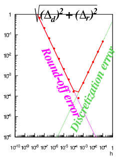
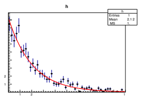
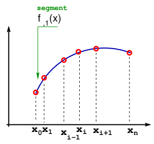
Computational Physics

numerical methods with C++ (and UNIX)







Fernando Barao
Instituto Superior Tecnico, Dep. Fisica
email: barao at lip.pt

Numerical methods

- System of linear equations
  - Gauss elimination
  - LU decomposition
  - Gauss-Seidel method
- Interpolation
  - Lagrange interpolation
  - Newton method
  - Neville method
  - Cubic spline
- Numerical derivatives
  - First derivative $\mathcal{O}(h^2)$, $\mathcal{O}(h^4)$
  - Second derivative $\mathcal{O}(h^2)$, $\mathcal{O}(h^4)$
  - Derivative by interpolation
- Numerical integration
  - Newton-Cotes: trapezoidal and Simpson rules
  - Gaussian quadrature
- Monte-Carlo methods

Numerical methods

- System of linear equations
  - Gauss elimination
  - LU decomposition
  - Gauss-Seidel method
- Interpolation
  - Lagrange interpolation
  - Newton method
  - Neville method
  - Cubic spline
- Numerical derivatives
  - First derivative $\mathcal{O}(h^2)$, $\mathcal{O}(h^4)$
  - Second derivative $\mathcal{O}(h^2)$, $\mathcal{O}(h^4)$
  - Derivative by interpolation
- Numerical integration
  - Newton-Cotes: trapezoidal and Simpson rules
  - Gaussian quadrature
- Monte-Carlo methods

Computational Physics

Numerical derivatives

Fernando Barao, Phys Department IST (Lisbon)

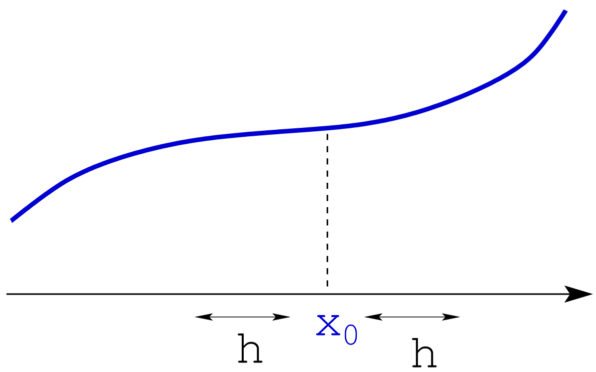
Numerical methods

- ✔ System of linear equations
  - ▶ Gauss elimination
  - ▶ LU decomposition
  - ▶ Gauss-Seidel method
- ✔ Interpolation
  - ▶ Lagrange interpolation
  - ▶ Newton method
  - ▶ Neville method
  - ▶ Cubic spline
- ✔ Numerical derivatives
  - ▶ First derivative $\mathcal{O}(h^2)$, $\mathcal{O}(h^4)$
  - ▶ Second derivative $\mathcal{O}(h^2)$, $\mathcal{O}(h^4)$
  - ▶ Derivative by interpolation
- ✔ Numerical integration
  - ▶ Newton-Cotes: trapezoidal and Simpson rules
  - ▶ Gaussian quadrature
- ✔ Monte-Carlo methods

functions: Taylor expansion

✔ A function can be approximated by a polynomial of order n

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



✔ coefficients ($h=0$)

$$f(x_0) = a_0$$

$$f_h^{'}(x_0) = a_1$$

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

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

Numerical 1st derivative

- The differentiation of a function is one of the most basic tasks in physics: $\frac{d\vec{v}}{dt} = \frac{\vec{F}}{m}$
- **forward difference**

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

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

- **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:

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

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

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

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

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

Numerical 1st derivative (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 $f(x_0 \pm 2h)$

$$\begin{aligned} f(x_0+h) - f(x_0-h) &= 2hf'(x_0) + \tfrac{1}{3}f'''(x_0) + O(h^3) \\ f(x_0+2h) - f(x_0-2h) &= 4hf'(x_0) + \tfrac{16h^3}{3!}f'''(x_0) + O(h^5) \end{aligned}$$

- 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*:

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

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

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 derive finite difference approximations for unevenly spaced data
  - ☞ We can approximate the derivative of $f(x)$ by the derivative of an interpolant

