrt{N}} \sqrt{\frac{1}{N}\sum_{i=1}^N \left(f(x_i)\right)^2 - \left(\frac{1}{N}\sum_{i=1}^N f(x_i)\right)^2}}$$

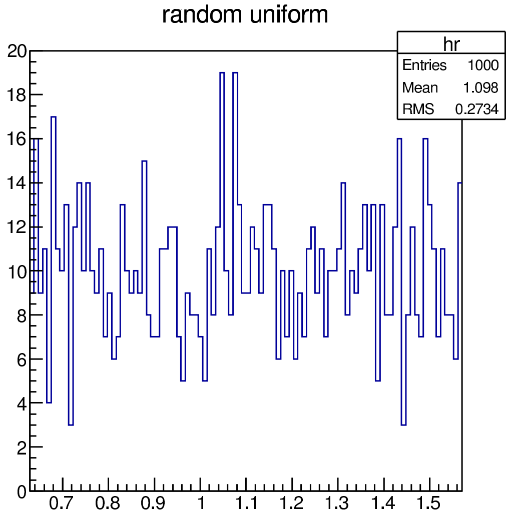
MC integration (cont.)

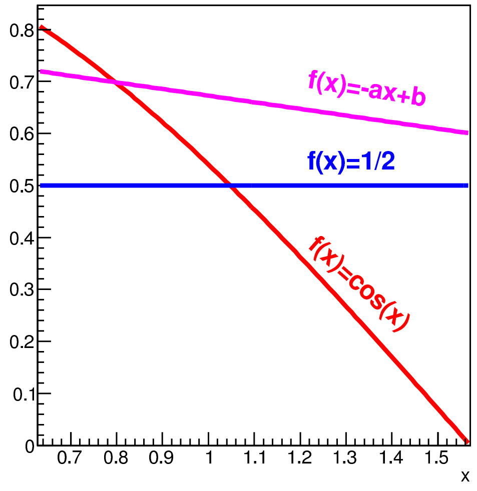
Let's compute the integrals of the functions:

$$\int_{0.2\pi}^{0.5\pi} \frac{dx}{2} = 0.471239$$
$$\int_{0.2\pi}^{0.5\pi} cos(x)\, dx = 0.412215$$
$$\int_{0.2\pi}^{0.5\pi} (ax+b)\, dx = 0.622035$$

Throwing 100 random variable uniformly distributed we obtain the following results:

$$\int_{0.2\pi}^{0.5\pi} \frac{dx}{2} = 0.471239 \pm 0.000000$$
$$\int_{0.2\pi}^{0.5\pi} cos(x)\, dx = 0.413671 \pm 0.007098$$
$$\int_{0.2\pi}^{0.5\pi} (ax+b)\, dx = 0.622280 \pm 0.001037$$





MC integration: algorithm

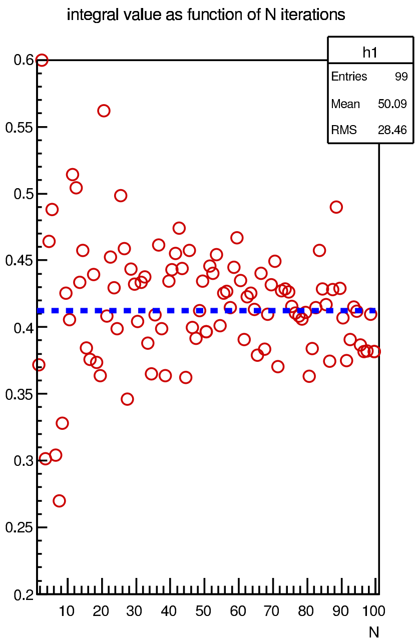
```
double xmin=TMath::Pi()*0.2;
double xmax=TMath::Pi()*0.5;
int N = 1000;

TF1 *f1 = new TF1("f1","TMath::Abs(cos(x))",xmin,xmax);
TF1 *f2 = new TF1("f2","0.5",xmin,xmax);
TF1 *f3 = new TF1("f3","-0.4/TMath::Pi()*x+0.8",xmin,xmax);
(...)
for (int i=0; i<N; i++) {
   double x = xmin + (xmax-xmin)*gRandom->Uniform();
   F1 += f1->Eval(x);
   F2 += f2->Eval(x);
   F3 += f3->Eval(x);
   f1s += f1->Eval(x) * f1->Eval(x);
   f2s += f2->Eval(x) * f2->Eval(x);
   f3s += f3->Eval(x) * f3->Eval(x);
 }
double f1m = F1/N; //mean
double f2m = F2/N;
double f3m = F3/N;
```

