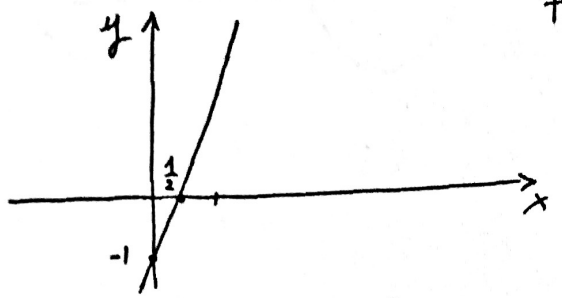
Nonlinear Equations - intro

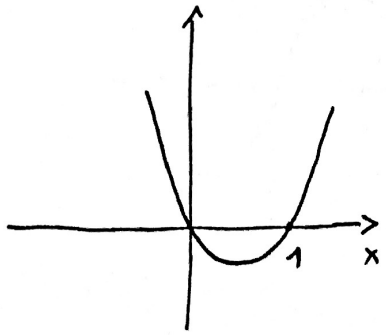
Consider the equation $y = \underbrace{2x - 1}_{f(x)}$



We are interested in finding the zeros of such equations, i.e., the value of x for which $y = 0$. Let us call such x α.

$$2\alpha = 1 \quad \Rightarrow \alpha = \frac{1}{2}$$

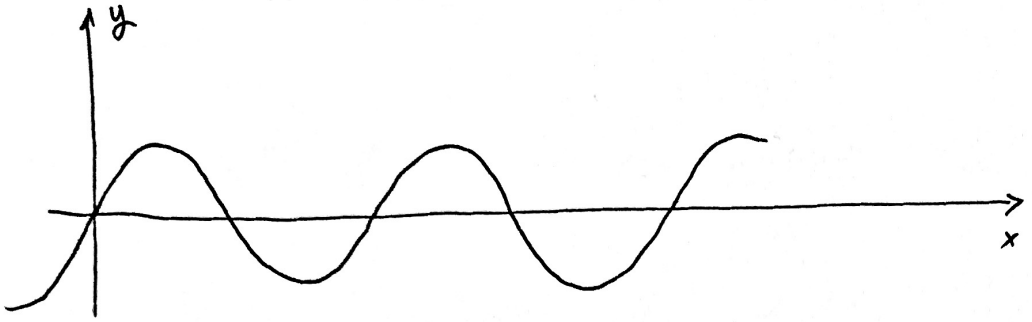
Up to now, no problem. Ok let's make things a bit more interesting



$$y = x(x-1)$$
$$= x^2 - x$$

Same story, right? $\alpha_1 = 0$
$\alpha_2 = 1$

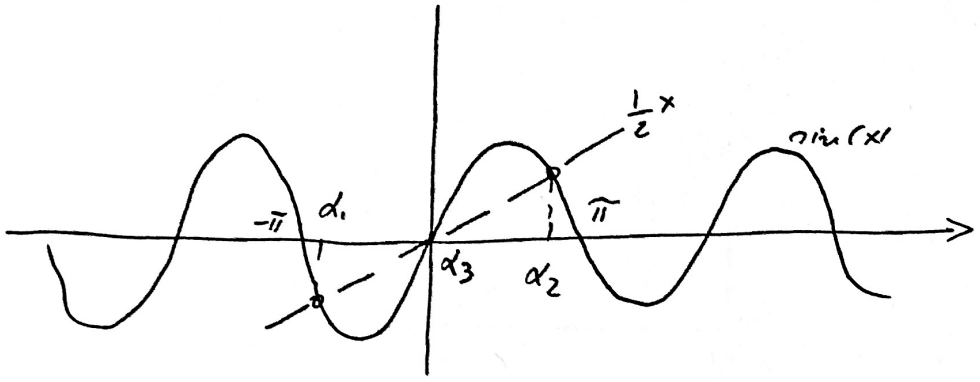
Ok, how about this one?



$y = \underbrace{\sin(x)}_{f(x)}$ $\qquad \sin(\alpha) = 0 \iff \alpha_k = k\pi \quad k \in \mathbb{Z}$

Now consider $y = \overbrace{\sin(x) - \frac{1}{2}x}^{f(x)}$.

Now we are in trouble... Basically the equation $y = 0$ defines the points of intersection of two functions: $\sin(x)$ and $\frac{1}{2}x$.



$\begin{cases} -\alpha_1 = \alpha_2 \simeq 2 \\ \alpha_3 = 0 \end{cases}$

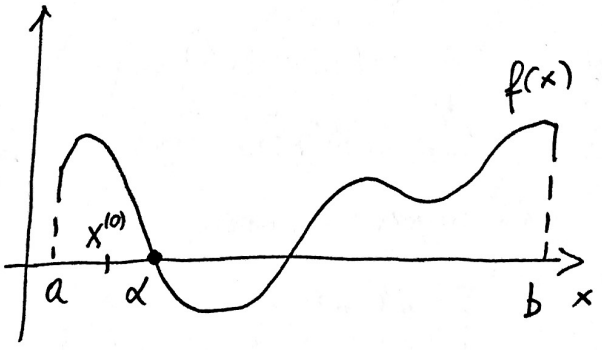
How do we compute α_1? (or α_2)

Nonlinear Equations

We are interested in computing the ZEROS of a real function f, i.e., the ROOTS of the equation

$$f(x) = 0 \qquad x \in [a,b]$$

Methods for the numerical approximation of a zero of f are usually iterative.
Let α be a zero, i.e., $f(\alpha)=0$

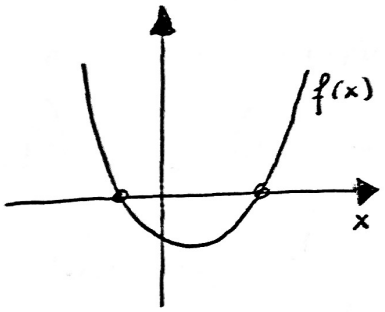


Given an initial guess $x^{(0)}$ (eventually sufficiently close to α), the aim is to generate

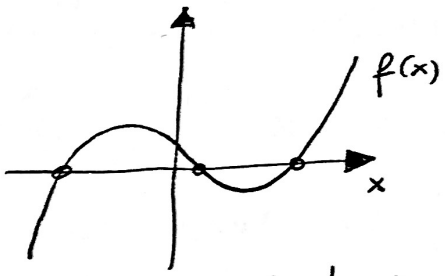
a sequence of numbers $x^{(k)}$ $(k=1,2,\dots)$ that converges to α, i.e.,

$$\lim_{k\to\infty} x^{(k)} = \alpha \qquad \left(\lim_{k\to\infty} |x^{(k)}-\alpha| = 0\right)$$

Remark: (Zeros of polynomial functions): can be computed analytically for polynomials up to order 4 (Abel-Ruffini theorem)



(order 2)
$f(x) = ax^2+bx+c$



cubic (order 3)
$f(x) = ax^3+bx^2+cx+d$

Remark: Roots of polynomials of any order can be computed by determining the eigenvalues of the companion matrix: (numerically)

$$C = \begin{bmatrix} 0 & 0 & \cdots & 0 & -b_0 \\ 1 & \ddots & & \vdots & \vdots \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & \cdots & 1 & -b_{n-1} \end{bmatrix}$$

$f(x) = x^n + b_{n-1}x^{n-1} + \dots + b_1 x + b_0$ (monic polynomial) of degree n

Indeed, the characteristic polynomial of G that defines the eigenvalues coincides with $f(\lambda)$, i.e.,

$$\det(G - \lambda I) = \lambda^n + b_{n-1}\lambda^{n-1} + \dots + b_1\lambda + b_0 = 0$$

$\Rightarrow \lambda$ is a zero of $f(x) \Leftrightarrow \lambda$ is an eigenvalue of G

For functions $f(x)$ that are more general than polynomials we follow an iterative approach, i.e., given $x^{(0)}$ we define an algorithm that generates $x^{(1)}, x^{(2)}, \dots$ hopefully convergent to the root α.

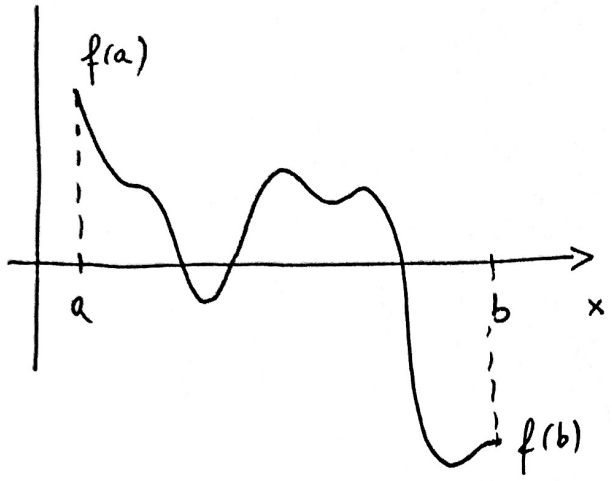
First of all, how can we be sure that such root exists?

<u>Theorem</u>: Let $f(x) \in C^0([a,b])$ with $f(a)f(b) < 0$.
Then it exists $\alpha \in]a,b[$ such that

$$f(\alpha) = 0$$

$\Rightarrow$ continuous functions with the property $f(a)f(b) < 0$ must have AT LEAST one zero in $[a,b]$
This is a sufficient condition (not necessary).

Example:



$f(a)f(b)<0$

Definition A sequence of numbers $x^{(0)}, x^{(1)}, \dots$ is said to converge to α with order $p \geq 1$ if it exists $G > 0$ such that

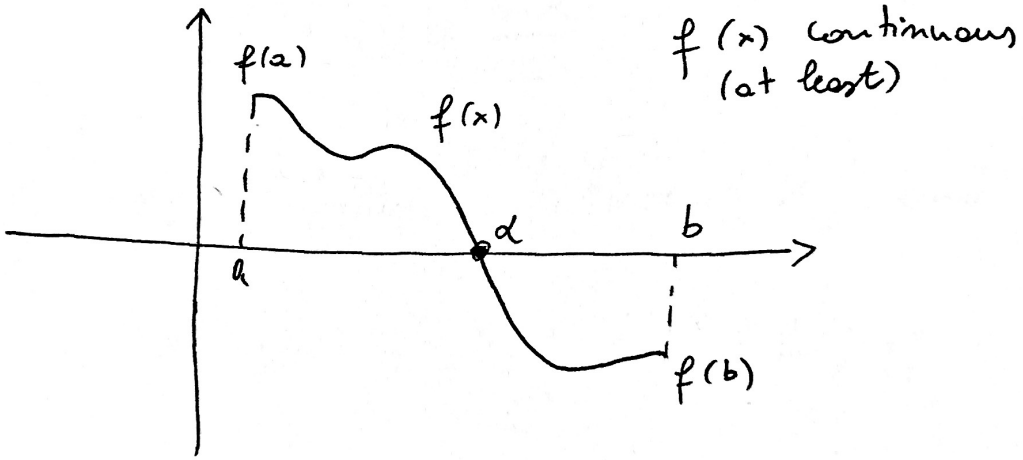
$$\frac{|x^{(k+1)} - \alpha|}{|x^{(k)} - \alpha|^p} \leq G \qquad \forall\, k > k_0$$

If $p = 1$ (first order convergence) we must have $G < 1$ (otherwise the sequence does not converge). In this case ($p=1$), G is called convergence factor.

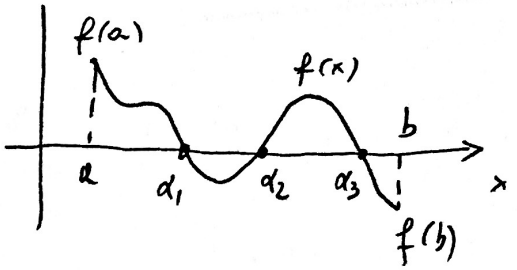
Geometric approaches to root-finding

Bisection method

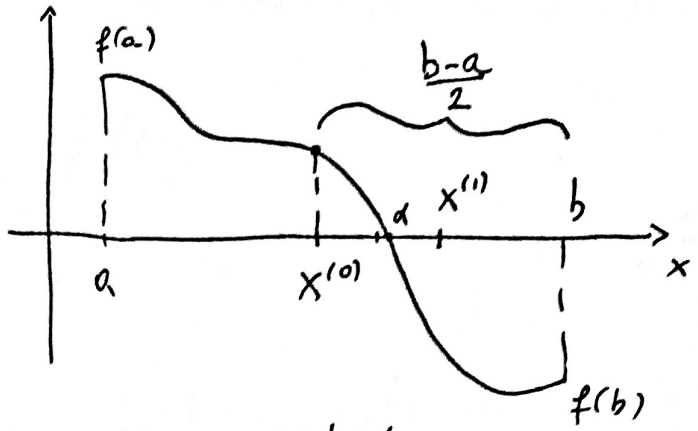
Now we start talking about algorithms to determine zeros of $f(x)$. To this end, suppose we provide two numbers a, b such that $f(a)f(b) < 0$



Remark: It is possible that we have more zeros in $[a,b]$



Consider the point $x^{(0)} = \frac{a+b}{2}$



Clearly, the distance between $x^{(0)}$ and α is at most $\frac{b-a}{2}$.

$$|x^{(0)} - \alpha| \leq \frac{b-a}{2}$$

Similarly the distance between $x^{(1)}$ and α is

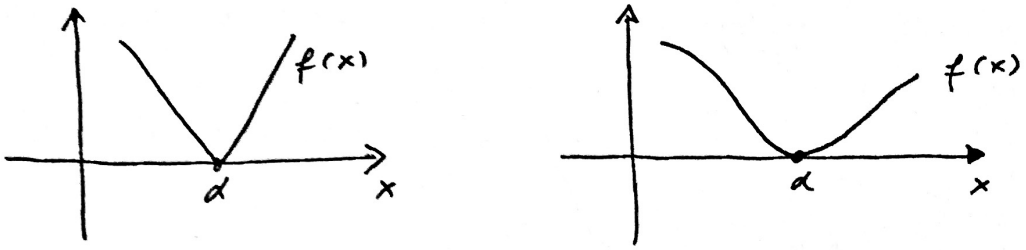
$$|x^{(1)} - \alpha| \leq \frac{b-a}{2^2} \quad \left(\text{half of } \frac{b-a}{2}\right)$$

In general: $$\boxed{|x^{(k)} - \alpha| \leq \frac{(b-a)}{2^{k+1}}}$$

Therefore $\lim_{k\to\infty} |x^{(k)} - \alpha| = 0$ (i.e. $x^{(k)} \to \alpha$),

which means that the sequence generated by the bisection method is convergent.

mark There are few unfortunate cases when the bisection method does not work:



We cannot compute such zeros with the bisection methods.

Remark . (Stopping criterion)

$$\frac{(b-a)}{2^{K+1}} \leq \varepsilon \qquad (\varepsilon \text{ set by the user})$$

In this way we guarantee that $x^{(K)}$ is close to α to order ε, i.e., $|x^{(K)} - \alpha| \leq \varepsilon$

The minimum number of iterations to achieve such tolerance ε is

$$K_{min} \geq \log_2 \left[\frac{(b-a)}{\varepsilon} \right] - 1$$

Remark ("speed" of convergence) How many iterations do we need to "gain" a decimal digit (i.e. to make the error 10 times smaller)?

$$|x^{(k)} - \alpha| \simeq \frac{b-a}{2^{k+1}} \qquad |x^{(j)} - \alpha| \simeq \frac{b-a}{2^{j+1}}$$

Suppose that $|x^{(k)} - \alpha| \cong \frac{|x^{(j)} - \alpha|}{10}$ $(k > j)$

This implies:

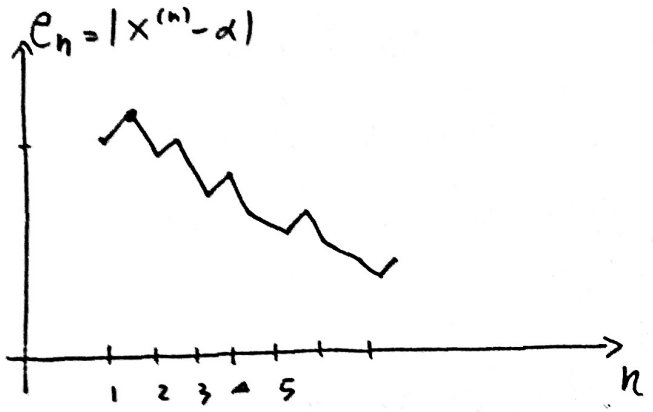
$$\frac{(b-a)}{2^{k+1}} \simeq \frac{(b-a)}{10 \cdot 2^{j+1}} \Rightarrow 2^{k-j} = 10$$

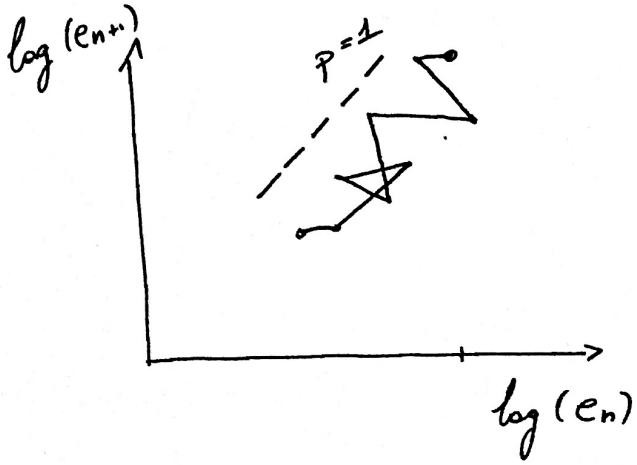
$$\Rightarrow (k-j) = \log_2(10) \simeq 3.3$$

Therefore, on average, every 3.3 iterations we gain a decimal digit in the estimate of the ROOT, i.e., the error becomes 10 times smaller

Remark (Convergence of the bisection method)

We do not have a monotone reduction of the error $\Rightarrow$ we cannot define rigorously a convergence order for the bisection method



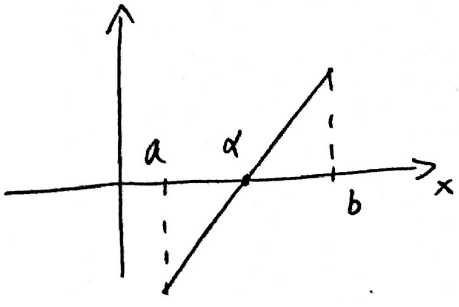


$$\frac{|x^{(n+1)} - \alpha|}{|x^{(n)} - \alpha|^p} \leq C$$

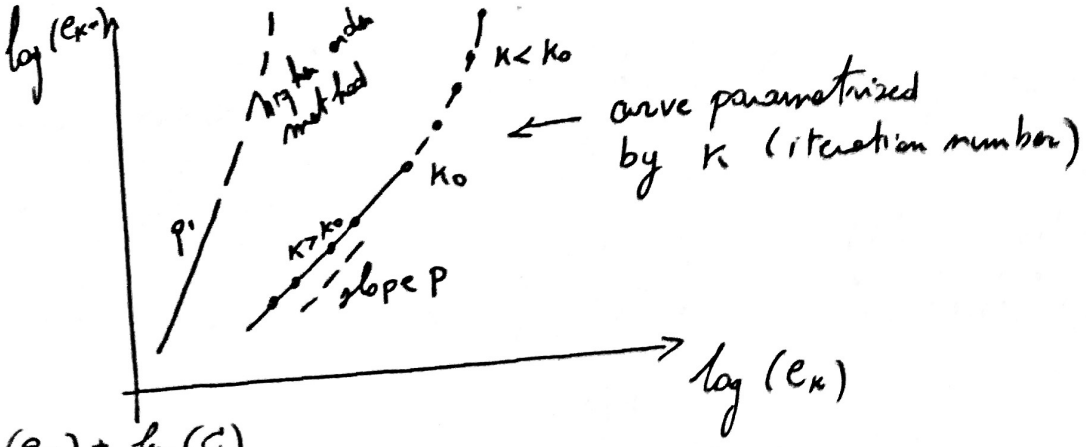
p ↘ convergence order!

on average we get $p = 1$

Remark: How many iterations do we need to compute this zero with the bisection method?



Remark (convergence rate of higher-order methods)



$$\frac{e_{k+1}}{e_k^p} \simeq G$$

$$\Rightarrow \ln(e_{k+1}) = p\ln(e_k) + \ln(G)$$

$\Rightarrow$ higher order methods converge faster, i.e., in less iterations.

Remark: To construct higher-order methods we need to include information from $f(x)$ at more points, and possibly derivative information (if f is differentable).

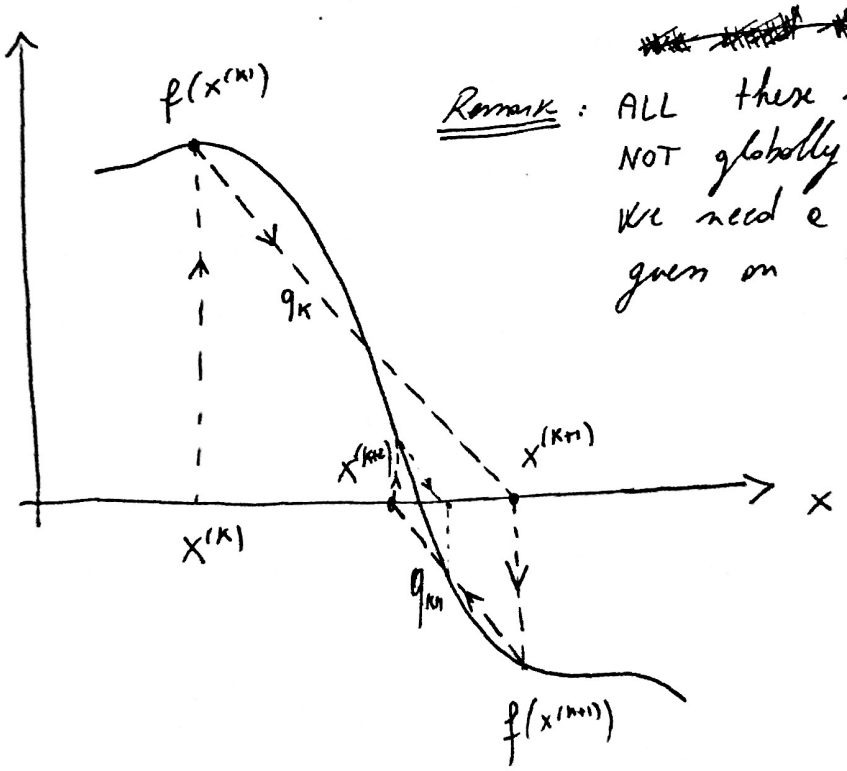
Methods of chord, secant and Newton

All these methods generate sequences in the form:

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

(this is actually a DISCRETE DYNAMICAL SYSTEM)

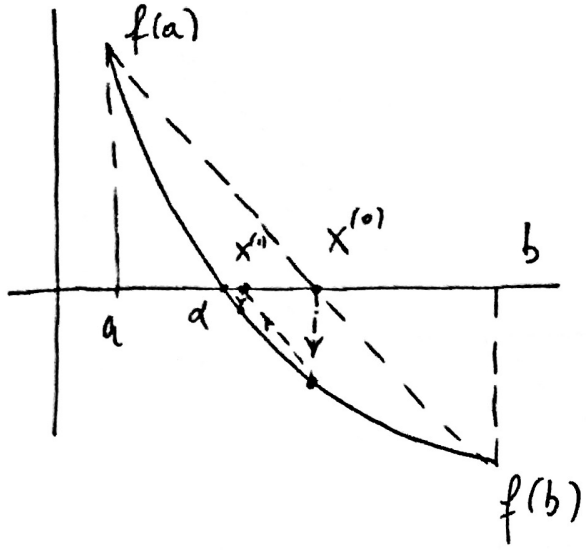
where q_k is the slope of a suitable line that passes through the point $(x^{(k)}, f(x^{(k)}))$ and allows us to determine $x^{(k+1)}$.



Remark: ALL these methods are NOT globally convergent, i.e., we need a good initial guess on the zeros.

How do we choose q_k ?

The chord method



$$q_K = \frac{f(b)-f(a)}{b-a}$$

(slope of the chord that passes through $(a, f(a))$ and $(b, f(b))$.)

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

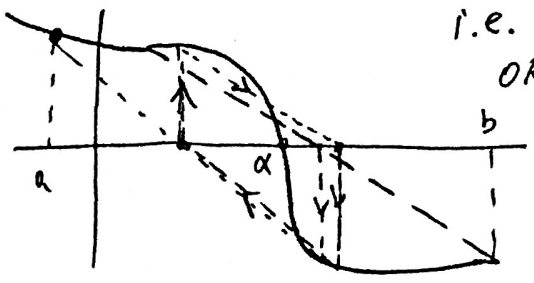
here x^0 is determined with a bisection step but we do not necessarily need such x^0. $\Rightarrow$ x^0 can be anything close to α.

The sequence has <u>convergence order $p=1$</u>
(we will prove this when we will talk about fixed point iterations).

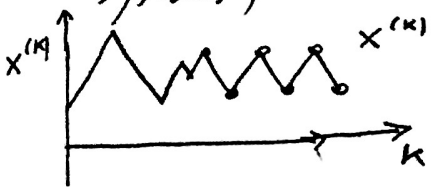
The method is NOT globally convergent, i.e., we need to pick a and b close enough to α. More precisely, we need to make sure that $\text{sign}(q) = \text{sign}(f'(\alpha))$ and that

$$|q| > \frac{|f'(\alpha)|}{2}$$

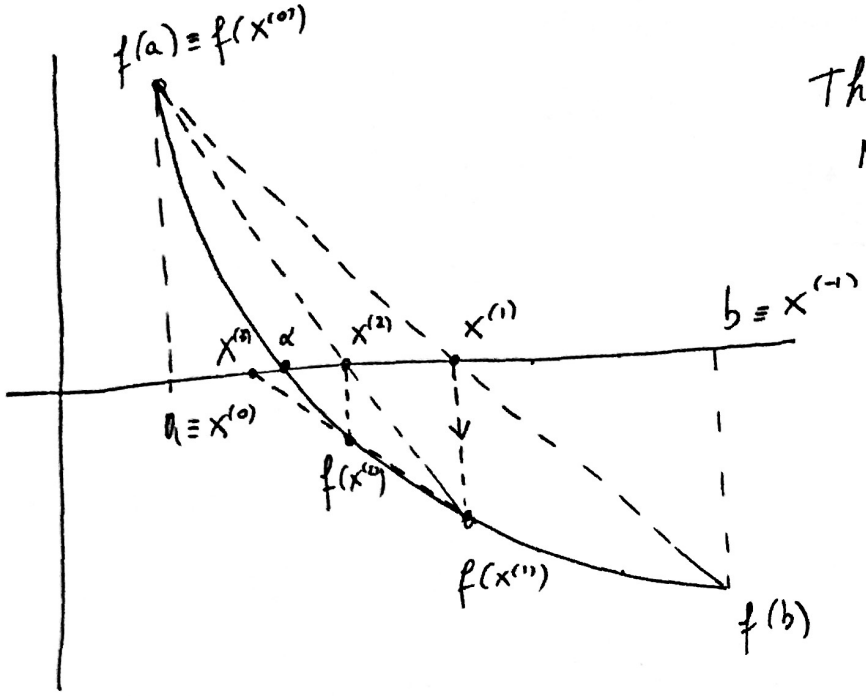
Otherwise we may end up in a situation like this:



i.e. A PERIOD 2 ORBIT (discrete dynamical systems)



Secant Method

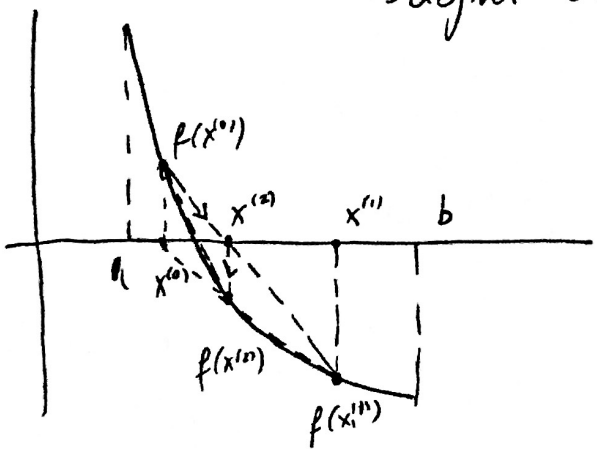


The method is NOT globally convergent

$$q_K = \frac{f(x^{(K)}) - f(x^{(K-1)})}{x^{(K)} - x^{(K-1)}}$$

we need two initial guesses to start the secant method (usually $x^{(0)} = a$, $x^{(-1)} = b$).

Any two initial guesses close enough to α would do the job. (even if they are both to the right or to the left of α)



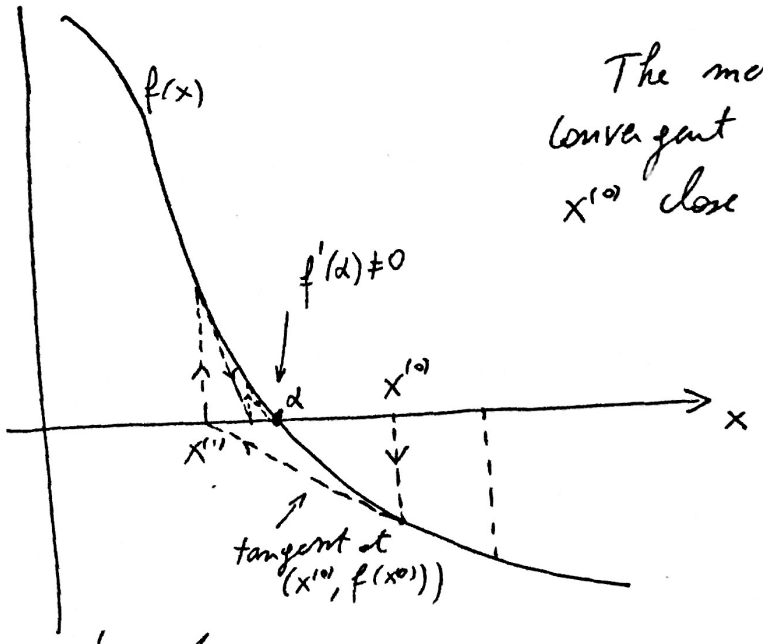
Remark: (Convergence order) If we choose the two initial guesses close enough to α and if $f'(\alpha) \neq 0$ then one can prove that $p = \frac{1+\sqrt{5}}{2} \simeq 1.63$ (GOLDEN RATIO) (SUPERLINEAR CONVERGENCE).

