# Scientific Computing

A.A. 2024-2025

Docente: E. Messina

Dispense tratte dal corso Magistrale di Informatica
a cura dello studente **S. Cerrone**
Si ringrazia **S. Scisciola** per buona parte di questi appunti

# Sommario

**1. Interpolazione polinomiale ..... 3**
Condizionamento di una matrice ..... 4
Forma di Lagrange ..... 4
L'unicità della forma di Lagrange ..... 5
Formula baricentrica ..... 6
Resto dell'interpolazione ..... 6
Maggiorazione dell'errore ..... 7
Spline cubiche interpolanti ..... 7
Costruzione di una spline cubica interpolante ..... 8
Spline naturali ..... 9
Errore della spline ..... 10
**2. Stima di integrali ..... 11**
Quadratura ..... 11
Formula dei trapezi ..... 12
Studio dell'errore ..... 13
Stima dell'errore ..... 14
Formula dei trapezi composta ..... 14
Errore nella formula di Simpson ..... 16
**3. Risoluzione sistemi lineare ..... 17**
Metodi diretti ..... 17
Backward Substitution ..... 17
Forward Substitution ..... 18
Fattorizzazione LU ..... 18
Proprietà della fattorizzazione LU ..... 18
Norme di vettori e di matrici ..... 19
Condizionamento di un sistema lineare ..... 20
Dimostrazione: numero di condizionamento ..... 20
Algoritmo di Thomas ..... 21
Metodi indiretti o iterativi ..... 22
Tale successione è convergente? Se sì, converge alla soluzione di $Ax = b$? ..... 23
Costruzione della matrice T ..... 23
Metodo di Jacobi ..... 24
Metodo di Gauss-Seidel ..... 26
Rappresentazione di matrici sparse ..... 27
Jacob per matrici sparse ..... 27
**4. Risoluzione numerica di equazioni differenziali ordinarie ..... 29**
Metodo di Eulero ..... 29
Calcolo dell'errore ..... 30
Numerical ODEs ..... 31
Sistemi "famosi" ..... 31

# 1. Interpolazione polinomiale

Interpolare significa costruire una curva che passa attraverso un insieme di punti dati. L'obiettivo dell'interpolazione è ottenere informazioni dai dati (**misure sperimentali**).

Usiamo la rappresentazione polinomiale dove una funzione $f(x)$ può essere espressa come $p_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$. Il polinomio è univocamente determinato dai coefficienti $a_i$

La condizione da rispettare per determinare gli $n+1$ coefficienti è tale che $p_n(x_i) = y_i$ per $i = 0, \dots, n$ questo rappresenterà un sistema lineare esprimibile in forma matriciale:

$$\begin{cases} a_0 + a_1x + a_2x^2 + \cdots + a_nx^n = y_0 \\ \vdots \\ a_0 + a_1x + a_2x^2 + \cdots + a_nx^n = y_n \end{cases} \Rightarrow \underbrace{\begin{pmatrix} 1 & x & x^2 & \dots & x^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x & x^2 & \dots & x^n \end{pmatrix}}_{matrice\ di\ Vandermonde} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{matrix} y_0 \\ \vdots \\ y_n \end{matrix}$$

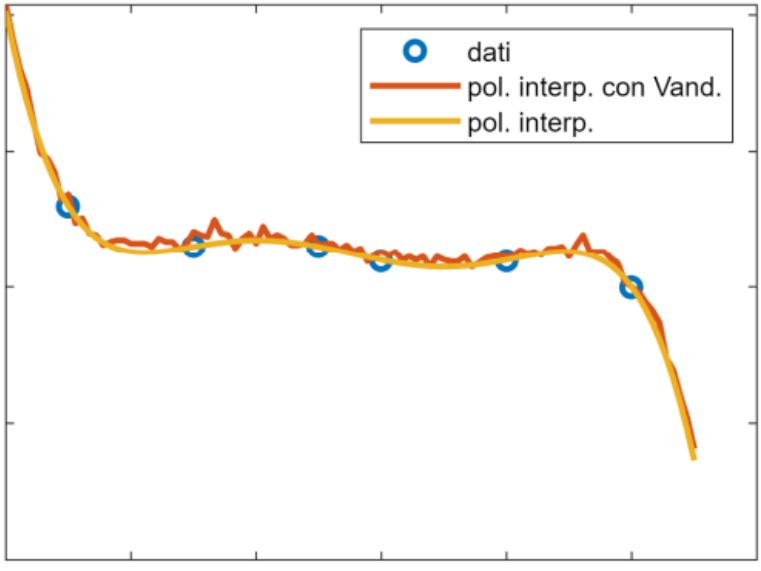
È dimostrabile che il determinante della matrice di Vandermonde è tale che

$$\det(V) = \prod_{i>j} (x_i - x_j) \neq 0$$

E questo dimostra che il sistema ammette una ed una sola soluzione: $p_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$ è unico.

**La matrice di Vandermonde è però una matrice <u>mal condizionata</u>.** Il condizionamento di un problema non dipende dal metodo usato per risolvere il problema ma da una perturbazione nei dati. Che per quanto l'errore sia piccolo mi porterà a risolvere un sistema diverso da $Va = y$ ovvero $\tilde{V}\tilde{a} = \tilde{y}$ dove $\tilde{V} = V + \delta V$ e $\tilde{y} = y + \delta y$; l'errore è rappresentato prorpio da $\delta V$ e $\delta y$.

Il problema si dice bencondizionato se l'errore sui risultati rimane piccolo.




Per problemi mal condizionati piccole variazioni nei dati, inevitabili per l'errore di rappresentazione, inducono grosse variazioni nei risultati (indipendentemente dall'algoritmo utilizzato).

## Condizionamento di una matrice

Risolviamo il sistema lineare $Ax = b$ supponendo $\delta A = 0$ (la dimostrazione è più semplice se non consideriamo gli errori sulla matrice). Sia $A(x + \delta x) = b + \delta b$, voglio risolvere il sistema $\frac{||\delta x||}{||x||}$ (errore sul risultato) rispetto $\frac{||\delta b||}{||b||}$ (errore sul dato). Mi aspetto che l'errore sul risultato sia della stessa grandezza dell'errore sul dato. Abbiamo

$$Ax + A\delta x = b + \delta b \overset{Ax=b}{\Longrightarrow} A\delta x = \delta b \Rightarrow \delta x = A^{-1}\delta b$$

Utilizziamo la norma, sfruttando la proprietà $\big||By|\big| \leq \big||B|\big| \cdot \big||y|\big|$:

$$\big||\delta x|\big| = \big||A^{-1}\delta b|\big| \leq \big||A^{-1}|\big| \cdot \big||\delta b|\big| \Rightarrow \frac{\big||\delta x|\big|}{\big||x|\big|} \leq \frac{\big||A^{-1}|\big|}{\big||x|\big|} \cdot \big||\delta b|\big|$$

Tenendo conto del fatto che $\big||b|\big| \leq \big||A|\big|\big||x|\big| \Rightarrow \frac{1}{\big||x|\big|} \leq \frac{||A||}{||b||}$ possiamo scrivere

$$\frac{\big||\delta x|\big|}{\big||x|\big|} \leq \big||A|\big| \cdot \big||A^{-1}|\big| \frac{\big||\delta b|\big|}{\big||b|\big|} \Rightarrow \frac{\big||\delta x|\big|}{\big||x|\big|} \leq K(A) \frac{\big||\delta b|\big|}{\big||b|\big|}$$

dove $K(A) = \big||A|\big| \cdot \big||A^{-1}|\big|$ è detto numero di condizionamento della matrice $A$.

