

Numerical Methods

Davide Marchesi

March 21, 2022

Contents

0	Introduction	3
1	Finite Arithmetic Fundamentals	4
2	Non Linear Equations	7
2.1	Bisection Method	7
2.2	Newton's Method	9
2.2.1	Extension of the Newton's Method to Non-Linear Equations Systems	10
3	Numerical Methods for Linear Systems	12
4	Data Approximation and Interpolation	13
4.1	Lagrange's characteristic Polynomials	15
4.2	Interpolation with Cubic Splines	19
5	Numerical Integration and Differentiation	21
5.1	Numerical Integration	21
5.2	Numerical Differentiation	28

Purpose

This Document will be a theoretical explanation **made by myself** of the numerical methods that could be used to solve complex mathematical problems using a virtual machine.

The purpose is to give, for everyone who has a sufficient knowledge of both mathematics and *MATLAB* language, the means to make numerical analysis of practical physical problems.

All the informations in this document are both learned following the "*Engineering' Numerical and Analitical Methods*" course at *Politecnico di Milano*, and studied in deep by myself.

For all the practical code examples please refer to my personal account on *GitHub*:

<https://github.com/davidemarchesi/NumericalMethods>

0 Introduction

The Numerical analysis is a modern approach to all that Physical/Mathematical problems which don't have an analytic resolution, or at least it results to be too complex to be solved in that way.

Take as example a simply temperature distribution problem in a complex space, the equation to resolve will be:

$$\frac{\partial T}{\partial t} - \mu \Delta T = f + \text{B.C.} + \text{I.C.}$$

This equation in fact doesn't have generally a resolution, and so we could use these numerical analysis to solve it.

More in general when facing a Physical problem, first we have to find a mathematical model to describe it, and this will be characterized by an error of the model e_M , and then to solve this mathematical problem.

Here comes into play the Numerical Analysis method, which has the aim to solve in the best possible way, with the lower possible values of the error given by the analysis itself e_N , these models.

1 Finite Arithmetic Fundamentals

A basical as important concept to understand before starting using the Numerical Methods on calculators is that Mathematics as we know it, doesn't exist in the computer world.

While in a theoretical way is possible for us to use and represent irrational numbers ($\sqrt{2}$ or π , for example), this is not possible for computers.

In fact every virtual language will be characterized by a finite number of relevant digits, and so by a minimum and a maximum number which is representable with them (for ex. in MATLAB they respectively are $x_{min} = 2.23 \cdot 10^{-308}$ and $x_{max} = 1.80 \cdot 10^{308}$).

So in this virtual world we have abandoned the classical set of *Real Numbers* and we are now working with what is called the *Floating Numbers* set:

Theoretical/Physical problems $\rightarrow x \in \mathbb{R}$

Numerical models $\rightarrow x \in \mathbb{F}$

Now lets take a look on the properties and on the representation that the floating numbers have:

Df. Floating Point Rappresentation

$$\begin{aligned} \text{if } y \in (F) \rightarrow y &= (-1)^S \cdot (0.a_1a_2...a_t) \cdot \beta^e \\ &= (-1)^S \cdot (a_1a_2...a_t) \cdot \beta^{e-t} \end{aligned}$$

Where:

$$s = \text{sign} \rightarrow s = 0, 1$$

$$\beta = \text{base} \rightarrow \beta \geq 2$$

$$m = a_1a_2...a_t \rightarrow m = \text{mantissa}$$

and:

$$0 \leq a_1 \leq \beta - i \qquad a_i \neq 0$$

Here an example of this rappresentation:

if $x = 0.01235$, then its floating will be $\text{fl}(x) = (-1)^0(1235) \cdot 10^{-6}$
 $\rightarrow s = 0, \beta = 10, e = -2, t = 4$

So more in general a Floating Space \mathbb{F} it's characterized by:
 β , which is the base on which we are working;
 t , which is the number of relevant digits;
 L , the minimum exponent;
 U , the maximum exponent;

(in *MATLAB* case, $\mathbb{F}_{MATLAB}(2, 53, -1024, 1024)$)

Df. Array Error

The array error is the error given by a trasposion of a number from its real representation to its floating representation. It's defined as:

$$\frac{|x - \text{fl}(x)|}{|x|} \leq \frac{1}{2} \varepsilon_M \rightarrow \varepsilon_M = \beta^{1-t}$$

It's immediate to understand in this way that the lower the approximation is, the lower the error is (as i take more relevant digits in my calculator).

NB. in \mathbb{F} are not valid all the properties of \mathbb{R}
The commutative property is still valid:

$$\begin{aligned}\text{fl}(x + y) &= \text{fl}(y + x) \\ \text{fl}(x \cdot y) &= \text{fl}(y \cdot x)\end{aligned}$$

But are no more valid: **associative** and **distributive** properties (in fact the results changes in base of how the formula is written, this due to the fact that i have different errors).

Now imagine that facing a Physical problem, we have yet built the Mathematical model which describes it, and we need a Numerical Method to have a quantitative solution;

the question is: how can we understand that our Numerical Method is a good one, or a bad one?

To individuate a good numerical method, we have to control that it has the following properties:

1. Convergence

Given a mathematical problem $F(x, d)$, (where x is the analitical solution, and d the set of data), and its approximation $F_h(x_h, d_h)$, if $\lim_{h \rightarrow 0} x_h = x$, it has the property the property of convergence.

2. Consistency

If the limit of the approximate function, for $h \rightarrow 0$, used on analytical data, tends to 0:

$$\lim_{h \rightarrow 0} F_h(x, d) = 0$$

This, is called consistency.

3. Stability

If the data are changed a little, also the results are changed only a little, which means:

$$\begin{aligned} F_h(\overline{x_h}, d_h + \varepsilon) &= 0 \\ F_h(x_h, d_h) &= 0 \\ \text{if } \varepsilon \rightarrow 0 \text{ so } (\overline{x_h} - x_h) &\rightarrow 0 \end{aligned}$$

(**NB.** in the numerical calculus if consistency and stability are verified, so it's convergence.)

In the end of all, after solving our problem in a numerical way, we will have a final solution which will have intrinsically an error e_N given by two different types of sub-errors:

e_T = the truncation error, given from a theoretical point of view (like for example the Taylor approximation);

e_A = the approximation error, given intrinsically by the calculator, for all the motivations that are explained at the beginning of this chapter (acceptable **only** if there is stability).