Newton's Method

Suppose that $f(x) \in C^{(1)}([a,b])$ and that $f'(\alpha) \neq 0$ (SIMPLE ROOT).

$$q_k = f'(x_1^{(k)}) \quad \Rightarrow \quad x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$$

(slope of f at $x^{(k)}$)



The method is convergent if we pick $x^{(0)}$ close enough to α

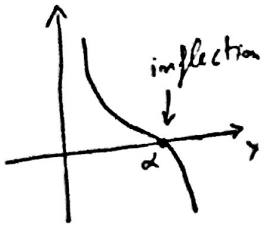
Remark: Alternative derivation:

$$f(x^{(k+1)}) \simeq f(x^{(k)}) + f'(x^{(k)})(x^{(k+1)} - x^{(k)}) + \cdots$$

$$\| \quad f(\alpha) = 0 \quad \rightarrow \quad x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$$

Remark (Convergence order) Suppose $f \in C^{(2)}$ and $f''(\alpha) \neq 0$ (i.e. α is not an inflection point)

Then: $$\lim_{k \to \infty} \frac{|x^{(k+1)} - \alpha|}{|x^{(k)} - \alpha|^2} = \frac{|f''(\alpha)|}{2|f'(\alpha)|}$$



$f'(\alpha) \neq 0$
$f''(\alpha) = 0$

$\Rightarrow$ The Newton method converges with order 2. (proof later).

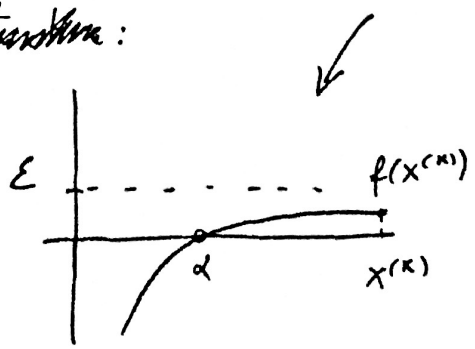
Remark (stopping criterion for $x^{(k)}$) - ANY METHOD WE DISCUSSED SO FAR

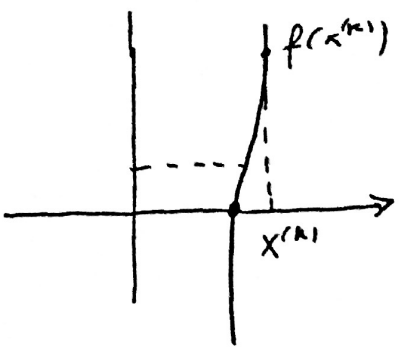
We do not know α, so we can't figure out k such that $|x^{(k)} - \alpha| \leq \varepsilon$ (ε is a threshold). Two criteria:

① Control on the increment (usually this is the condition implemented / that is)
$$|x^{(k)} - x^{(k-1)}| \leq \varepsilon_1$$

② Control on the residual $|f(x^{(k)})| \leq \varepsilon_2$

This can be excessively optimistic or too restrictive





Newton's method for nonlinear systems of equations

So far we discussed numerical methods to determine the ZEROS of scalar functions in one variable, i.e. $f:[a,b]\to\mathbb{R}$. Next, we consider the problem of determining the ZEROS of vector-valued functions of many variables, i.e. $\vec{f}: D\subseteq\mathbb{R}^n\to\mathbb{R}^n$. In other words, we would like to find the zeros of a nonlinear system of equations in the form:

$$\bar{f}(\bar{x}) = \bar{0} \qquad\qquad \bar{x}\in D\subseteq\mathbb{R}^n$$

In an expanded notation the previous system looks like:

$$\begin{cases} f_1(x_1, \dots, x_n) = 0 \\ \vdots \\ f_n(x_1, \dots, x_n) = 0 \end{cases}$$

Example: (linear systems)

In the case of linear systems we have that $\bar{f}(\bar{x})$ is linear in $\bar{x}$, i.e., it has the general form

$$\bar{f}(\bar{x}) = A\bar{x} - \bar{b}$$

For example,

$$\begin{cases} x_1 - 3x_2 - 1 = 0 \\ 3x_1 + x_2 + 5 = 0 \end{cases} \Rightarrow \begin{bmatrix} 1 & -3 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -5 \end{bmatrix}$$

Remark (Geometric meaning of rootfinding for nonlinear systems of equations)

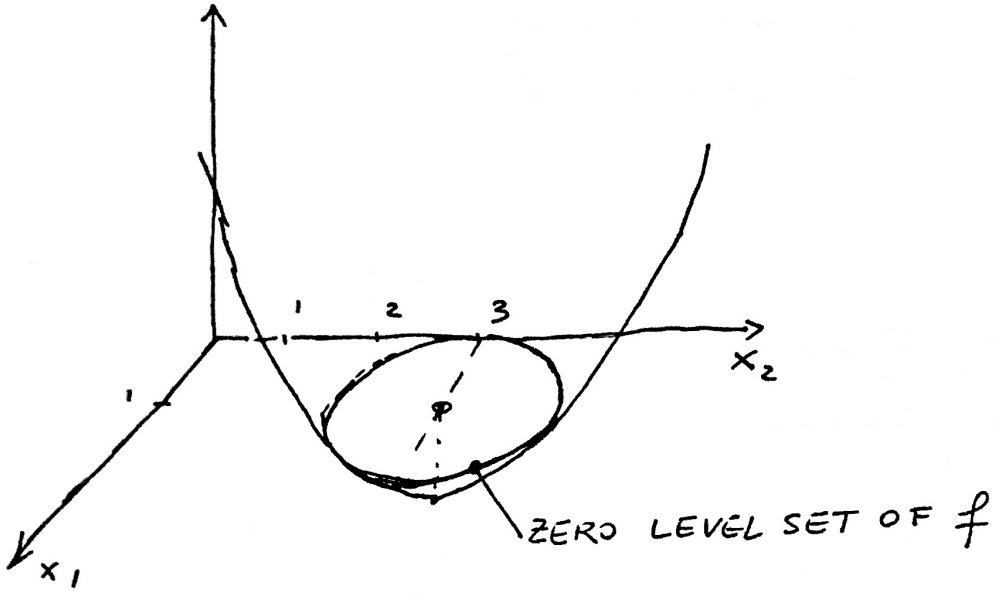
We are basically intersecting the ZERO LEVEL SETS of all surfaces defined by

$y_i = f_i(x_1, \cdots, x_n) \quad i = 1, \cdots, n$

The zero level set is the intersection of $f_i(x_1, \cdots, x_n)$ with the "hyperplane" $y_i = 0$

Example: (zero level set)

$$f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 3)^2 - 1$$

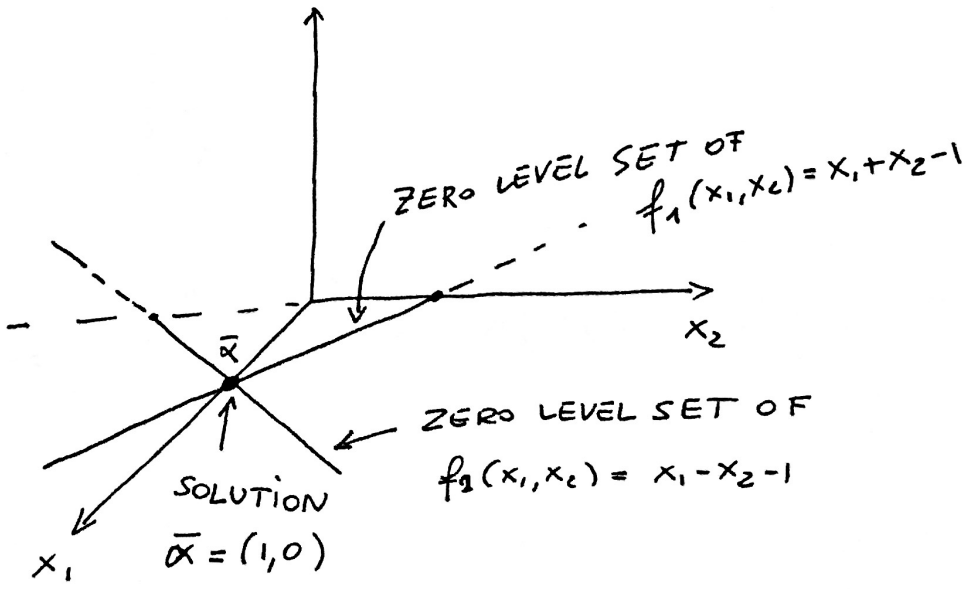


Example (solution to linear system)

$$\begin{cases} x_1 + x_2 - 1 = 0 \\ x_1 - x_2 - 1 = 0 \end{cases}$$

$x_2 = -x_1 + 1$

$x_2 = x_1 - 1$



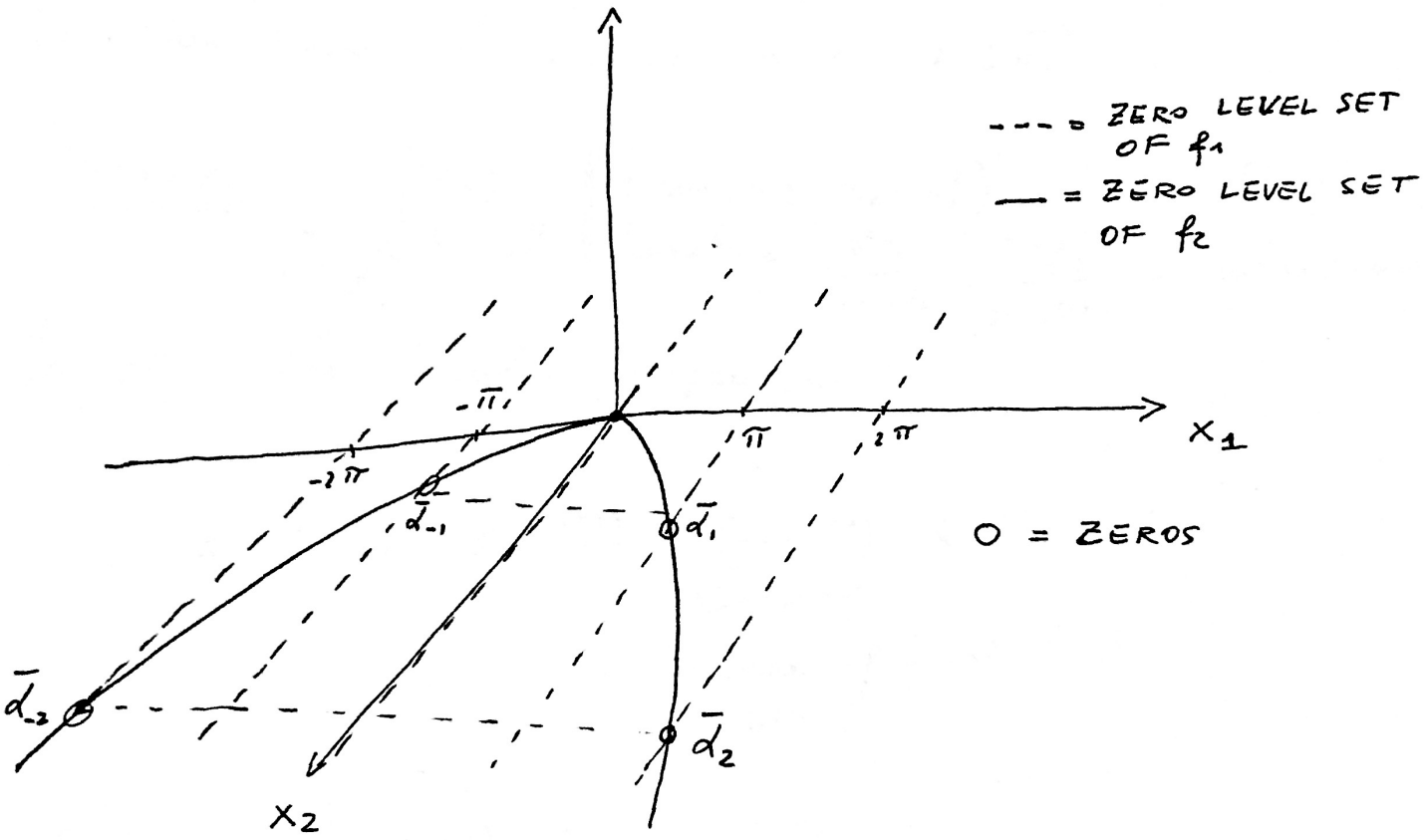
Example (Solution to nonlinear systems)

$$\begin{cases} \sin(x_1) = 0 \\ x_2 - x_1^2 = 0 \end{cases} \Rightarrow$$

the solutions are at the intersections between zero level sets of

$f_1(x_1, x_2) = \sin(x_1)$

$f_2(x_1, x_2) = x_2 - x_1^2$



All zeros are in the form $\bar{x}_k = (k\pi, k^2\pi^2)$ $k \in \mathbb{Z}$

Many methods have been developed to compute the solution to a nonlinear system of equations. Newton's method is one of the simplest.

To derive the Newton's method, let us assume that $\bar{f} \in C^{(1)}(D)$ $D \subseteq \mathbb{R}^n$, i.e., that the vector-valued function $\bar{f}(\bar{x})$ is differentiable with continuous derivatives in $D \subseteq \mathbb{R}^n$.

Remark: $\bar{f}(\bar{x}) = (\sin(x_1), x_2 - x_1^2)$ is $C^{(\infty)}$ in $\mathbb{R}^2$
In fact, all derivatives $\frac{\partial^2 f_1}{\partial x_1 \partial x_2}$, $\frac{\partial^4 f_2}{\partial^3 x_1 \partial x_2}$, ... are continuous

Consider the Taylor expansion:

$$\bar{f}(\bar{x}^{(k+1)}) = \bar{f}(\bar{x}^{(k)}) + \underbrace{J_{\bar{f}}(\bar{x}^{(k)})}_{\text{Jacobian of } \bar{f} \text{ calculated at } \bar{x}^{(k)}} (\bar{x}^{(k+1)} - \bar{x}^{(k)}) + \dots$$

Setting $\bar{f}(\bar{x}^{(k+1)}) = \bar{0}$ yields:

$$\boxed{\bar{x}^{(k+1)} = \bar{x}^{(k)} - J_{\bar{f}}^{-1}(\bar{x}^{(k)})\, \bar{f}(\bar{x}^{(k)})}$$

vector-valued discrete dynamical system.

$J_{\bar{f}}^{-1}(\bar{x}^{(k)})$ is the inverse of the jacobian matrix at $\bar{x}^{(k)}$.

Remark (what is a Jacobian matrix?)

Consider n multivariate functions of n variables

$$f_1(x_1, \cdots, x_n) \quad , \cdots , \quad f_n(x_1, \cdots, x_n)$$

The Jacobian of $\bar{f}(\bar{x})$ is a matrix that collects the partial derivatives of f_i with respect to any variable x_j, i.e.,

$$J_{\bar{f}}(\bar{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

Example: $f_1(x_1, x_2) = \sin(x_1)$ $\quad$ $f_2(x_1, x_2) = x_2 - x_1^2$

$$J_{\bar{f}}(\bar{x}) = \begin{bmatrix} \cos(x_1) & 0 \\ -2x_1 & 1 \end{bmatrix}$$

$\det(J_{\bar{f}}(\bar{x})) = \cos(x_1) \Rightarrow$ the Jacobian is singular at $x_1 = \frac{\pi}{2} + k\pi \quad k \in \mathbb{Z}$ (see previous figure)

① Clearly the Jacobian matrix has to be nonsingular for the method to be applicable (otherwise the inverse does not exist).

Remark: At each iteration we need to solve the linear system:

$$\boxed{J_{\bar{f}}(\bar{x}^{(k)})\,(\bar{x}^{(k+1)} - \bar{x}^{(k)}) = -\,\bar{f}(\bar{x}^{(k)})}$$

(DO NOT COMPUTE J_f^{-1} explicitly)

② In particular, $J_f(\bar{\alpha})$ ($\bar{\alpha}$ is the ZERO WE ARE AFTER) must be nonsingular. The continuity of the derivatives ($\bar{f}$ is $C^{(1)}$) guarantees that if $J_f(\bar{\alpha})$ is nonsingular then there exists a neighborhood of $\bar{\alpha}$ where $J_f(\bar{x})$ is nonsingular.

③ As in the scalar case, we need to select $\bar{x}^{(0)}$ close enough to $\bar{\alpha}$

Theorem (convergence of Newton's method for systems)

Let $\bar{f}: \mathbb{R}^n \to \mathbb{R}^n$ be $C^{(1)}$ in a convex open set $D \subseteq \mathbb{R}^n$, $\bar{\alpha} \in D$. Suppose that there exists $R, L, G \in \mathbb{R}^+$ such that

$$\| J_{\bar{f}}^{-1}(\bar{\alpha}) \| \leq G$$

$$\| J_{\bar{f}}(\bar{x}) - J_{\bar{f}}(\bar{y}) \| \leq L \| \bar{x} - \bar{y} \| \qquad \forall \bar{x}, \bar{y} \in B(\bar{\alpha}, R)$$

($\bar{\alpha}$: CENTER; B: BALL; R: RADIUS)

Then there exists $r > 0$ such that for any $\bar{x}^{(0)} \in B(\bar{\alpha}, r)$ the sequence

$$\bar{x}^{(k+1)} = \bar{x}^{(k)} - J_{\bar{f}}^{-1}(x^{(k)})\, \bar{f}(x^{(k)})$$

converges to $\bar{\alpha}$ with order 2, i.e.,

$$\| \bar{x}^{(k+1)} - \bar{\alpha} \| \leq GL \| \bar{x}^{(k)} - \bar{\alpha} \|^2 \qquad \forall k \geq k_0$$

Fixed-point iterations

Geometric approaches to rootfinding can be studied and analyzed by using a general framework, i.e., the theory of fixed points. The basic idea is the following. Given a nonlinear function

$$f : [a,b] \longrightarrow \mathbb{R}$$

we transform the problem $f(\alpha) = 0$ into an <u>equivalent</u> problem in the form $\phi(\alpha) = \alpha$, where ϕ is a suitable AUXILIARY FUNCTION.

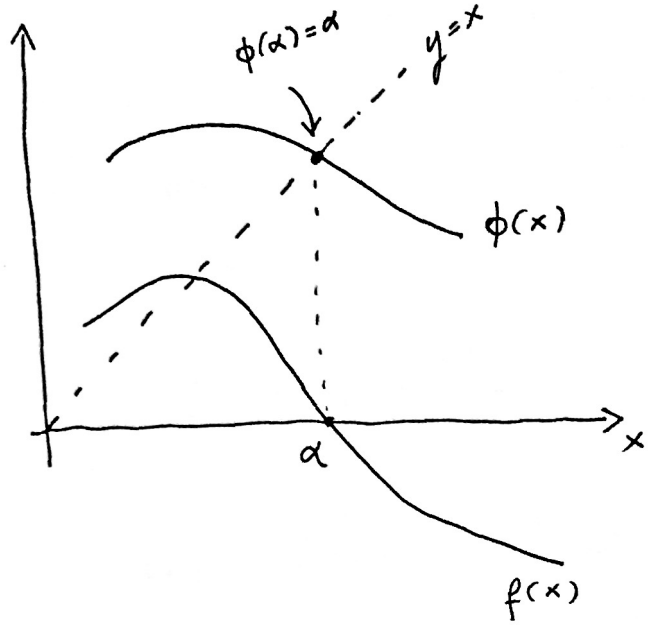
$$f(\alpha) = 0 \quad \longleftrightarrow \quad \phi(\alpha) = \alpha$$

(α is a ZERO OF $f(x)$) (α is A FIXED POINT OF $\phi(x)$)

<u>Example</u>: Chord method and Newton's method.

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

ZERO OF f IS A FIXED POINT OF ϕ



Determining the zeros of a function f is thus equivalent to finding the fixed points of a suitable auxiliary function. This can be done by the following algorithm:

$$x^{(k+1)} = \phi(x^{(k)}) \qquad (x^{(0)} \text{ given})$$

Remark: (ϕ is not unique given f). There are many iteration functions ϕ that can be constructed for a given $f(x)$. Consider, for example,

$$\phi(x) = x + F(f(x))$$

where F is any continuous function such that $F(0) = 0$. Examples of such function F could be:

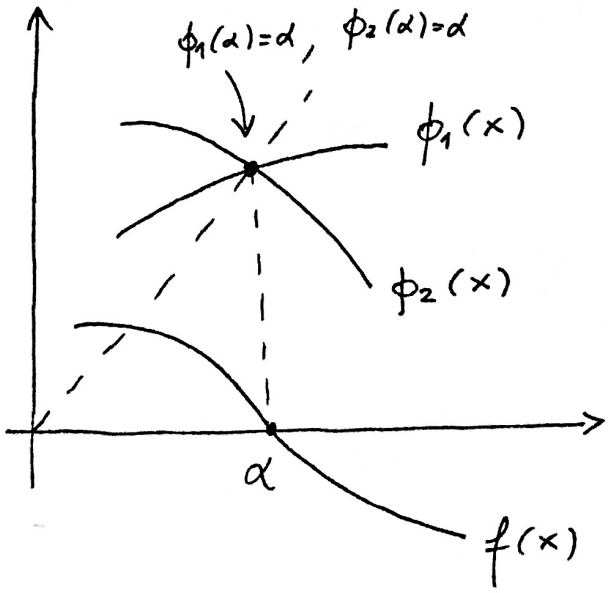
$$F(x) = x$$

$$F(x) = e^{-x} - 1$$

This means that the zeros of a given function $f(x)$ can be (in principle) computed by determining the fixed points of different auxiliary functions. In the example above, we have:

$$\phi_1(x) = x + f(x)$$

$$\phi_2(x) = x + e^{-f(x)} - 1$$



α is a fixed point of both ϕ_1 and ϕ_2

Remark: The choice of the auxiliary function ϕ can influence stability, convergence and convergence rate of fixed point iterations.

Example: $f(x) = x^4 - 4$ has 2 real zeros $\alpha_{1,2} = \pm\sqrt{2}$

Let us consider the following auxiliary functions:

$$\phi_1(x) = x + f(x)$$

$$\phi_2(x) = x - \frac{f(x)}{11}$$

Both are such that $f(\alpha) = 0 \Downarrow \phi(\alpha) = \alpha$

Set $x^{(0)} = 1$. Then:

$x^{(k+1)} = \phi_1(x^{(k)})$ diverges

$x^{(k+1)} = \phi_2(x^{(k)})$ converges to $\sqrt{2}$

(Try it in your computer)

Why is this happening?

Theorem (sufficient conditions for convergence of fixed point iterations)

Consider the sequence $x^{(k+1)} = \phi(x^{(k)})$, $x^{(0)}$ given. Then:

① If $\phi: [a,b] \to [a,b]$ is continuous then there exists at least one fixed point in $[a,b]$. Moreover, if $\phi(x)$ is a contraction, i.e.

$$\exists\ L < 1 \ : \ |\phi(x_1) - \phi(x_2)| \leq L\,|x_1 - x_2| \qquad \forall\, x_1, x_2 \in [a,b]$$

then the fixed point is unique in $[a,b]$ and globally attracting (any $x^{(0)} \in [a,b]$ will converge to such unique fixed point).

② If $\phi \in C^{(1)}([a,b])$ and $\exists\ \kappa < 1 : |\phi'(x)| \leq \kappa$ for all $x \in [a,b]$ then we have a globally attracting unique fixed point in $[a,b]$.

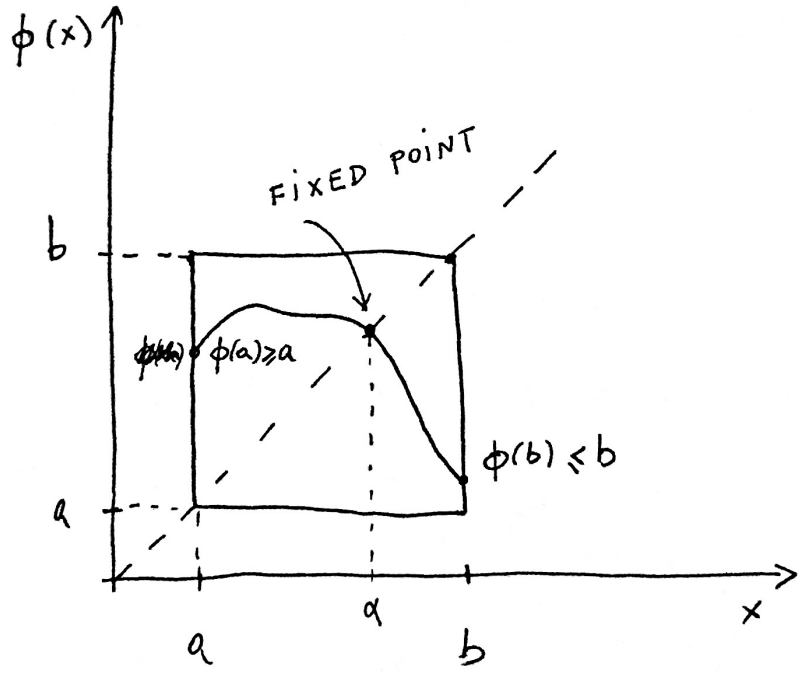
Proof

① Let $g(x) = \phi(x) - x$. g is continuous by our assumptions on ϕ. In addition, since the range of ϕ is bounded, we have:

$$g(a) = \phi(a) - a \geq 0$$

$$g(b) = \phi(b) - b \leq 0$$

The reason for such inequalities is the following:



$\Rightarrow$ $g(a) \geq 0$, $g(b) \leq 0$ with continuous g implies that there exists at least one zero of g in $[a,b]$, i.e., one fixed point of $\phi(x)$.

Now, let us assume that ϕ is a contraction, i.e.,

$$|\phi(x_1) - \phi(x_2)| \leq L\,|x_1 - x_2| \qquad x_1, x_2 \in [a,b]$$
$$L < 1$$

We want to prove that in this case (the fixed point) is unique. Let us proceed by contradiction. Suppose we have two fixed points α_1 and α_2 in $[a,b]$. Then

$$|\underbrace{\phi(\alpha_1)}_{\alpha_1} - \underbrace{\phi(\alpha_2)}_{\alpha_2}| \leq L\,|\alpha_1 - \alpha_2| < |\alpha_1 - \alpha_2| \quad (L<1)$$

Therefore we have $|\alpha_1 - \alpha_2| < |\alpha_1 - \alpha_2|$ which is impossible. Therefore $\alpha_1 = \alpha_2$, i.e., the fixed point is unique if ϕ is contracting.

Now, let us prove the globally attracting property, i.e., converge to the fixed point disregarding the initial guess $x^{(0)}$.

$$|x^{(k+1)} - \alpha| = |\phi(x^{(k)}) - \phi(\alpha)| \leq L\,|x^{(k)} - \alpha|$$
$$\leq L^2\,|x^{(k-1)} - \alpha|$$
$$\vdots$$
$$\leq L^{k+1}\,|x^{(0)} - \alpha|$$

Therefore

$$\frac{|x^{(k+1)} - \alpha|}{|x^{(0)} - \alpha|} \leq L^{k+1} \qquad (L<1)$$

$\Rightarrow$ $x^{(k)}$ converges to α disregarding $x^{(0)}$ (simply take the limit left and right)

Therefore if ϕ is continuous and contracting we have a unique globally attracting fixed point and the sequence $x^{(k)}$ converges to that point with order 1.

(2) Let $\phi \in C^{(1)}([a,b])$ with $|\phi'(x)| < 1$ for all $x \in [a,b]$. Let us prove by contradiction that there exists only one fixed point in $[a,b]$. To this end, assume that α_1, α_2 are two fixed points

$$|\alpha_1 - \alpha_2| = |\phi(\alpha_1) - \phi(\alpha_2)| = |\phi'(\eta)(\alpha_1 - \alpha_2)| = \underbrace{|\phi'(\eta)|}_{<1} |\alpha_1 - \alpha_2|$$

mean value theorem ($\eta \in [a,b]$)

$\Rightarrow |\alpha_1 - \alpha_2| < |\alpha_1 - \alpha_2|$ impossible, therefore $\alpha_1 = \alpha_2$

Let us prove that $x^{(k)}$ converges to α disregarding $x^{(0)}$.

$$x^{(k+1)} - \alpha = \phi(x^{(k)}) - \phi(\alpha) = \phi'(\eta^{(k)})(x^{(k)} - \alpha)$$

$\eta^{(k)}$ is some point between $x^{(k)}$ and α ⊛

$$\Rightarrow \frac{|x^{(k+1)} - \alpha|}{|x^{(k)} - \alpha|} = |\phi'(\eta^{(k)})| < 1$$

$\eta^{(k)} \in [x^{(k)}, \alpha]$ or $\eta^{(k)} \in [\alpha, x^{(k)}]$

Taking the limit

$$\lim_{k \to \infty} \frac{|x^{(k+1)} - \alpha|}{|x^{(k)} - \alpha|} = |\phi'(\alpha)|$$

(here we used the continuity of ϕ')

This means that if $\phi'(\alpha) \neq 0$ then the sequence converges with order 1.
Also $|\phi'(\alpha)| < 1$ is the convergence factor
As we will see, if $\phi'(\alpha) = 0$ then the sequence converges with order higher than one.

#

As a corollary of part ② of the previous theorem we have

Theorem (Ostrowsky) Let α be a fixed point of $\phi \in C^{(1)}([a,b])$. If $|\phi'(\alpha)| < 1$ then there exists a neighborhood of α $I_\alpha^\delta = \{x \in [a,b] \mid |x-\alpha| \leq \delta\}$ such that

$$\lim_{k \to \infty} x^{(k)} = \alpha \quad \text{for any } x^{(0)} \in I_\alpha^\delta$$

Example: Consider again $f(x) = x^4 - 4$ and the auxiliary functions:

$$\phi_1 = x + f(x)$$

$$\phi_2 = x - \frac{f(x)}{11}$$

$\alpha_{1,2} = \pm\sqrt{2}$ (real zeros)

It can be shown that $|\phi_1'(x)| > 0 \quad \forall x > 0$ while $|\phi_2'(x)| < 1 \quad \forall x \in [0, 1.765]$

$\Rightarrow$ if we pick $x^{(0)} = 1$ ϕ_1 diverges while ϕ_2 converges to a unique fixed point in $[0, 1.765]$, i.e., $\sqrt{2}$.