Tale valore è sicuramente $\geq 1$ essendo $\big||A|\big| \cdot \big||A^{-1}|\big| \leq \big||AA^{-1}|\big| = \big||I|\big| = 1$.

$K(A)$ rappresenta un fattore di amplificazione dell'errore. Praticamente il numero di condizionamento ci da la misura di quanto l'errore sui dati venga amplificato sul risultato.

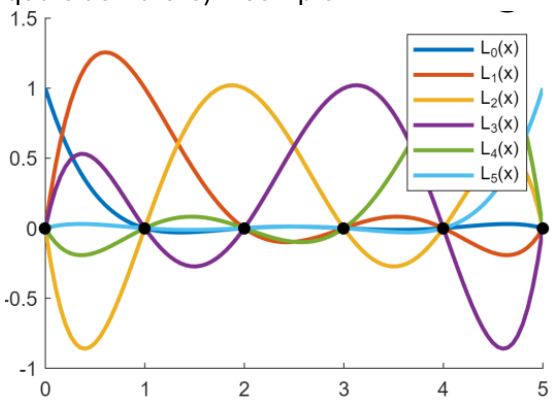
Matlab ha una sua funzione per calcolare il condizionamento ed è la funzione cond.

## Forma di Lagrange

Quindi quando si ha che fare con un problema mal condizionato è meglio non usare la matrice di coefficienti di Vandermonde. Al suo posto possiamo usare la forma più compatta di polinomio interpolante; ovvero la **forma di Lagrange**:

$$p_n(x) = \sum_{k=0}^{n} L_k(x) y_k \quad \text{con} \quad L_k(x) = \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{x - x_j}{xk - x_j}$$

La caratteristica dei $L_k$ con $k = 0,1, \dots, n$, conosciuti come **polinomi fondamentali di Lagrange** è che assumo valore 1 per $j = k$ e 0 per tutti gli altri valori di $j$ (nei valori non conosciuti può assumere qualsiasi valore). Esempio:




$$L_k(x_i) = \begin{cases} 0 & se\ i \neq k \\ 1 & se\ i = k \end{cases}$$

$$\Rightarrow$$

$$p_n(x) = \sum_{k=0}^{n} y_k \cdot L_k(x)$$

N.B. posso costruire un $L_k$ per ogni indice; quindi, potenzialmente, potrei avere un numero di $L_k$ pari a $n+1$.

$$\boldsymbol{L_k(x) = c_k(x - x_0) \dots (x - x_{k-1})(x - x_{k+1}) \dots (x - x_n)}$$

Sfruttando la condizione $L_k(x_k) = 1$ possiamo calcolare la costante $c_k$ nel seguente modo:

$$L_k(x_k) = c_k(x - x_0) \dots (x - x_{k-1})(x - x_{k+1}) \dots (x - x_n) = 1 \Rightarrow c_k = \frac{1}{\prod_{\substack{j=0 \\ j \neq k}}^{n} (x_k - x_j)}$$

Una volta calcolato $c_k$ avremo che

$$L_k(x) = \frac{\prod_{\substack{j=0 \\ j \neq k}}^{n} (x - x_j)}{\prod_{\substack{j=0 \\ j \neq k}}^{n} (x_k - x_j)} = \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{x - x_j}{x_k - x_j}$$

Quindi il polinomio possiamo scriverlo come $p_n(x) = b_0 L_0(x) + b_1 L_1(x) + \cdots + b_n L_n(x)$. I coefficienti $b_i$ possono essere calcolati sempre con le condizioni di interpolazione $p_n(x_i) = y_i, i = 0, \dots, n$ dove

$$y_0 = p_n(x_0) = b_0 \underbrace{L_0(x_0)}_{1} + b_1 \underbrace{L_1(x_0)}_{0} + \cdots + b_n \underbrace{L_n(x_0)}_{0}$$

$$y_1 = p_n(x_1) = b_0 \underbrace{L_0(x_1)}_{0} + b_1 \underbrace{L_1(x_1)}_{1} + \cdots + b_n \underbrace{L_n(x_0)}_{0}$$

$$\vdots$$

$$y_n = p_n(x_n) = b_0 \underbrace{L_0(x_n)}_{0} + b_1 \underbrace{L_1(x_n)}_{0} + \cdots + b_n \underbrace{L_n(x_n)}_{1}$$

In forma compatta (polinomio interpolante della forma di Lagrange) possiamo scrivere:

$$p_n(x) = \sum_{k=0}^{n} y_k L_k(x) \quad con\ L_k(x) = \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{x - x_j}{x_k - x_j}$$

La forma di Lagrange ha la seguente proprietà derivata dall'unicità: $\sum_{k=0}^{n} L_k(x) = 1$.

## L'unicità della forma di Lagrange

**Dimostrazione**: Supponiamo per assurdo che il polinomio non sia unico. Dunque, se anche $q(x)$ di grado al più $n$ è tale che $q(x_i) = y_i$ con $i = 0, \dots, n$ allora $p_n(x) - q(x)$ è un polinomio di grado $\leq n$ che si annulla in $n+1$ punti e ciò implica che $p_n(x) - q(x) = \dot{0}$ e ciò è assurdo.

**Conseguenze dell'unicità**:

- Se gli $n+1$ dati appartengono ad un polinomio $P(x)$, allora $p_n(x) = \dot{P}(x)$
- Se le ordinate dei punti dati sono tutte uguali ad 1, allora $p_n(x) \equiv 1$
- La somma dei polinomi fondamentali di Lagrange è uguale ad 1, qualunque sia la distribuzione dei nodi $\sum_{k=0}^{n} L_k(x) = 1$ (polinomio interpolante $f(x) = 1$)
- Il polinomio interpolante dipende dai dati

## Formula baricentrica

La formula baricentrica riduce il calcolo computazionale sfruttando la struttura dei polinomi fondamentali di Lagrange, più precisamente sfruttando il fatto che molti calcoli vengono ripetuti su tutti i polinomi fondamentali, questi vengono chiamati pesi e calcolati una sola volta prima di procedere con il calcolo degli $L_k$.

$$\text{Sia } \omega_{n+1}(x) = (x-x_0)(x-x_1)\dots(x-x_n) = \prod_{j=0}^{n}(x-x_j) \Rightarrow L_k(x) = \frac{\omega_{n+1}(x)}{(x-x_k)} \cdot \underbrace{\frac{1}{\prod_{\substack{j=0 \\ j\neq k}}^{n}(x_k-x_j)}}_{\omega_{n+1}(x_k)}$$

$$\Rightarrow p_n(x) = \sum_{k=0}^{n} y_k L_k(x) = \sum_{k=0}^{n} \frac{\omega_{n+1}(x)}{(x-x_k)\omega_{n+1}(x_k)} y_k = \omega_{n+1}(x) \sum_{k=0}^{n} \frac{w_k}{(x-x_k)} y_k$$

Dove $w_k = \frac{1}{\omega_{n+1}(x_k)}$ sono i pesi baricentrici mentre $\omega_{n+1}$ prende il nome di peso nodale

Nella pratica viene usata la seguente formula:

$$p_n(x) = \frac{\sum_{k=0}^{n} \frac{w_k}{(x-x_k)} y_k}{\sum_{k=0}^{n} \frac{w_k}{(x-x_k)}}$$

**Confronto di costo computazionale**: Considerando un $\Theta(n^2)$ per la costruzione dei pesi $w_k$ con $k = 0, \dots, n$; se si effettuano $m$ valutazioni $p_n(x^*)$ avremo per la formula di Lagrange $T(n) = \Theta(mn^2)$ mentre per la formula baricentrica $T(n) = \Theta(mn)$

## Resto dell'interpolazione

Per calcolare l'errore (o resto) per Lagrange si usa $r_n(x) = |f(x) - p_n(x)|$ dove $f \in C^{n+1} \subseteq [a,b]$, quindi $f$ è continua e $x_0, x_1, \dots, x_n \in [a,b]$.