2 Non Linear Equations

The general solving process for *non linear equations* can be summarized in finding, if it exists, the 'zero' of a generic function $f(x)$.

In order to do this, some generic methods were developed, and i will discuss two of them: the *Bisection Method* and *Newton's Method*.

2.1 Bisection Method

The *Bisection Method* exploits the **zeros theorem**, so the algorithm will be build starting by simply it:

$$\text{"If } f \in \mathbb{C}^0([a, b]) \text{ and } f(a) \cdot f(b) < 0 \text{ so } \exists \alpha \text{ s.t. } f(\alpha) = 0\text{"}$$

So if this hypothesis are verified, an algorithm can be build in this way:

0.I set a desired tolerance for the resolution: $\text{tol} < \epsilon$ (or either a maximum number of iterations M which I am disposed to do);

1. I take a value $x^{(k)}$ in the interval $[a, b]$;

\rightarrow is $f(x^{(k)}) < \text{tol}$?

2.**yes** If the answer is **yes**, the resolution stops here, and $\alpha = x^{(k)}$;

2.**no** If the answer is **no**, I'll take a value $x^{(k+1)} \in [a, x^{(k)}]$, or either $x^{(k+1)} \in [x^{(k)}, b]$, and restart the algorithm from the point 1 until I don't have a value that fits my initial conditions.

Watching the algorithm, how the intervals in point 2 are selected?

Simply applying the zeros theorem: the value of $x^{(k)}$ is substituted to b for example, and then if the expression $f(a) \cdot f(x^{(k)}) < 0$ is verified, the zero will be into the interval $[a, x^{(k)}]$, otherwise it will be into the other one $[x^{(k)}, b]$.

Moreover, as from the name itself of the method suggest, to find every $x^{(k)}$ will be taken using the *bisection formula*:

$$x^{(k)} = \frac{a^{(k)} + b^{(k)}}{2}$$

(where $a^{(k)}$ and $b^{(k)}$ surely represent the extremes of the interval in the k-esim interaction)

At the end of all I'll have: $\lim_{k \rightarrow \infty} x^{(k)} = \alpha$

Weaknesses and Highlights of the model

One of the Weaknesses of the problem is surely that it individuates only a zero, the one included in the interval set, and not all the zeros of a function.

Then there is also the fact that the method converges to a solution only with zeros with *odd* multiplicity, and not to the one with *even* multiplicity, where the *zeros theorem* is not applicable ($f(a) \cdot f(b) > 0$).

Last weakness is the fact that the convergence is not monotonous. Observe that the distance between the solution and the $x^{(k)}$ value is expressible by: $e^{(k)} = |x^{(k)} - \alpha|$.

So we do not have any certainty that $e^{(k+1)} < e^{(k)}$.

To make an example of this, let's take a function with $\alpha = 4$, if the starting interval taken is $[0, 10]$, after one iteration we will have $x^{(0)} = 5$ and $e^{(0)} = 1$;

But then with a second iteration we will have the values $x^{(1)} = 2.5$ and $e^{(1)} = 1.5$.

Surely at the end of all the method will be convergent, but this will lead us to do a not negligible higher value of iterations if compared with the next method that will be presented.

On the other hand we have a big highlight in these method: without having informations on the function, i know before starting the method the error i will committ at the k-esim iteration, so having set a tolerance i know already the number of iterations necessary. That's why:

$$I^{(k)} = [a^{(k)}, b^{(k)}]; \text{ with } x^{(k)} = \frac{a^{(k)} + b^{(k)}}{2}$$

$$|x^{(k)} - \alpha| < \frac{1}{2} |I^{(k)}| = \frac{1}{2} \frac{1}{2} |I^{(k+1)}| = \frac{1}{2^{k+1}} |I^{(0)}| = \frac{1}{2^{k+1}} (b - a)$$

A priori i can set the tolerance s.t. : $\frac{(b-a)}{\text{tol}} < 2^{k+1}$

And the related number of iterations will be: $k > \log_2(\frac{b-a}{\text{tol}}) - 1$.

NB. The *Bisection Method* can also be used to find the intersection between two functions (if it exists) creating a function that comprehends both, and putting it equals to zero. For example:

$$\text{given } f(x) \text{ and } g(x) \rightarrow F(x) = f(x) + g(x) = 0$$

2.2 Newton's Method

This particular method needs, if compared with the *Bisection Method*, needs more informations about the function we want to analyze. It is based on the *Taylor's approximation theorem*, and the algorithm developed will be:

$$f(x^{(k+1)}) - f(x^{(k)}) = f'(x^{(k)})(x^{(k+1)} - x^{(k)})$$

So at the end of the algorithm, we will have that $f(x^{(k+1)}) = 0$, and following to this:

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

So basically what we are doing is simply moving to the α point using the intercepts of the derivatives of the function step by step (please for a practical example refer to my Github page cited at the beginning of the paper).

It's interesting to watch closer how this method converges to the numerical solution:

Convergence Theorem

If $f \in \mathbb{C}^\infty([a, b])$ s.t $f(\alpha) = 0$ with $\alpha \in [a, b]$, and that $f'(\alpha) \neq 0$, we have that:

$$1. \forall k \geq 1 \quad |x^{(k)} - \alpha| < \eta;$$

$$2. \text{ (convergence) } \lim_{k \rightarrow \infty} x^{(k)} = \alpha;$$

$$3. \lim_{k \rightarrow \infty} \frac{x^{(k+1)} - \alpha}{(x^{(k)} - \alpha)^2} = \frac{f''(\alpha)}{(f'(\alpha))^2}$$

This last condition leads to the *Quadratic Convergence Property*:

$$\frac{e^{(k+1)}}{(e^{(k)})^2} = c$$

(Otherwise if this last third condition is not verified, but only the two above, we have a simple convergence).

Lemma: Newton's method converges also for zeros with multiplicity > 1 , but in this case only in a linear way (we cannot have the quadratic convergence); To regain the quadratic convergence the *modified Newton's method* can be used, which algorithm will be:

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

Now let's watch to the **stop criteria**:

The first criteria could be one based on the *increment*, which watches to the x in the k -esim interaction

$$|x^{(k+1)} - x^{(k)}| < \text{toll}$$

And this one become problematic if applied to functions which are particularly pending.

Another one is a criteria based on the *residual* of the function

$$|f(x^{(k+1)})| < \text{toll}$$

And this one, on the contrary with the other, has problems with flat functions. It's clear that to adjust this problem, both criteria could be used.