Theorem (Convergence order of fixed-point Iterations)

Let I_α^δ be a neighborhood of α and

$\phi \in C^{p+1}(I_\alpha^\delta)$ $\quad p \geq 1$ (integer).

If $\dfrac{d^i \phi(\alpha)}{dx^i} = 0 \quad i = 1, \cdots, p$ and $\dfrac{d^{p+1}\phi(\alpha)}{dx^{p+1}} \neq 0$ (FINITE)

then the sequence $x^{(k+1)} = \phi(x^{(k)})$ converges to α with order $p+1$ for any $x^{(0)} \in I_\alpha^\delta$

$$\lim_{k\to\infty} \frac{(x^{(k+1)} - \alpha)}{(x^{(k)} - \alpha)^{p+1}} = \frac{1}{(p+1)!} \frac{d^{p+1}\phi(\alpha)}{dx^{p+1}}$$

Proof Consider the Taylor series

$$x^{(k+1)} - \alpha = \phi(x^{(k)}) - \phi(\alpha) = \sum_{i=1}^{p} \frac{1}{i!} \frac{d^i \phi(\alpha)}{dx^i} (x^{(k)} - \alpha)^i +$$

$$+ \frac{1}{(p+1)!} \frac{d^{p+1}\phi}{dx^{p+1}} (\eta^{(k)}) (x^{(k)} - \alpha)^{p+1}$$

$\eta^{(k)} \in [x^{(k)}, \alpha]$ or $[\alpha, x^{(k)}]$

By continuity.

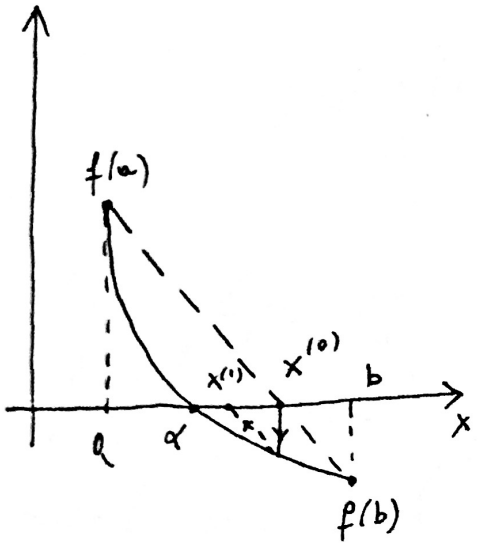
$$\lim_{k\to\infty} \frac{(x^{(k+1)} - \alpha)}{(x^{(k)} - \alpha)^{p+1}} = \frac{1}{(p+1)!} \frac{d^{p+1}\phi}{dx^{p+1}} (\alpha)$$

#

Remark: We do not need to assume that $\left| \frac{d^{p+1}\phi}{dx^{p+1}}(\alpha) \right| < 1$. Remember that this is necessary for convergence only if $p=0$ (Ostrowsky theorem).

Convergence Results for some fixed point methods

The chord method



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

$$\Rightarrow x^{(k+1)} = \phi(x^{(k)})$$

$$\phi(x) = x - \frac{(b-a)}{f(b)-f(a)} f(x)$$

(AUXILIARY FUNCTION)

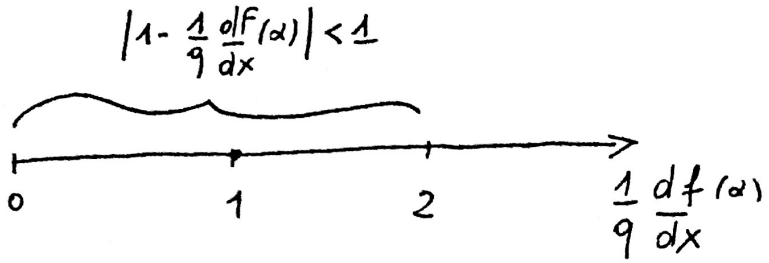
$$\frac{d\phi}{dx} = 1 - \frac{(b-a)}{f(b)-f(a)} \frac{df}{dx}$$

If $\frac{df}{dx}(\alpha) = 0$ (multiple zero) we have $\frac{d\phi}{dx}(\alpha) = 1$

and the chord method is not granted to converge.

Let $q = \frac{f(b)-f(a)}{b-a}$. In order to have $\left|\frac{d\phi}{dx}(\alpha)\right| < 1$

we must have $\left|1 - \frac{1}{q}\frac{df}{dx}(\alpha)\right| < 1$



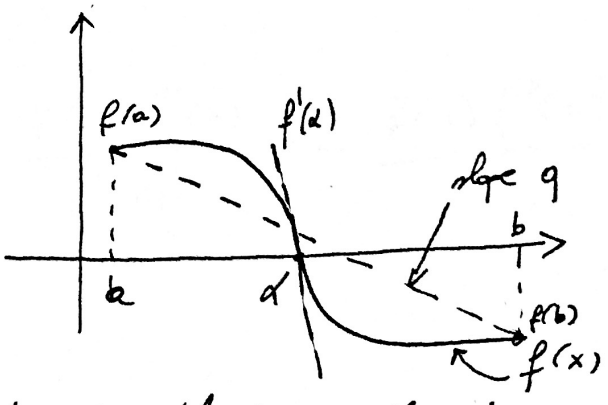
This means that

$$0 < \frac{df}{dx}(\alpha) \cdot \frac{1}{q} < 2$$

(1) $\Rightarrow$ the slope of the chord q must be of the same sign as $\frac{df}{dx}(\alpha)$ and must be such that:

$$|q| > \frac{1}{2}\left|\frac{df}{dx}(\alpha)\right|$$

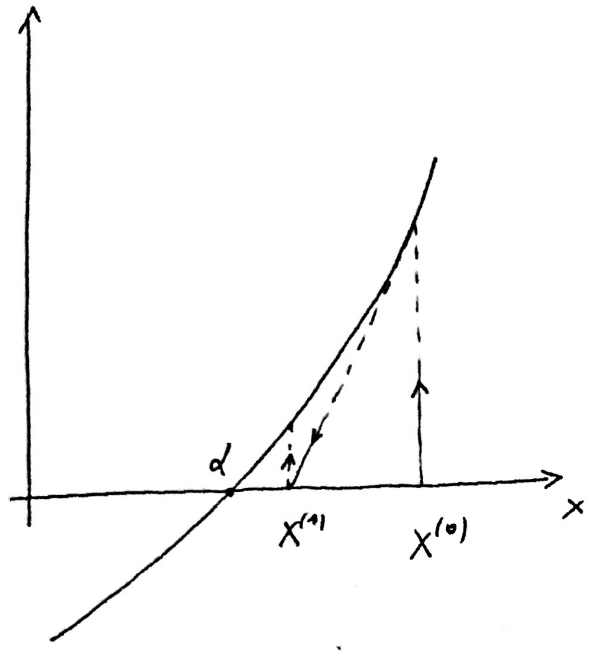
Example:



The chord method will not converge here as $|q| < \frac{1}{2}|f'(\alpha)|$

(2) $\Rightarrow$ The chord method converges with ORDER 1 since $\phi'(\alpha) \neq 0$

Newton's method



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

$$\Rightarrow x^{(k+1)} = \phi(x^{(k)})$$

$$\phi(x) = x - \frac{f(x)}{f'(x)}$$

(auxiliary function)

$$\phi'(x) = 1 - \frac{f'(x)}{f'(x)} + \frac{f(x)}{f'(x)^2} f''(x)$$

$$= \frac{f(x)}{f'(x)^2} f''(x)$$

$$\phi''(x) = \frac{f'(x)}{f'(x)^2} f''(x) + \frac{f(x) f'''(x)}{f'(x)^2} - 2 \frac{f(x) f''(x)^2}{f'(x)^3}$$

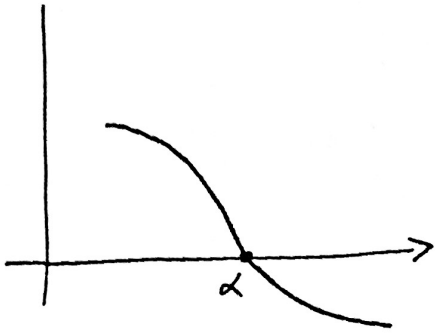
$$\Rightarrow \phi'(\alpha) = 0$$

$$\phi''(\alpha) = \frac{f''(\alpha)}{f'(\alpha)} \Rightarrow$$

if $f''(\alpha) \neq 0$ then the Newton's method converges with ORDER 2

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

Question: Can the Newton's method converge with order 3? Yes, if $f'(\alpha) \neq 0$ and $f''(\alpha) = 0$ (simple zero at inflection point of $f(x)$):



How to stop fixed point iterations

- Control on the residual $\quad |f(x^{(k)})| < \varepsilon$

  could be too restrictive ↙ ↘ could be excessively optimistic

- Control on the increment

$$x^{(k+1)} - x^{(k)} = (x^{(k+1)} - \alpha) + (\alpha - x^{(k)})$$

$$= (\alpha - x^{(k)})\left(1 - \phi'(\eta^{(k)})\right)$$

In fact $\quad x^{(k+1)} - \alpha = \phi(x^{(k)}) - \phi(\alpha) = \phi'(\eta^{(k)})(x^{(k)} - \alpha)$

(mean value theorem)

This implies that:

$$\alpha - x^{(k)} = \frac{1}{1 - \phi'(\eta^{(k)})} \left(x^{(k+1)} - x^{(k)} \right)$$

($\eta^{(k)}$ between α and $x^{(k)}$)

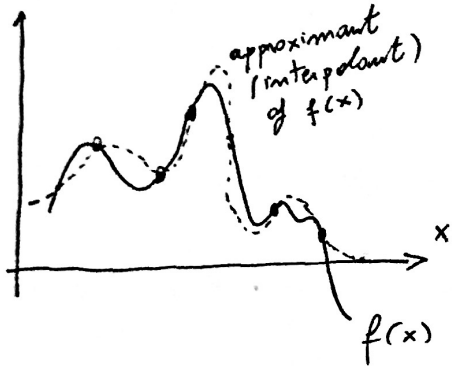
$$\simeq \frac{1}{1 - \phi'(\alpha)} \left(x^{(k+1)} - x^{(k)} \right)$$

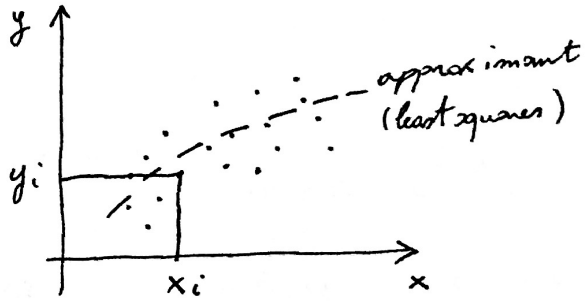
If $\phi'(\alpha) = 0$ (e.g. Newton's method) then the control on the increment gives us a very good estimate of the distance to the zero. On the other hand, if $\phi'(\alpha) \simeq 1$ then the test on the increment has issues.

Example: the chord method with a double root ($f'(\alpha) = 0$) has $\phi'(\alpha) = 1$

Approximation of functions and data

In many mathematical problems there is the need to approximate a function $f(x)$ or a set of data points $\{x_i, y_i\}_{i=0,\cdots,n}$ with another function that is much more simple.





This allows us to perform operations such as integration, differentiation, solve differential equations, etc. To approximate $f(x)$ we need to identify:

① The class of approximating functions, i.e., in which function space we look for an approximant:

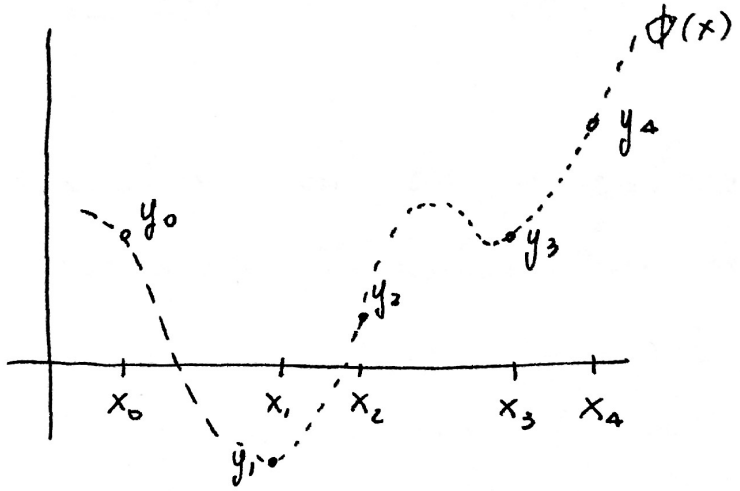
- Polynomials
- Trigonometric functions
- Rational functions

(2) The criterion to select a particular element within the class of approximating functions. For example:

- Interpolation
- Projection
- Least squares

Interpolation

Suppose we have available $n+1$ data points (x_i, y_i) $i = 0, \cdots, n$, where the nodes x_i are all distinct



We look for an approximant $\Phi(x)$ that satisfies the following INTERPOLATION

CONDITIONS

$$\phi(x_i) = y_i \qquad i = 0, \dots, n$$

Examples:

1) $\phi(x) = \sum_{k=0}^{n} a_k x^k$ (polynomial interpolants)

2) $\phi(x) = \sum_{k=-\frac{n}{2}}^{n/2} c_k e^{ikx}$ $\left(e^{ikx} = \cos(kx) + i \sin(kx)\right)$ (trigonometric interpolants)

$$= \sum_{k=1}^{m/2} \alpha_k \sin(kx) + \beta_k \cos(kx) + \alpha_0$$

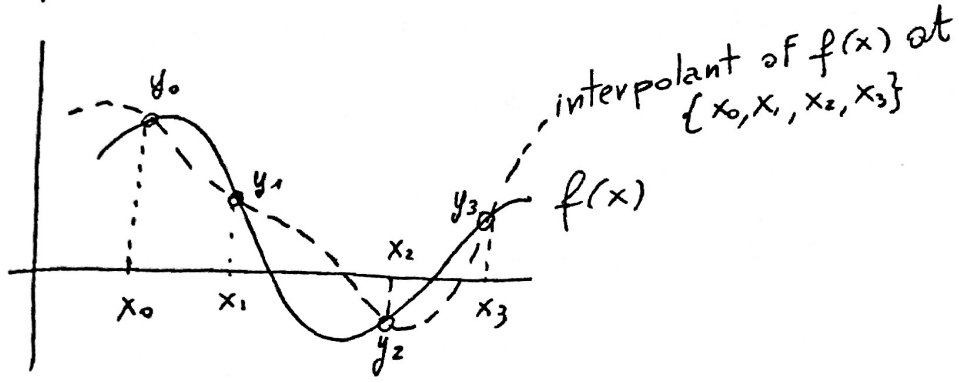
3) $\phi(x) = \dfrac{\sum_{i=0}^{h} a_i x^i}{\sum_{j=0}^{q} b_j x^j}$ (Rational functions)

$h + q = n - 1$

$((h+1) + (q+1) = n+1)$

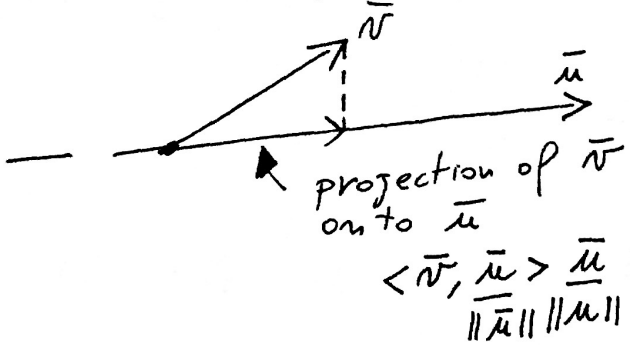
Remark: Interpolation can be generalized to any nonlinear transformation between vector spaces. For example, if X and Y are two linear spaces, e.g., $\mathbb{R}^n$ or $C^{(0)}([a,b])$, and $x_i \in X$ $y_i \in Y$, then we can construct an interpolant $H: X \to Y$ such that $H(x_i) = y_i$

Remark: We can also interpolate a function $f(x)$, provided we sample it at suitable nodes



Projection

The projection method relies on a generalization of the concepts we learned in linear algebra.



A function $f(x)$ is indeed a VECTOR in a LINEAR SPACE of infinite dimension

Example: $C^{(0)}([-1,1])$ continuous functions in $[-1,1]$

$L_2([-1,1])$ square integrable functions in $[-1,1]$

($f(x) \in L_2([-1,1])$ if $\int_{-1}^{1} f(x)^2 dx < \infty$)

$C^{(0)}$ and L_2 are closed under addition and subtraction, we have the neutral element with respect to addition, we can define scalar multiplication, etc..., In other words C^0 and L_2 are vector spaces.

We can define the inner product (scalar product):

$$(\phi_1(x), \phi_2(x)) = \int_{-1}^{1} \phi_1(x)\,\phi_2(x)\,dx \qquad \phi_1, \phi_2 \in C^{(0)}([-1,1])$$

(L_2 inner product)

on the norm:

$$\|\phi_1\|_2^2 = (\phi_1(x), \phi_1(x)) = \int_{-1}^{1} \phi_1(x)^2 dx$$

(norm induced by the inner product)

(This measures the LENGTH of the function $\phi_1(x)$)

With the INNER PRODUCT $(,)$ we can project and the function $f(x)$ onto a BASIS $\{\phi_1(x), \dots, \phi_n(x)\}$ (ϕ_i are simple functions, for example polynomials)

$$f(x) \simeq \sum_{k=1}^{n} a_k \phi_k(x)$$

Suppose that $(\phi_i, \phi_j) = \delta_{ij}$, then

$$a_k = \int_{-1}^{1} \phi_k(x) f(x)\, dx \qquad k = 1, \cdots, n$$

Thus, approximating $f(x)$ through projection reduces to computing integrals.

Least squares

Consider the error norm

$$\left\| f(x) - \sum_{k=1}^{n} a_k \phi_k(x) \right\|_2^2 = \int_{-1}^{1} \left(f(x) - \sum_{k=1}^{n} a_k \phi_k(x) \right)^2 dx$$

$$\min_{a_1, \cdots, a_n} \left\| f(x) - \sum_{k=1}^{n} a_k \phi_k(x) \right\|_2^2 \Rightarrow \sum_{k=1}^{n} M_{jk} a_k = b_j \qquad j = 1, \cdots, n$$

$$M\bar{a} = \bar{b}$$

$$M_{jk} = \int_{-1}^{1} \phi_j(x) \phi_k(x)\, dx \qquad b_j = \int_{-1}^{1} f(x) \phi_j(x)\, dx$$

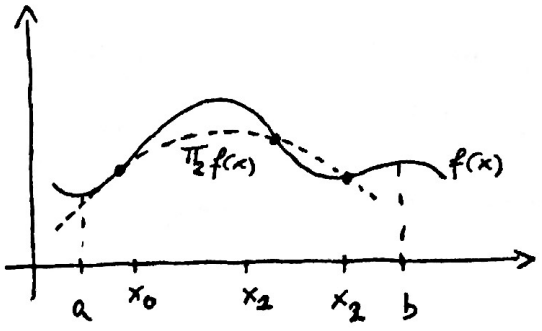
Polynomial interpolation (Global)

Consider a set of nodes $\{x_0, \dots, x_n\}$ in $[a,b]$ and let $y_i = f(x_i)$ be the values of $f(x)$ at x_i. Assume that $x_i \neq x_j$ for $i \neq j$.

Proposition For any set of couples $\{x_i, f(x_i)\}$ $i = 0, \dots, n$ with distinct x_i, there exists a unique polynomial $\Pi_n f(x)$ of degree less or equal than n such that

$$\Pi_n f(x_i) = f(x_i) \qquad i = 0, \dots, n$$

(INTERPOLATION CONDITION)



Proof Consider the $n+1$ pairs $\{x_i, y_i\}$ where $y_i = f(x_i)$ $i=0,\cdots,n$ and the polynomial

$$\Pi_n f(x) = a_0 + \cdots + a_n x^n$$

By imposing the interpolation conditions $\Pi_n f(x_i) = y_i$ we obtain the following system for $[a_0, \cdots, a_n]^T$

$$\underbrace{\begin{bmatrix} 1 & x_0 & \cdots & x_0^n \\ 1 & x_1 & \cdots & x_1^n \\ \vdots & \vdots & & \vdots \\ 1 & x_n & & x_n^n \end{bmatrix}}_{\text{Vandermonde matrix } V} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

If $x_i \neq x_j$ for $i \neq j$ (distinct nodes) then the Vandermonde matrix is non singular (but very badly conditioned usually)

$$\det(V) = \prod_{0 \leq i < j \leq n} (x_i - x_j) = \prod_{j=0}^{n-1} \prod_{i=j+1}^{n} (x_i - x_j) \neq 0$$

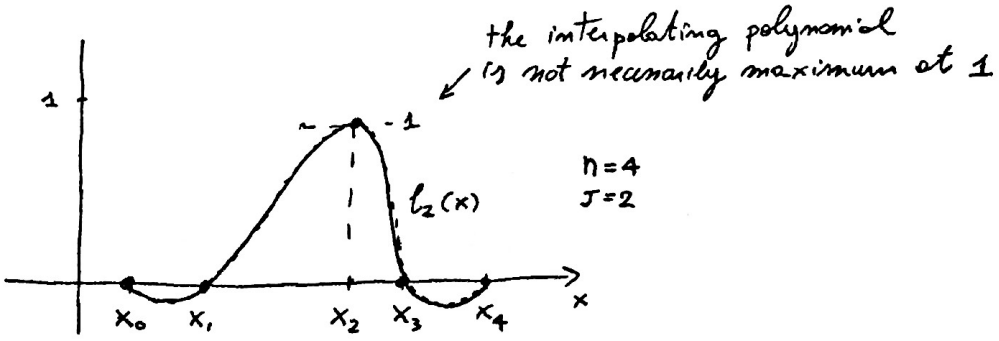
Therefore V can be inverted to obtain $[a_0, \cdots, a_n]^T$ uniquely, i.e., $\Pi_n f(x)$ is unique.

Lagrangian polynomial interpolation

To construct $\Pi_n f(x)$ effectively, let us first consider the set of polynomials interpolating

$$\{x_i, \delta_{ij}\} \quad i=0,\cdots,n \qquad j=0,\cdots,n$$

Example



Such polynomials satisfy

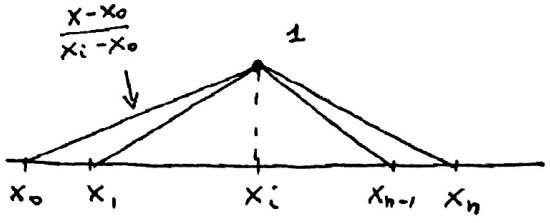
$$l_i(x_j) = \delta_{ij} \qquad \text{(CARDINAL BASIS)}$$

and they can can be written as:

$$l_i(x) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{(x-x_j)}{(x_i-x_j)}$$

LAGRANGE CHARACTERISTIC POLYNOMIALS

Remark (Geometrical interpretation)



$l_i(x)$ is the product of all lines above

Remark (SPAN): Each Lagrange polynomial is of order n, and we have $n+1$ linearly independent ones. This means that

$$\mathbb{P}_n = \text{span}\{l_0(x), \dots, l_n(x)\}$$

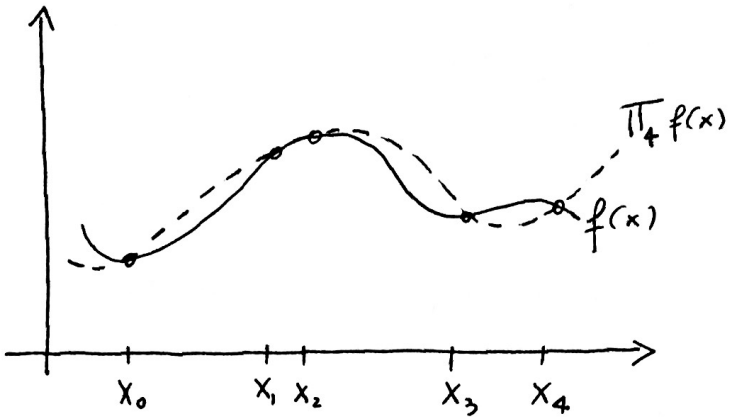
i.e. the set of Lagrange polynomials associated with a set of nodes spans the space of polynomials of order n.

$$\Pi_n f(x) = \sum_{k=0}^{n} f(x_k)\, l_k(x) \qquad \left(\begin{matrix}\text{INTERPOLATION}\\ \text{FORMULA}\end{matrix}\right)$$

Note that $\Pi_n f(x_j) = \sum_{i=0}^{n} f(x_i) \underbrace{l_i(x_j)}_{\delta_{ij}} = \sum_{i=1}^{n} f(x_i)\delta_{ij} = f(x_j)$

i.e. $\Pi_n f(x)$ interpolates $f(x)$ at $\{x_0, \dots, x_n\}$.

Interpolation Error



Replacing $f(x)$ with a polynomial interpolant $\Pi_n f(x)$ generates an error that depends on $f(x)$ as well as on the number and LOCATION of the interpolation nodes.

Theorem Let $f \in C^{(0)}([a,b])$ and $\Pi_n f(x)$ the n-th order interpolating polynomial of $f(x)$ at $\{x_0, \dots, x_n\}$. Then,

$$\| f(x) - \Pi_n f(x) \|_\infty \leq (1 + \Lambda_n) \inf_{\Psi \in \mathbb{P}_n} \| f(x) - \Psi(x) \|_\infty$$

Ψ: best approximating polynomial

$$\Lambda_n = \max_{x \in [a,b]} \lambda_n(x)$$

(Lebesgue constant)

$$\lambda_n(x) = \sum_{j=0}^{n} |\ell_j(x)|$$

(Lebesgue function)

#

Remark: (optimal interpolation nodes) The theorem tells us that a good set of interpolation nodes $\{x_0, \cdots, x_n\}$ MINIMIZES the Lebesgue constant. For a given interval, say $[-1, 1]$, this is a $(n+1)$ dimensional optimization problem

$$\min_{(x_0, \cdots, x_n) \in [-1,1]^{n+1}} \Lambda_n(x_0, \cdots, x_n)$$

The minimization can be performed in a greedy way by adding one point at a time.

Remark (lower bound on Λ_n). No matter how well we try to optimize Λ_n, there exists a lower bound that grows logarithmically with n (SEMINAL RESULT IN APPROXIMATION THEORY)

$$\Lambda_n \geq \frac{2}{\pi} \log(n+1) + G \qquad n \to \infty$$

$\Rightarrow$ Polynomial interpolation somehow tends to diverge from the best approximating polynomial as we increase the number of nodes, no matter where they are.

Remark The Lebesgue constant can provide a guideline to understand whether certain sets of points will result in well behaved interpolating polynomials

Uniform grids (evenly-spaced) in $[-1,1]$ ($n+1$ points)

$$\frac{2^{n-2}}{n^2} \leq \Lambda_n \leq \frac{2^{n+3}}{n} \qquad \Lambda_n \sim \frac{2^{n+1}}{en(\log n + \gamma)}$$

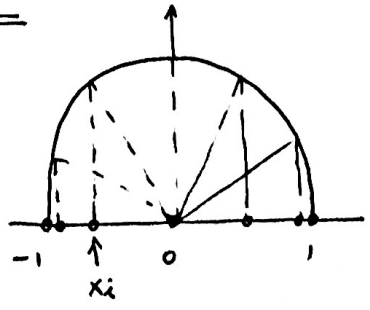
2.7182 NAPIER NUMBER

0.5477 EULER CONSTANT

$n = 65 \qquad \Lambda_n \leq 10^{18} \times 4.5$

Chebyshev - Gaun - Lobatto grids

$$x_i = \cos\left(\frac{\pi}{n} i\right) \qquad i = 0, \dots, n$$



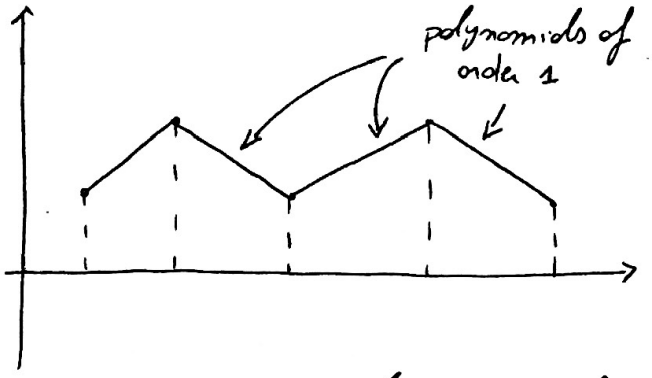
$$\Lambda_n \leq \frac{2}{\pi}\left(\log(n) + \gamma + \log\frac{8}{\pi}\right) + \frac{\pi}{72n^2}$$

$\gamma \to 0.5477$

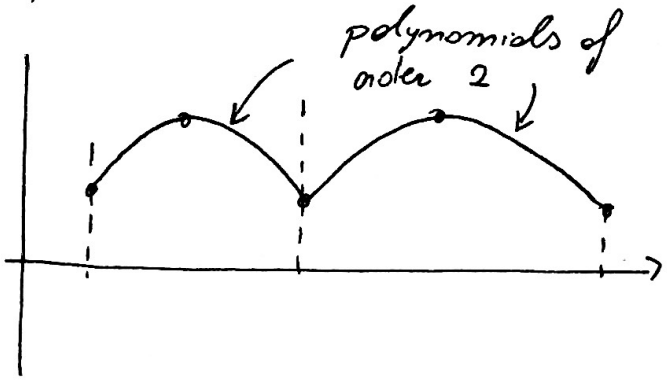
$n \simeq 65 \qquad \Lambda_n \leq 3.6$

Piecewise polynomial interpolation

- Piecewise Lagrangian interpolation



piecewise linear



piecewise quadratic

Note that in both cases we obtain a continuous function that has discontinuous derivatives at some nodes. In many applications, e.g., computer graphics, it is desirable to have polynomial approximations of data and functions which are at least differentiable in X. => SPLINES

Approximation by splines

Spline functions allow for a piecewise interpolation (OR REGRESSION) with global smoothness.