**Teorema**. Siano $x_0, x_1, \dots, x_n$ nodi distinti nell'intervallo $[a,b]$ e $f \in C^{n+1}[a,b]$. Allora, per ogni $x \in [a,b]$, esiste un punto $\xi = \xi(x) \in (a,b)$, tale che

$$r_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}\omega_{n+1}(x)$$

**Dimostrazione**:

Per $x = x_i$, per $i = 0, \dots, n$ risulta $r_n(x) = 0$ essendo $\omega_{n+1}(x_i) = (x_i - x_0)\dots(x_i - x_1)\dots(x_i - x_n) = 0$, per cui $r_n(x_i) = f(x_i) - p_n(x_i) = y_i - y_i = 0$.

Mentre per $x \neq x_i$, per $i = 0, \dots, n$ si considera la funzione $g(t) = \underbrace{r_n(t)}_{f(t)-p_n(t)} - \frac{\omega_{n+1}(t)}{\omega_{n+1}(x)} r_n(x)$

Si usa il **teorema di Rolle**: "Sia $f$ una funzione continua definita in un intervallo chiuso $[a,b]$, derivabile in ogni punto dell'intervallo aperto $(a,b)$, tale che $f(a) = f(b)$. Allora esiste un punto $x \in (a,b)$ in cui $f'(c) = 0$".

$g$ è continua e derivabile in tutti i punti dell'intervallo $(a,b)$ quindi applicando il teorema di Roll a due punti successivi per $n+2$ punti $x_0, x_1, \dots, x_n, x$ avrò $n+1$ intervalli per cui si annulla $g'(t)$.
Siano $\xi_o(x), \xi_1(x), \dots, \xi_n(x)$ tali punti:

$$g'(t) = f'(t) - p_n'(t) - \frac{\omega_{n+1}'(t)}{\omega_{n+1}(x)} r_n(x)$$

$g'(t)$ è ancora continua (derivata e polinomio mantengono la continuità) e derivabile; quindi, si può riapplicare il teorema di Roll e ricavarci $g''(t)$ che sarà ancora continua e derivabile. Procedendo in questo modo arriveremo alla derivata $(n+1)$-esima che si annullerà in un punto.
Dunque: $\exists \xi(x) \in (a,b)$ tale che $g^{(n+1)}\big(\xi(x)\big) = 0$

$$g^{(n+1)}(t) = f^{(n+1)}(t) - p_n^{(n+1)}(t) - \frac{\omega_{n+1}^{(n+1)}(t)}{\omega_{n+1}(x)} r_n(x) = f^{(n+1)}(t) - \frac{r_n(x)}{\omega_{n+1}(x)} (n+1)!$$

Essendo $p_n^{(n+1)}(t) = 0$ poiché è la derivata $(n+1)$ esima di un polinomio di grado $n$ e $\omega_{n+1}^{(n+1)}(t) = \big((t-t_0)\dots(t-t_n)\big)^{n+1} = (n+1)!$
Dunque per la condizione del teorema di Roll "$\exists \xi(x) \in (a,b)$ tale che $g^{(n+1)}\big(\xi(x)\big) = 0$" risulta:

$$g^{(n+1)}\big(\xi(x)\big) = f^{(n+1)}\big(\xi(x)\big) - \frac{r_n(x)}{\omega_{n+1}(x)} (n+1)! = 0 \Rightarrow r_n(x) = \frac{f^{(n+1)}\big(\xi(x)\big)}{(n+1)!} \omega_{n+1}(x) \blacksquare$$

## Maggiorazione dell'errore

$$M_{n+1} = \max_{x \in [a,b]} \left| f^{(n+1)}(x) \right| \Rightarrow |r_n(x)| \leq \frac{M_{n+1}}{(n+1)!} \max_{x \in [a,b]} |\omega_{n+1}(x)|$$

Poiché $x \in [a,b]$ la stima sarà sicuramente minore o uguale dell'intervallo: $|x - x_i| \leq b$

Questa stima dell'errore non basta per capire il fenomeno poiché c'è il problema dell'errore sui calcoli (ad esempio le approssimazioni per i float point) che ovviamente dipende dalla complessità della funzione.

# Spline cubiche interpolanti

Spline: strumento flessibile usato per disegnare che veniva vincolato su dei punti che minimizzava le oscillazioni tra i vari punti.

Le spline possono avere più del grado tre ma la spline cubica è quella che offre il miglior equilibrio tra calcolo computazionale e accuratezza del grafico.

Consideriamo una partizione dell'intervallo $[a,b]$: $a = x_0 < x_1 < \dots < x_n = b$.

$s(x)$ è una spline cubica interpolante se:

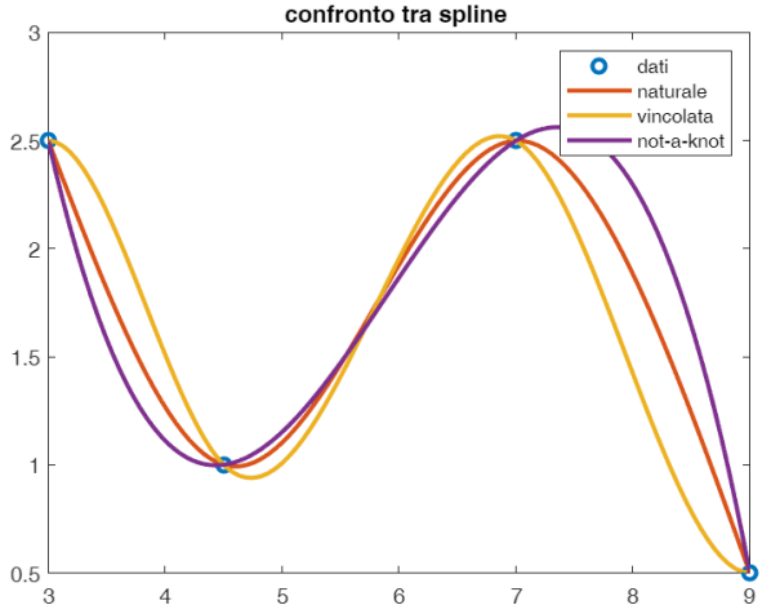
1) $s(x)|_{(x_i, x_{i+1})} = s_i(x_{i+1})$ è un polinomio di grado al più tre
2) $\left.\begin{array}{l} s_i(x_{i+1}) = s_{i+1}(x_{i+1}) \\ s_i'(x_{i+1}) = s_{i+1}'(x_{i+1}) \\ s_i''(x_{i+1}) = s_{i+1}''(x_{i+1}) \end{array}\right\}$ per $i = 0, \dots, n-2$
3) $s(x_i) = y_i$ per $i = 0, \dots, n$ (perché voglio che sia un polinomio interpolante)

Questo dice che i due polinomi successivi non solo devono coincidere nel punto ma anche nelle loro derivate. Fino ad un grado in meno del polinomio (per le cubiche, quindi, devono coincidere sia la derivata prima che la seconda). Questo ovviamente deve valere per tutte gli intervalli scelti (che per $n+1$ punti saranno $n$ intervalli e quindi avrò $n$ polinomi).