2.2.1 Extension of the Newton's Method to Non-Linear Equations Systems

Consider a system of N equations with N unknowns:

$$f_1(x_1, x_2, \dots, x_N) = 0;$$

$$f_2(x_1, x_2, \dots, x_N) = 0;$$

$$f_3(x_1, x_2, \dots, x_N) = 0;$$

The Newton method can find the zero to which this system converges (starting in a defined area).

(**N.b.** the method doesn't give us all the solutions of the system, but only the one which is strictly connected with the area in which we are starting the analysis).

The system will be written in this way:

$$\vec{f} = \begin{pmatrix} f_1 \\ f_2 \\ \dots \\ f_N \end{pmatrix}; \quad \vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_N \end{pmatrix};$$

So the system can be written like:

$$\vec{f}(\vec{x}) = 0$$

Implementing the algorithm it's clear that we cannot refer to a simple derivative for the system, we have to use the **Jacobian** of $\vec{f}(\vec{x})$:

$$J_f(\vec{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_N} \\ \frac{\partial f_2}{\partial x_1} & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \frac{\partial f_N}{\partial x_1} & \dots & \dots & \frac{\partial f_N}{\partial x_N} \end{bmatrix}$$

So the algorithm becomes:

$$\begin{aligned} & \text{given } \vec{x}^{(0)}, \text{ for } k = 0, 1, \dots \\ \delta \vec{x} &= -J_f^{-1}(\vec{x}^{(k)}) \vec{f}(\vec{x}^{(k)}) \\ \vec{x}^{(k+1)} &= \vec{x}^{(k)} + \delta \vec{x} \end{aligned}$$

In the practice due to the fact that the calculus of the inverse matrix is very heavy from the computational point of view, i can transform the algorithm in order that it will be a linear system that has to be resolved at every step:

$$J_f(\vec{x}^{(k)}) \delta \vec{x} = -\vec{f}(\vec{x}^{(k)})$$

observe that for this type of problem, the convergence conditions are the same, but in a vector definition:

- The function must be of \mathbb{C}^2 class;
- The Jacobian must be Invertible (that is the request related to the fact that the derivate must be $\neq 0$);
- I must be in the neighbourhood of the solution.

A last observation that is possible to do in this paragraph is about how to use both methods together;

In fact is possible to implement a code in order to use in a first moment the *bisection method* in order to restrict, with a pair of iterations, the interval in which we could have the solution, and then to use the *Newton's method* to have a quickly convergence to it.

3 Numerical Methods for Linear Systems

4 Data Approximation and Interpolation

In this chapter will be discussed the basic theories behind the data approximation and interpolation.

We will consider the study of the simplest set of data, the **monovariate** ones, which actually are data where the change of a quantity is strictly related to only one unknown (certainly this type of analysis is only a superficial one, but all the observations made could be extended, with the proper adjustments, to more general theories).

With the word **Interpolation** we are actually meaning to find a function that can describe correctly the data, tracking a 'pattern' which actually fits with them.

This in the reality could be a very powerful instrument, that can be used for example to find links between two variables starting with a set of experimental data, or for example to extract intermediate or subsequent data which were not directly measured in a practical way.

A lot of times the interpolation process can lead us to very big variations from the best pattern that fits the data due to his possible *instability*, so in this cases may be better to adopt an **Approximation** of them rather than studying a function which has to pass precisely in the set variables.

(So we can say that, given a set of data [basically points, or values], the difference between *interpolation* and *approximation* is that whether the first finds a pattern, a function, that passes perfectly in the given values, the second one wants only to follow the general trend of them).

Of course all this theory is not only limited to data, but also to functions: we may want to find a function which interpolate, or approximate, a general $f(x)$ function in a certain set of points.

The motivation to do this is simply to use a simpler equation to do much more things in an easy way in comparison to the starting general function (for example using polynomial is much more simpler to do integrations, or differentiation...). Here will be discussed mainly the '*Polynomial functions which interpolate or approximate arbitrary equations $f(x)$* ', others more complex mathematical instruments in this field (like the '*Fast Fourier Transformed*') will be only cited.

Given $(n + 1)$ points, that is, (x_k, y_k) for $k = 0, \dots, n$ distinct between them, does a polynomial which interpolates them exist?

i.e. : $\exists \Pi_n(x_k) = y_k$? (where Π stays for polynomial)

The answer is yes, if $\Pi_n \in \mathbb{P}^n$, where \mathbb{P}^n stays for the set of polynomials with n as maximum grade.

To better understand this lets make an example:

Consider $n + 1 = 2$, so basically we are considering only two points, then we can say that:

$$\begin{aligned}\exists |\Pi_1 \in \mathbb{P}^1; \\ \nexists |\Pi_0 \in \mathbb{P}^0; \\ \nexists |\Pi_2 \in \mathbb{P}^2;\end{aligned}$$

Imagine two points in the space: in the first equation we are interpolating them with a straight line, which surely exists, and which is one and only; in the second case we are trying to interpolate two distinct points with only a point, so a polynomial of this grade simply cannot do this, it doesn't exist; in the third case, we are trying to interpolate the two points with a parabola, which surely will be possible, but it won't be unique as we could define more than one to pass in both of them; and so on...

So now is much more clear the precedent condition of why the polynomial with which we are interpolating the $n + 1$ points has to be of grade n .

The needed polynomial will have to be s.t. :

$$\Pi_n(x) = \begin{cases} a_0 + a_1x_0 + \dots + a_nx_0^n = y_0 \\ a_0 + a_1x_1 + \dots + a_nx_1^n = y_1 \\ \dots \\ a_0 + a_1x_n + \dots + a_nx_n^n = y_n \end{cases}$$

That are $(n + 1)$ unknowns (the a_i) in $(n + 1)$ equations.

Which can be written as the following system:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \dots & & & & \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \dots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \dots \\ y_n \end{bmatrix}$$

At the end of all, if we are considering distinct points the system will be *invertible* so we will have that $\exists a_k$ for $k = 0, \dots, n$. On the other hand this type of approach, even if possible and correct, it's too much heavy for the computational point of view, so we need to follow a different path.

4.1 Lagrange's characteristic Polynomials

In this part will be discussed the existence and uniqueness of a very particular and interesting set of polynomials for the Interpolation process. To do this we will have 2 consecutive steps: the first, where it will be demonstrate by 'construction' the existence of them, and then in the second step their uniqueness.