Definition Let $\{x_i\}_{i=0,\dots,n}$ $(n+1)$ distinct nodes in $[a,b]$ $\quad a = x_0 < x_1 < \cdots < x_n = b$

The function $S_K(x)$ $(x \in [a,b])$ is a SPLINE of degree K if

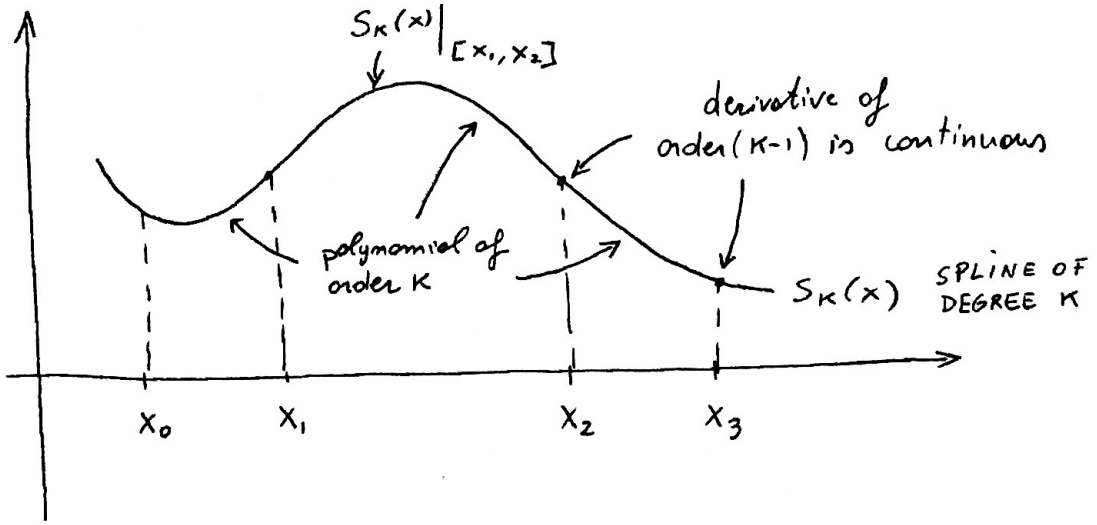
1) $S_K(x)\Big|_{[x_j, x_{j+1}]} \in \mathbb{P}_K \quad j = 0, \dots, n-1$

($S_K(x)$ restriction of $S_K(x)$ to $x \in [x_j, x_{j+1}]$; $\mathbb{P}_K$: polynomial of order K)

2) $S_K(x) \in C^{(K-1)}([a,b])$ (global smoothness)

From this definition it follows that the derivative of a spline of degree K is a spline of degree $K-1$, while the integral of a spline (primitive function) is a spline of degree $K+1$.

Remark: Piecewise lagrangian interpolants are not splines.



Remark Any polynomial of degree K is a SPLINE of degree K. For example, a global Lagrangian interpolant is a spline.

Remark (Degrees of freedom of a spline of degree K)

Denote by $S_{K,i}(x) = S_K(x)\Big|_{[x_i, x_{i+1}]}$ $i = 0, \cdots, n-1$

From condition 1) we have that $S_{K,i}(x)$ is a polynomial of order K in $x \in [x_i, x_{i+1}]$.

Therefore:

$$S_{K,i}(x) = \sum_{q=0}^{K} a_{qi} (x - x_i)^q \qquad x \in [x_i, x_{i+1}]$$

a_{qi}: polynomial coefficients $(K+1)$

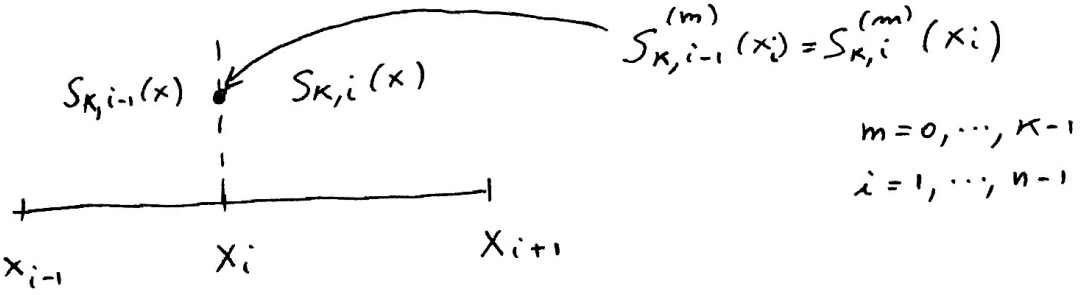
n: number of elements

$\Rightarrow$ We need $(K+1) \cdot n$ coefficients to identify the spline in $[a,b]$.

By imposing the regularity requirements

$$S_K(x) \in C^{(K-1)}([a,b])$$

at all nodes (the inner ones) $\{x_1, \cdots, x_{n-1}\}$



We obtain $(n-1)K$ conditions. Therefore we need to set

$$\underbrace{n(K+1)}_{\text{degrees of freedom}} - \overbrace{(n-1)K}^{\text{regularity requirements}} = (n+K) \quad \text{degrees of freedom.}$$

<u>Remark</u> (Interpolatory cubic splines $S_3(x)$)

In this case we have $K=3$ and $n+3$ degrees of freedom, $n+1$ of which can be set through interpolation conditions at $\{x_i, y_i\}_{i=0,\cdots,n}$. Thus we are left with only 2 degrees of freedom: We can set them as:

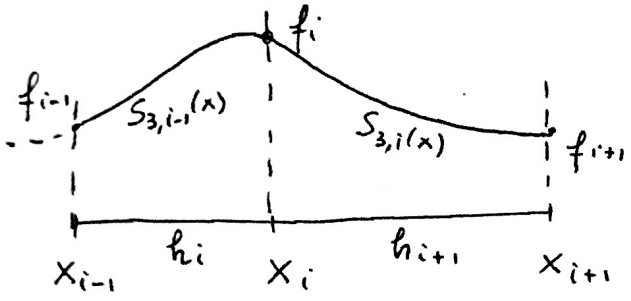
1) $S_3''(x_0) = S_3''(x_n) = 0$ (natural splines)

2) $S_3'''(x)$ is continuous at x_1 and x_{n-1} (not-a-knot splines) (IMPLEMENTED IN MATLAB AS SPLINE())

Interpolatory cubic splines.

Consider $n+1$ distinct nodes $a = x_0 < x_1 < \cdots < x_n = b$ and data $\{y_i\}_{i=0,\dots,n}$ (note: y_i could be $f(x_i)$, for some $f(x)$. In this case, we approximate $f(x)$ with an interpolatory cubic spline).

Introduce the notation



$f_i = S_3(x_i)$

$m_i = S_3'(x_i)$

$M_i = S_3''(x_i)$

Since $S_{3,i-1}(x)$ is a polynomial of order 3 we have that $S''_{3,i-1}(x)$ is LINEAR in x.

(LAGRANGIAN INTERPOLANT OF M_{i-1} and M_i at x_{i-1}, x_i)

$$S''_{3,i-1}(x) = M_{i-1}\frac{x_i - x}{h_i} + M_i\frac{x - x_{i-1}}{h_i} \qquad x \in [x_{i-1}, x_i]$$

Integrating twice with respect to x yields

$$S_{3,i-1}(x) = M_{i-1}\frac{(x_i - x)^3}{6h_i} + M_i\frac{(x - x_{i-1})^3}{6h_i} + C_{i-1}(x - x_{i-1}) + B_{i-1}$$

where $C_{i-1} = \frac{f_i - f_{i-1}}{h_i} - \frac{h_i}{6}(M_i - M_{i-1})$

$B_{i-1} = f_{i-1} - M_{i-1}\frac{h_i^2}{6}$ (INTERPOLATION CONDITIONS)

Remark: (calculation of C_{i-1} and B_{i-1}) (CONTINUITY of $S_{3,i-1}(x)$)

$S_{3,i-1}(x_{i-1}) = f_{i-1} = M_{i-1}\frac{h_i^2}{6} + B_{i-1}$

$S_{3,i-1}(x_i) = f_i = M_i\frac{h_i^2}{6} + C_{i-1}h_i + B_{i-1}$

$\Rightarrow B_{i-1} = f_{i-1} - \frac{M_{i-1}}{6}h_i^2$

$\Rightarrow C_{i-1} = \frac{f_i - f_{i-1}}{h_i} - \frac{h_i}{6}(M_i - M_{i-1})$

By imposing the continuity of the first-order derivative at the (INNER) interpolation nodes we obtain $\left(S'_{3,i-1}(x_i) = S'_{3,i}(x_i)\right)$

$$\mu_i M_{i-1} + 2M_i + \lambda_i M_{i+1} = d_i \qquad (M\text{-continuity system})$$

$i = 1, \dots, n-1$

$$\begin{cases} \mu_i = \dfrac{h_i}{h_i + h_{i+1}} \\ \lambda_i = \dfrac{h_{i+1}}{h_i + h_{i+1}} \\ d_i = \dfrac{6}{h_i + h_{i+1}}\left(\dfrac{f_{i+1} - f_i}{h_{i+1}} - \dfrac{f_i - f_{i-1}}{h_i}\right) \end{cases}$$

THESE COEFFICIENTS DEPEND ONLY ON THE GRID (h_i) AND ON THE DATA WE ARE INTERPOLATING.

The M-continuity system has $n-1$ equations and $n+1$ unknowns $\{M_0, \dots, M_n\}$.

We need to impose 2 additional conditions, for example

$$M_0 = S_3''(x_0) = 0 \qquad M_n = S_3''(x_n) = 0$$

(NATURAL SPLINES)

This yields the system:

$$\begin{bmatrix} 2 & 0 & 0 & \cdots & 0 \\ \mu_1 & 2 & \lambda_1 & \cdots & 0 \\ 0 & \mu_2 & 2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \lambda_{n-1} \\ 0 & \cdots & 0 & \mu_{n-1} & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ \vdots \\ M_n \end{bmatrix} = \begin{bmatrix} 0 \\ d_1 \\ \vdots \\ d_{n-1} \\ 0 \end{bmatrix}$$

(M-continuity system for natural splines)

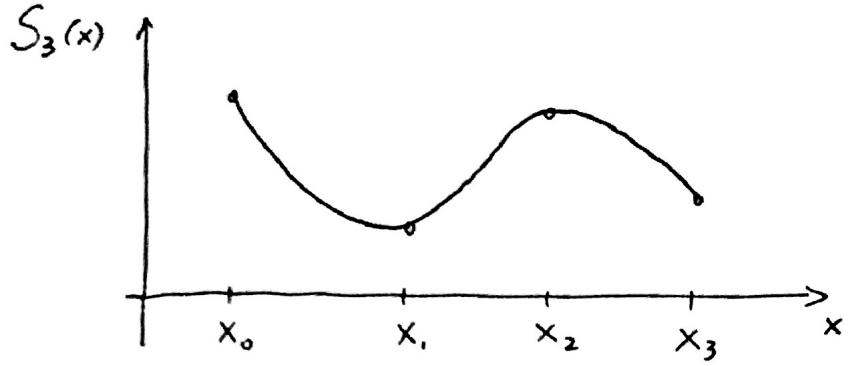
TRIDIAGONAL MATRIX (can be inverted with $O(N)$ operations)

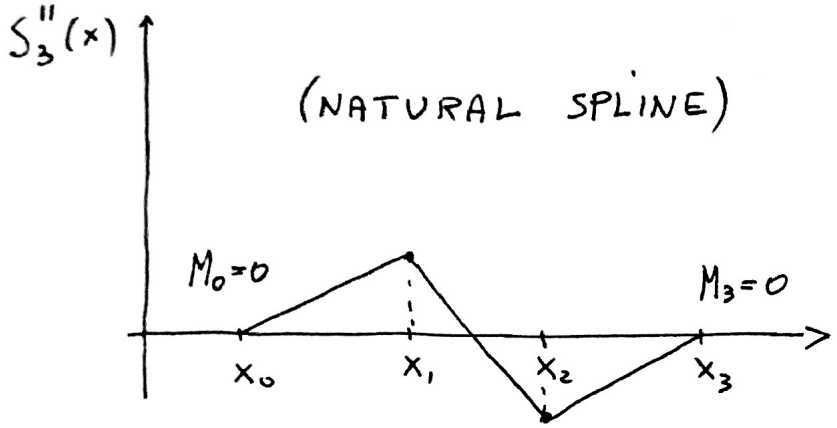
Once $\{M_0, \dots, M_n\}$ are available we can plot the third-order piecewise polynomials

$$S_{3,i-1}(x) = M_{i-1}\frac{(x_i - x)^3}{6h_i} + M_i\frac{(x - x_{i-1})^3}{6h_i} + C_{i-1}(x - x_{i-1}) + B_{i-1}$$

(restriction of the spline to the interval $\longrightarrow x \in [x_{i-1}, x_i]$)

Remark: (Natural splines vs not-a-knot splines)

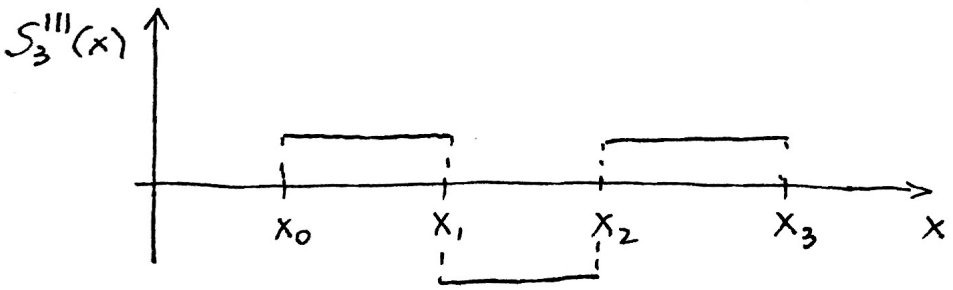




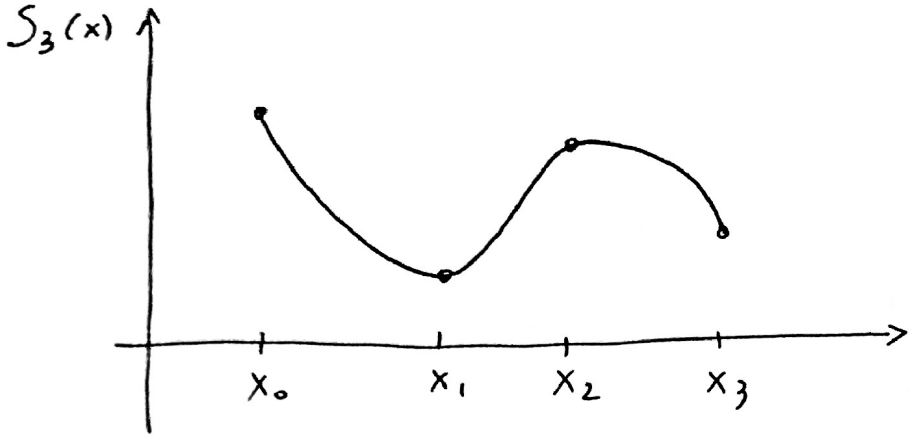
$S_3''(x)$ is PIECEWISE LINEAR in x

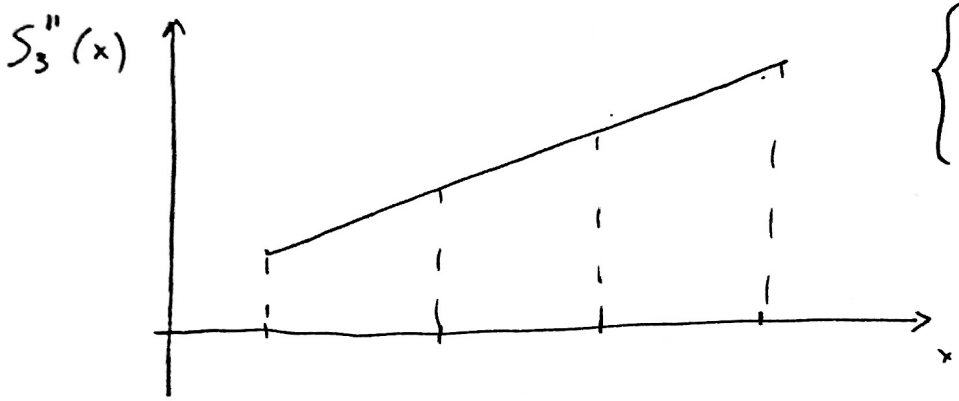
($S_3'''(x)$ is piecewise constant and discontinuous)

$\Rightarrow$ x_0 and x_3 are inflection points since $M_0 = S_3''(x_0) = 0$ and $M_3 = S_3''(x_3) = 0$ (by construction). There is another inflection point between x_1 and x_2 but such point depends on the data we are interpolating.



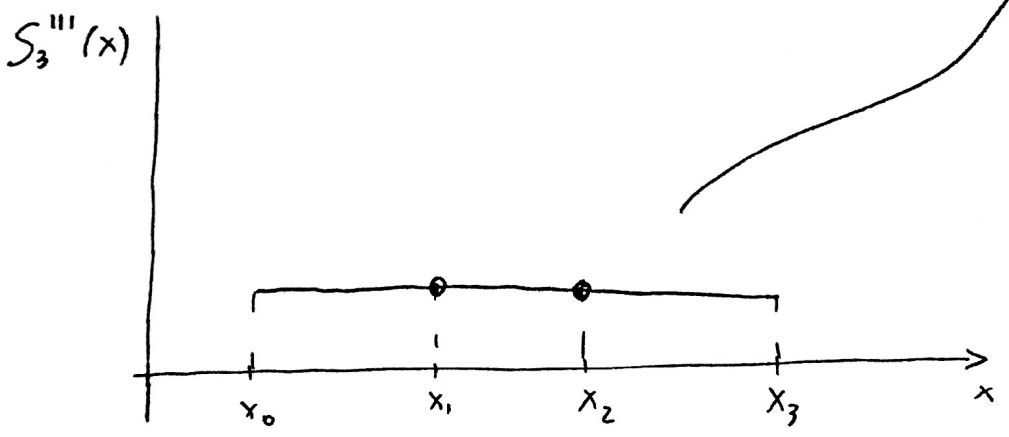
Let us see what happens in the not-a-knot case:





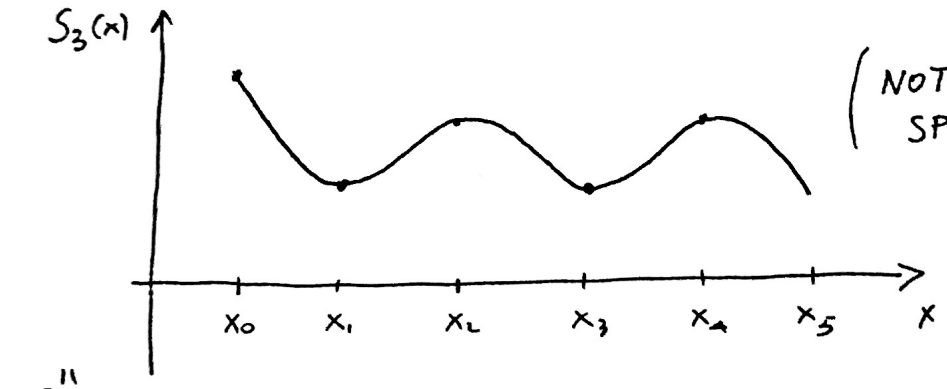
$$\begin{cases} S_{3,0}'''(x_1) = S_{3,1}'''(x_1) \\ S_{3,2}'''(x_2) = S_{3,1}'''(x_2) \end{cases}$$

(third-order derivative continuous at x_1 and x_2)

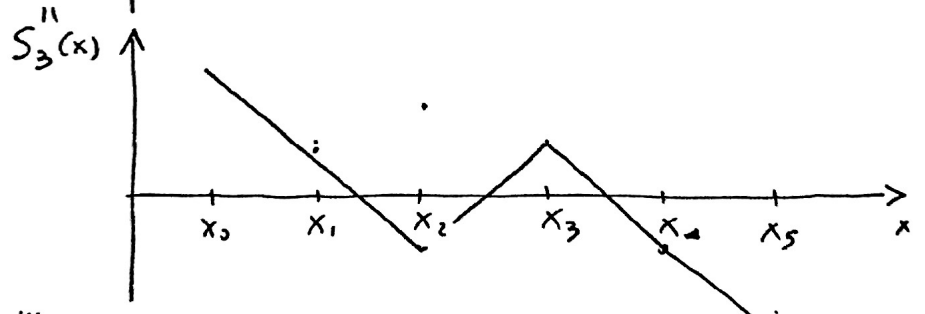


In this way, x_0 and x_3 do not contribute at all to the construction of the spline, hence the name "not-a-knot".

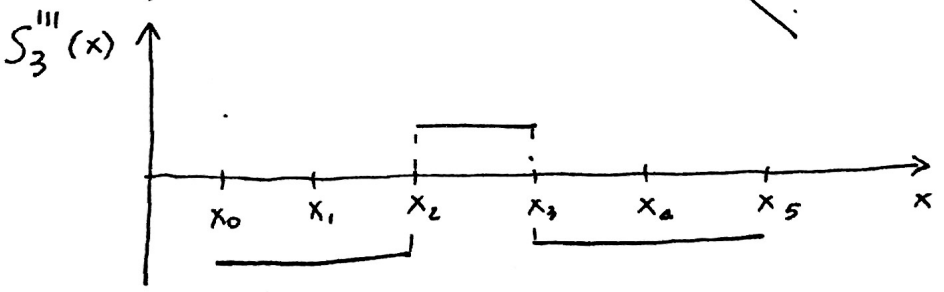
Note that in the specific example we are considering, in which we are interpolating data at 4 points, the not-a-knot spline is a GLOBAL INTERPOLATING POLYNOMIAL OF ORDER 3. In fact, the third derivative ($S_3'''(x)$) of the spline $S_3(x)$ is constant and continuous across the whole domain. Things are different with more than 4 nodes:



(NOT-A-KNOT SPLINE)

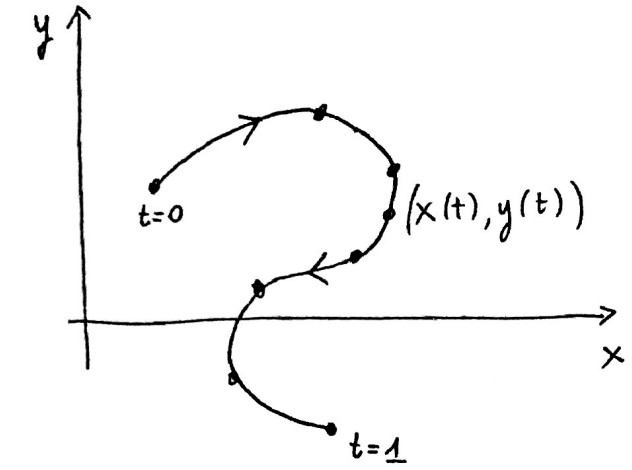


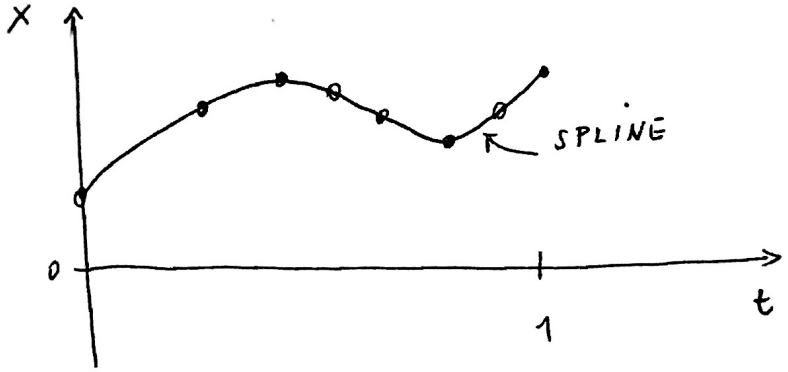
SLOPE IN $S_3''(x)$ is THE SAME IN $[x_0, x_1]$ and $[x_1, x_2]$ ($[x_3, x_4]$ and $[x_4, x_5]$)

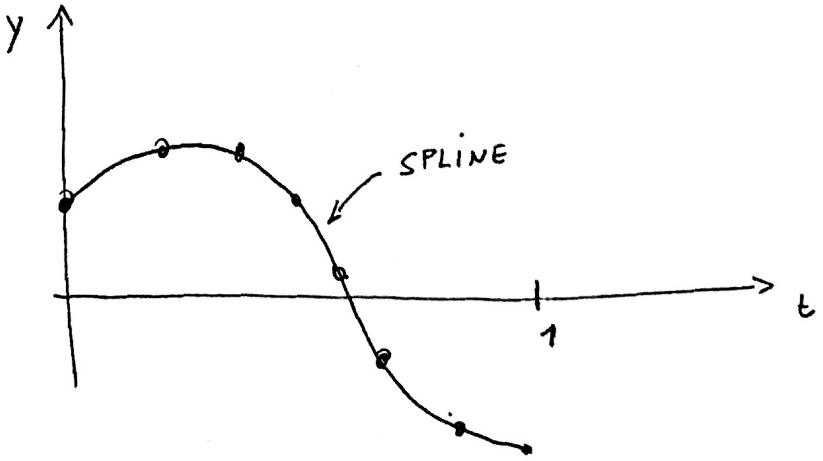


(continuity of $S_3'''(x)$ at x_4 and x_1 by construction)

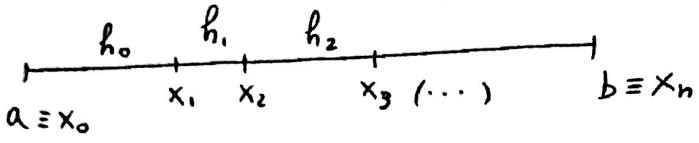
Example (using splines to represent curves in the plane)







Remark (Error estimates for interpolatory cubic splines) Let $f(x) \in C^{(4)}([a,b])$ and consider any set of nodes in $[a,b]$ (distinct nodes).



Let $h = \max\{h_i\}$ and $S_3(x)$ the interpolatory cubic spline of $f(x)$ at $\{x_0, \cdots, x_n\}$ Then:

① $\| f(x) - S_3(x) \|_\infty \leq \frac{5}{384} h^4 \| f^{(4)}(x) \|_\infty$

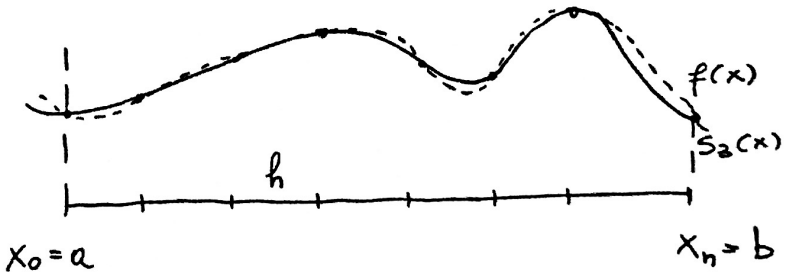
② $\| f'(x) - S_3'(x) \|_\infty \leq \frac{1}{24} h^3 \| f^{(4)}(x) \|_\infty$

③ $\| f''(x) - S_3''(x) \|_\infty \leq \frac{3}{8} h^2 \| f^{(4)}(x) \|_\infty$

This implies that convergence of cubic splines is UNIFORM as we increase the number of interpolation nodes.

(integration with cubic splines)

Remark: The error estimate (1) tells us the following: if we replace a function with an interpolatory cubic spline and we integrate we obtain a quadrature formula that converges with order 4 in the number of points. In fact, consider an evenly-spaced grid



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

$$\left| \int_a^b f(x)\,dx - \int_a^b S_3(x)\,dx \right| \leq \int_a^b |f(x) - S_3(x)|\,dx$$

$$\leq \int_a^b \max_{x\in[a,b]} |f(x) - S_3(x)|\,dx$$

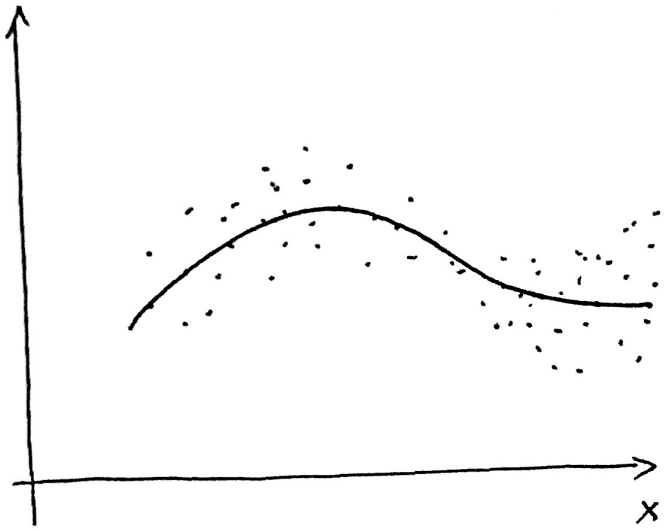
$$\leq (b-a) \cdot \frac{5}{384} \frac{(b-a)^4}{n^4} \|f''''(x)\|_\infty$$

$$\Rightarrow \left| \int_a^b f(x)dx - \int_a^b S_3(x)\,dx \right| \leq \frac{5(b-a)^5}{384\,n^4} \|f''''(x)\|_\infty$$

(IF WE DOUBLE THE NUMBER OF POINTS THE ERROR HAS AN UPPER BOUND THAT IS 16 TIMES SMALLER)

The least-squares method

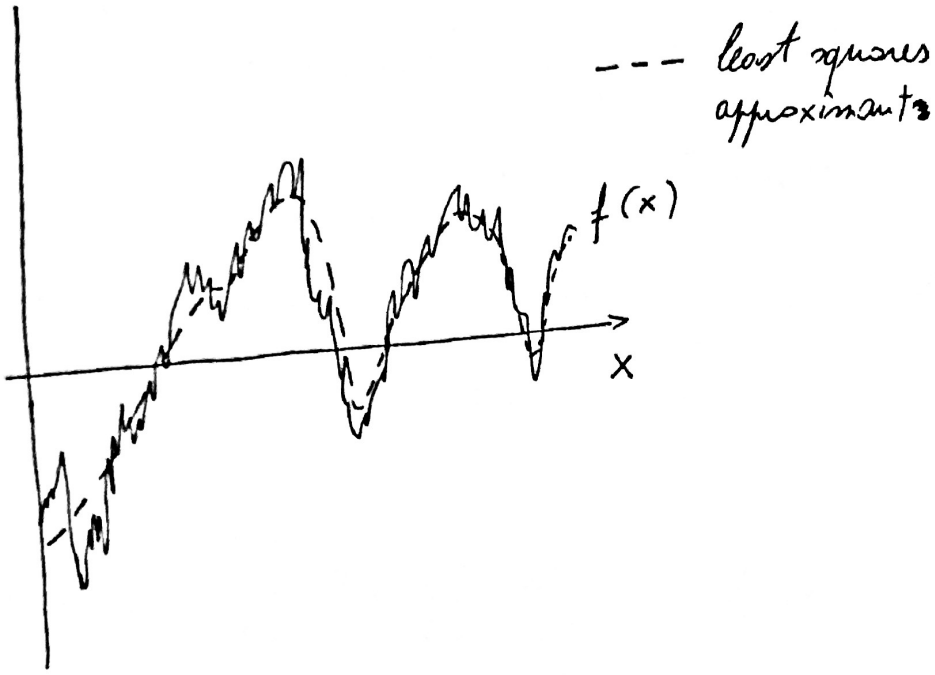
Suppose we are interested in approximating (or modeling) the following data set



LEAST SQUARES APPROXIMANT (can be a polynomial, a spline, a rational function, etc..)