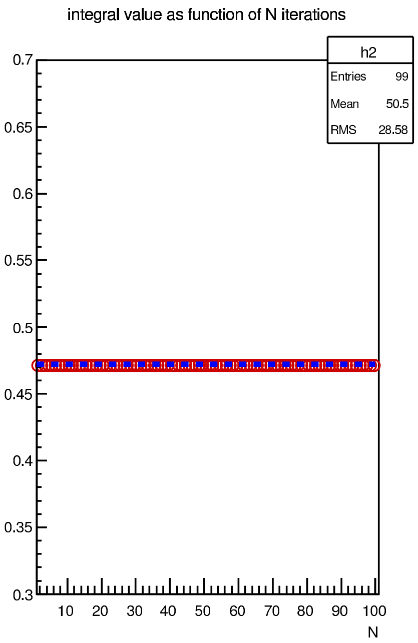
algorithm

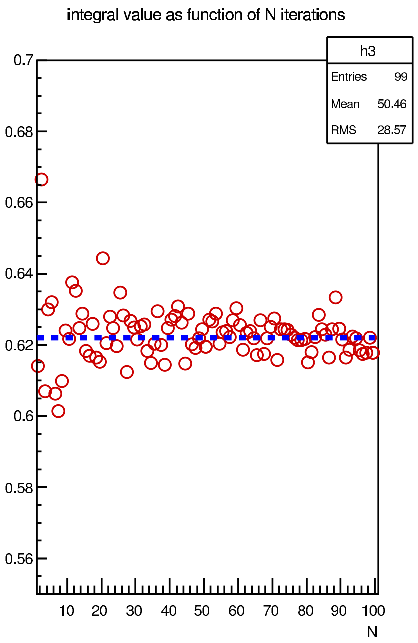
```
// integrals
double I1 = f1m*(xmax-xmin);
double I2 = f2m*(xmax-xmin);
double I3 = f3m*(xmax-xmin);

// variances
double Var1 = f1s/N - f1m*f1m;
double Var2 = f2s/N - f2m*f2m;
double Var3 = f3s/N - f3m*f3m;

// errors
double E1 = (xmax-xmin)/sqrt(N)*sqrt(Var1);
double E2 = (xmax-xmin)/sqrt(N)*sqrt(Var2);
double E3 = (xmax-xmin)/sqrt(N)*sqrt(Var3);
```

MC integration (cont.)

Let's check the integral value as function of the number of random variables generated N







$$F = \int_{0.2\pi}^{0.5\pi} cos(x)\, dx$$

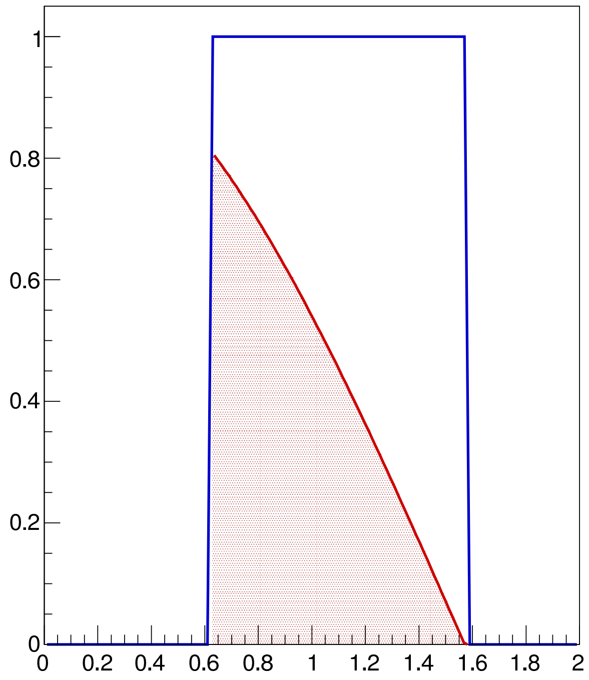
$$F = \int_{0.2\pi}^{0.5\pi} \frac{dx}{2}$$

$$F = \int_{0.2\pi}^{0.5\pi} (ax + b)\, dx$$

MC integration: von Neumann

- ✔ method introduced by von Neumann
- ✔ for integrating the function we define an envelope with an area $A = (x_{max} - x_{min}) * f_{max}$
- ✔ generate two random variables
  - ☞ x_R in in range $[x_{min}, x_{max}]$
  - ☞ f_R in in range $[0, f_{max}]$
- ✔ count the number of events N_R that $\boxed{f_R \leq f(x_R)}$
- ✔ the integral $\boxed{I = (x_{max} - x_{min})\, f_{max}\, \frac{N_R}{N}}$
- ✔ the integral error

$$\boxed{\sigma_I = \frac{(x_{max}-x_{min})\, f_{max}}{N} \sqrt{N_R\left(1 - \frac{N_R}{N}\right)}}$$



Using 100 randoms:
$$\int_{0.2\pi}^{0.5\pi} cos(x)\, dx = 0.435 \pm 0.037$$

radioactive decay

- ✔ Monte-Carlo methods can be applied to radioactive decays since these are random processes

  the number of decay counts in a time interval **t**, if counted several times, will be different