Existence Theorem

Given $n + 1$ points (x_k, y_k) with $k = 0, \dots, n$ and $x_k \neq x_j$, if $k \neq j$, then : \exists polynomial $\Pi_n(x) \in \mathbb{P}^n$, s.t. $\Pi_n(x_k) = y_k$ for $k = 0, \dots, n$.

In general we have seen that $\Pi_n(x) = a_0 + a_1x + \dots + a_nx^n$, as $\Pi_n = \text{Span}\{1, x, x^2, \dots, x^n\}$ (That basically means that every element $\Pi_n(x)$ is a linear combination of the Base above).

BUT this is not the only base possible for the n -th polynomial space.

I can actually build a base with different functions so defined:

$$\mathbb{L}_i(x) = \prod_{j=0(\neq i)}^n \frac{x - x_j}{x_i - x_j} \quad \text{with } i = 0, \dots, n$$

These are called *Lagrange's charactesistic Polynomials* associated to the i -th node, and they have the following properties:

They are $n + 1$ polynomials of grade n , and they they assume the following values

$$\mathbb{L}_i(x_j) = \begin{pmatrix} 1 & \text{if } j = i \\ 0 & \text{if } j \neq i \end{pmatrix}$$

See some examples:

$\mathbb{L}_i(x_i)$ with $x_i = x_j$ will be the product of $\frac{x_i - x_j}{x_i - x_j}$, that will give 1;

$\mathbb{L}_i(x_i)$ with $x_i \neq x_j$ will be the product of $\frac{x_j - x_j}{x_i - x_j}$, that will give 0;

So the functions are 1 only in one of the nodes, and 0 in the remaining.

This fact makes for us possible to write the interpolating with the various $\mathbb{L}_i(x)$ multiplied for the appropriate coefficients:

$$\Pi_n(x) = \sum_{i=0}^n y_i \mathbb{L}_i(x) \quad \rightarrow \quad \textbf{Lagrange's Interpolating}$$

obs.: this is given by the fact that they are linearly independent functions of a Space, so they constitute a Base of it (from the df. of Base, 'linearly independent elements with the dimension of the Space itself').

From now we have clearly understood that a polynomial is representable in this way:

$$\Pi_n(x) = y_0 \mathbb{L}_0(x) + y_1 \mathbb{L}_1(x) + \dots + y_n \mathbb{L}_n(x)$$

Why can we say that this type of base selection is better than the 'classical' one?

Firstly because in the previous case we had to assembly a linear system and to solve it to find the coefficients;

Then also because the calculation of the coefficients y_i is immediate due to the fact that they are equal to the ordinate of the selected points.

Therefore we have built the interpolating and demonstrated its existence to interpolate the nodes.

Uniqueness Theorem

Here we will do a reductio ad absurdum. The hypothesis is that exist two interpolating of grade n s.t. :

$$\begin{aligned} \exists \Pi_n^1(x_j) &= y_j \\ \exists \Pi_n^2(x_j) &= y_j \end{aligned} \quad \text{s.t.} \quad \Pi_n^1 \neq \Pi_n^2$$

Let be $\tilde{\Pi}_n(x) = \Pi_n^1(x) - \Pi_n^2(x)$, $\tilde{\Pi}_n(x) \in \mathbb{P}^n$

So if I evaluate the function in the x_j points, i will have:

$$\tilde{\Pi}_n(x_j) = \Pi_n^1(x_j) - \Pi_n^2(x_j) = y_j - y_j = 0$$

And from the algebra, a Polynomial of grade n which nullifies itself in $n + 1$ points is identically null, BUT this is a contradiction of the starting hypothesis, because the polynomials must be different, but their difference is null, and this, is impossible.

Making a resume, we can interpolate $n + 1$ points using a polynomial of grade n , which we can find in an easier way using as a vectorial base the Lagrange's Polynomials, of which we have demonstrated the existence and uniqueness, and that assume value of 1 in only one point (at the end, is like if this vectorial base is correlated with an identical matrix).

The advantages of using these type of polynomials is clear, but how good does this base work out of the nodes?

Lagrange Interpolation Error: given an interval I and $n + 1$ nodes of interpolating $x_i \in I$ with $i = 0, \dots, n$, if $f: i \rightarrow \mathbb{R}$ is of class $\mathbb{C}^{(n+1)}(I)$, then:

$$\begin{aligned} \forall x \in I, \quad \xi \text{ s.t.} \quad E^n f &= f(x) - \Pi_n f(x) \\ \text{so} \quad E^n f &= \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^n (x - x_i) \end{aligned}$$

So if the error is too much big i can add points, but is not for sure that the error will go down, in fact we don't know the limit: $\lim_{n \rightarrow \inf} E^n f(x) = ?$

In addition to this, given n equally spaced nodes ($x_{i+1} - x_i = h$, with h constant, $\forall i$), it's demonstrable that $\exists f \in \mathbb{C}^{(n+1)}$ s.t. :

$$\lim_{n \rightarrow \inf} \max_{x \in I} |E^n f(x)| = \inf$$

The method is not stable every time, we have to select the nodes in a different way, not equally spaced; one method could be the one given by *Chebyshev-Gauss-Lobatto* (abbr.: CGL).

CGL nodes:

Set an reference interval $\hat{I} = [-1, 1]$, then the points are taken with $\hat{x}_i = -\cos \frac{i}{n}\pi$ (the process basically consists in taking a semi-circumference in the interval, then dividing it into n spaces with an equal angle , and then projecting into the abscissa).

With this production the points will no longer be equally spaced, then we transform them in the desired interval $[a, b]$ in this way:

$$x_i = \left(\frac{a+b}{2}\right) + \left(\frac{b-a}{2}\right)\hat{x}_i$$

(obs.: $x_0 = a, \dots, x_n = b$)

For all the *CGL nodes* can be demonstrated that:

$$\lim_{n \rightarrow \infty} \max_{x \in I} |E^n f(x)| = 0 \quad \forall f \in \mathbb{C}^{(n+1)}(I)$$

With this points selection, once the interpolating function is selected, in a set interval, i have a good interpolation.