Clearly, it is not a good idea to compute an interpolant in this case as the interpolating function can be highly oscillatory. It may not even possible to compute such interpolant, e.g. if we have multiple data points at the same interpolation node.

Remark (Approximation of rough functions)



The key idea of the least squares method is to compute an approximant of a data set or a function $f(x)$ by minimizing a suitable error norm, usually the L_2 norm (discrete or continuous).

Least-squares approximation of data

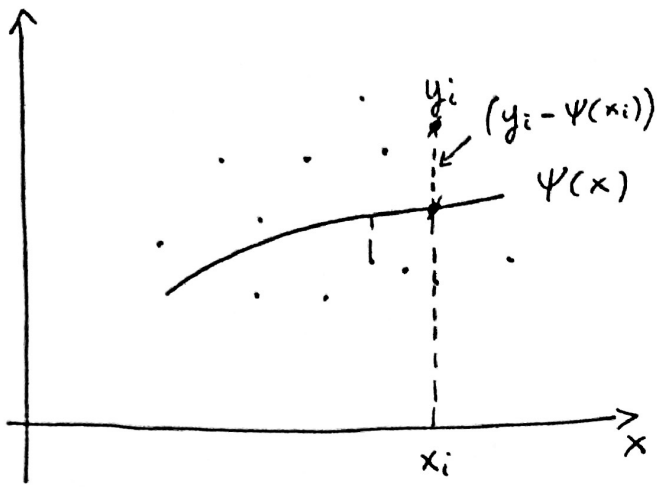
Suppose we have available $(n+1)$ data points $\{x_i, y_i\}_{i=0,\cdots,n}$, where x_i are not necessarily distinct. We look for an approximant of such data set in the form:

$$\psi(x) = \sum_{k=0}^{m} a_k \underbrace{\phi_k(x)}_{\substack{\text{known} \\ \text{basis functions}}} \qquad (m+1) \leq (n+1)$$

To determine the unknown coefficients defining $\psi(x)$, i.e., $\{a_0, \cdots, a_m\}$ we minimize the distance between $\psi(x)$ and the available data points:

$$\min_{a_0,\cdots,a_m} \underbrace{\sum_{p=0}^{n} \left(y_p - \psi(x_p)\right)^2}_{E(a_0,\cdots,a_m)} \qquad \begin{matrix}(\text{DISCRETE} \\ L_2 \text{ NORM})\end{matrix}$$

$E(a_0, \cdots, a_m)$ ERROR FUNCTION



The minimum of the error function $E(a_0, \cdots, a_m)$ can be determined by setting equal to zero the gradient, i.e.,

$$\frac{\partial E}{\partial a_K} = 0 \Rightarrow \frac{\partial}{\partial a_K} \sum_{p=0}^{n} \left(y_p - \sum_{J=0}^{m} a_J \phi_J(x_p) \right)^2 = 0$$

$$\Rightarrow 2 \sum_{p=0}^{n} \left(y_p - \sum_{J=0}^{m} a_J \phi_J(x_p) \right) \phi_K(x_p) = 0$$

$$\Rightarrow \sum_{J=0}^{m} \left(\sum_{p=0}^{n} \phi_K(x_p)\, \phi_J(x_p) \right) a_J = \sum_{p=0}^{n} y_p \phi_K(x_p) \qquad K = 0, \cdots, m$$

(LINEAR SYSTEM OF EQUATIONS IN $a_0, \cdots, a_m$)

The last system is known as system of NORMAL EQUATIONS and it can be written

in the following matrix-vector form:

$$B^T B\, a = B^T y$$

$$B = \underset{n}{\begin{bmatrix} \phi_0(x_0) & \cdots & \phi_m(x_0) \\ \vdots & & \vdots \\ \phi_0(x_n) & \cdots & \phi_m(x_n) \end{bmatrix}}^{m} \qquad a = \begin{bmatrix} a_0 \\ \vdots \\ a_m \end{bmatrix}_m \qquad y = \begin{bmatrix} y_0 \\ \vdots \\ y_n \end{bmatrix}_n$$

Remark: The matrix $B^T B$ is positive semi-definite and symmetric. If the columns of B are linearly independent $B^T B$ is INVERTIBLE. (and positive definite)

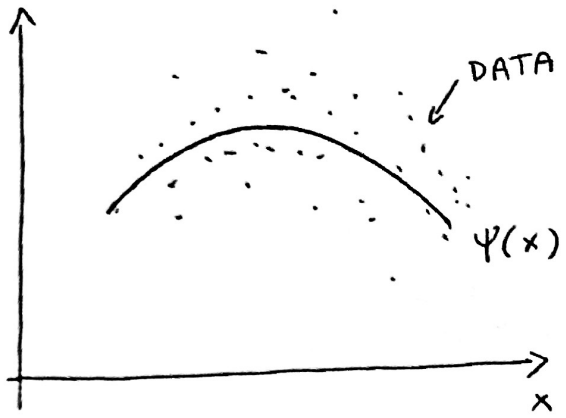
Remark: If $n = m$ and the nodes x_i are distinct then the least-squares approximant is an interpolant. (provided B is invertible)

Remark: (Example of a low-order polynomial model)

Consider $\phi_0(x) = 1$ $\phi_1(x) = x$ and $\phi_2(x) = x^2$

$$\psi(x) = \sum_{k=0}^{2} a_k \phi_k(x)$$

$$\psi(x) = a_0 + a_1 x + a_2 x^2$$



Note that if we have available only 3 data points (x_0, y_0) (x_1, y_1) and (x_2, y_2) (x_i distinct.) then

$$B = \begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix}$$

(Vandermonde matrix ($\det B \neq 0$))

$$\Rightarrow \underbrace{B^T B a = B^T y}_{\text{(normal equations for least-squares approximation)}} \overset{B \text{ invertible}}{\Longleftrightarrow} \underbrace{B a = y}_{\text{(interpolation conditions)}} \Rightarrow \psi(x) \text{ interpolates the data}$$

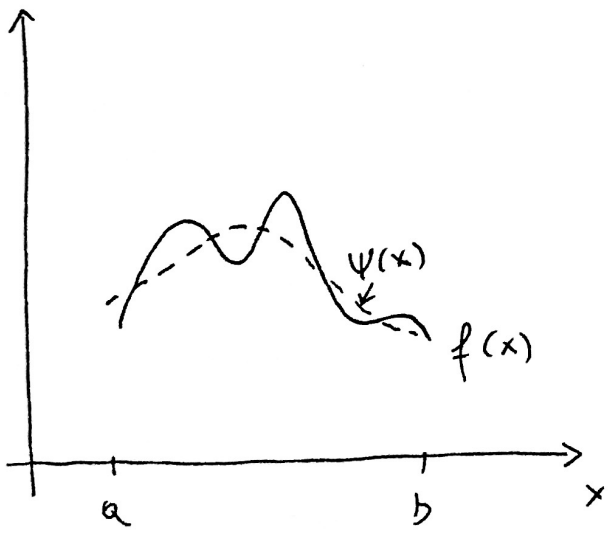
In other words, unisolvent interpolation problems can be computed by least-squares. (if B is invertible)

Least-squares approximation of functions

Consider a function $f(x)$ and an approximant of f in the form

$$\psi(x) = \sum_{k=0}^{m} a_k \phi_k(x)$$

We would like to construct $\psi(x)$ by minimizing the distance between $f(x)$ and $\psi(x)$ in the L_2 norm. To this end, suppose $f(x)$ and $\psi(x)$ are defined in $x \in [a, b]$.



$$(h(x), g(x))_{L_2} = \int_a^b f(x) g(x)\, dx$$

(inner product)

$$\| h(x) \|_{L_2}^2 = \int_a^b h(x)^2\, dx$$

Define the error norm:

$$E(a_0, \cdots, a_m) = \| f(x) - \psi(x) \|_{L_2}^2$$

$$= \| f(x) - \sum_{k=0}^{m} a_k \phi_k(x) \|_{L_2}^2$$

$$= \int_a^b \left(f(x) - \sum_{k=0}^{m} a_k \phi_k(x) \right)^2 dx$$

A necessary condition for a global minimum is:

$$\frac{\partial E}{\partial a_p} = 0 \quad p = 0, \cdots, m \quad \Rightarrow \quad \sum_{k=0}^{m} (\phi_p, \phi_k)_{L_2} a_k = (f, \phi_p) \quad p = 0, \cdots, m$$

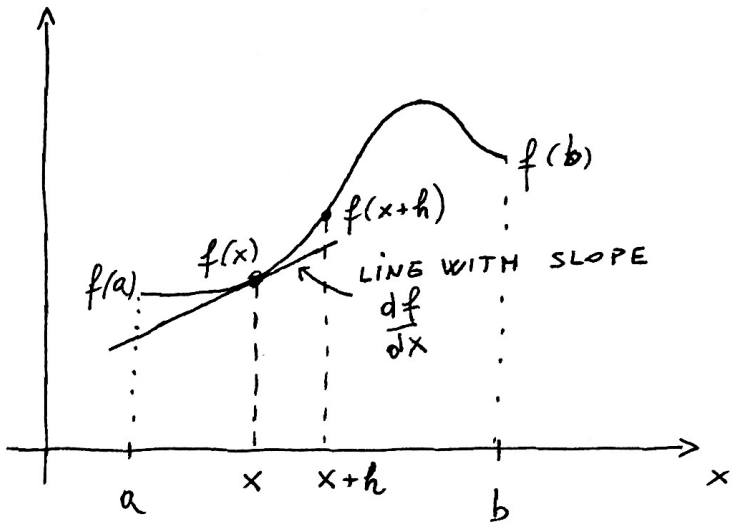
This yields:

$$\underbrace{\begin{bmatrix} (\phi_0, \phi_0)_{L_2} & \cdots & (\phi_0, \phi_m)_{L_2} \\ \vdots & & \vdots \\ (\phi_m, \phi_0)_{L_2} & \cdots & (\phi_m, \phi_m)_{L_2} \end{bmatrix}}_{\text{MASS MATRIX}} \begin{bmatrix} a_0 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} (f, \phi_0)_{L_2} \\ \vdots \\ (f, \phi_m)_{L_2} \end{bmatrix}$$

(symmetric and positive definite)

Numerical differentiation

Consider a function $f : [a,b] \to \mathbb{R}$ ($f \in C^{(1)}([a,b])$). We are interested in approximating $f'(x)$ at an arbitrary point $x \in [a,b]$.



$$f'(x) = \frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

(the limit exists since $f \in C'([a,b])$)

One can use many different methods to approximate numerically the derivative of a function.

In particular, we will study:

- Finite difference methods: They are based on a LOCAL polynomial approximation (interpolant) of $f(x)$

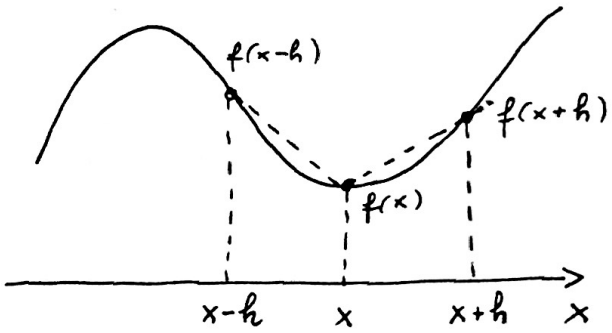
- Pseudospectral methods: They are based on a GLOBAL polynomial approximation of $f(x)$. The polynomiel is constructed by interpolating $f(x)$ at ~~three~~ Gauss points.

- Methods that use interpolating splines: (for example interpolating cubic splines)

In all these methods we replace $f(x)$ with a polynomial (either locally or globally), differentiate the polynomial, an use the derivative of the polynomial as an approximation to the derivative of $f(x)$.

Finite differences

These methods are based on replacing $f(x)$ with a piecewise Lagrangian polynomial interpolant. The simplest one is piecewise LINEAR



By differentiating the piecewise linear polynomial we see that we can define two different approximations of $f'(x)$ at x:

$$\frac{df}{dx} \simeq \frac{f(x+h)-f(x)}{h}$$

(FORWARD FIRST-ORDER FINITE DIFFERENCE FORMULA)

$$\frac{df}{dx} \simeq \frac{f(x)-f(x-h)}{h}$$

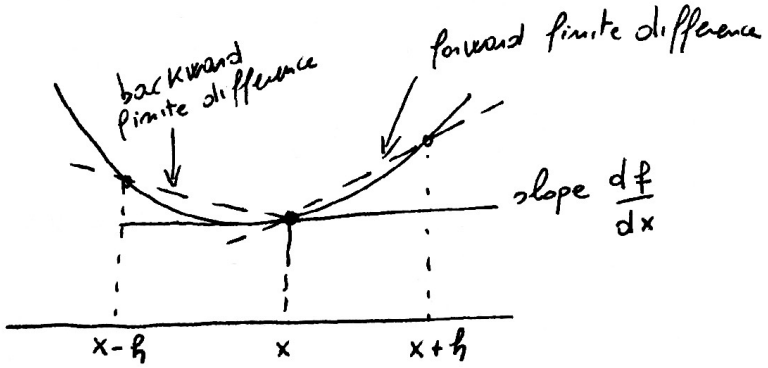
(BACKWARD FIRST-ORDER FINITE DIFFERENCE)

Why "first-order"? Because the error in the approximation of the derivative goes to zero linearly in h. To show this, suppose $f \in C^{(2)}([a,b])$. By using the mean value theorem we have:

$$f(x+h) = f(x) + h f'(x) + \frac{h^2}{2} f''(\xi) \qquad \xi \in [x, x+h]$$

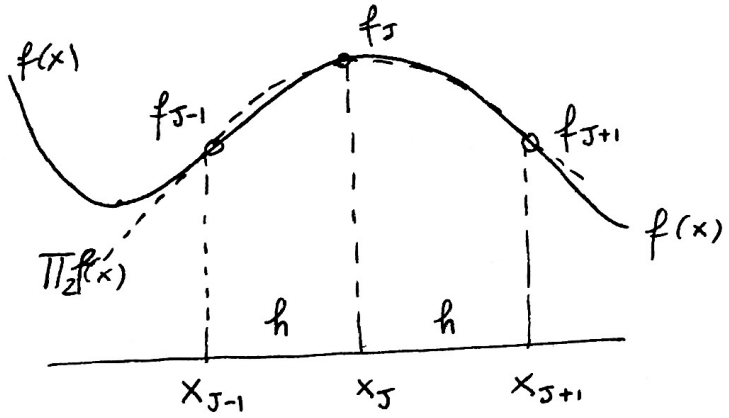
$$\Rightarrow \underbrace{\left| f'(x) - \frac{f(x+h) - f(x)}{h} \right|}_{\text{error} \xrightarrow[h \to 0]{} 0 \text{ (linearly in } h)} = \frac{h}{2} \left| f''(\xi) \right|$$

Remark: Note that forward and backward first-order finite differences are different.



Higher order finite difference methods are obtained by approximating $f(x)$ locally with a piecewise (Lagrangian) polynomial of higher order (i.e. involving more nodes). The derivatives of $f(x)$ are then approximated by differentiating such local polynomials.

Second-order finite differences



$$f_J = f(x_J)$$

$$\Pi_2 f(x) = \frac{(x-x_J)}{(x_{J-1}-x_J)}\frac{(x-x_{J+1})}{(x_{J-1}-x_{J+1})} f_{J-1} + \frac{(x-x_{J-1})}{(x_J-x_{J-1})}\frac{(x-x_{J+1})}{(x_J-x_{J+1})} f_J + \frac{(x-x_{J-1})}{(x_{J+1}-x_{J-1})}\frac{(x-x_J)}{(x_{J+1}-x_J)} f_{J+1}$$

$$= \frac{1}{2h^2}(x-x_J)(x-x_{J+1}) f_{J-1} - \frac{1}{h^2}(x-x_{J-1})(x-x_{J+1}) f_J + \frac{1}{2h^2}(x-x_{J-1})(x-x_J) f_{J+1}$$

$$\frac{d}{dx}\Pi_2 f(x) = \frac{1}{2h^2}\left(2x\, f_{J-1} - x_J f_{J-1} - x_{J+1} f_{J-1}\right) -$$
$$- \frac{1}{h^2}\left(2x\, f_J - x_{J-1} f_J - x_{J+1} f_J\right) +$$
$$+ \frac{1}{2h^2}\left(2x\, f_{J+1} - x_{J-1} f_{J+1} - x_J f_{J+1}\right)$$

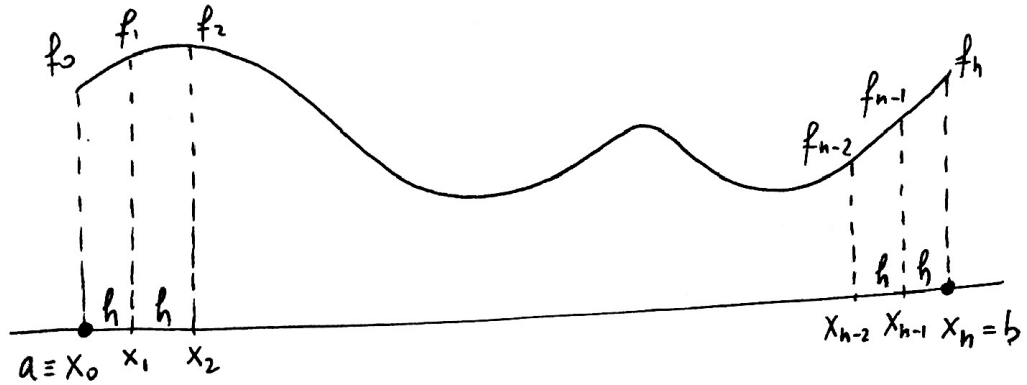
Depending on where we evaluate $\frac{d}{dx}\Pi_2 f(x)$, we have different formulae:

$$\left.\frac{d}{dx}\Pi_2 f(x)\right|_{x=x_J} = \left|\begin{array}{l} \frac{1}{2h^2} f_{J-1}\left(2x_J - x_J - x_{J-1}\right) - \\ -\frac{1}{h^2} f_J\left(2x_J - x_{J-1} - x_{J+1}\right) + \\ +\frac{1}{2h^2} f_{J+1}\left(2x_J - x_{J-1} - x_J\right) \end{array}\right.$$
$$= \frac{f_{J+1} - f_{J-1}}{2h} \quad \left(\begin{array}{l}\text{CENTERED SECOND-ORDER}\\ \text{FINITE DIFFERENCES}\end{array}\right)$$

$$\left.\frac{d}{dx}\Pi_2 f(x)\right|_{x=x_{J-1}} = \frac{1}{2h}\left(-3 f_{J-1} + 4 f_J - f_{J+1}\right) \quad \left(\begin{array}{l}\text{RIGHT-SIDED FINITE}\\ \text{DIFFERENCE}\end{array}\right)$$

$$\left.\frac{d}{dx}\Pi_2 f(x)\right|_{x=x_{J+1}} = \frac{1}{2h}\left(3 f_{J+1} - 4 f_J + f_{J-1}\right) \quad \left(\begin{array}{l}\text{LEFT-SIDED FINITE}\\ \text{DIFFERENCE}\end{array}\right)$$

Remark: The last two formulae can be used, e.g., to approximate the derivative at the endpoints of the interval (x_0 and x_n)



$$\left.\frac{df}{dx}\right|_{x_0} \simeq \frac{1}{2h}\left(-3f_0 + 4f_1 - f_2\right)$$

$$\left.\frac{df}{dx}\right|_{x_n} \simeq \frac{1}{2h}\left(3f_n - 4f_{n-1} + f_{n-2}\right)$$

We can use centered differences at all other nodes $\{x_1, \dots, x_{n-1}\}$. These formulae are second order accurate with respect to h, meaning that the approximation error in the derivative decreases quadratically with h. (mesh size)

Differentiation matrices

Consider an evenly spaced grid $x_0, \cdots, x_n$ and corresponding function values $f_0, \cdots, f_n$. the derivative of $f(x)$ at the nodes $x_0, \cdots, x_n$ can be obtained by applying a matrix to the vector $f = [f_0, \cdots, f_n]'$. The structure of such matrix depends on the particular finite difference scheme we use. For example, in the case of second-order finite differences, we have

$$\left.\frac{df}{dx}\right|_{x_0} \simeq \frac{1}{2h}(-3f_0 + 4f_1 - f_2)$$

$$\left.\frac{df}{dx}\right|_{x_i} = \frac{f_{i+1} - f_{i-1}}{2h} \qquad i = 1, \cdots, n-1$$

$$\left.\frac{df}{dx}\right|_{x_n} = \frac{1}{2h}(3f_n - 4f_{n-1} + f_{n-2})$$

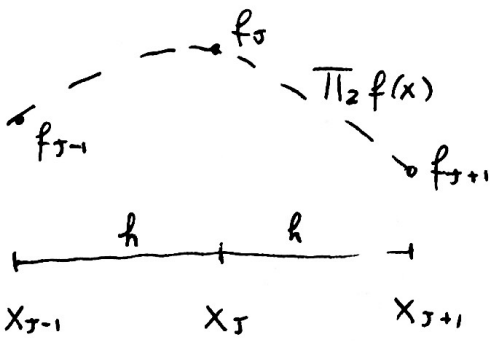
This yields the following matrix-vector representation of the derivative of $f(x)$

$$\begin{bmatrix} f'(x_0) \\ \vdots \\ f'(x_n) \end{bmatrix} \simeq \frac{1}{2h} \underbrace{\begin{bmatrix} -3 & 4 & -1 & 0 & \cdots & 0 \\ -1 & 0 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & & -1 & 0 & 1 \\ 0 & \cdots & & 1 & -4 & 3 \end{bmatrix}}_{D^{(1)}} \begin{bmatrix} f_0 \\ \vdots \\ f_n \end{bmatrix}$$

$D^{(1)}$ differentiation matrix (second-order finite differences)

$$df = D^{(1)} * f$$

Approximation of second-order derivatives (second-order finite differences)



We have seen that

$$\frac{d\Pi_2 f(x)}{dx} = \frac{1}{2h^2}(2x f_{J-1} - (x_J + x_{J+1}) f_{J-1}) -$$
$$- \frac{1}{h^2}(2x f_J - (x_{J-1} + x_{J+1}) f_J) +$$
$$+ \frac{1}{2h^2}(2x f_{J+1} - (x_J + x_{J-1}) f_{J+1})$$

$$\Rightarrow \frac{d^2 \Pi_2 f(x)}{dx^2} = \frac{f_{J-1} - 2 f_J + f_{J+1}}{h^2}$$

(at any point x_{J-1}, x_J, or x_{J+1})

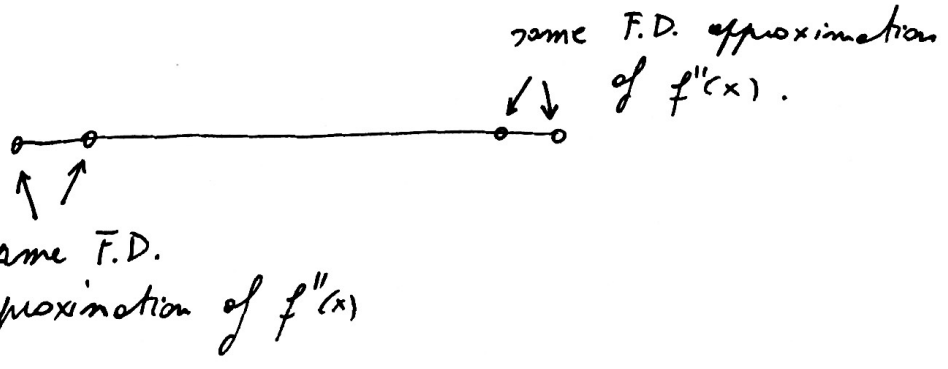
(centered finite difference approximation of $f''(x)$)

The last formula is first-order accurate in h. Note that the formula can be rearranged to yield right-sided and left sided derivatives as:

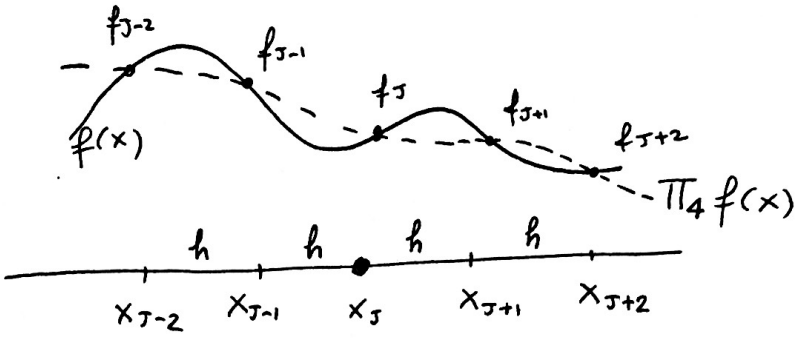
$$\frac{d^2 \Pi_2 f}{dx^2} = \frac{f_j - 2f_{j+1} + f_{j+2}}{h^2} \qquad \text{(right sided)}$$

$$\frac{d^2 \Pi_2 f}{dx^2} = \frac{f_{j-2} - 2f_{j-1} + f_j}{h^2} \qquad \text{(left-sided)}$$

Remark: The second-order derivative of the second-order finite difference approximation is the same at the end points of the interval and the points right next to them:



Fourth-order finite differences



$$\left.\frac{d\,\Pi_4 f(x)}{dx}\right|_{x=x_J} = \frac{1}{12h}\left(f_{J-2} - 8f_{J-1} + 8f_{J+1} - f_{J+2}\right)$$

(centered finite difference)

By evaluating $\frac{d\Pi_4 f(x)}{dx}$ at $x = x_{J-2}$ or $x = x_{J+2}$ we obtain other fourth-order finite difference formulae.

Pseudospectral methods

Global Lagrangian interpolants through suitable set of nodes can yield a well-behaved polynomial approximant. For example, we have seen that Lagrangian interpolation through Gauss-Chebyshev Lobatto nodes:

$$x_j = \cos\left(\frac{\pi}{N} j\right) \qquad j = 0, \cdots, N$$

yields a small Lebesgue constant Λ_N and an accurate approximant. Gauss-Chebyshev-Lobatto nodes are, by definition, the zeros of the polynomial

$$q(x) = (1-x^2)\frac{dT_N(x)}{dx} \qquad (N+1 \text{ nodes})$$

$$T_N(x) = \cos(N \arccos(x))$$

Similarly, one can define Gauss-Legendre-Lobatto nodes as zeros of the polynomial:

$$q(x) = (1-x^2)\frac{dL_N(x)}{dx} \qquad L_N(x) \text{ Legendre polynomial of degree } N$$

The interesting thing is that the Lagrange characteristic polynomials can be computed analytically for many different Gauss-Lobatto grids. For example:

$$l_j(x) = \frac{-1}{N(N+1)} \frac{(1-x^2)}{(x-x_j)} \frac{L_N'(x)}{L_N(x_j)} \qquad \text{(Gauss-Lobatto-Legendre)}$$

$$l_j(x) = \frac{(-1)^{n+j+1} (1-x^2) T_N'(x)}{d_j N^2 (x-x_j)} \qquad \text{(Gauss-Lobatto-Chebyshev)}$$

$$d_0 = d_N = 2$$
$$d_1, \cdots, d_{N-1} = 1$$

This means that we can compute the interpolant of $f(x)$

$$\Pi_N f(x) = \sum_{j=0}^{N} f(x_j) l_j(x)$$

analytically for different Gauss-Lobatto grids. The pseudospectral derivative of $f(x)$ is

defined to be the derivative of $\Pi_N f(x)$

$$\frac{d}{dx}\Pi_N f(x) = \sum_{k=0}^{N} f(x_k)\frac{dl_k(x)}{dx}$$

Evaluating this expression at the Gauss-Lobatto nodes yields:

$$\frac{d\Pi_N f(x_i)}{dx} = \sum_{j=0}^{N} f(x_j)\underbrace{\frac{dl_j(x_i)}{dx}}_{D_{ij}^{(1)}}$$

$D_{ij}^{(1)}$ is the first-order pseudospectral differentiation matrix $(dF = D^{(1)} * F)$

Example: (Gauss-Chebyshev-Lobatto nodes) - $N+1$ nodes $\{x_0, \cdots, x_N\}$

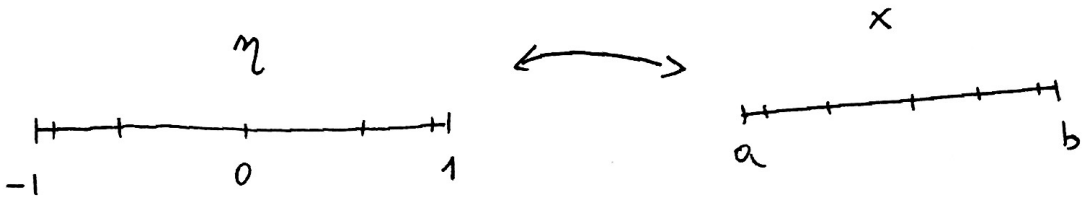
$$D_{ij}^{(1)} = \begin{cases} -\frac{2N^2+1}{6} & i=j=0 \\ \frac{d_i}{d_j}\frac{(-1)^{i+j}}{x_i - x_j} & i \neq j \\ -\frac{x_i}{2(1-x_i^2)} & i=j \text{ (not 0 nor N)} \\ \frac{2N^2+1}{6} & i=j=N \end{cases}$$

$x_j = -\cos\left(\frac{\pi}{N} j\right)$ $\quad j=0,\cdots,N$

$d_0 = d_N = 2$

$d_1, \cdots, d_{N-1} = 1$

Remark: Gauss - Chebyshev - Lobatto nodes are in $[-1,1]$. If we are interested in differentiating a function in $[a,b]$ we can consider the following map:



$$x = \frac{b-a}{2}\eta + \frac{a+b}{2}$$

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

$$f(x) = f(x(\eta)) = f\left(\frac{b-a}{2}\eta + \frac{a+b}{2}\right)$$

$$\frac{df}{d\eta} = \frac{df}{dx}\frac{dx}{d\eta} \quad \Rightarrow \quad \frac{df}{dx} = \frac{df}{d\eta}\frac{d\eta}{dx} = \frac{2}{(b-a)}\frac{df}{d\eta}$$

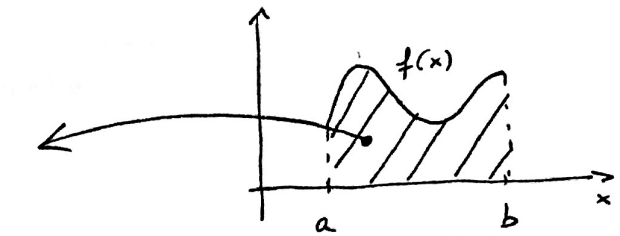
$$\frac{df}{dx} = \frac{2}{(b-a)}\underbrace{\frac{df}{d\eta}}_{\text{PSEUDOSPECTRAL}} \quad \Rightarrow \quad \underbrace{D_x^{(1)} = \frac{2}{(b-a)} D_\eta^{(1)}}$$

$D_x^{(1)}$: DIFFERENTIATION MATRIX in x; $D_\eta^{(1)}$: DIFF. MAT IN η

Numerical Integration

Let $f(x)$ be a real integrable function over the interval $[a,b]$. We are interested in computing:

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

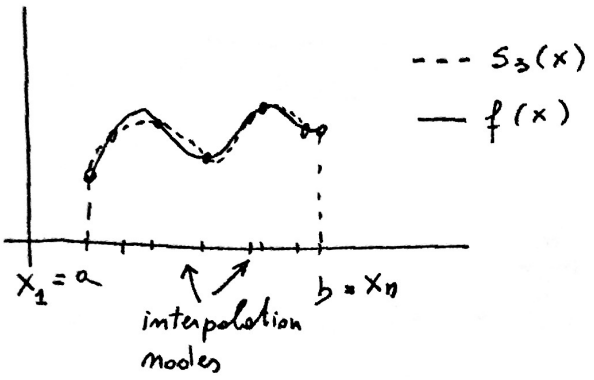


This can be done analytically only in few cases. A formula that provides an approximation of $\int_a^b f(x)\,dx$ is said to be a QUADRATURE FORMULA (or a numerical integration formula).