In definitiva avrò $4n$ coefficienti da determinare con $3(n-1)$ condizioni per il punto 2 e $n+1$ condizioni per il punto 3.

Ci sono vari tipi di spline che aggiungono alcune condizioni rispetto a quelle descritte precedentemente, le principali sono:

1) spline naturale $s''(x_0) = s''(x_n) = 0$
2) spline vincolata $s'(x_0) = f_0, s'(x_n) = f_n$
3) spline not_a_knot (quella usata in MatLab) $s_0'''(x_1) = s_1'''(x_1)$ e $s_{n-2}'''(x_{n-1}) = s_{n-1}'''(x_{n-1})$ (quindi coincidono con le derivate prime, seconde e anche terze; dunque, saranno un solo polinomio per gli intervalli $[x_0, x_1]$ e $[x_1, x_2]$ e un unico polinomio per $[x_{n-2}, x_{n-1}]$ e $[x_{n-1}, x_n]$




## Costruzione di una spline cubica interpolante

Sappiamo che

$$s(x) = \begin{cases} s_0(x) \; se \; x \in [x_0, x_1] \\ s_1(x) \; se \; x \in [x_1, x_2] \\ \vdots \\ s_{n-1}(x) \; se \; x \in [x_{n-1}, x_n] \end{cases}$$

Sia $j = 0,1, \dots, n-1$ scriviamo la spline nella seguente forma:

$$s_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$$

Dove $a_j, b_j, c_j$ e $d_j$ sono coefficienti da determinare per.

Sappiamo dalla proprietà 3 che $s_j(x_j) = y_j$, quindi possiamo ricavare la <u>condizione di interpolazione</u>:

$$s_j(x_j) = a_j + b_j \underbrace{(x_j - x_j)}_{=0} + c_j \underbrace{(x_j - x_j)^2}_{=0} + d_j \underbrace{(x_j - x_j)^3}_{=0} = y_j \Rightarrow a_j = y_j$$

Dalla proprietà 2 invece abbiamo che $s_j(x_{j+1}) = s_{j+1}(x_{j+1})$. Sia $h_j = x_{j+1} - x_j$ per $j = 0, \dots, n-1$ abbiamo (0) $y_j + b_j h_j + c_j h_j + c_j h_j^2 + d_j h_j^3$; inoltre, per definizione

$$s_{j+1}(x) = a_{j+1} + b_{j+1}(x - x_{j+1}) + c_{j+1}(x - x_{j+1})^2 + d_{j+1}(x - x_{j+1})^3$$

Quindi $s_{j+1}(x_{j+1}) = a_{j+1} = y_{j+1}$ *e per la* (0) abbiamo la **condizione di continuità**:

$$(\boldsymbol{a}) \quad \boldsymbol{y_j + b_j h_j + c_j h_j^2 + d_j h_j^3 = y_{j+1}}$$

Segue la **continuità della derivata prima**:

$$\begin{matrix} s_j'(x) = b_j + 2c_j(x - x_j) + 3d_j(x - x_j)^2 \\ s_{j+1}'(x) = b_{j+1} + 2c_j(x - x_{j+1}) + 3d_j \end{matrix} \Rightarrow s_j'(x_{j+1}) = s_{j+1}'(x_{j+1}) \Rightarrow (\boldsymbol{b}) \quad \boldsymbol{b_j + 2c_j h_j + 3d_j h_j^2 = b_{j+1}}$$

E la **continuità della derivata seconda**:

$$\begin{matrix} s_j''(x) = 2c_j + 6d_j(x - x_j) \\ s_{j+1}''(x) = 2c_{j+1} + 6d_{j+1}(x - x_{j+1}) \end{matrix} \Rightarrow \frac{s_j''(x_{j+1})}{2} = \frac{s_{j+1}''(x_{j+1})}{2} \Rightarrow (\boldsymbol{c}) \quad \boldsymbol{c_j + 3d_j h_j = c_{j+1}}$$

Sostituiamo $d_j = \frac{c_{j+1} - c_j}{3h_j}$ (ricavato dalla (c)) nella condizione di continuità (a):

$$(1) \quad y_j + b_j h_j + \frac{h_j^2}{3}(2c_j + c_{j+1}) = y_{j+1}$$

Adesso, sostituiamo $d_j$ nella condizione di continuità delle derivate:

$$(2) \quad b_{j+1} = b_j + h_j(c_j + c_{j+1})$$

Ricaviamo $b_j$ dalla (1) e riduciamolo in seguito di un indice:

$$b_j = \frac{y_{j+1} - y_j}{h_j} - \frac{h_j^2}{3}(2c_j + c_{j+1}) \Rightarrow b_{j-1} = \frac{y_j - y_{j-1}}{h_{j-1}} - \frac{h_{j-1}^2}{3}(2c_{j-1} + c_j)$$

Sostituiamo questo $b_{j-1}$ nella (2) con l'indice ridotto di uno, dopo vari raggruppamenti e semplificazioni troveremo la seguente equazione valida per $j = 1, \dots, n-1$:

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = 3\frac{y_{j+1} - y_j}{h_j} - 3\frac{y_j - y_{j-1}}{h_{j-1}}$$

Questa equazione per $j = 1, \dots, n-1$ avrà i seguenti valori (si noti che per $j = 0$ e $j = n$ l'equazione non ha senso):

$$\begin{matrix} per\ j = 1: & h_0c_0 + 2(h_0 + h_1)c_1 + h_1c_2 = 3\frac{y_2 - y_1}{h_1} - 3\frac{y_1 - y_0}{h_0} \\ \vdots & \vdots \\ j = n-1: & h_{n-2}c_{n-2} + 2(h_{n-2} + h_{n-1})c_{n-1} + h_{n-1}c_n = 3\frac{y_n - y_{n-1}}{h_{n-1}} - 3\frac{y_{n-1} - y_{n-2}}{h_{n-2}} \end{matrix}$$

In questa forma è evidente che ci troviamo in un sistema di $n-1$ equazioni dove possiamo ricavare i $c_j$. Ovviamente ci mancano l'indice 0 e $n$ che non possiamo ricavare allo stesso modo degli altri indici.

## Spline naturali

Ma possiamo sfruttare il fatto che $s''(x_0) = 2c_0 = 0$ e che $s''(x_n) = 2c_n = 0$, quindi $c_0 = c_n = 0$. Adesso abbiamo tutte le informazioni per scrivere il seguente sistema lineare $Ac = y$:

$$\begin{bmatrix} 1 & 0 & 0 & & & & \\ h_0 & 2(h_0 + h_1) & h_1 & \cdots & & 0 & \\ 0 & h_1 & 2(h_1 + h_2) & & & & \\ & \vdots & & \ddots & & \vdots & \\ & & & & 2(h_{n-3} + h_{n-2}) & h_{n-2} & 0 \\ & 0 & & \cdots & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \\ & & & & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-2} \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} 0 \\ \delta_1 \\ \delta_2 \\ \vdots \\ \delta_{n-2} \\ \delta_{n-1} \\ 0 \end{bmatrix}$$

Dove $\delta_j = 3\left(\frac{y_{j+1}-y_j}{h_j} - \frac{y_j-y_{j-1}}{h_{j-1}}\right)$ per $j = 1, \dots, n-1$. Si noti che la matrice $A$ è tridiagonale, a diagonale strettamente dominante ed il sistema **ammette una ed una sola soluzione** che può essere efficientemente calcolata usando l'algoritmo di Thomas.