The fact that the selection of the points in the interpolation process is so important can be seen clearly watching to the **Lagrangian' stability**:

$$\begin{aligned} f(x) &\rightarrow \Pi^n f(x) \\ \tilde{f}(x) &\rightarrow \Pi^n \tilde{f}(x) \end{aligned}$$

Where \tilde{f} is the perturbed function, and $\varepsilon(x) = f(x) - \tilde{f}(x)$ the perturbation, then

$$\begin{aligned} \|\Pi^n f(x) - \Pi^n \tilde{f}(x)\| &= \max_{x \in I} |\Pi^n f(x) - \Pi^n \tilde{f}(x)| \\ &= \max_{x \in I} \left| \sum_{i=0}^n f(x_i) \mathbb{L}_i(x) - \sum_{i=0}^n \tilde{f}(x_i) \mathbb{L}_i(x) \right| \\ &= \max_{x \in I} \left| \sum_{i=0}^n (f(x_i) - \tilde{f}(x_i)) \mathbb{L}_i(x) \right| \\ &\leq \max_{i=0, \dots, n} |f(x_i) - \tilde{f}(x_i)| \max_{x \in I} \left| \sum_{i=0}^n \mathbb{L}_i(x) \right| \end{aligned}$$

As we can see from the second factor of the last equation line, the stability of the interpolation, a priori depends on the selection of the nodes we do at the beginning, this due to the fact that $\mathbb{L}_i(x)$ value, depends by the nodes themselves, and on the other hand is independent from the selection of the interpolating function.

This second factor is called **Lebesgue's constant**:

$$\Lambda_n = \max_{x \in I} \left| \sum_{i=0}^n \mathbb{L}_i(x) \right| \approx \frac{2^{n+1}}{e n \lg(n+\gamma)} \text{ with } \gamma \approx 0,5.$$

Λ_n express us the fact that the error is controlled with the perturbation, and the constant itself, so we can have a very little perturbation, but if the *Lebesgue's constant* is not small, and so the point selection was not made with accuracy, we will have a not negligible error.

ex.:

- 20 equally spaced nodes: $\Lambda_2 0^E \approx 20000$;
- 20 CGL nodes: $\Lambda_2 0^{CGL} \approx 3$;

Now lets do a practical example; normally, when we are doing measurement, data are not taken with a CGL disposition, they usually are taken with a uniform progress (for obvious measurement and instrument constrictions). The question now is: what can I do with this type of data?

In this situation (equally spaced nodes) is observable that as long as the interpolating polynomials i am using, have a low grade, i can have good results; but, after a certain number of k , the grade of the polynomials becomes too big and oscillation start appearing, that then, increasing more and more the grade, will explode.

In this case we could decide to interpolate the function looking for polynomial low grade approximation, but in more subintervals of the data.

df.: $\Pi_1^{H(x)}$ is called *Linear Composit Interpolating*.

Given $n + 1$ points x_i equally spaced in the interval $[a, b]$, therefore:

$$\Pi_1^H(x) = f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} (x - x_i), \text{ with } x \in [x_i, x_{i+1}]$$

Expressing the *convergence* :

$$\max_{x \in I} |E^H(x)| = \max_{x \in I} |f(x) - \Pi_1^H f(x)| \leq \frac{H^2}{8} \max_{x \in I} |f'(x)|$$

(valid for $f \in \mathbb{C}^2([a, b])$, typically if $f \notin \mathbb{C}^2$, but is anyway valid $f \in \mathbb{C}^1$, there is convergence, but not quadratic).

What is said for the linear interpolating could be extended for whichever grade (remember that in this case is not the grade, but the number of subintervals that we are considering that makes the mainly difference).

In a more general way, can be written:

$$\max_{x \in I} |f(x) - \Pi_p^H(x)| \leq C H^{p+1} \max_{x \in I} |f^{(p+1)}(x)|$$

Note that the higher the polynomial grade is, the stricter the conditions on the function are ($f \in \mathbb{C}^{p+1}$).

The problems related with this type of approach are given by the fact that I can do the interpolation only locally, but globally i will have problems on the continuity of the function; imagine that for every interval there are straight lines, or parabolas (for example): every time, with all probability, I will have discontinuity problems in the common points between an interval and the successive one.

This is why was implemented the following interpolation approach.

4.2 Interpolation with Cubic Splines

With this method i want to use the '*Intervals approach*' explained above, but trying to maintain the continuity of the function derivatives, we don't want them to operate in an independent way one from each other.

For every subinterval I_j with $j = 1, \dots, n$ there's a polynomial of grade 3

$$P_j(x) = a_0^j + a_1^j x + a_2^j x^2 + a_3^j x^3$$

That corresponds in having $4n$ unknowns.

Then, how said, we want to impose not only the continuity of the cubics in the touching point, but also the continuity of the first and second derivate.

Be $S_j^3(x)$ the cubic which lives in the interval $I_j(x_{j-1}, x_j)$ with $j = 1, \dots, n$, then the condition to impose are:

- **continuity condition of S** : $S_j^3(x_j) = S_{j+1}(x_j)$ $(n - 1)$ conditions;
- **continuity condition of S'** : $(S_j^3(x_j))' = (S_{j+1}(x_j))'$ $(n - 1)$ conditions;
- **continuity condition of S''** : $(S_j^3(x_j))'' = (S_{j+1}(x_j))''$ $(n - 1)$ conditions;
- **interpolation condition (also at the extremes)**: $S_j^3(x_j) = f(x_j) = y_j$ $(n + 1)$ conditions.

The sum of the conditions is $4n - 2$, but the unknowns are $4n$, what are the additional conditions to put? Here some of the possible ones:

- Natural Cubic Splines : $S''(x_0) = 0, \quad S''(x_n) = 0;$
- NOT-A-KNOT Cubic Splines : $(S_1^3(x_1))''' = (S_2^3(x_1))''' , \quad (S_{n-1}^3(x_n))''' = (S_n^3(x_n))'''.$

The esteem of convergence for the *composite interpolation* ($p = 3$) is:

$$\lim_{H \rightarrow 0, n \rightarrow \inf} \max_{x \in I} |f(x) - \Pi_3^H f(x)| \leq CH^4 \max_{x \in I} |f^{(4)}(x)|$$

And for the *splines*:

$$\lim_{H \rightarrow 0, n \rightarrow \inf} \max_{x \in I} |f(x) - S^3(x)| = \tilde{C}H^4 \max_{x \in I} |f^{(4)}(x)|$$

In the particular case of the splines, also the derivatives converge (but in a slower way):

$$\lim_{H \rightarrow 0, n \rightarrow \inf} \max_{x \in I} |f^{(r)}(x) - (S^3)^{(r)}(x)| = \tilde{C} H^{4-r} \max_{x \in I} |f^{(4)}(x)|$$

Mention of Trigonometric Interpolation

Be $f : [a, b] \rightarrow \mathbb{R}$ a periodical function s.t. $f(a) = f(b)$;

Define n points x_i s.t. $x_i = a + \frac{h \cdot i}{n+1}$ with $h = \frac{b-a}{n+1}$, for $i = 0, \dots, n$ (where $a = x_0$, and $b = x_{n+1}$);

Therefore we look for a function $\tilde{f}(x)$ s.t. $\tilde{f}(x_i) = f(x_i) \forall i = 0, \dots, n$.