Perhaps, the simplest method to develop a quadrature formula is to replace $f(x)$ with an approximant that is easy to integrate, e.g., polynomials. If the approximating function is an interpolant then the quadrature formula is called

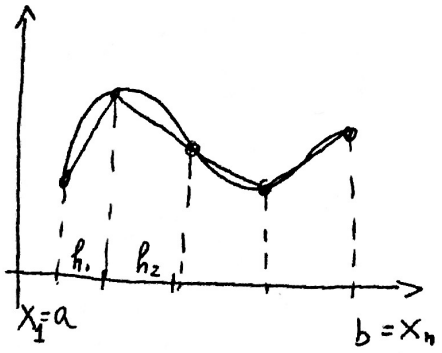
INTERPOLATORY QUADRATURE FORMULA.

Examples: ① Interpolatory cubic splines



$$\int_a^b f(x)\,dx \simeq \int_a^b S_3(x)\,dx = \sum_{j=1}^{n-1} \underbrace{\int_{x_j}^{x_{j+1}} S_3(x)\,dx}_{\text{integrals of cubic polynomials}}$$

② Trapezoidal rule (composite)

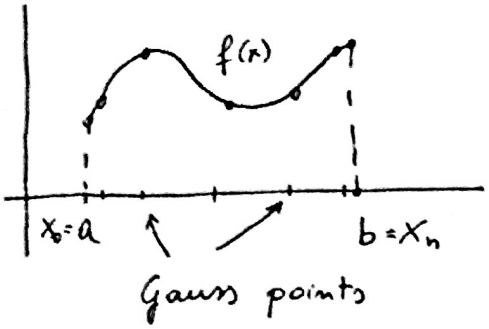


Interpolation with piecewise linear polynomials

$$\int_a^b f(x)\,dx = \sum_{j=1}^{n-1} \left(\frac{f(x_j) + f(x_{j+1})}{2}\right) h_j$$

$$h_j = (x_{j+1} - x_j)$$

③ Gaussian Quadrature

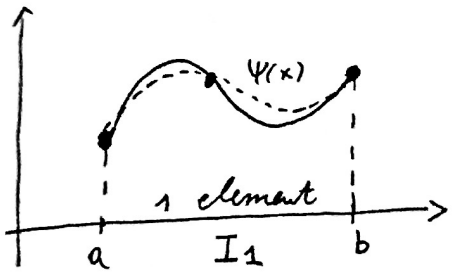


$$\int_a^b f(x)\,dx \simeq \int_a^b \Pi_n f(x) = \sum_{j=0}^{n} f(x_j) \underbrace{\int_a^b l_j(x)\,dx}_{w_j}$$

w_j integration weights

Remark: Interpolatory quadrature rules can be GLOBAL or LOCAL (composite rules).

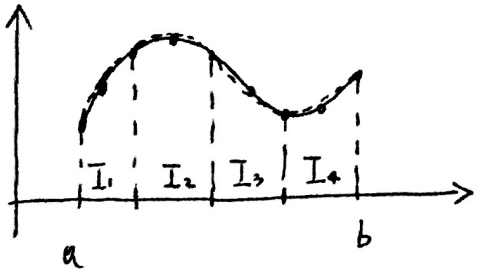
GLOBAL



$$\int_a^b f(x)\,dx \simeq \int_a^b \Psi(x)\,dx$$

↘ approximant (GLOBAL)

LOCAL (COMPOSITE)



$$\int_a^b f(x)\,dx \simeq \sum_{i=1}^{4} \int_{I_i} \Psi_i(x)\,dx$$

↘ LOCAL APPROXIMANT

There are many different types of interpolatory quadratures. Here we will study

① NEWTON-COTES FORMULAE
(they are based on polynomial interpolants constructed on evenly spaced grids)

② GAUSSIAN FORMULAE
(they are based on polynomial interpolants constructed on Gauss points)

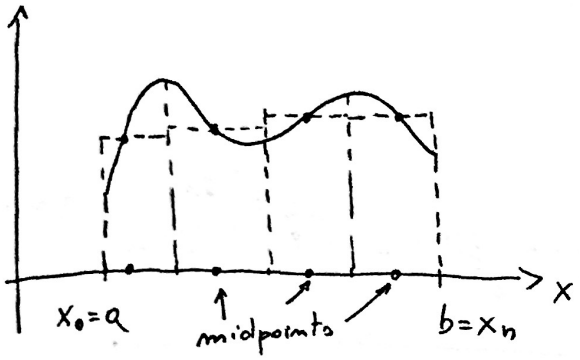
① and ② can be global or composite.

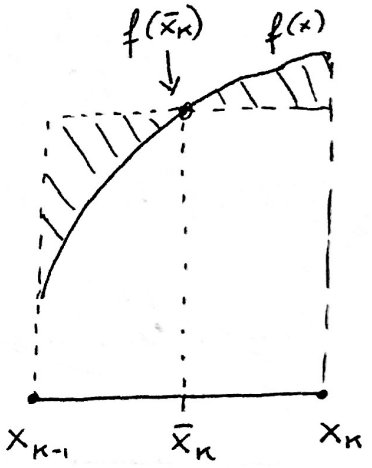
Definition: (DEGREE OF EXACTNESS) The degree of exactness of a quadrature formula is the smallest order of the polynomial that can be integrated exactly by the formula.

In other words, a quadrature formula has degree of exactness p if it can integrate exactly polynomials of degree p or less.

Composite Newton-Cotes formulae

Midpoint rule





$$\int_{x_{k-1}}^{x_k} f(x)\,dx \simeq h\, f(\bar{x}_k)$$

$$\bar{x}_k = \frac{x_k + x_{k-1}}{2}$$

$$\int_a^b f(x)\,dx = \sum_{k=1}^{m} \int_{x_{k-1}}^{x_k} f(x)\,dx$$

$$\simeq h \sum_{k=1}^{m} f(\bar{x}_k)$$

(equal weight quadrature formula)

ERROR ESTIMATE:

$$\int_{x_{k-1}}^{x_k} f(x)\,dx = \int_{\bar{x}_k - \frac{h}{2}}^{\bar{x}_k + \frac{h}{2}} f(x)\,dx$$

(Expand $f(x)$ with Taylor series around $\bar{x}_k$)

$$= h\, f(\bar{x}_k) + \underbrace{f'(\bar{x}_k) \int_{\bar{x}_k - \frac{h}{2}}^{\bar{x}_k + \frac{h}{2}} (x - \bar{x}_k)\,dx}_{=0} + \frac{1}{2} f''(\xi_k) \int_{\bar{x}_k - \frac{h}{2}}^{\bar{x}_k + \frac{h}{2}} (x - \bar{x}_k)^2\,dx$$

$\xi_k \in [x_{k-1}, x_k]$

$$= h\, f(\bar{x}_k) + \frac{h^3}{24} f''(\xi_k)$$

Therefore we obtain

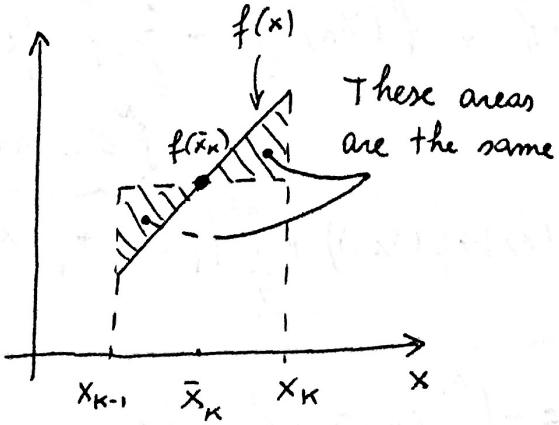
$$\sum_{k=1}^{n} \int_{x_{k-1}}^{x_k} f(x)\,dx = h \sum_{k=1}^{n} f(\bar{x}_k) + \frac{h^3}{24} \underbrace{\sum_{k=1}^{n} f''(\xi_k)}$$

it can be shown that there exists $\xi \in [a,b]$ such that

$$\sum_{k=1}^{n} f''(\xi_k) = n f''(\xi) = \frac{(b-a)}{h} f''(\xi)$$

$$\Rightarrow \boxed{\int_a^b f(x)\,dx - h \sum_{k=1}^{n} f(\bar{x}_k) = \frac{(b-a)\,h^2}{24} f''(\xi)}$$

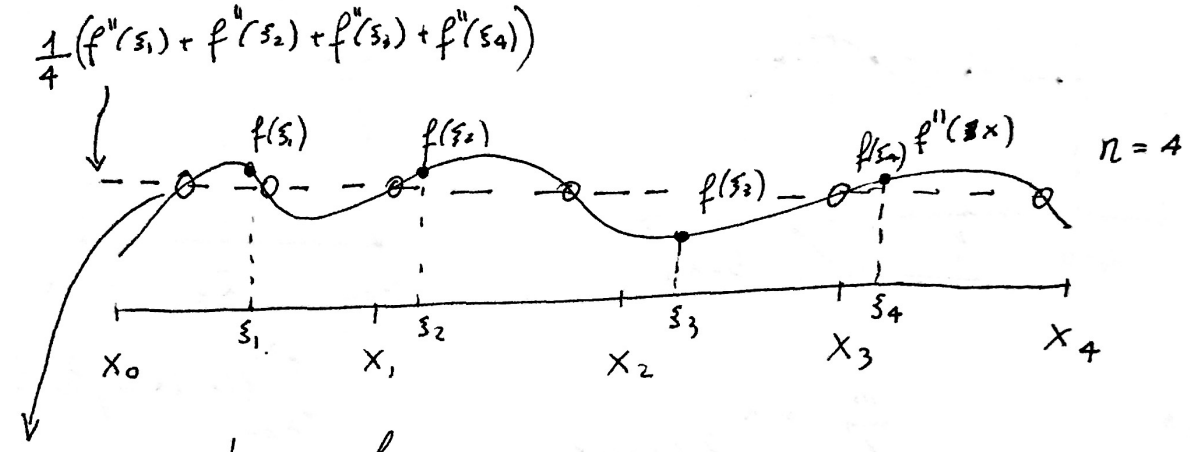
Thus, the composite midpoint rule is second-order accurate in h and it has degree of exactness 1 (i.e. it can integrate exactly polynomials of order 1). The reason is clear:



$f(x)$ LINEAR IMPLIES $f''(x) = 0$

Therefore, according to the error estimate above, the error is zero, i.e., the midpoint rule integrates exactly linear functions.

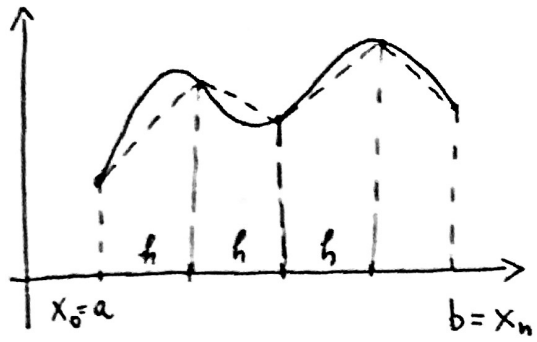
Remark: The reason why $\exists\, \xi \in [a,b]$ (not unique) such that $\frac{1}{n}\sum_{k=1}^{n} f''(\xi_k) = f''(\xi)$ is clear

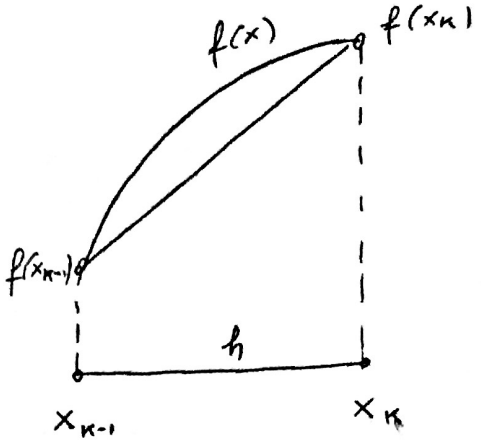


0 points where

$$\frac{1}{n}\sum_{k=1}^{n} f''(\xi_k) = f''(\xi) \qquad n = \frac{(b-a)}{h}$$

Trapezoidal rule





$$\int_a^b f(x)\,dx \simeq h \sum_{k=1}^{n} \frac{f(x_{k-1}) + f(x_k)}{2}$$

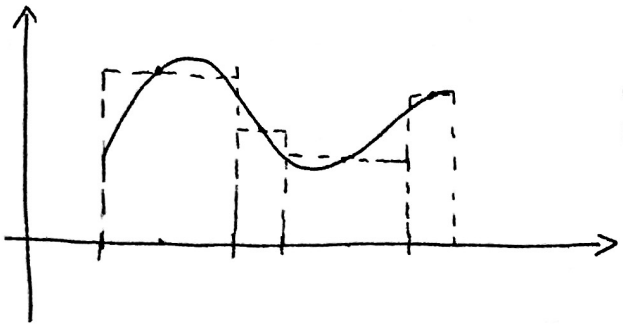
ERROR ESTIMATE:

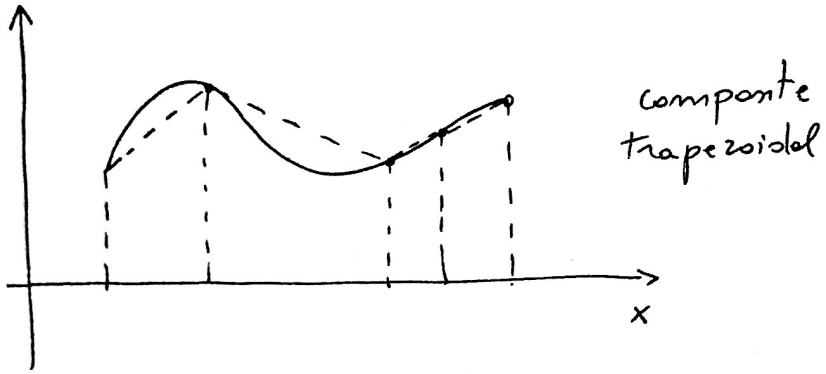
$$\int_a^b f(x)\,dx - h \sum_{k=1}^{n} \frac{f(x_{k-1}) + f(x_k)}{2} = -\frac{(b-a)}{12} h^2 f''(\xi) \qquad \xi \in [a,b]$$

Thus, the trapezoidal rule is second-order accurate and has degree of exactness 1 (Exactly as the midpoint rule). ($f(x)$ linear $\Rightarrow f''(x) = 0$)

Remark: The elements in the midpoint rule and in the trapezoidal rule do not need to be evenly spaced.

For example, we can have





We assumed that the elements were evenly spaced just because the error estimates look nicer.

(TRAPEZOIDAL RULE)

Error estimate. Set $\bar{x}_K = \frac{x_K + x_{K-1}}{2}$

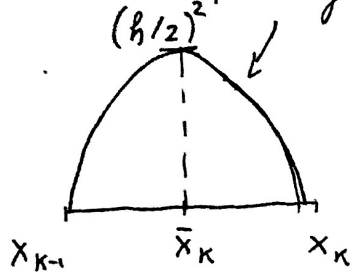
$$\int_{x_{K-1}}^{x_K} (x-\bar{x}_K) f'(x)\,dx = \frac{h}{2}\left(f(x_K)+f(x_{K-1})\right) - \int_{x_{K-1}}^{x_K} f(x)\,dx$$

This error can be written as:

$$\int_{x_{K-1}}^{x_K} (x-\bar{x}_K) f'(x)\,dx = \frac{1}{2}(x-\bar{x}_K)^2 f'(x)\Big|_{x_{K-1}}^{x_K} - \frac{1}{2}\int_{x_{K-1}}^{x_K} (x-\bar{x}_K)^2 f''(x)$$

$$= \frac{1}{2}\left(\frac{h}{2}\right)^2 \int_{x_{K-1}}^{x_K} f'(x) - \frac{1}{2}\int_{x_{K-1}}^{x_K} (x-\bar{x}_K)^2 f''(x)$$

$$= \frac{1}{2}\int_{x_{K-1}}^{x_K} \underbrace{\left[\left(\frac{h}{2}\right)^2 - (x-\bar{x}_K)^2\right]}_{\text{positive function}} f''(x)\,dx$$



Remark: if $w(x)$ is a positive function then

$$\int_{x_{K-1}}^{x_K} w(x) f''(x) = f''(\xi_K) \int_{x_{K-1}}^{x_K} w(x)\,dx$$

(ξ_K in $[x_{K-1}, x_K]$)

$$\int_{x_{K-1}}^{x_K} \overbrace{\left[\left(\frac{h}{2}\right)^2 - (x-x_K)^2\right]}^{w(x)}\,dx = \frac{h^3}{4} - \frac{2}{3}\left(\frac{h}{2}\right)^3 = \frac{h^3}{4} - \frac{h^3}{12} = \frac{h^3}{6}$$

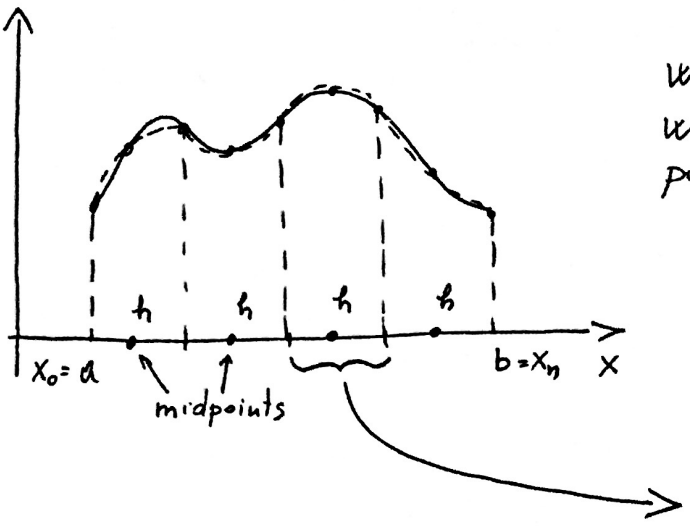
Therefore we obtain

$$\int_{x_{K-1}}^{x_K} f(x)\,dx - \frac{h}{2}\left(f(x_K)+f(x_{K-1})\right) = -\frac{h^3}{12} f''(\xi_K)$$

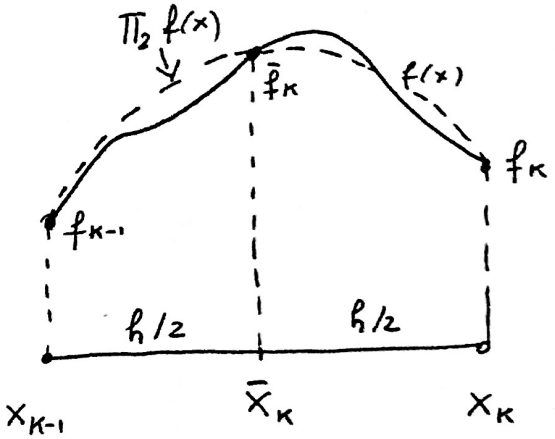
as before, $\frac{\sum_{K=1}^{n} f''(\xi_K)}{n} = f''(\xi)$

$$\Rightarrow \int_a^b f(x)\,dx - \sum_{K=1}^{n} \frac{h}{2}\left(f_K + f_{K-1}\right) = -\frac{h^2(b-a)}{12} f''(\xi)$$

Cavalieri - Simpson rule



we approximate $f(x)$ with a piecewise quadratic polynomial



$$\Pi_2 f(x) = \frac{2(x-\bar{x}_K)(x-\bar{x}_K)}{h^2} f_{K-1} + \frac{4(x_{K-1}-x)(x-x_K)}{h^2} \bar{f}_K + \frac{2(x-\bar{x}_K)(x-x_{K-1})}{h^2} f_K$$

$$\int_{\bar{x}_K - \frac{h}{2}}^{\bar{x}_K + \frac{h}{2}} \Pi_2 f(x)\,dx = \frac{h}{6}\left(f_{K-1} + 4\bar{f}_K + f_K \right)$$

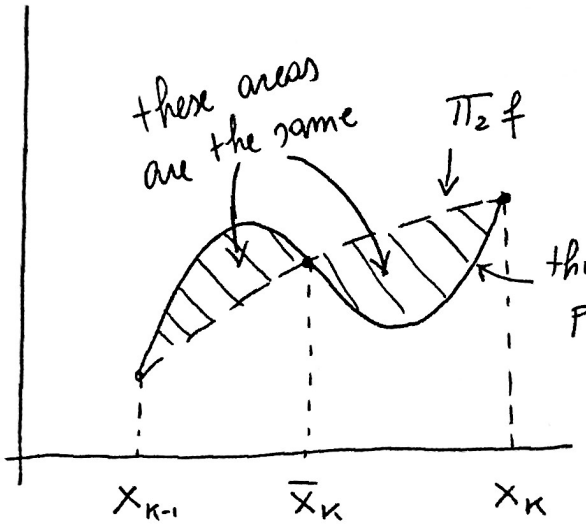
$$\Rightarrow \int_a^b f(x)\,dx \simeq \frac{h}{6} \sum_{K=1}^{n} \left(f_{K-1} + 4\bar{f}_K + f_K \right)$$

ERROR ESTIMATE

$$\int_a^b f(x)\,dx - \frac{h}{6}\sum_{k=1}^{m}\left(f_{k-1} + 4\bar{f}_k + f_k\right) = -\frac{(b-a)}{180}\,\frac{h^4}{2}\,f''''(\xi)$$

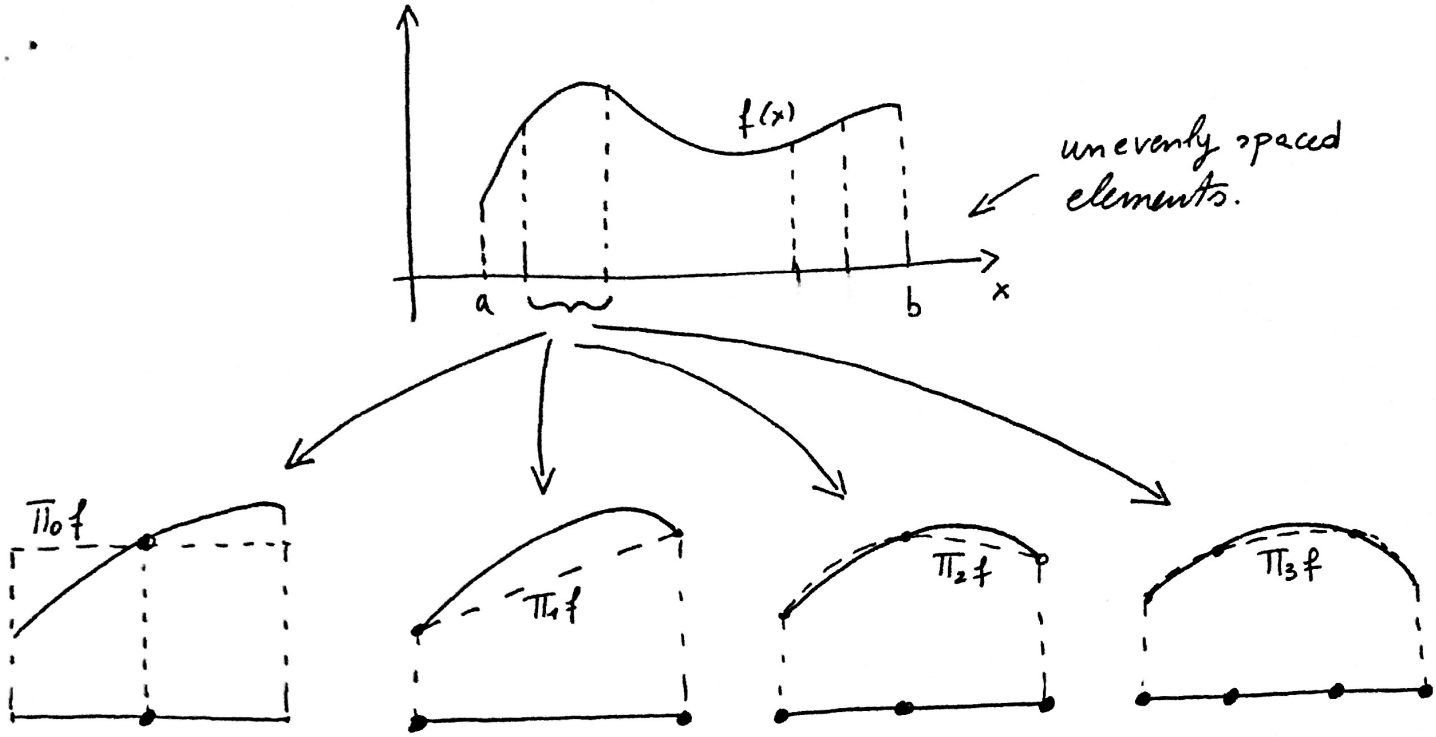
$\xi \in [a,b]$

Therefore the composite Cavalieri-Simpson rule is fourth order accurate in h. The degree of exactness is 3. The reason is the following:



$f(x)$ CUBIC implies that $f''''(x) = 0$. Therefore, based on the error estimate above, the Simpson's rule integrates exactly third order polynomials

More general Newton-Cotes formulae can be constructed by approximating $f(x)$ in each element with an polynomial interpolant constructed on an evenly-spaced grid.



$N=1$ (midpoint rule)

$N=2$ (trapezoidal rule)

$N=3$ (Simpson rule)

$N=4$

If N is odd we have degree of exactness N

If N is even we have degree of exactness $N-1$

Gaussian Quadrature

The degree of exactness of a Newton-Cotes formula with $n+1$ points $\{x_0, \cdots, x_n\}$ is

- n if n is odd (trapezoidal)
- $n+1$ if n is even (Simpson)

A natural question is whether suitable choices of nodes exist such that the degree of exactness is $n + M$ for some $M \in \mathbb{N}$, $M > 0$.
Without loss of generality we restrict our attention to nodes defined in the interval $[-1, 1]$.
Any interval $[a,b]$ can be mapped onto $[-1, 1]$ by using a linear transformation:

$$x = \frac{b-a}{2}\eta + \frac{b+a}{2} \qquad x \in [a,b], \quad \eta \in [-1,1]$$

$$\Rightarrow \int_a^b f(x)\,dx = \frac{(b-a)}{2}\int_{-1}^{1} f(x(\eta))\,d\eta$$

Theorem (Jacobi). Let $\{x_0, \dots, x_n\}$ be nodes in $[-1,1]$. For any given $M>0$ ($M \in \mathbb{N}$) the quadrature formula

$$\int_{-1}^{1} f(x)\, w(x)\, dx \simeq \sum_{k=0}^{n} f(x_k)\, w_k$$

has degree of exactness $n+M$ if and only if the polynomial $p_{n+1}(x) = \prod_{i=0}^{n} (x - x_i)$ is orthogonal to all polynomials of order $M-1$ in $L^2_w([-1,1])$ i.e., if

$$(p_{n+1}, q)_{L^2_w} = \int_{-1}^{1} p_{n+1}(x)\, q(x)\, w(x)\, dx = 0 \quad \forall q(x) \in \mathbb{P}_{M-1}$$

Remark: To determine the NODES $\{x_0, \dots, x_n\}$ we need to identify a polynomial with zeros $\{x_0, \dots, x_n\}$ that is orthogonal to all polynomials of degree $M-1$.

Example: (Gauss-Legendre quadrature). It is known that Legendre polynomials are $L_n(x)$ defined by the three-term relation

$$L_{n+1}(x) = \frac{2n+1}{n+1} \times L_n(x) - \frac{n}{n-1} L_{n-1}(x)$$

$$L_0(x) = 1$$

$$L_{-1}(x) = 0$$

and are orthogonal in $L^2([-1,1])$.