- Per $j = 0, \dots, n-1$ si determinano;

$$a_j = y_j$$
$$b_j = \frac{y_{j+1} - y_j}{h_j} - \frac{h_j^2}{3}(2c_j + c_{j+1})$$
$$c_j \text{ dal sistema } Ac = y$$
$$d_j = \frac{c_{j+1} - c_j}{3h_j}$$

- Per valutare $s(x)$ in un punto $x^*$ basta trovare l'intervallo in cui si trova, se $x^* \in [x_j, x_{j+1}]$ si calcola $s_j(x^*) = a_j + b_j(x^* - x_j) + c_j(x^* - x_j)^2 + d_j(x^* - x_j)^3$, con l'algoritmo di Horner.

## Errore della spline

**Teorema**. Sia $f \in C^4([a,b])$ e si supponga nota una partizione di $[a,b]$ in sottointervalli di ampiezza $h_j$. Sia $s(x)$ la spline cubica interpolante $f$. Allora

$$\left\|f^{(k)} - s^{(k)}\right\|_\infty \leq CH^{4-k}\left\|f^{(4)}\right\|_\infty, \qquad k = 0,1,2,3$$

con $C_0 = \frac{5}{384}, C_1 = \frac{1}{24}, C_2 = \frac{3}{8}$ e $C_3 = \frac{\beta+\beta^{-1}}{2}$, dove $H = \max_i h_i$ e $\beta = \frac{H}{\min_i h_i}$.

Quindi in questo caso l'errore diminuisce all'aumentare del numero di nodi, cosa che non si aveva con la forma di Lagrange.

# 2. Stima di integrali

Consideriamo una funzione $f$ continua nell'intervallo $[a, b]$. Vogliamo approssimare l'integrale limitato nell'intervallo (praticamente vogliamo approssimare l'area sotto la curva $f$), ossia:

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

L'approssimazione dell'integrale è utile nel caso in cui non si conosca $F$ oppure non si ha la funzione completa (magari abbiamo solo dei punti misurati sperimentalmente).

## Quadratura

Come ben noto calcolare l'integrale di un polinomio è facile proprio per la linearità dell'integrale

$$\int_a^b \left(a_0 + a_1 x + \cdots + a_k x^k\right)dx = a_o \int_a^b 1dx + a_1 \int_a^b xdx + \cdots + a_k \int_a^b x^k dx$$

Quindi possiamo risolvere il problema di calcolare l'integrale di una funzione $f(x)$ calcolando l'integrale di un polinomio interpolante che approssima $f(x)$.

Siano $x_0, x_1, \dots, x_n$, $n+1$ punti tali che: $a = x_0 < x_1 < \cdots < x_n = b$. Abbiamo, in corrispondenza di questi punti, delle misurazioni $f(x_0)$,$f(x_1)$, … , $f(x_n)$.

Supponiamo che i punti siano equidistanti: $h = x_{i+1} - x_i$ per $i = 0, \dots, n$; avremo il seguente polinomio interpolante:

$$p_n(x) = \sum_{k=0}^{n} f(x_k)L_k(x) \quad \text{dove} \quad L_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i}$$

Ora, essendo $f \in C^{n+1}[a, b]$ abbiamo, per $\xi \in (a, b)$:

$$e_n(x) = f(x) - p_n(x) = \frac{f^{(n+1)}\left(\xi(x)\right)}{(n+1)!} \cdot \omega_{n+1}(x)$$

Ed è anche noto che $f(x) = p_n(x) + e_n(x)$. Passiamo agli integrali:

$$I = \int_a^b f(x)dx = \int_a^b \left(p_n(x) + e_n(x)\right)dx = \int_a^b p_n(x)dx + \int_a^b e_n(x)dx$$

Dove $\int_a^b p_n(x)\,dx = I_n$ e $\int_a^b e_n(x)\,dx = r_n$ dove $r_n$ è conosciuto come errore di quadratura.

Iniziamo con il definire $I_n$:

$$I_n = \int_a^b p_n(x)dx = \sum_{k=0}^{n} w_k \cdot f(x_k) \quad \text{dove} \quad w_k = \int_a^b L_k(x)dx$$

I $w_k$ sono detti pesi della formula di quadratura. Si noti che se i pesi sono dati dalle basi di Lagrange, allora $I_n$ è detta **formula di quadratura interpolata**.

Per quanto riguarda $r_n$, invece:

$$r_n = \int_a^b e_n(x)dx = \int_a^b \left(\frac{f^{(n+1)}\left(\xi(x)\right)}{(n+1)!}\omega_{n+1}(x)\right)dx \quad \text{se } f \in C^{n+1}[a, b]$$

Se $x_k = x_0 + kh$ per $k = 0, \dots, n$ con $x_0 = a$ e $x_n = b$, allora la formula di quadratura interpolatoria si chiama **formula di Newton-Cotes**

## Formula dei trapezi

Calcoliamo la formula di Newton-Cotes per $n = 1$, ovvero con solo 2 punti: $x_0 = a$ e $x_1 = b$.
Quindi, sia $h = b - a$, abbiamo:

$$L_0(x) = \frac{x - x_1}{x_0 - x_1} = -\frac{x - x_1}{h} \qquad L_1(x) = \frac{x - x_0}{x_1 - x_0} = \frac{x - x_0}{h}$$

Ne consegue:

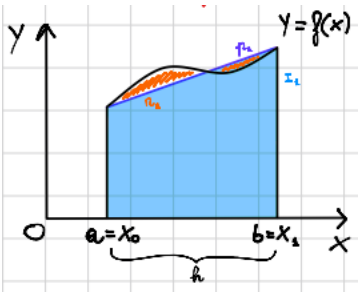
$$w_0 = \int_{x_0}^{x_1} L_0(x)dx$$
$$= -\int_{x_0}^{x_1} \frac{x - x_1}{h} dx \underbrace{=}_{x = x_0 + s\cdot h} -\int_{x_0}^{x_1} \frac{x_0 + s\cdot h - x_1}{h} \underbrace{d(x_0 + s\cdot h)ds}_{\text{per l'integrale di sostituzione}} \underbrace{=}_{x_1 = x_0 + h}$$
$$-\int_{x_0}^{x_0+h} \frac{x_0 + s\cdot h - (x_o + h)}{h} [d(x_0) + d(s\cdot h)]ds = -\int_{x_0}^{x_0+h} \frac{s\cdot h - h}{h} [0 + h]ds$$
$$= -\int_{x_0}^{x_0+h} (s-1)h\, ds \underbrace{=}_{(*)} -h\int_0^1 (s-1)\, ds = h\int_0^1 (1-s)\, ds = h\left[s - \frac{s^2}{2}\right]\Big|_0^1 = \frac{h}{2}$$

$$(*)\begin{matrix} x_0 + h \Rightarrow 1 \\ x_0 \Rightarrow 0 \end{matrix} \quad \begin{matrix} x = x_1 \Rightarrow x_1 = x_0 + sh \Rightarrow x_0 + h = x_0 + sh \Rightarrow h = sh \Rightarrow s = 1 \\ x = x_0 \Rightarrow x_0 = x_0 + sh \Rightarrow 0 = sh \Rightarrow s = 0 \end{matrix}$$

$$w_1 = \int_{x_0}^{x_1} L_1(x)dx = \int_{x_0}^{x_1} \frac{x - x_0}{h} dx \underbrace{=}_{x = x_0 + s\cdot h} \int_{x_0}^{x_1} \frac{x_0 + s\cdot h - x_0}{h} d(x_0 + s\cdot h)ds \underbrace{=}_{(*)} \int_0^1 s\, ds = h\left[\frac{s^2}{2}\right]\Big|_0^1 = \frac{h}{2}$$