- ✔ Every nuclei decays with constant probability in a time interval **dt**

$$\boxed{P = \lambda\, dt = \frac{dt}{\tau}}$$

τ, lifetime of the nuclei

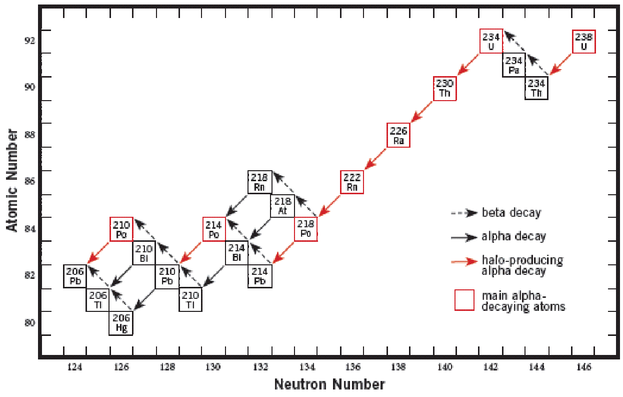
- ✔ The analytical solution will provide a model for the average behaviour, but nothing about the random behaviour

  The decay law is given by:

$$N(t+dt) = N(t) - \lambda\, dt\, N(t) \Rightarrow \frac{dN}{N} = -\frac{d}{\tau}$$

$$\int_{N_0}^{N} \frac{dN}{N} = -\int_0^t \frac{dt}{\tau} \Rightarrow \boxed{N(t) = N_0\, e^{-\frac{t}{\tau}}}$$

U238 chain decay



radioactive decay

N_0 (initial number of nuclei)

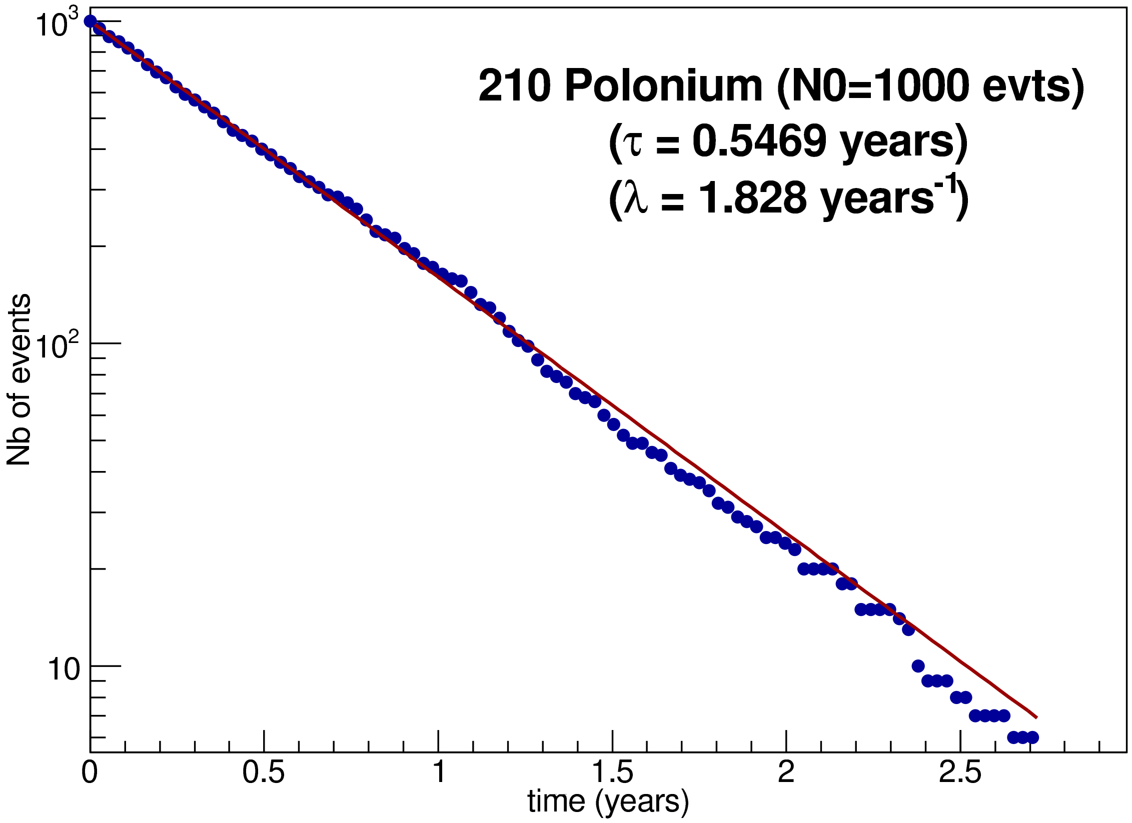
λ (decay constant)

dt (time step)

algorithm

```
N = N0
LOOP from t=Ti to Tf, step dt

   LOOP over each remaining parent nucleus

     //Generate a random number R between [0,1] from a uniform distribution
     R = uniform_RandomNumber()

     // Decide if the nucleus decays (lambda=1/tau)
     // => reduce the number of parents by 1
     IF( R < lambda dt ) N = N-1

   END LOOP over nuclei

   // calculate theoretical value
   Ntheory = N0 * exp(-lambda t)

   Record t, N and Ntheory

END LOOP over time
```

210Po decay


210 Polonium (N0=1000 evts)
(τ = 0.5469 years)
(λ = 1.828 years^{-1})
Nb of events
10^3
10^2
10
0
0.5
1
1.5
2
2.5
time (years)