(weight function $w(x) = 1$)

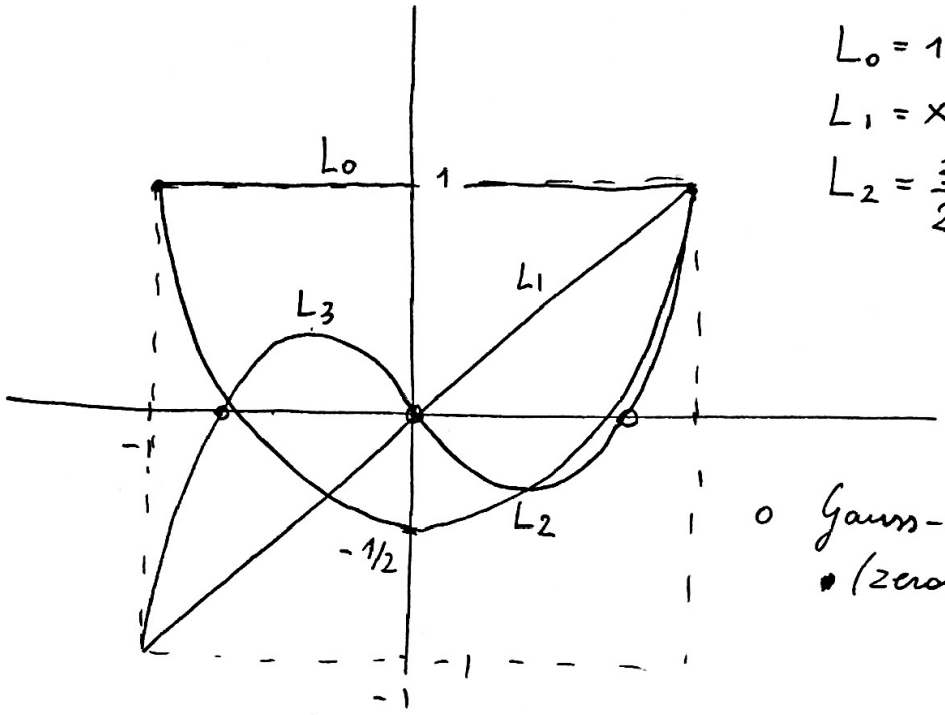
$$\int_{-1}^{1} L_n(x) L_m(x) \, dx = \delta_{nm} \frac{2}{2n+1}$$

Any polynomial of degree $M-1$ can be represented in terms of Legendre polynomials as

$$q(x) = \sum_{k=0}^{M-1} b_k L_k(x)$$

If we set $p_{n+1}(x) = L_{n+1}(x)$ then $\{x_0, \dots, x_n\}$ are ZEROS OF THE LEGENDRE POLYNOMIAL OF DEGREE $n+1$. (Recall that $p_{n+1} = \prod_{j=0}^{n} (x - x_j)$)

Remark (Legendre polynomials)



$L_0 = 1$

$L_1 = x$

$L_2 = \frac{3}{2}x^2 - \frac{1}{2}$

o Gauss-Legendre points

• (zeros of $L_3(x)$)

With such points we can integrate exactly a polynomial of order 5

By applying Jacobi's theorem we have

$$\int_{-1}^{1} L_{n+1}(x)\, q(x)\, dx = \sum_{K=0}^{M-1} b_K \int_{-1}^{1} L_{n+1}(x) L_K(x)\, dx$$

$$= 0 \iff M-1 = n \text{ (or less)}$$

$$\Rightarrow M = n+1$$

$\Rightarrow$ Gauss Legendre quadrature has degree of exactness $n + M = n + n + 1 = 2n+1$.

In other words, with $n+1$ points (ZEROS OF Legendre polynomial $L_{n+1}(x)$) we integrate exactly polynomials of degree $2n+1$

Example : $n = 5 \Rightarrow$ 6 points (zeros of $L_6(x)$)
we integrate exactly a polynomial of degree 11.

Remark : The maximum degree of exactness in quadrature formulae of interpolatory type with $n+1$ nodes is $2n+1$.
and it is achieved by Gaussian integration.

Now that we have all quadrature nodes $\{x_0, \cdots, x_n\}$, how do we construct the Gauss quadrature formula?

$$\int_{-1}^{1} f(x)\,dx \approx \sum_{k=0}^{n} f(x_k) \underbrace{\int_{-1}^{1} l_k(x)\,dx}_{w_k}$$

The Lagrange characteristic polynomials ($l_k(x)$) associated with a grid defined by zeros of Legendre polynomials is given by

$$l_k(x) = \frac{L_{n+1}(x)}{(x - x_k)\, L'_{n+1}(x_k)}$$

The integrals of such polynomials gives us the integration weights

$$w_k = \int_{-1}^{1} l_k(x)\,dx = \frac{2}{(1 - x_k^2)\, L'_{n+1}(x_k)^2}$$

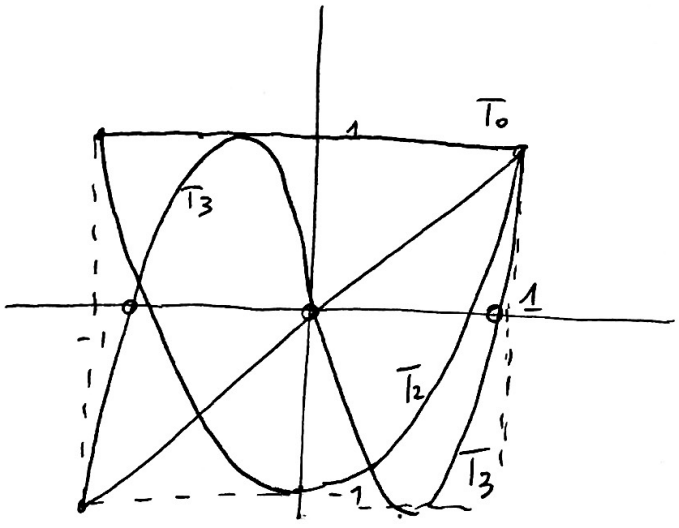
Example (Gauss - Chebyshev quadrature)

$$\int_{-1}^{1} f(x) \underbrace{\frac{1}{\sqrt{1-x^2}}}_{w(x)} dx \simeq \sum_{k=0}^{n} f(x_k) w_k$$

$$w_k = \int_{-1}^{1} l_k(x) \frac{1}{\sqrt{1-x^2}} dx$$

$\{x_0, \dots, x_n\}$ are zeros of the Chebyshev polynomial of degree $(n+1)$, $T_{n+1}(x)$.

Chebyshev polynomials are orthogonal in $L^2_w([-1,1])$ relative to the weight $w(x) = \frac{1}{\sqrt{1-x^2}}$.



$$\int_{-1}^{1} T_i(x) T_j(x) \frac{1}{\sqrt{1-x^2}} dx = \| T_i \|^2_{L^2_w} \delta_{ij}$$

$$\| T_i \|^2_{L^2_w} = \begin{cases} \pi & i = 0 \\ \frac{\pi}{2} & i > 0 \end{cases}$$

① Gauss Chebyshev points $T_{n+1}(x) = 0$ (left figure: $T_3(x) = 0$)

② Lagrange polynomials $l_k(x) = \dfrac{T_{n+1}(x)}{(x - x_k) T'_{n+1}(x)}$

③ Integration weights $w_k = \int_{-1}^{1} l_k(x) \frac{1}{\sqrt{1-x^2}} dx = \frac{\pi}{n+1}$

Gauss and Gauss-Lobatto integration (with $n+1$ nodes)

$$\int_{-1}^{1} f(x)\,w(x)\,dx \simeq \sum_{k=0}^{n} f(x_k)\,w_k \qquad w_k = \int_{-1}^{1} l_k(x)\,w(x)\,dx$$

	GAUSS (EXACTNESS $2n+1$)		GAUSS-LOBATTO (EXACTNESS $2n-1$)	
	LEGENDRE	CHEBYSHEV	LEGENDRE	CHEBYSHEV
x_k	$L_{n+1}(x)=0$	$T_{n+1}(x)=0$	$(1-x^2)\frac{dL_n(x)}{dx}=0$	$(1-x^2)\frac{dT_n(x)}{dx}=0$
$l_k(x)$	$\frac{L_{n+1}(x)}{(x-x_k)L'_{n+1}(x)}$	$\frac{T_{n+1}(x)}{(x-x_k)T'_{n+1}(x)}$	$-\frac{1}{n(n+1)}\frac{(1-x^2)}{(x-x_k)}\frac{L'_n(x)}{L_n(x_k)}$	$\frac{(-1)^{n+k+1}(1-x^2)T'_n(x)}{di\,n^2(x-x_k)}$
w_k	$\frac{2}{(1-x_k^2)L'_{n+1}(x_k)^2}$	$\frac{\pi}{n+1}$	$\frac{2}{n(n+1)L_n(x_k)^2}$	$\frac{\pi}{dim}$

$d_0 = d_n = 2$

$d_1 = \dots = d_{n-1} = 1$

Weight functions:

- Legendre $\quad w(x) = 1$
- Chebyshev $\quad w(x) = \frac{1}{\sqrt{1-x^2}}$

Gauss elimination method and the LU factorization

The Gauss elimination method with pivoting by row induces a factorization of the matrix A in the form:

$$PA = LU$$

where P is a permutation matrix, L is lower triangular and U is upper triangular. P keeps track of all the row permutations we have performed in the pivoting steps, L has all coefficients we used to perform elimination.

Example: To illustrate the LU factorization, consider the following matrix

$$A = \begin{bmatrix} 1 & 1 & 3 \\ -3 & 0 & 1 \\ 2 & 2 & -1 \end{bmatrix} \qquad \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

This vector keeps track of all permutations we do in the pivoting steps

$$\begin{bmatrix} 1 & 1 & 3 \\ -3 & 0 & 1 \\ 2 & 2 & -1 \end{bmatrix} \xrightarrow[\begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}]{\text{pivoting}} \begin{bmatrix} -3 & 0 & 1 \\ 1 & 1 & 3 \\ 2 & 2 & -1 \end{bmatrix} \xrightarrow{\text{elimination}} \begin{bmatrix} -3 & 0 & 1 \\ 0 & 1 & \frac{10}{3} \\ 0 & 2 & -\frac{1}{3} \end{bmatrix}$$

To perform elimination, we multiplied the first row by $-\frac{1}{3}$ and $-\frac{2}{3}$ and subtracted it respectively from the second and the third row.

We store $-\frac{1}{3}$ and $-\frac{2}{3}$ in the ~~zeros~~ spots where we have zeros, i.e.,

$$\begin{bmatrix} -3 & 0 & 1 \\ \begin{bmatrix} -\frac{1}{3} \\ -\frac{2}{3} \end{bmatrix} & 1 & \frac{10}{3} \\ & 2 & -\frac{1}{3} \end{bmatrix} \xrightarrow[\begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}]{\text{pivoting}} \begin{bmatrix} -3 & 0 & 1 \\ \begin{bmatrix} -\frac{2}{3} \\ -\frac{1}{3} \end{bmatrix} & 2 & -\frac{1}{3} \\ & 1 & \frac{10}{3} \end{bmatrix}$$

ELIMINATION: Multiply the ~~second~~ first row of the block matrix $\begin{bmatrix} 2 & -\frac{1}{3} \\ 1 & \frac{10}{3} \end{bmatrix}$ by $+\frac{1}{2}$ and subtract it from the third row

$$\begin{bmatrix} -3 & 0 & 1 \\ -\frac{2}{3} & 2 & -\frac{1}{3} \\ -\frac{1}{3} & \frac{1}{2} & \frac{21}{6} \end{bmatrix} \qquad \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

permutation vector

We claim that $PA = LU$ where:

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

permutation matrix corresponding to the permutation vector

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{2}{3} & 1 & 0 \\ -\frac{1}{3} & \frac{1}{2} & 1 \end{bmatrix} \qquad U = \begin{bmatrix} -3 & 0 & 1 \\ 0 & 2 & -\frac{1}{3} \\ 0 & 0 & \frac{21}{6} \end{bmatrix}$$

In fact: $$PA = \begin{bmatrix} -3 & 0 & 1 \\ 2 & 2 & -1 \\ 1 & 1 & 3 \end{bmatrix}$$

$$LU = \begin{bmatrix} -3 & 0 & 1 \\ 2 & 2 & -1 \\ 1 & 1 & \frac{18}{6} \end{bmatrix}$$

Remark (Computing determinants of matrices)

The Laplace rule requires

$\lfloor n!e \rfloor - 2$ operations $\Rightarrow$ not viable for $n > 15$

(e: napier number; $\lfloor \cdot \rfloor$: lowest integer less or equal than $n!e$)

If we use the LU factorization we have

$$PA = LU \Rightarrow \det(P)\det(A) = \underbrace{\det(L)}_{=1}\det(U)$$

$$\boxed{\det(A) = (-1)^s \prod_{i=1}^{n} u_{ii}}$$

s is the total number of permutations

The cost of computing $\det(A)$ with LU factorization is:

$$\underbrace{\frac{2}{3}n^3 - \frac{n^2}{2} - \frac{n}{6}}_{\text{LU factorization}} + \underbrace{n+1}_{(-1)^s \prod_{i=1}^{n} u_{ii}} \cong \frac{2}{3}n^3 - \frac{n^2}{2} + \frac{5}{6}n$$

(n is the size of A)

Example: The LU factorization of

$$A = \begin{bmatrix} 1 & 1 & 3 \\ -3 & 0 & 1 \\ 2 & 2 & -1 \end{bmatrix}$$ yields $$P = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

$(\Rightarrow s = 2)$
2 permutations of the rows of the identity matrix

$$U = \begin{bmatrix} -3 & 0 & 1 \\ 0 & 2 & -\frac{1}{3} \\ 0 & 0 & \frac{21}{6} \end{bmatrix}$$

$$\Rightarrow \det(A) = (-1)^2 \left(-3 \times 2 \times \frac{21}{6} \right) = -21$$

Remark (Tridiagonal systems)

The LU factorization of a tridiagonal matrix A (nonsingular)

$$A = \begin{bmatrix} a_1 & c_1 & & & 0 \\ e_2 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & e_n & a_n \end{bmatrix}$$

assume that all principal minors are nonsingular (i.e. we do not do pivoting)

yields bidiagonal matrices L and U

$$L = \begin{bmatrix} 1 & & & \\ \beta_2 & \ddots & & \\ & \ddots & \ddots & \\ & & \beta_n & 1 \end{bmatrix} \qquad U = \begin{bmatrix} \alpha_1 & c_1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & c_{n-1} \\ 0 & & & \alpha_n \end{bmatrix}$$

Example: $A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 2 & 3 \\ 0 & 1 & 4 \end{bmatrix}$. It is clear that L is bi diagonal as well as U.

The coefficients of L and U, β_2 and α_k, can be obtained by imposing $LU = A$

This yields:

$\alpha_1 = a_1 \qquad \alpha_1 \beta_2 = e_2 \qquad \beta_2 c_1 + \alpha_2 = a_2 \qquad (\dots)$

i.e.

$\alpha_1 = a_1 \qquad \beta_i = \dfrac{e_i}{\alpha_{i-1}} \qquad \alpha_i = a_i - \beta_i c_{i-1} \quad i = 2, \dots, n$

$\Rightarrow$ LU for tridiagonal matrices is computed with $O(n)$ OPERATIONS

With these coefficients available we can solve the linear system with $O(n)$ operations

To this end, let $Ax = b$ be the tridiagonal system. The LU factorization of A is computed at linear cost in n

$$LUx = b$$

Define $Ux = y$. This yields the system $Ly = b$, which can be solved for y using forward substitution at linear cost in n.

$$Ly = b \xrightarrow{\text{forward substitution}} y_1 = b_1 \qquad y_i = b_i - \beta_i y_{i-1} \quad i = 2, \cdots, n$$

$$Ux = y \xrightarrow{\text{backward substitution}} x_n = \frac{y_n}{\alpha_n} \qquad x_i = \frac{1}{\alpha_i}(y_i - c_i x_{i+1}) \quad i = n-1, \cdots, 1$$

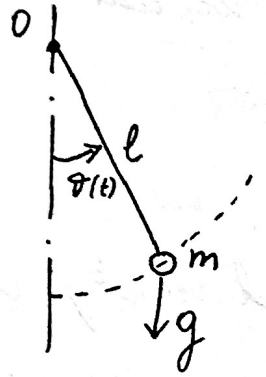
This method is known as THOMAS ALGORITHM and it allows us to compute the solution to tri-diagonal system with $O(n)$ operations instead of $O(n^3)$.

Remark: The M-continuity system for cubic interpolatory splines is tridiagonal.

Differential Equations

A differential equation is an equation involving one or more derivatives of an unknown functions

Example : (Nonlinear pendulum)



$$\frac{d^2\theta(t)}{dt^2} = -\frac{g}{l}\sin(\theta(t))$$

(nonlinear second-order ordinary differential equation)

To compute a unique $\theta(t)$ we need two additional conditions. For example, we can set $\theta(0) = \theta_0$ (initial position of the pendulum) and $\frac{d\theta(0)}{dt} = \dot{\theta}_0$ (initial velocity of the pendulum). This yields an INITIAL VALUE problem. Note that the pendulum equation

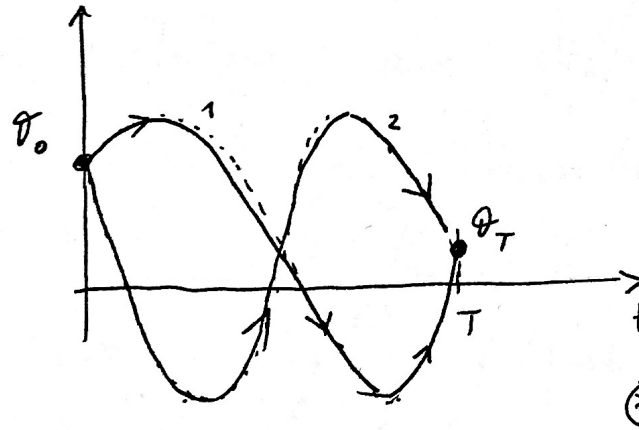
can be written as a system of first-order ordinary differential equations. To this end, simply define

$$x_1(t) = \vartheta(t)$$
$$x_2(t) = \frac{d\vartheta(t)}{dt}$$

$$\Rightarrow \begin{cases} \dfrac{dx_1(t)}{dt} = x_2(t) \\ \dfrac{dx_2(t)}{dt} = -\dfrac{g}{\ell}\sin(x_1(t)) \\ x_1(0) = \vartheta_0 \quad x_2(0) = \dot{\vartheta}_0 \end{cases}$$

Nonlinear pendulum (equations in a first-order form)

If we set $x_1(0) = \vartheta_0$ and $x_2(T) = \dot{\vartheta}_T$ then we have a TWO-POINT BOUNDARY VALUE problem. Basically we are aiming at determining which trajectory passes through the points $\vartheta(0) = \vartheta_0$ and $\vartheta(T) = \vartheta_T$. (Such trajectory may not be unique)

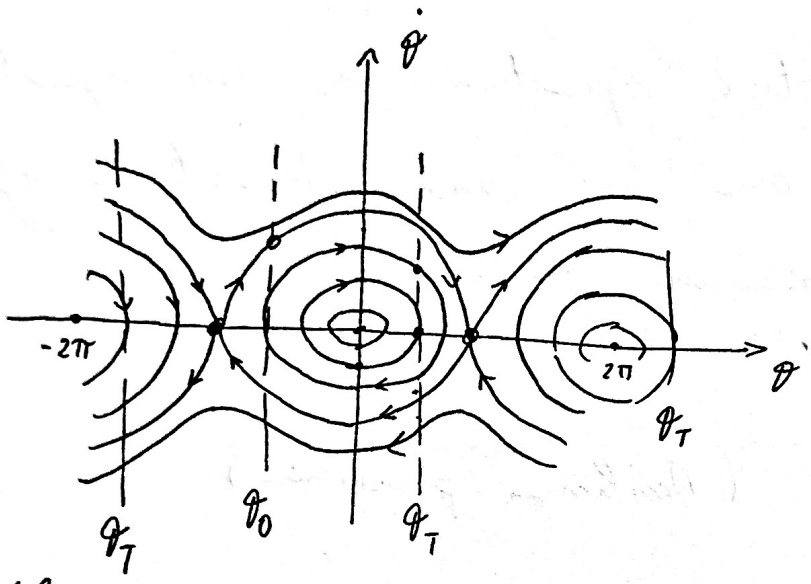


⇓

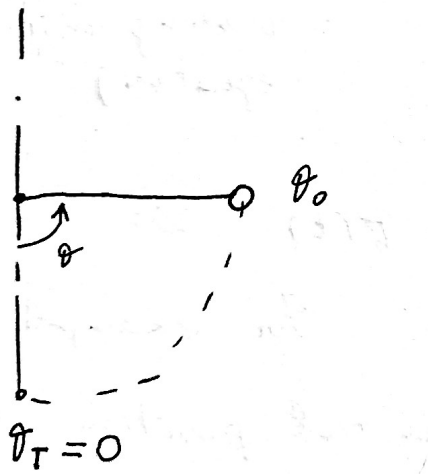
two point boundary value problems for nonlinear odes might have multiple solutions (*)[1]

(*) The multiplicity can be due to symmetry but not necessarily

(*)[1] The phase portrait of the nonlinear pendulum is



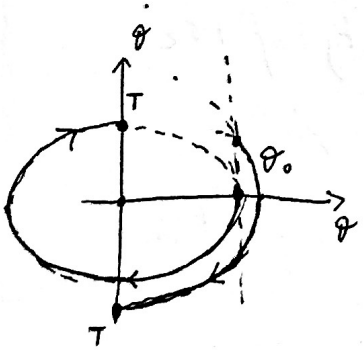
there are many ways we can go from θ_0 to $\theta_T \pm 2k\pi$

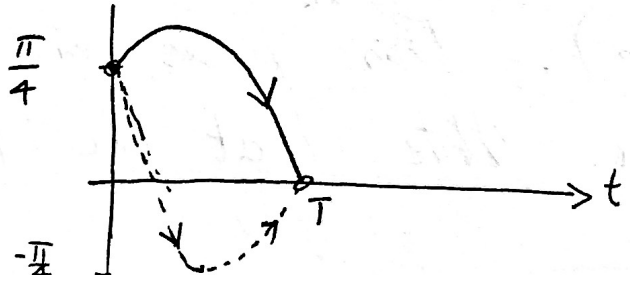


initial velocity upward

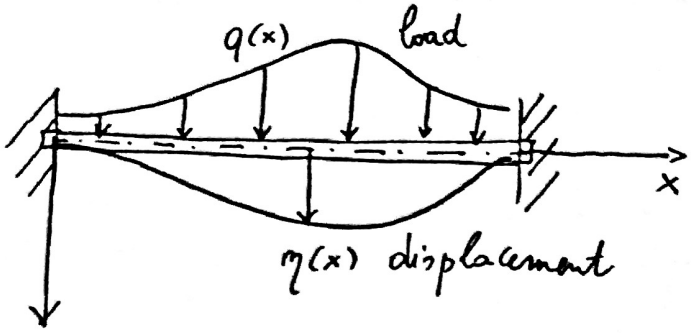
initial velocity downward (or zero)

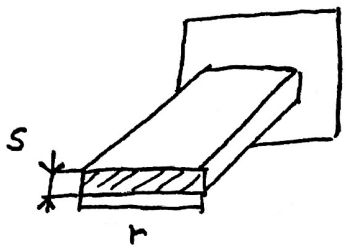
the two dynamics can pass through $\theta = 0$ at the same time





Example (Bending of a fully clamped rectangular beam)





$I = \frac{1}{12} r s^3$ (MOMENTUM OF INERTIA)

$$\begin{cases} \dfrac{d^2}{dx^2}\left(EI \dfrac{d^2\eta(x)}{dx^2} \right) = q(x) \\ \eta(0) = \eta(L) = 0 \\ \dfrac{d\eta}{dx}(0) = \dfrac{d\eta}{dx}(L) = 0 \end{cases}$$

$E \rightarrow$ modulus of elasticity of the material

Boundary value problem for a fourth-order LINEAR ordinary differential equation.

Remark: Numerical methods for initial value problems are different from numerical methods for boundary value problems. We will focus mostly on initial value problems (Cauchy problems) for systems of first-order ODEs in a normal form, i.e., systems in the form $\frac{dy}{dt} = f(y,t)$.

Initial value problem for systems of first-order nonlinear ODEs

Consider the following system of first-order ordinary differential equations in a normal form:

$$\frac{dy(t)}{dt} = f(y(t), t) \qquad \begin{array}{l} y \in \mathbb{R}^n \\ f \in \mathbb{R}^n \\ t \in [0, T] \end{array}$$

where $f(y,t)$ is $C^{(1)}$ in $y \in \mathbb{R}^n$. The system can be written in an expanded form as

$$\begin{cases} \dfrac{dy_1(t)}{dt} = f_1(y_1(t), \dots, y_m(t), t) \\ \vdots \\ \dfrac{dy_m(t)}{dt} = f_n(y_1(t), \dots, y_m(t), t) \end{cases}$$

We supplement these equations with the initial condition $y(0) = (y_1(0), \dots, y_m(0))$.

Theorem (Existence and uniqueness of the solution)

Consider the Cauchy problem

$$\begin{cases} \dfrac{dy(t)}{dt} = f(y(t), t) \\ y(0) = y_0 \end{cases}$$

where $\dfrac{\partial f_i}{\partial y_j}$ are continuous in some open set $D \subset \mathbb{R}^n$. Then for $y_0 \in D$ there exists a unique solution $y(t)$ on some time interval about $t=0$. ($y(t)$ is $C^{(1)}$ in such interval)

Remark If $\dfrac{\partial f_i}{\partial y_j}$ are continuous and bounded in $\mathbb{R}^n$ then the solution exists and it is unique for any time interval $[0, T]$.

Example:

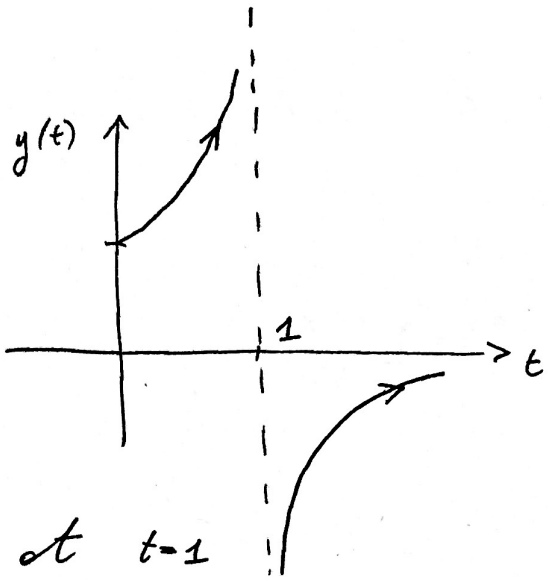
$$\begin{cases} \dfrac{dy(t)}{dt} = y(t)^2 \\ y(0) = 1 \end{cases}$$

here $f(y,t) = y^2$

$\dfrac{\partial f}{\partial y} = 2y$ which is unbounded and continuous

The solution is

$$y(t) = \frac{1}{1-t}$$



Note that $y(t)$ blows up at $t=1$

This is not surprising since the existence and uniqueness theorem guarantees that $y(t)$ is exists and is unique in some time interval ~~around~~ about $t=0$.

Example: (pendulum equations)

$$\frac{dy_1(t)}{dt} = y_2(t) \qquad y_1(0) = y_{10}$$

$$\frac{dy_2(t)}{dt} = -\sin(y_1(t)) \qquad y_2(0) = y_{20}$$

Here we have $f_1(y_1, y_2) = y_2$ $f_2(y_1, y_2) = -\sin(y_1)$

All partial derivatives are continuous and bounded. Therefore, the solution to the initial value problem exists and is unique for any finite time.

To compute the numerical solution to the Cauchy problem

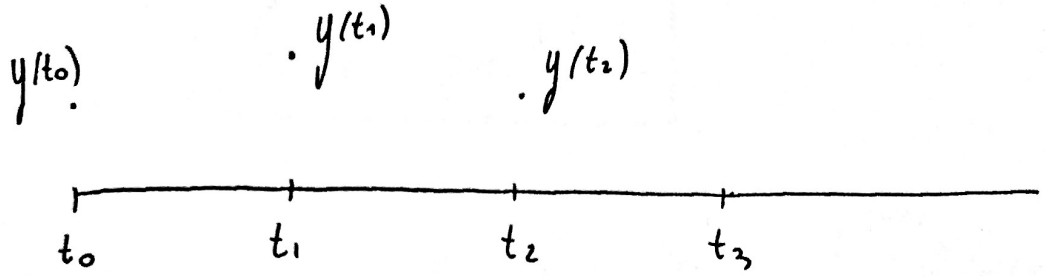
$$\begin{cases} \dfrac{dy}{dt} = f(y,t) \\ y(0) = y_0 \end{cases} \qquad y \in \mathbb{R}^n, \; f \in \mathbb{R}^n$$

it is convenient to integrate the ODE in time:

$$\int_0^t \frac{dy(\tau)}{dt} \, d\tau = \int_0^t f(y(\tau), \tau) \, d\tau$$

$$\Rightarrow \quad y(t) = y(0) + \int_0^t f(y(\tau), \tau) \, d\tau$$

More generally, suppose we have available the solution $y(t)$ at a certain number of time instants $t_0 = 0$, t_1, t_2, t_3, ... The formula above yields



$$y(t_1) = y(t_0) + \int_{t_0}^{t_1} f(y(\tau), \tau)\, d\tau$$

$$y(t_2) = y(t_1) + \int_{t_1}^{t_2} f(y(\tau), \tau)\, d\tau$$

$$\vdots$$

$$\Rightarrow \quad y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(y(\tau), \tau)\, d\tau$$

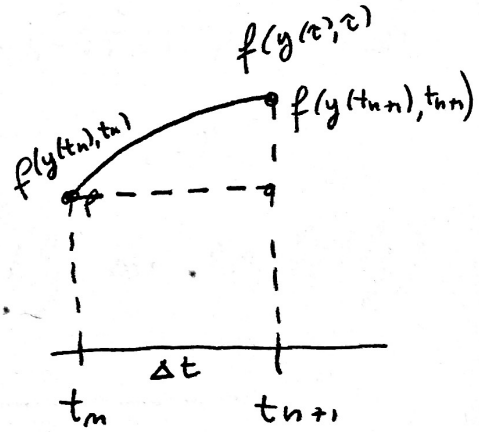
A simple way to develop a numerical scheme for the system of ODEs is therefore to approximate the integral $\int_{t_n}^{t_{n+1}} f(y(\tau), \tau)\, d\tau$ by a quadrature rule, for example the trapezoidal rule. As we will see, this yields the Crank-Nicolson method.