Quindi $w_0 = w_1 = \frac{h}{2}$. Di conseguenza:

$$I = w_0 f(x_0) + w_1 f(x_1) = \frac{h}{2}[f(x_0) + f(x_1)]$$




Questa formula è detta **formula dei trapezi** perché il polinomio interpolante per $n = 1$ è la retta passante per i punti $(a, f(a))$ e $(b, f(b))$. In effetti stiamo calcolando l'area del trapezio blu in figura.

Si utilizzano polinomi di grado basso perché i polinomi di grado alto oscillano troppo. Infatti, l'errore dipende dalla funzione (che non possimao controllare) e da $h$. Sia $f \in C^2[a, b]$ allora:

$$e_1(x) = f(x) - p_1(x) = \frac{f^{(2)}(\xi(x))}{2}(x - x_0)(x - x_1) \Rightarrow r_1 = \int_{x_0}^{x_1} \frac{f^{(2)}(\xi(x))}{2}(x - x_0)(x - x_1)dx$$
$$= -\frac{h^3}{12}\cdot f^2(\xi) \ \text{ con } \ \xi \in (a, b) \text{ che, poiché svolto l'integrale, non dipende più da } x$$

Dunque, per intervalli $h$ grandi, l'errore $r_n$ è grande.

Prendiamo ad esempio $I = \int_0^{100} \sin(x)$ e $r_1 = -\frac{100^3}{2}$. L'errore è un valore positivo, quindi consideriamo il valore assoluto: $|r_1| \leq \frac{100^3}{2}$ che è evidentemente troppo alto. Per diminuire l'errore si deve prendere un $h$ più piccolo, ma non possiamo usare polinomi di grado più alto perché avremmo il problema dell'oscillazione (si procede con qualcosa di molto simile alla spline).

Consideriamo una partizione $a = x_0 < x_1 < \cdots < x_n = b$ (poiché gli integrali sono indipendenti tra loro non ho nemmeno bisogno di punti equidistanti):

$$I = \int_a^b f(x)dx = \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \cdots + \int_{x_{n-1}}^{x_n} f(x)dx$$

Sia $h_i = x_i - x_{i-1}$ per $i = 1, \dots, n$, prendiamo l'integrale $I$ dividendoli di $n$ intervalli usando polinomi di grado 1 (si usa la notazione $I_1^n$):

$$I_1^n = \frac{h_1}{2}\big(f(x_0) + f(x_1)\big) + \frac{h_2}{2}\big(f(x_1) + f(x_2)\big) + \cdots + \frac{h_n}{2}\big(f(x_{n-1}) + f(x_n)\big)$$

Questo è utile quando I dati sono di origine sperimentale e quando I punti non sono equidistanti, ma se posso scegliere la disposizione mi conviene scegliere dei punti equidistanti.

Infatti, se $h_1 = h_2 = \cdots = h_n$ (chiamiamoli $h$) la formula sarà:

$$I_1^n = \frac{h}{2}\big[\big(f(x_0) + f(x_1)\big) + \big(f(x_1) + f(x_2)\big) + \cdots \text{è}\big(f(x_{n-1}) + f(x_n)\big)\big]$$
$$= \frac{h}{2}\left[f(x_0) + 2\sum_{k=1}^{n-1} f(x_k) + f(x_n)\right]$$

## Studio dell'errore

Controlliamo se abbiamo effettivamente una riduzione dell'errore nell'intervallo $[x_i, x_{i+1}]$:

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx \frac{h}{2}[f(x_i) + f(x_{i+1})]$$

Ma, se $f \in C^2[a,b]$ allora per $\xi_i \in (x_i, x_{i+1})$:

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx \frac{h}{2}[f(x_i) + f(x_{i+1})] - \frac{h^3}{12}f^{(2)}(\xi_i) \Rightarrow I = \sum_{i=0}^{n}\int_{x_i}^{x_{i+1}} f(x)dx = I_1^n + e_1^n$$

Dove, per $\xi_i \in (x_i, x_{i+1})$:

$$e_1^n = \sum_{i=0}^{n-1}\left(-\frac{h^3}{12}f^{(2)}(\xi_i)\right) = -\frac{h^3}{12}\sum_{i=0}^{n-1} f^{(2)}(\xi_i)$$

Per il teorema della media discreta: $\exists \xi \in (a,b) : f^{(2)}(\xi) = \frac{1}{n}\sum_{i=0}^{n-1} f^{(2)}(\xi_i)$ possiamo scrivere

$$e_1^n = -\frac{h^3}{12}\sum_{i=0}^{n-1} f^{(2)}(\xi_i) = -\frac{h^3}{12}n \cdot f^{(2)}(\xi) \quad \text{con} \quad \xi \in (a,b)$$

A questo punto cerchiamo di capire quanto è piccolo l'errore; dato che abbiamo scelto $n$ sotto intervalli equidistanti, sappiamo che $h = \frac{b-a}{n} \Rightarrow n = \frac{b-a}{h}$. Quindi:

$$e_1^n = -\frac{h^3}{12}n \cdot f^{(2)}(\xi) = -\frac{h^2}{12}(b-a)f^{(2)}(\xi)$$

Non possiamo conoscere $\xi$ e talvolta non possiamo nemmeno calcolare la derivata seconda. Però dalla teoria vediamo che se $n$ è abbastanza grande ed $h$ abbastanza piccolo avremo $\lim_{h\to 0} e_1^n = 0$.

## Stima dell'errore

Possiamo calcolare una stima matematica dell'errore. Sappiamo che

$$e_1^n = -\frac{h^3}{12} n \cdot f^{(2)}(\xi) = -\frac{h^2}{12}(b-a) f^{(2)}(\xi) \text{ dove } h = \frac{b-a}{n}$$

Andiamo a maggiorare come segue:

$$|e_1^n| \leq -\frac{h^2}{12}(b-a) \max_{a \leq x \leq n} \left| f^{(2)}(x) \right|$$

Talvolta $\max_{a \leq x \leq n} \left| f^{(2)}(x) \right|$ è facile da calcolare a mano, altre, invece, potremmo essere costretti a graficare $f^{(2)}(x)$ e vedere dove sta il massimo così da non dover effettuare calcoli probabilmente complicati per trovarlo matematicamente (MatLab sennò che lo teniamo a fare!).