This function we are looking for, will be a trigonometrical function, given by the combination of sin and cos:

$$\tilde{f}(x) = \frac{a_0}{2} + \sum_{k=1}^{\frac{n}{2}} a_k \cos(kx) + b_k \sin(kx) \quad \begin{array}{ll} \text{For } a_k & k = 0, \dots, \frac{n}{2} \\ \text{For } b_k & k = 1, \dots, \frac{n}{2} \end{array}$$

This, is called **FFT**, which means *Fast Fourier Transformer*, that is a discreet way to write the Fourier's series.

To resume, the aim of this method is to represent periodical signals with trigonometric function linearly combined.

Mention of the Method of the Quadratic Minimum

I could also be, during my data analysis, no more interested in having a function that fits perfectly with the points I obtained, so in this case i would like to do an approximation:

Given $n + 1$ points, i look for the polynomial $\hat{P}(x)$ of grade $m \leq n$ s.t.

$$\sum_{i=0}^n (y_i - \hat{P}(x_i))^2 \leq \sum_{i=0}^n (y_i - P(x_i))^2, \quad \forall p \in \mathbb{P}^m$$

where $\hat{P}(x)$ is the m -th grade polynomial which makes minimum the *mean quadratic error*.

(obs.: for construction of the minimum quadratics, when $m = n$, i have the interpolating polynomials, so if i want to do an approximation and not an interpolation of the points, I'll have to consider $m < n$. Note that on MATLAB exists a function called 'polyfit', that takes as an input the coordinates of the points, and the grade m of the polynomial we want to use to make interpolation, and when $m = n$ it looks for the Interpolating, but when $m < n$ it uses the minimum quadratic error).

5 Numerical Integration and Differentiation

5.1 Numerical Integration

In this part we will examine numerical methods developed in order to be able to estimate the integral of a function, even if this function has the *primitive* undefined, controlling the tolerance, the error and the convergence at the same time.

Will be studied only a restricted, but important part of this theory; the analysis begins watching to the simplest methods to give an esteem of the integral:

Medium Point Method

$$I = \int_a^b f(x)dx \quad I_0 = (b-a)f\left(\frac{a+b}{2}\right)$$

(Which needs the evaluation of the function only in 1 point);

Trapezium Method

$$I_1 = \frac{(b-a)}{2}[f(a) + f(b)]$$

(Which needs the evaluation of the function in 2 points, and can be interpreted also as the integral of the linear interpolating);

Simpson's Method

$$I_2 = \frac{(b-a)}{2}[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)]$$

(Which needs the evaluation of the function in 3 points, and can be also interpreted as the integral of the quadratic interpolating);

This method could be generalized taking an increasing number of n points, interpolating them and calculating the integral of the interpolating function. *BUT* this won't be successfull at long term, as in the previous chapter said, due to the fact that the interpolating function with a great number n of equally-spaced nodes will no longer have a good approximation (due to the generation of instabilities in the border of the function).

It's clear that the method could be improved both taking *non*-equally-spaced nodes or increasing the number of intervals, but how good will the integral be approximated? We can quantify the difference as follows (Firstly we will consider the *Medium Point Method*):

$$\text{Given } f \in \mathbb{C}([a, b]), \quad I = \int_a^b f(x)dx, \quad I_0 = (b-a)f(x_m) \quad \text{with } x_m = \frac{a+b}{2}$$

$$\text{Then } I - I_0 = \int_a^b f(x)dx - \int_a^b f(x_m)dx = \int_a^b (f(x) - f(x_m))dx$$

And from the Taylor approximation we know that:

$$f(x) = f(x_m) + (x - x_m)f'(x_m) + \frac{(x - x_m)^2}{2}f''(\xi(x)) \quad \text{s.t. } \xi(x) \in [x, x_m]$$

Then we can write that:

$$I - I_0 = \int_a^b (x - x_m)f'(x_m)dx + \int_a^b \frac{(x - x_m)^2}{2}f''(\xi(x))dx$$

Where the first integral is equal to zero (in fact it's the integral of the straight line passing in the medium point in the interval $[a, b]$), and the second can be valued with the *Medium Value Theorem*:

$$\int_a^b f(x)g(x)dx = f(\eta) \int_a^b g(x)dx, \quad \exists \eta \in [a, b] \text{ if } g(x) > 0$$

So at the end the error will be dependent on the amplitude of the intervals:

$$E_0 = I - I_0 = \frac{(b - a)^3}{24}f''(\eta)$$

(as: $\int_a^b \frac{(x - x_m)^2}{2}f''(\xi(x))dx = \frac{f''(\eta)}{2} \int_a^b (x - x_m)^2dx = \frac{(b - a)^3}{24}f''(\eta)$)

Following the same model used for the 'Medium Point Method' above, the other methods error values, will be:

$$E_1 = I - I_1 = -\frac{(b - a)^3}{12}f''(\eta) \quad (\text{Trapezium Method})$$

$$E_2 = I - I_2 = -\frac{(b - a)^5}{16 \cdot 180}f^{(IV)}(\eta) \quad (\text{Simpson's Method})$$

Now that we have an idea about the error we commit integrating the functions, lets watch the **Composite Squaring Formulas**:

Given $f: [a, b] \rightarrow \mathbb{R}$, we divide the interval $[a, b]$ in 'n' sub-intervals like follows

$$I_j = [x_{j-1}, x_j] \quad \text{s.t. } x_j - x_{j-1} = H = \frac{b - a}{n}$$

So the *Medium Point Composite Formula* is obtained applying the Medium Point formula in each sub-interval:

$$I_0^C = \sum_{j=1}^N (x_j - x_{j-1})f\left(\frac{x_{j-1} + x_j}{2}\right) = \sum_{j=1}^N Hf\left(\frac{x_{j-1} + x_j}{2}\right)$$