Euler methods

Consider $$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(y(\tau), \tau)\, d\tau$$

We approximate the integral from t_n to t_{n+1} by assuming that the integrand is constant.

We have different choices:

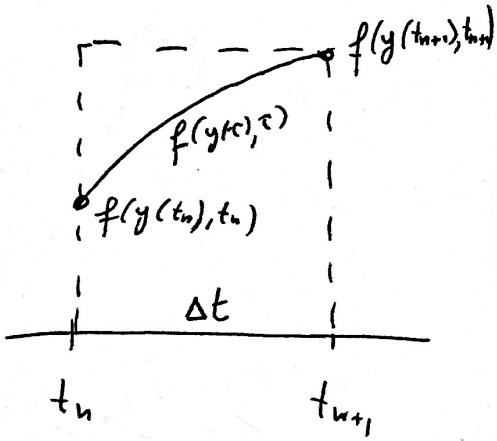


(1) $$\int_{t_n}^{t_{n+1}} f(y(\tau),\tau)\, d\tau \simeq f(y(t_n),t_n)\,\Delta t$$

this yields $$\boxed{y_{n+1} = y_n + \Delta t\, f(y_n, t_n)}$$

EULER FORWARD METHOD (EXPLICIT)

(2) $$\int_{t_n}^{t_{n+1}} f(y(\tau),\tau)\, d\tau \simeq f(y(t_{n+1}),t_{n+1})\,\Delta t$$



this yields

$$\boxed{y_{n+1} = y_n + \Delta t\, f(y_{n+1}, t_{n+1})}$$

EULER BACKWARD METHOD (IMPLICIT)

Note that in the backward Euler method we need a nonlinear solver to determine y_{n+1} from y_n. In fact, we need to solve the nonlinear algebraic equation

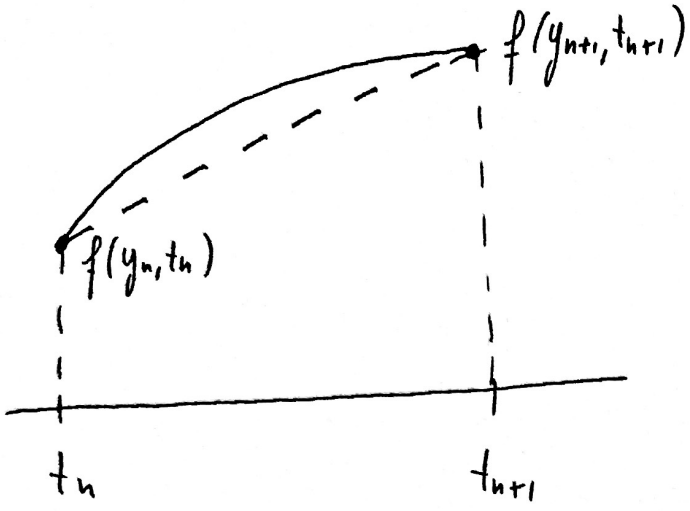
$$y_{n+1} - \Delta t\, f(y_{n+1}, t_{n+1}) - y_n = 0 \quad (\text{for } y_{n+1})$$

Remark: Euler methods are ONE-STEP methods since they require only y_n to determine y_{n+1} (implicitly or explicitly)

Crank-Nicolson and Heun methods

The Crank-Nicolson scheme can be obtained by approximating the integral $\int_{t_n}^{t_{n+1}} f(y(\tau), \tau)\, d\tau$ with the trapezoidal rule. This yields:

$$\int_{t_n}^{t_{n+1}} f(y(\tau), \tau)\, d\tau \simeq \frac{\Delta t}{2}\left(f(y_{n+1}, t_{n+1}) + f(y_n, t_n) \right)$$



This yields:

$$\boxed{y_{n+1} = y_n + \frac{\Delta t}{2}\left(f(y_{n+1}, t_{n+1}) + f(y_n, t_n)\right)}$$

Crank-Nicolson
(implicit - one step)

If we replace $f(y_{n+1}, t_{n+1})$ with ~~$f(y_n + \Delta t f(y_n, t_n))$~~
$f(\underbrace{y_n + \Delta t f(y_n, t_n)}, t_{n+1})$ we obtain the scheme:

we approximated y_{n+1} with Euler forward

$$\boxed{y_{n+1} = y_n + \frac{\Delta t}{2}\left(f(y_n + \Delta t f(y_n, t_n), t_{n+1}) + f(y_n, t_n)\right)}$$

Two-stages explicit RUNGE KUTTA ← Heun method (explicit - one step)

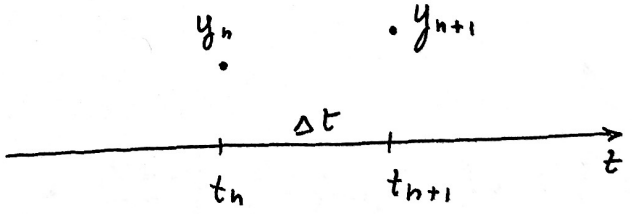
Multistep Methods

A q-step method is a method for which the solution to the ODE system

$$\frac{dy}{dt} = f(y,t)$$
$$y(0) = y_0$$

at time t_{n+1}, i.e., y_{n+1} depends on $y_n, y_{n-1}, \cdots, y_{n-q}$

Example: (1-step method)



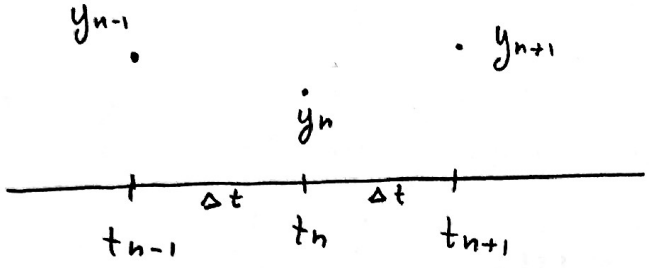
$$y_{n+1} = y_n + \frac{\Delta t}{2}\left[f(y_n, t_n) + f(y_{n+1}, t_{n+1}) \right]$$

(Crank-Nicolson)

$$y_{n+1} = y_n + \Delta t\, f(y_n, t_n)$$

(Euler-Forward)

Example (2-step method)



1) $$y_{n+1} = y_{n-1} + \int_{t_{n-1}}^{t_{n+1}} f(y(\tau),\tau)\, d\tau$$

$$= y_{n-1} + 2\Delta t\, f(y_n, t_n)$$

(explicit)

(approximation of the integral with the MIDPOINT RULE)

2) Alternatively, we can interpolate $f(y(\tau),\tau)$ with a second-order polynomial at t_{n-1}, t_n, t_{n+1}, and then integrate. This yields the Simpson quadrature rule, and the scheme:

$$y_{n+1} = y_{n-1} + \frac{\Delta t}{3}\left[f(y_{n-1}, t_{n-1}) + 4 f(y_n, t_n) + f(y_{n+1}, t_{n+1}) \right]$$

(implicit)

Adams-Bashforth Methods

Adams-Bashforth methods are explicit multistep methods to compute the solution to ODE systems in the form

$$\frac{dy}{dt} = f(y,t)$$
$$y(0) = y_0$$

The key idea is the following. Given the time instants $t_n, t_{n-1}, \dots$ and the corresponding solution vectors $y_n, y_{n-1}, \dots$ we use EXTRAPOLATION of the interpolant of $f(y(\tau),\tau)$ at $t_n, t_{n-1}, t_{n-2}, \dots$ to compute an approximation of $\int_{t_n}^{t_{n+1}} f(y(\tau),\tau)\, d\tau$.

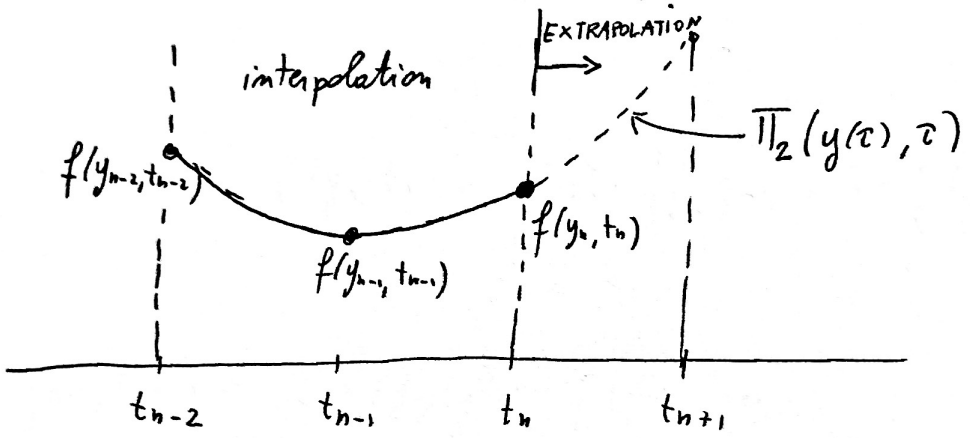
In other words, we consider

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(y(\tau),\tau)\, d\tau$$

where $$\int_{t_n}^{t_{n+1}} f(y(\tau),\tau)\,d\tau \simeq \int_{t_n}^{t_{n+1}} \Pi_q(y(\tau),\tau)\,d\tau$$

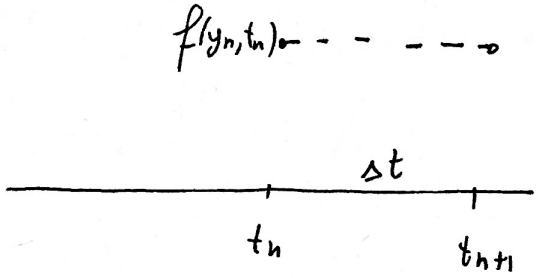
and $\Pi_q(y(\tau),\tau)$ is the polynomial interpolant

of $f(y_n,t_n), f(y_{n-1},t_{n-1}), \cdots, f(y_{n-q},t_{n-q})$

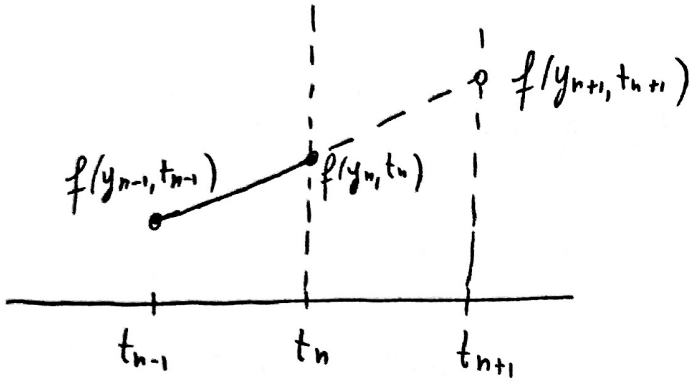


Examples: $q=0 \Rightarrow \Pi_0(y(\tau),\tau) = f(y_n,t_n)$

$y_{n+1} = y_n + \Delta t\, f(y_n,t_n)$ one-step Adams Bashforth coincides with Euler forward



Example : (two-steps Adams-Bashforth)



$$\Pi_1(y(\tau), \tau) = f(y_n, t_n) + \frac{t - t_n}{t_{n-1} - t_n}\left(f(y_{n-1}, t_{n-1}) - f(y_n, t_n)\right)$$

$$\Rightarrow \Pi_1(y^*_{n+1}, t_{n+1}) = 2f(y_n, t_n) - f(y_{n-1}, t_{n-1})$$

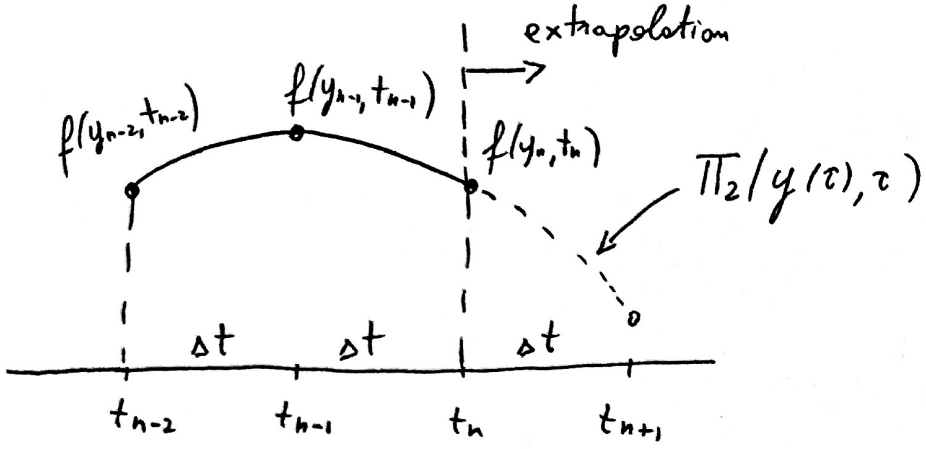
$$\Pi_1(y_n, t_n) = f(y_n, t_n)$$

$$\Rightarrow \int_{t_n}^{t_{n+1}} \Pi_1(y(\tau), \tau)\, d\tau = \frac{\Delta t}{2}\left(3f(y_n, t_n) - f(y_{n-1}, t_{n-1})\right)$$

This yields the (second-order) two-steps Adams-Bashforth method

$$y_{n+1} = y_n + \frac{\Delta t}{2}\left(3f(y_n, t_n) - f(y_{n-1}, t_{n-1})\right)$$

Example: (three steps Adams-Bashforth method)

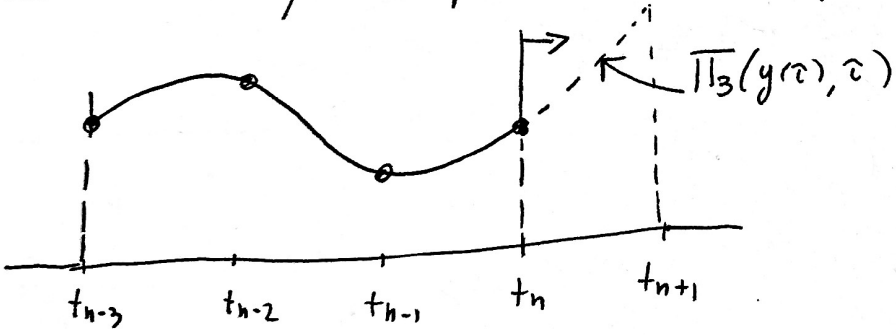


$$\int_{t_n}^{t_{n+1}} \Pi_2(y(\tau), \tau)\, d\tau = \frac{\Delta t}{12}\left(23 f(y_n, t_n) - 16 f(y_{n-1}, t_{n-1}) + 5 f(y_{n-2}, t_{n-2})\right)$$

$$\Rightarrow y_{n+1} = y_n + \frac{\Delta t}{12}\left(23 f(y_n, t_n) - 16 f(y_{n-1}, t_{n-1}) + 5 f(y_{n-2}, t_{n-2})\right)$$

(third-order explicit)

Example (four-steps Adams-Bashforth method)

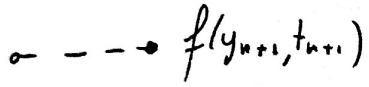


$$y_{n+1} = y_n + \frac{\Delta t}{24}\left(55 f(y_n, t_n) - 59 f(y_{n-1}, t_{n-1}) + 37 f(y_{n-2}, t_{n-2}) - 9 f(y_{n-3}, t_{n-3})\right)$$

Adams-Moulton Methods

These are implicit multistep methods in which the integral from t_n to t_{n+1} is approximated by replacing $f(y(\tau),\tau)$ with the interpolating polynomial at $y_{n+1}, y_n, y_{n-1}, \ldots$

Ex

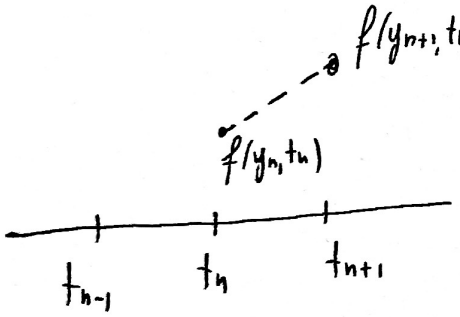


a)

$$y_{n+1} = y_n + \Delta t\, f(y_{n+1}, t_{n+1})$$

(Backward Euler)

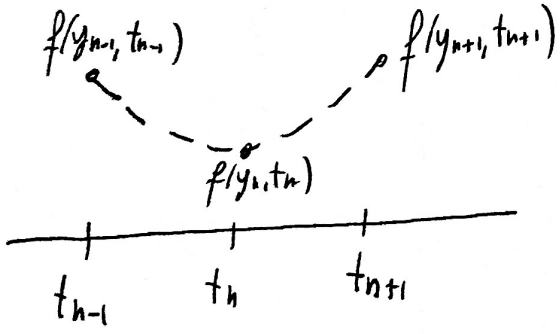
b)



$$y_{n+1} = y_n + \frac{\Delta t}{2}\left(f(y_{n+1}, t_{n+1}) + f(y_n, t_n)\right)$$

(Crank-Nicolson)

c)



$$y_{n+1} = y_n + \frac{\Delta t}{12}\left(5f(y_{n+1}, t_{n+1}) + 8f(y_n, t_n) - f(y_{n-1}, t_{n-1})\right)$$

(third-order Adams-Moulton)

Analysis of one-step explicit methods

Any one-step explicit method for the numerical approximation of

$$\begin{cases} \dfrac{dy(t)}{dt} = f(y(t), t) \\ y(0) = y_0 \end{cases}$$

can be written in the following general form:

$$y_{n+1} = y_n + \Delta t \underbrace{\Phi(y_n, f(y_n, t_n), t_n, \Delta t)}_{\text{increment function}}$$

Examples:

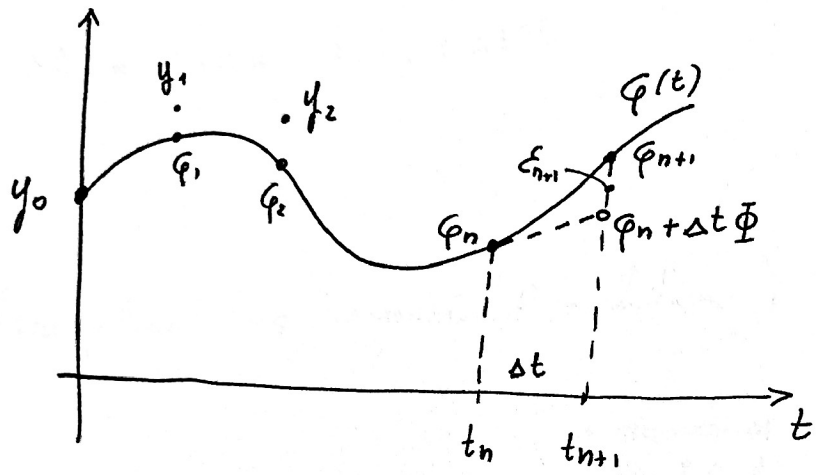
1) $\Phi = f(y_n, t_n)$ (Euler forward method)

2) $\Phi = \dfrac{f(y_n, t_n) + f(y_n + \Delta t f(y_n, t_n), t_{n+1}^{t_n + \Delta t})}{2}$

(Heun method)

Remark (LOCAL TRUNCATION ERROR). Let $\varphi(t)$ be the exact solution to $\frac{dy(t)}{dt} = f(y(t),t)$

$y(0) = y_0$



If we substitute $\varphi_n = \varphi(t_n)$ into the one-step method we obtain

$$\varphi_{n+1} = \varphi_n + \Delta t\,\Phi + \varepsilon_{n+1}$$

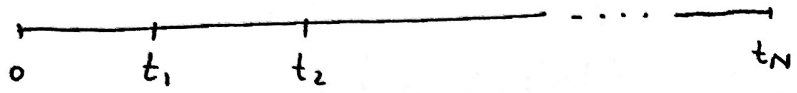
We define the <u>local truncation error</u> $\tau_{n+1}(\Delta t)$ as $\tau_{n+1}(\Delta t) = \frac{\varepsilon_{n+1}}{\Delta t}$

$$\Rightarrow \varphi_{n+1} = \varphi_n + \Delta t\,(\Phi + \tau_{n+1}(\Delta t))$$

$$\left(\frac{\varphi_{n+1} - \varphi_n}{\Delta t} = \Phi(\varphi_n, f(\varphi_n, t_n), t_n, \Delta t) + \tau_{n+1}(\Delta t) \right)$$

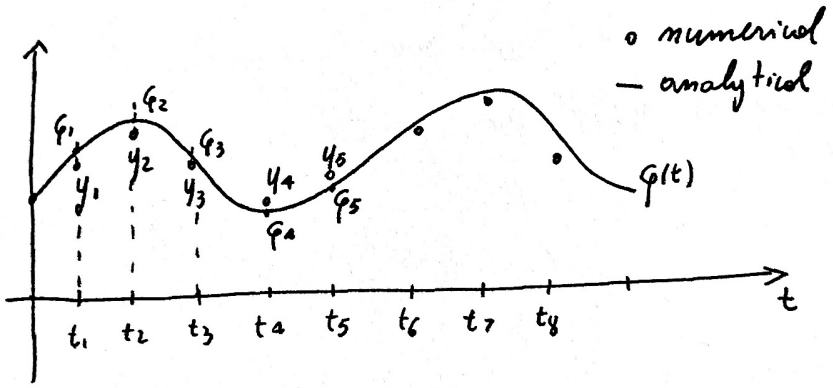
↑ this depends on φ, etc.. as well

Remark (GLOBAL TRUNCATION ERROR)



$$\tau(\Delta t) = \max_{i=1,\dots,N} |\tau_i(\Delta t)|$$

Remark : (Analytical vs numerical solution)



Remark (Consistent and Convergent schemes)

A numerical scheme is said to be

CONSISTENT if

$$\lim_{\Delta t \to 0} \tau(\Delta t) = 0$$

If $\tau(\Delta t)$ goes to zero as $(\Delta t)^p$ then the scheme is said to be consistent with order p.

A numerical scheme is said to be convergent if

$$|y_i - \varphi_i| \leq C_i(\Delta t) \qquad \lim_{\Delta t \to 0} C_i(\Delta t) = 0$$

$$i = 1, \dots, N$$

If $C_i(\Delta t) = O(\Delta t^p)$ then we say that the method converges with order p.

Remark: Euler forward method converges with order 1. Heun method converges with order 2.

Convergence analysis of the Euler forward method

The local truncation error of the Euler forward method is:

$$\tau_{n+1}(\Delta t) = \frac{\varphi_{n+1} - \varphi_n - \Delta t\, f(\varphi_n, t)}{\Delta t} \qquad \varphi_{n+1} = \varphi(t_{n+1}),\ \varphi_n = \varphi(t_n)$$

$$= \frac{\varphi_{n+1} - \varphi_n}{\Delta t} - f(\varphi_n, t)$$

By using the Taylor series expansion of φ_{n+1}

$$\varphi_{n+1} = \varphi_n + \Delta t\, \varphi'(t_n) + \frac{\Delta t^2}{2} \varphi''(\xi_n) \qquad \xi_n \in [t_n, t_n + \Delta t]$$

$$\Rightarrow \frac{\varphi_{n+1} - \varphi_n}{\Delta t} = \underbrace{\varphi'(t_n)}_{f(\varphi_n, t_n)} + \frac{\Delta t}{2} \varphi''(\xi_n)$$

Therefore the local truncation error can be expressed as

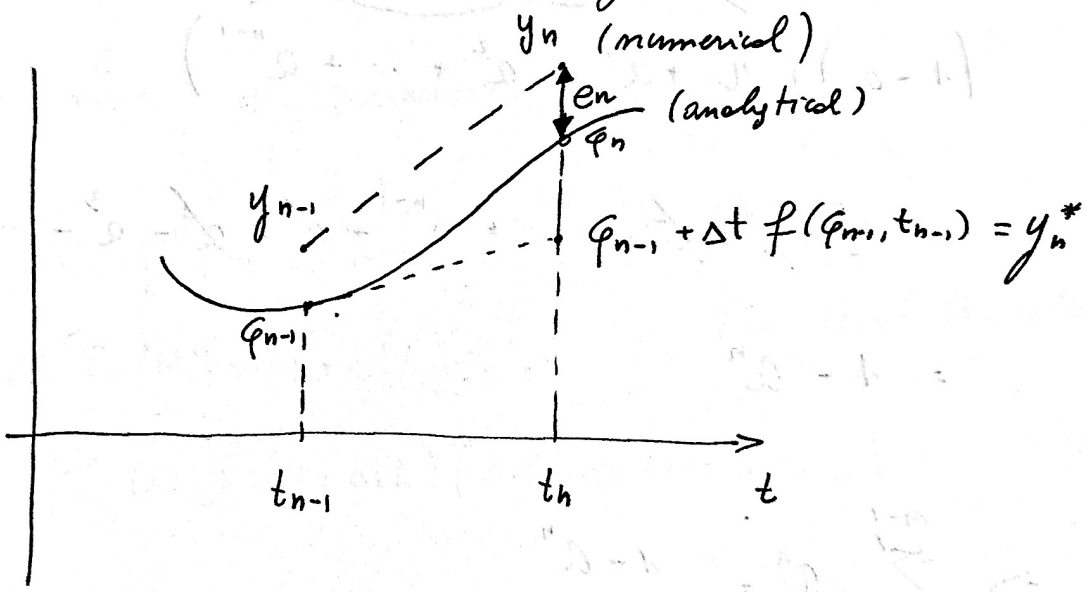
$$\tau_{n+1}(\Delta t) = \frac{\Delta t}{2} \varphi''(\xi_n)$$

This implies that the global truncation error is

$$\tau(\Delta t) = \frac{\Delta t}{2} \max_{t \in [0,T]} |\varphi''(t)| \qquad \left(\frac{\Delta t}{2} \max_{i=1,\dots,N} |\varphi''(\xi_i)| \right)$$

$\Rightarrow$ The Euler forward method is consistent with order 1.

Now let us prove convergence



$$\begin{aligned} e_n &= |y_n - \varphi_n| \\ &= |y_n - y_n^* + y_n^* - \varphi_n| \\ &\leq |\varphi_n - y_n^*| + |y_n^* - y_n| \end{aligned}$$

$$\begin{aligned} y_n^* - y_n &= \varphi_{n-1} - y_n + \Delta t\, f(\varphi_{n-1}, t_{n-1}) \\ &= \varphi_{n-1} - y_{n-1} + \Delta t \left(f(\varphi_{n-1}, t_{n-1}) - f(y_{n-1}, t_{n-1}) \right) \end{aligned}$$

$$\Rightarrow \quad |y_n^* - y_n| \leq e_{n-1} + \Delta t \, |f(\varphi_{n-1}, t_{n-1}) - f(y_{n-1}, t_{n-1})|$$

$$\leq e_{n-1} + \Delta t \, L \, e_{n-1}$$

$\downarrow$ LIPSCHITZ CONSTANT

$$\Rightarrow \quad |y_n^* - y_n| \leq (1 + \Delta t L) e_{n-1}$$

Now we have a recursion:

$$e_n \leq |\varphi_n - y_n^*| + |y_n^* - y_n|$$

$$\leq \Delta t |\tau_n(\Delta t)| + (1 + \Delta t L) e_{n-1} \qquad (\tau_n \text{ local truncation error})$$

$$\leq \Delta t |\tau_n(\Delta t)| + (1 + \Delta t L)(\Delta t |\tau_{n-1}| + (1 + \Delta t L) e_{n-2})$$

$$\leq \Delta t |\tau_n(\Delta t)| + (1 + \Delta t L) \Delta t |\tau_{n-1}(\Delta t)| + (1 + \Delta t L)^2 (|\tau_{n-2}(\Delta t)| + (1 + \Delta t L) e_{n-3})$$

$$\leq \left(\sum_{k=0}^{n-1} (1 + \Delta t L)^k \right) \Delta t \, \underbrace{\tau(\Delta t)}_{\text{global truncation error}} \qquad (e_0 = 0 \text{ initial condition})$$

Recall that: (GEOMETRIC PROGRESSION)

$$\sum_{k=0}^{n-1} (1 + \Delta t L)^k = \frac{(1 + \Delta t L)^n - 1}{\Delta t L} \qquad (1 + \Delta t L) \leq e^{\Delta t L}$$

$$\Rightarrow \quad e_n \leq \frac{e^{n \Delta t L} - 1}{\Delta t L} \tau(\Delta t) = \frac{e^{n \Delta t L} - 1}{L} \frac{\Delta t}{2} \max_{t \in [0,T]} |\varphi''(t)|$$

Recall that

$$\sum_{k=0}^{n-1} a^k = \frac{1-a^n}{1-a}$$

To prove this consider

$$(1-a)\overbrace{(a^0 + a^1 + a^2 + \cdots + a^{n-1})}^{\sum_{k=0}^{n-1} a^k}$$

$$= a^0 + \cancel{a^1} + \cancel{a^2} + \cdots + a^{n-1} - \cancel{a^1} - \cancel{a^2} - a^3 - \cdots a^n$$

$$= 1 - a^n$$

$$\Rightarrow \sum_{k=0}^{n-1} a^k = \frac{1-a^n}{1-a}$$

$$e^{nL\Delta t} - 1 = \cancel{1} + nL\Delta t + \frac{n^2 L^2 \Delta t^2}{2} - \cancel{1} + \cdots$$

$$= nL\Delta t + \cdots$$

$$= TL + \cdots$$

(where T is the total integration time

Note that $n\Delta t = T$ (total integration time).

Therefore,

$$e_n \leq \frac{e^{TL} - 1}{L} \frac{\Delta t}{2} \max_{i=1,\cdots,n} |\varphi''(\xi_i)| \qquad \xi_i \in [t_i, t_{i+1}]$$

$\Rightarrow$ The Euler forward method converges with order 1. In fact,

$$\max_{i=1,\cdots,n} |y_i - \varphi_i| = \max_{i=1,\cdots,n} e_i \leq \frac{e^{TL} - 1}{L} \frac{\Delta t}{2} \max_{j=1,\cdots,n} |\varphi''(\xi_j)|$$