Sia $M_n = \max_{a \leq x \leq n} \left| f^{(2)}(x) \right|$, facciamo una stima a priori. Sia $tol$ una tolleranza, vogliamo sapere quanto deve essere grande $n$ affinché l'errore $|e_1^n| < tol$; ovvero

$$\frac{h^2}{12}(b-a) \max_{a \leq x \leq n} \left| f^{(2)}(x) \right| \Rightarrow \frac{(b-a)^3}{12n} < \frac{tol}{M_n}$$

Supponiamo ad esempio di avere il seguente integrale:

$$\int_0^{100} \sin(x)\, dx \Rightarrow \frac{100^3}{12n^2} \leq \frac{tol}{1} \Rightarrow 12n^2 \geq \frac{100^3}{tol} \Rightarrow n \geq \sqrt{\frac{100^3}{12\, tol}}$$

Quanto velocemente si riduce l'errore se scelgo $\frac{h}{2}, \frac{h}{4}, \ldots$?

Sostituiamo in $e_1^n$ il valore $\frac{h}{2}$ al posto di $h$:

$$e_1^n = -\frac{h^2}{12}(b-a) f^{(2)}(\xi) \Rightarrow e_1^{2n} \cong \frac{1}{4} e_1^n$$

Dobbiamo usare il "circa" perché il valore $\xi$ cambierà con una nuova distribuzione di punti, che è cambiata per permettere che ora $h$ sia $\frac{h}{2}$ (i punti raddoppiano).

# Formula dei trapezi composta

Poniamoci nel caso di voler calcolare $I(f) = \int_a^b f(x)dx$ quando i punti sono equidistanti.

$$I_1^n(f) = \frac{h}{2}\left( f(x_0) + 2\sum_{i=1}^{n} f(x_i) + f(x_n) \right)$$

Se $f \in C^2(a,b)$ abbiamo $E_1^n(f) = -\frac{h^2}{12}(b-a) \cdot f''(\eta)$ con $\eta \in (a,b)$ e $h = \frac{b-a}{n}$.

Per comprendere quale formula è migliore andiamo a vedere quante valutazioni di funzione dobbiamo fare. Più diminuisce $h$ più aumentano i nodi e ho maggiore accuratezza ma anche un maggior costo computazionale, poiché dobbiamo valutare le funzioni in più nodi.

**Nota**: Se il polinomio è di grado $\leq 1$ la formula dei trapezi è esatta!

Una formula di quadratura ha **grado di esattezza** (o precisione) $\boldsymbol{d}$ se è esatta (cioè se $E(f) = 0$) per $f = p_j(x)$ con $j = 0, \dots, d$ ed esiste almeno un polinomio di grado $d+1$ tale che $E(p_{d+1}) \neq 0$.

**Nota**: $E_1^n(f) = -\frac{h^2}{12}(b-a) \cdot f''(\eta)$. Se il grado del polinomio è 0 o 1 l'errore sarà nullo essendo la derivata seconda uguale a 0.

Il nostro scopo è migliorare la potenza di $h$ e la derivata di $f$ utilizzando la formula di Newton-Cotes (ci sono anche altri metodi che si potrebbero utilizzare). Ricordiamo che

$$I_n(f) = \sum_{i=0}^{n} w_i f(x_i) \quad \text{dove } w_i = \int_a^b L_i(x)dx$$

Consideriamo 3 punti: $x_0 = a, x_1 = \frac{a+b}{2}, x_2 = b$; e sia $h = \frac{b-a}{2}$. In questo caso, il polinomio che interpola $\big(x_0, f(x_0)\big), \big(x_1, f(x_1)\big), \big(x_2, f(x_2)\big)$ è $p_2(x) = f(x_0)L_0(x) + f(x_1)L_1(x) + f(x_2)L_2(x)$.

Dove:

$$L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}, \qquad L_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}, \qquad L_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}$$

Per ottenere i pesi di quadratura dobbiamo integrare i pesi di Lagrange:

$$w_0 = \int_{x_0}^{x_2} L_0(x)dx = \frac{h}{3}, \qquad w_1 = \int_{x_0}^{x_2} L_1(x)dx = \frac{4}{3}h, \qquad w_2 = \int_{x_0}^{x_2} L_2(x)dx = \frac{h}{3}$$

Di conseguenza, la formula di quadratura è la seguente:

$$I_2(f) = \frac{h}{3}\big(f(x_0) + 4f(x_1) + f(x_2)\big)$$

Tale formula prende il nome di **formula di Simpson** (o Cavalieri-Simpson).

Integrando un polinomio interpolante di grado 2, ho una riduzione dell'errore perché approssimo la funzione a una parabola. I calcoli per trovare l'errore nella formula di Simpson sono simili a quella della formula dei trapezi (un bel po' più tediosi quindi ce li risparmiamo):

$$E_2(f) = -\frac{h^5}{90} \cdot f^{(4)}(\xi) \quad \text{con } \xi \in (a,b)$$

Questa è la formula *semplice* su tutto l'intervallo $(a,b)$. Passiamo a quella composta dividendo l'intervallo in un numero **pari** (necessario che sia pari visto che lavoriamo su coppie di punti) di sotto intervalli (se il numero fosse dispari si potrebbe usare i trapezi per l'ultimo nodo e Simpson per tutti gli altri).

Prendiamo $n+1$ punti equidistanti tra loro tali che $n = 2m$ dove $m$ rappresenta il numero di coppie e $n$ il numero di sottointervalli:

$$I_2^n(f) = \frac{h}{3}\Big(\underbrace{f(x_0) + 4f(x_1) + f(x_2)}_{} + \underbrace{f(x_2) + 4f(x_3) + f(x_4)}_{} + \cdots + \underbrace{f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)}_{}\Big)$$

$$= \frac{h}{3}\left(f(x_0) + 4\underbrace{\sum_{j=1}^{m} f\big(x_{2j-1}\big)}_{\text{indici dispari}} + 2\underbrace{\sum_{j=1}^{m} f\big(x_{2j}\big)}_{\text{indici pari}} + f(x_n)\right)$$

## Errore nella formula di Simpson

Sappiamo che $E_2(f) = -\frac{h^5}{90} f^{(4)}(\xi)$ dove $h = \frac{b-a}{2}$. Ma nella forma composta

$$E_2^n(f) = \sum_{j=0}^{m} -\frac{h^5}{90} f^{(4)}(\xi_j) \quad \text{con } h = \frac{b-a}{n} \xrightarrow{\text{si dimostra}} E_2^n(f) = -\frac{h^4}{180}(b-a) \cdot f^{(4)}(\xi) \text{ per } \xi \in (a,b)$$

Ora se dimezzassimo $h$ l'errore si ridurrebbe di 16: $\left(\frac{h}{2}\right)^4 = \frac{h^4}{16}$. Inoltre, il grado di esattezza della formula di Simpson è $d = 3$ (essendo $f^{(4)}(\xi) = 0$ per polinomi di grado 0,1,2 e 3).

Tuttavia, la formula per stimare l'errore è poco pratica, infatti, con questa formula possiamo fare la massimo una stima a priori ma non possiamo di cerco calcolare l'errore automaticamente. Dunque, facciamo lo stesso lavoro fatto per la formula dei trapezi.