And the total error will be given by the sum of the singular errors in each j -th sub-interval:

$$E_0^C = I - I_0^C = \sum_{j=1}^N E_0^j = \sum_{j=1}^N \frac{H^3}{24}f''(\xi_j)$$

To analyze it better, we can majorize it:

$$\begin{aligned}
\sum_{j=1}^N \frac{H^3}{24} |f''(\xi_T)| &\leq \max_{x \in [a,b]} |f''(x)| \sum_{j=1}^N \frac{H^3}{24} \\
&= \max_{x \in [a,b]} |f''(x)| \frac{H^2}{24} \sum_{j=1}^N \frac{(b-a)}{N} \\
&= CH^2
\end{aligned}$$

This means that we have an *accuracy* level that's quadratic. (N.b. the *accuracy* describe us the way the error changes).

As made until this moment, these considerations can be made also for the remaining errors introduced at the beginning (Simpson's and Trapezium methods). Here a scheme of the composite we obtain at the end:

- **Medium Point Composite Formula**

$$I_0^C = H \sum_{j=1}^N f\left(\frac{x_{j-1} + x_j}{2}\right) \quad \rightarrow \quad |E_0^C| \leq \frac{(b-a)}{24} \max_{x \in [a,b]} |f''(x)| H^2$$

- **Trapezium Composite Formula**

$$\begin{aligned}
I_1^C &= \sum_{j=1}^N \frac{H}{2} [f(x_{j-1}) + f(x_j)] \quad \rightarrow \quad |E_1^C| \leq \frac{(b-a)}{12} \max_{x \in [a,b]} |f''(x)| H^2 \\
(\text{so: } I_1^C &= \frac{H}{2} (f(a) + f(b)) + \sum_{j=1}^{N-1} H f(x_j))
\end{aligned}$$

In '()' an Implementation detail to not calculate at every iteration the same value x_j on two different intervals.

- **Simpson's Composite Formula**

$$\begin{aligned}
I_2^C &= \frac{H}{6} \sum_{j=1}^N [f(x_j) + 4f\left(\frac{x_j + x_{j-1}}{2}\right) + f(x_{j-1})] \quad \rightarrow \quad |E_2^C| \leq CH^4 \\
(\text{where: } C &= \frac{(b-a)}{16 \cdot 180} \max_{x \in [a,b]} |f^{(IV)}(x)| H^4)
\end{aligned}$$

Simpson's method is much quicker than the others, *BUT* it requests (like also the other methods) a function well defined, that in this case means that it must be defined also for the 4-th derivate.

In these methods, one can be better than the other to be used, depending on what we are going to do. For example if we want to calculate the integral of a set of data without making before the interpolation of them, to bypass that, we can use the *Trapezium Method* which takes in consideration only the extremes of the intervals; on the other hand, if we want to calculate the integral (which is finite and defined) of a function which has extremes that tends to infinite, could be useful to use the *Medium Point Method*.

Now we will consider $f \in \mathbb{C}^{\mathbb{P}}([a, b])$, how the different formulas perform if $f \in \mathbb{P}^r$?

To look at it, firstly we have to introduce the definition of '**Accuracy Grade**', which is defined as the maximum polynomial grade s.t. the integration formula is exact, which means that the formula has a grade of accuracy p if : $I(f) - I_n(f) = 0 \quad \forall f \in \mathbb{P}^p$ (or in other words, the grade of accuracy can be defined when the error given by the calculation of the integral is $E_j = 0$).

In this way we can introduce the **Squaring Formulas**, that are exact integrals of the interpolating functions.

Interpolating Squaring Formulas

$$I_n = \sum_{j=0}^n \alpha_j f(x_j) \quad \text{where:} \quad \begin{array}{ll} \alpha_j \rightarrow & \text{'squaring load'}; \\ x_j \rightarrow & \text{'squaring nodes'}; \end{array}$$

Then we have:

- Medium Point: $n = 0, \quad \alpha_0 = b - a, \quad x_0 = \frac{b+a}{2}$;
- Trapezium: $n = 1, \quad \alpha_{0,1} = \frac{b-a}{2}, \quad x_0 = a, \quad x_1 = b$;
- Simpson: $n = 2, \quad \alpha_{0,2} = \frac{b-a}{6}, \alpha_1 = \frac{2}{3}(b-a), \quad x_0 = a, x_1 = \frac{a+b}{2}, x_2 = b$;

All of these are the exact integral of the interpolating of grade n . Lets examine them:

$$\begin{aligned} I_n &= \int_a^b \Pi^n f(x) dx = \int_a^b \sum_{j=0}^N f(x_j) \mathbb{L}_i(x) dx \\ &= \sum_{j=0}^N \left[\int_a^b \mathbb{L}_i(x) dx \right] f(x_j) \\ &= \sum_{j=0}^N \alpha_j f(x_j) \end{aligned}$$

So as we can see the loads α_j are the exact integral of the Lagrange's functions, and due to this is independent from the selection of the $N + 1$ nodes.

Here the example of the calculation of the load α_j for the Trapezium Method:

$$I_1 = [f(a) + f(b)] \cdot \left(\frac{b-a}{2}\right) \rightarrow x_0 = a, x_1 = b \quad \alpha_0 = \alpha_1 = \frac{b-a}{2}$$

Knowing that : $\mathbb{L}_i(x) = \prod_{j=0, j \neq i}^n \frac{x - x_j}{x_i - x_j}$, then :

$$\mathbb{L}_0(x) = \frac{x-b}{a-b}; \quad \mathbb{L}_1(x) = \frac{x-a}{b-a}; \quad \int_a^b \frac{x-b}{a-b} dx = \frac{b-a}{2}.$$

How to select the nodes in order to maximize the accuracy grade? To discover it we have to lay in the following theorem:

' Given $m > 0$, the squaring formula $I_n = \sum_{j=0}^n \alpha_j f(x_j)$ has the accuracy grade equals to $n+m$ if the formula, once that is exploded, it's of interpolating kind (which means $\alpha_j = \int_{-1}^1 \mathbb{L}_i(x) dx$), and the nodal polynomial, defined as $\omega_{n+1}(x) = \prod_{j=0}^n (x - x_j)$, associated to the nodes, is s.t. :

$\int_{-1}^1 \omega_{n+1}(x) p(x) dx = 0 \quad \forall p(x) \in \mathbb{P}_{m-1}$ '. This theorem tells us that if we find a certain distribution of point s.t. the associated nodal polynomial is orthogonal (obs. in fact we are looking, in the theorem, for $\int_I f(x)g(x)dx = 0$) to all the polynomials until the grade $m-1$, the *squaring formula* has an accuracy grade of $n+m$.