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 ($f''(x_i)$)

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

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

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:

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

Numerical 2nd derivative

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

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

- Adding the function expansions 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:

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

- The five-pont formula:

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

Computational Physics

Numerical integration

Fernando Barao, Phys Department IST (Lisbon)

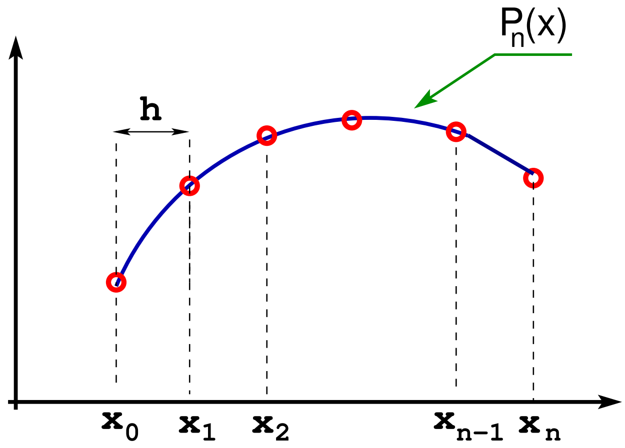
Numerical methods

- ✔ System of linear equations
  - Gauss elimination
  - LU decomposition
  - Gauss-Seidel method
- ✔ Interpolation
  - Lagrange interpolation
  - Newton method
  - Neville method
  - Cubic spline
- ✔ Numerical derivatives
  - First derivative $\mathcal{O}(h^2)$, $\mathcal{O}(h^4)$
  - Second derivative $\mathcal{O}(h^2)$, $\mathcal{O}(h^4)$
  - Derivative by interpolation
- ✔ Numerical integration
  - Newton-Cotes: trapezoidal and Simpson rules
  - Gaussian quadrature
- ✔ Monte-Carlo methods

Numerical integration: Newton-Cotes

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

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



- 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 $\mathbf{x_0}, \mathbf{x_1}, \cdots, \mathbf{x_n}$
- 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 integral can therefore be expressed as:

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

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

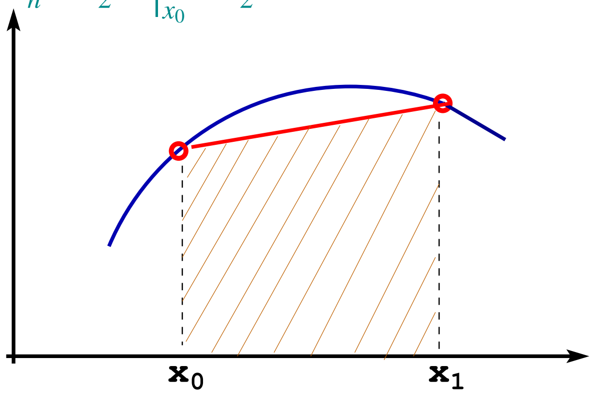
Trapezoidal rule

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

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

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

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



Trapezoidal rule error

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

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

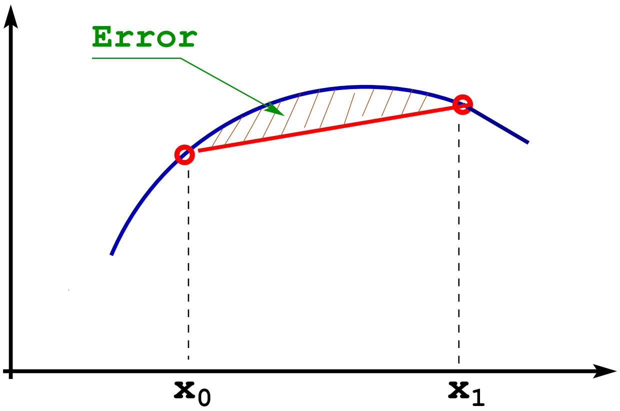
- ✔ For a given slice $[x_i, x_{i+1}]$, the truncation error associated to the linear approximation is

$$f(x) - P_1(x) = \frac{(x-x_i)(x-x_{i+1})}{(n+1)!} f''(\chi)$$

(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:

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

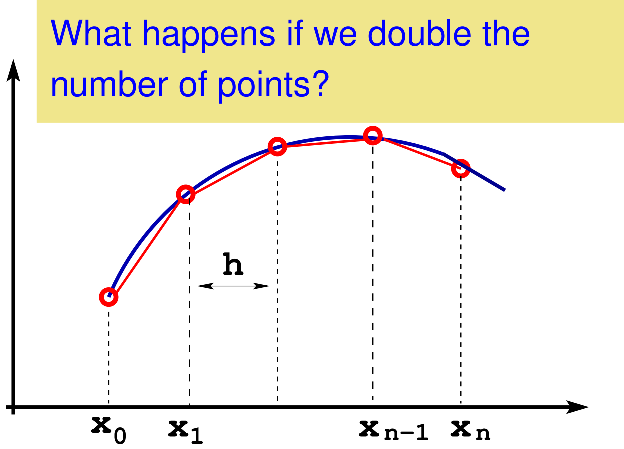
$$\frac{1}{2}(x - x_i)(x - x_{i+1})^2 \Big|_{x_i}^{x_{i+1}} - \frac{1}{2} \int_{x_i}^{x_{i+1}} (x - x_{i+1})^2 dx =$$

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

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

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**



- For every interval

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

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

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

- 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: Problem

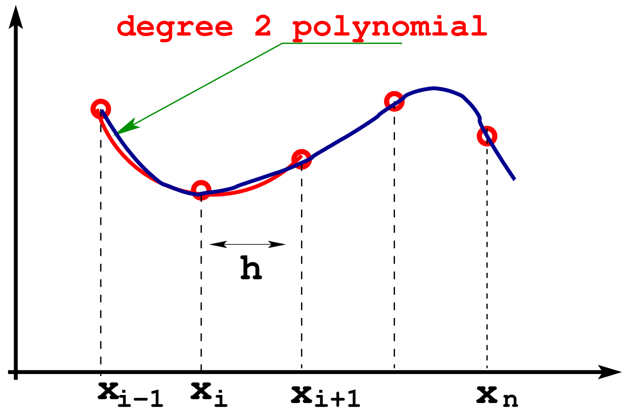
Calcular o integral

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

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

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 $h = \frac{b-a}{n}$,

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

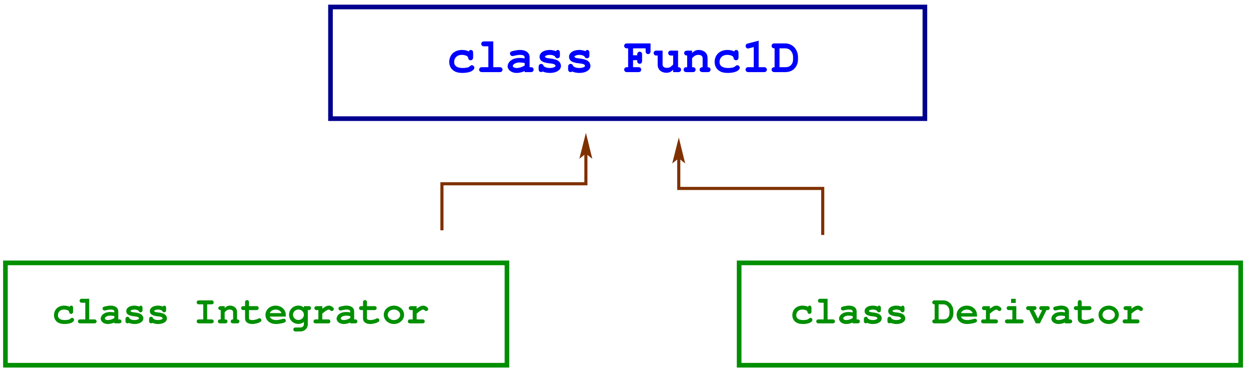
- ✔ Error:

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

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 $[x_{n-2}, x_{n-1}, x_n]$

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

C++ classes




```
class Func1D {
public:
 Func1D(TF1 *fp=NULL);
 // other constructors?
 ~Func1D();
 void Draw();
 double Evaluate();
protected:
 TF1 *p;
};
```

```
class Integrator: public Func1D {
public:
 Integrator(double fx0, double fx1, TF1 *fp=NULL) :
           x0(fx0), x1(fx1), Func1D(fp) {;}
 ~Integrator();
 void TrapezoidalRule(int n, double& Integral, double& Error);
 void SimpsonRule(int n, double& Integral, double& Error);
protected:
 double x0;
 double x1;
};
```