(Nb.: the definition's interval is $(-1, 1)$, like for Chebishev, but then it can be expanded).

This type of result can be used with the **Gaussian Squaring Formulas**. To talk about them we have firstly to introduce **Legendre's Polynomials**, which are particular polynomials s.t.:

$$\begin{aligned} L_0(x) &= 1 \\ L_1(x) &= x \\ L_{k+1}(x) &= \frac{2k+1}{k+1} x L_k(x) - \frac{k}{k+1} L_{k-1}(x) \end{aligned}$$

Where, whatever couple is taken in them, the elements are one orthogonal with the other one :

$$\int_{-1}^1 L_k(x) L_j(x) dx = 0 \quad \forall j \neq k$$

(Nb.: from now on we will distinguish the *Lagrange's* polynomials $[\mathbb{L}]$ from the *Legendre's* polynomials L).

It's demonstrable that the maximum grade of accuracy of an interpolating formula of grade n (so with $n+1$ nodes) is $2n+1$; this result is obtained when ω_{n+1} is proportional to the Legendre's polynomials of grade $n+1$.

We will look the demonstration to a more generic theorem: 'If ω_{n+1} is proportional to $\mathbb{L}_{n+1}(x)$ then the accuracy grade is $2n+1$ (or $m = n+1$).' (This hyp. is not strong as the one above due to the fact that we are not considering

that the accuracy grade is the maximum one that can be obtained.).
Here, the demonstration:

$$\int_{-1}^1 \omega_{n+1}(x)p(x)dx = 0 \quad \text{with: } p(x) \in \mathbb{P}_{m-1}, \quad \text{or } p(x) \in \mathbb{P}_n$$

Being : $\omega_{n+1}(x) = \lambda L_{n+1}(x)$ with $\lambda \in \mathbb{R}$

$$\text{And : } p(x) = \sum_{j=0}^n \beta_j L_j(x)$$

$$\text{Then : } \int_{-1}^1 \lambda L_{n+1}(x) \sum_{j=0}^n \beta_j L_j(x) dx = 0$$

Where in this last equation we have the product between two *Legendre's* polynomials in which one is \leq of the other, that means we have demonstrated that ω_{n+1} is proportional to $L_{n+1}(x)$, and that we are capable to find a squaring formula associated to it with an accuracy grade of $2n + 1$.

Gauss-Legendre Squaring Formula

Considering the grade n :

$$x_j \text{'s are } L_{n+1}(x) \text{ zeros;}$$

$$\alpha_j = \frac{2}{(1 + x_j)^2 [L_{n+1}'(x_j)]^2};$$

(Usually these values are printout)

An example of its application could be the following : considering the formula $f(x) = 2x + x^4$, knowing that the **G-L** formula of grade n is exact for the polynomials, with grade $2n + 1$, when we are going to select the nodes to integrate it, we already know that for $n = 2$ is correct, we do not need more points, it's only a waste of 'energy', as how said above, the integral is exact, and not subject to errors (remember that this is true only if we are watching polynomials).

If we want to apply this squaring formula to a generic function, we can do it, but we won't have a perfect integral, we will have an error described by an accuracy grade of :

$$I(t) - I_{G-L}(t) = \frac{2^{2n+3}((n+1)!)^4}{(2n+3)((2n+2)!)^3} f^{(2n+2)}(\xi) \quad \forall f$$

$$\text{So in general: } \lim_{n \rightarrow \infty} I(t) - I_{G-L} = 0$$

Until now we have considered squaring formulas with a constant interval (with n points, the interval is $H = \frac{b-a}{n}$).

Considering for example *Simpson's* or *Simpson's Composite* formula:

$$I(t) - I_S(t) = C(b-a)^5 f^{(5)}(\xi);$$

$$|I(t) - I_S^C(t)| \leq CH^4 \max_{x \in I} |f^{(5)}(\xi)|;$$

I could want for particular oscillating functions (for ex.) non-equal intervals; this due to the fact that taking the intervals constant i could encounter problems like for example *too big* intervals, not able to approximate in a good way the function, or for example *too small* intervals, that would lead to having much computational weight non-necessary.

Looking to the error: $E_S^C \leq \frac{b-a}{16 \cdot 180} H^2 \max_{x \in I} |f^{(5)}(x)| < \text{toll}(10^{-5})$.

Then, due to the fact that the error could be not calculable, or that could be too difficult to calculate it, we can introduce an *Error Estimator* S_H^C , s.t. $S_H^C \approx E_H^C$. With this type of error esteem, an algorithm like the following could be implemented:

$$\text{Initialized } H = \frac{b-a}{n}, \quad \text{if : } S_H < \text{toll} \rightarrow I = I_H;$$

$$\text{else : } H = \frac{H}{2} \rightarrow \int_a^{a+\frac{H}{2}} \text{ and } \int_{a+\frac{H}{2}}^b;$$

In other words this type of algorithm means that every integration step, the function is watched locally, going to refine it where it is oscillating.

How to determine the error estimator?

We know already that:

$$I - I_H = C(b-a)^5 f^{(5)}(\eta); I - I_H = C \frac{(b-a)^5}{16} f^{(5)}(\eta);$$

If we make the hypothesis that:

$$f^{(5)}(\eta) \approx f^{(5)}(\xi) \rightarrow \Delta I = I_H - I_{\frac{H}{2}} = C \frac{15}{16} (b-a)^5 f^{(5)}(\xi)$$

$$\text{Then} \rightarrow f^{(5)}(\xi) = \frac{\Delta I}{C} \frac{16}{15} \frac{1}{(b-a)^5}$$

$$\text{And} \rightarrow E_{\frac{H}{2}} = I - I_{\frac{H}{2}} = C \frac{(b-a)^5}{16} f^{(5)}(\xi) \approx \frac{\Delta I}{15}$$

Which means that an esteem of the error is given by : $E_{\frac{H}{2}} \approx \frac{\Delta I}{15}$.

5.2 Numerical Differentiation