Calcoliamo la formula di Simpson con $h$ e $\frac{h}{2}$ dove $h = \frac{b-a}{n}$ e $n = 2m$. Sappiamo che l'integrale vero è

$$I = I_2(h) + E_2(h) = I_2(h) - \frac{h^4}{180}(b-a) f^{(4)}(\xi)$$

$$I = I_2\left(\frac{h}{2}\right) + E_2\left(\frac{h}{2}\right) = I_2\left(\frac{h}{2}\right) - \frac{\left(\frac{h}{2}\right)^4}{180}(b-a) f^{(4)}(\bar{\xi})$$

Si noti che $\xi$ e $\bar{\xi}$ derivano dal teorema della media discreta ma non coincidono.

Supponiamo che $f^{(4)}(\xi) \cong f^{(4)}(\bar{\xi})$. Quindi, se $f \in C^{(4)}(a,b)$:

$$I_2(h) - \frac{h^4}{180}(b-a) f^{(4)}(\xi) \cong I_2\left(\frac{h}{2}\right) - \frac{h^4}{180} \cdot \frac{1}{16}(b-a) f^{(4)}(\xi)$$

$$\Rightarrow I_2(h) - I_2\left(\frac{h}{2}\right) \cong -\frac{h^4}{180}(b-a) f^{(4)}(\xi)\left(\frac{1}{16} - 1\right)$$

$$\Rightarrow I_2(h) - I_2\left(\frac{h}{2}\right) \cong \left(\frac{15}{16}\right)\frac{h^4}{180}(b-a) f^{(4)}(\xi)$$

Ne consegue che

$$E_2(h) \cong \frac{I_2\left(\frac{h}{2}\right) - I_2(h)}{15} \quad \text{essendo } E_2\left(\frac{h}{2}\right) = -\frac{h^4}{180}\frac{1}{16}(b-a) f^{(4)}(\xi)$$

La strategia da applicare è la stessa usata nella formula dei trapezi, cioè dividiamo in successione $h$ fino a raggiungere un errore minore di $tol$. Si noti che tutto questo è valido se $f^{(4)} \in C^4(a,b)$, e per verificarlo si può procedere come segue:

1) Possiamo usare dei test a priori che verifichino che fino alla derivata terza, il risultato sia 0
2) Possiamo verificare che la riduzione dell'errore abbia una convergenza di circa $\frac{1}{16}$
    - Se abbiamo risultati anomali, poi, andiamo a fare i dovuti test

L'utente dovrebbe essere sicuro di passare una funzione che sia $C^4(a,b)$

# 3. Risoluzione sistemi lineare

Consideriamo un sistema lineare $n \times n$ ($n$ equazioni e $n$ incognite) del tipo:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

Dove $x_i$ sono le $n$ incognite; $a_i$ sono i coefficienti del sistema e $b_i$ i termini noti. Possiamo scrivere tale sistema in forma matriciale: $Ax = b$; dove

- $A$ è la matrice dei coefficienti: $A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$
- $x$ è il vettore delle incognite trasposto: $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$
- $b$ è il vettore dei termini noti trasposto: $b = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$

Se $A$ è non singolare, cioè il determinante di $A$ è diverso da zero, allora il sistema lineare $Ax = b$ ammette un'unica soluzione (dato che $\det A \neq 0$ possiamo trovare l'inversa della matrice $A$).

La soluzione del sistema lineare sarà: $x = A^{-1}b$; ma calcolare l'inversa $A^{-1}$ e poi moltiplicarlo per il vettore $b$ non conviene a livello computazionale, per questo si usano altre strategie.

I metodi di risoluzione di sistemi lineari si classificano in due famiglie:

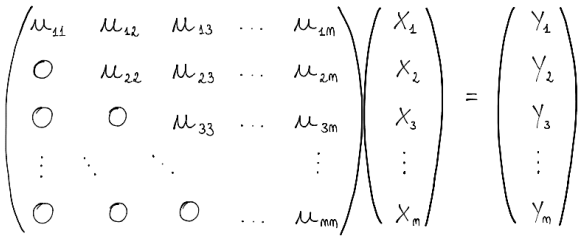
- **Metodi diretti**. Trovano la soluzione in un numero finito di passi
- **Metodi iterativi**. Teoricamente, per trovare la soluzione ci vogliono un numero infinito di passi, ma si definisce una condizione di stop.

## Metodi diretti

Analizziamo i due casi più semplici per risolvere il sistema lineare $Ax = b$

### Backward Substitution

Supponiamo che la matrice dei coefficienti sia una matrice triangolare superiore $Ux = y$, cioè:



N.B.: come nel caso generale anche $\det U \neq 0$ (quindi $U$ deve essere non singolare). Dato che $U$ è triangolare superiore, $\det U$ è semplicemente il prodotto degli elementi sulla diagonale principale. Quindi verificare che $\det U \neq 0$ significa controllare che $u_{ii} \neq 0 \; \forall i = 1, \dots, n$.

Tramite sostituzioni successive, possiamo ricavare tutte le incognite:

$$u_{nn}x_n = y_n \Rightarrow x_n = \frac{y_n}{u_{nn}}$$

$$u_{n-1,n-1}x_{n-1} + u_{n-1,n}x_n = y_{n-1} \Rightarrow x_{n-1} = \frac{y_{n-1} - u_{n-1,n}x_n}{u_{n-1,n-1}}$$

$$\vdots$$

Questo algoritmo prende il nome di **backward substitution**

### Forward Substitution

Supponiamo che la matrice dei coefficienti sia una matrice triangolare inferiore $Lx = b$, cioè:

$$\begin{pmatrix} l_{11} & 0 & 0 & \dots & 0 \\ l_{21} & l_{22} & 0 & \dots & 0 \\ l_{31} & l_{32} & l_{33} & \dots & 0 \\ \vdots & \ddots & \ddots & & \vdots \\ l_{m1} & l_{m2} & l_{m3} & \dots & l_{mm} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{pmatrix}$$

N.B.: come nel caso generale anche $\det L \neq 0$. Dato che $L$ è triangolare inferiore, $\det L$ è semplicemente il prodotto degli elementi sulla diagonale principale. Quindi verificare che $\det L \neq 0$ significa controllare che $l_{ii} \neq 0 \; \forall i = 1, \dots, n$.

Tramite sostituzioni successive, possiamo ricavare tutte le incognite:

$$l_{11}x_1 = b_1 \Rightarrow x_1 = \frac{b_1}{l_{11}}$$

$$l_{21}x_1 + l_{22}x_2 = b_2 \Rightarrow x_2 = \frac{b_2 - l_{21}x_1}{l_{22}}$$

$$\vdots$$

Questo algoritmo è chiamato **forward substitution**

## Fattorizzazione LU

Sia $Ax = b$ un sistema lineare con $A \in \mathbb{R}^{n \times m}, b \in \mathbb{R}^n$ e $\det A \neq 0$.

Anche usando metodi diretti per la soluzione (quindi senza approssimazione, che danno la soluzione esatta) ci sono errori dovuti alla memorizzazione o la propagazione.

Scriviamo la matrice $A$ come prodotto di due matrici: $A = LU$ dove $L$ è una matrice triangolare inferiore ($l_{ii} = i$ per $i = 1, \dots, n$) ed $U$ una matrice triangolare superiore. Quindi scomponiamo il problema in due sottoproblemi più semplici (infatti essendo matrici triangolari la soluzione è immediata):

$$Ax = b \Leftrightarrow L\underbrace{Ux}_{y} \Rightarrow Ly = b \Rightarrow Ux = y$$

La matrice $LU$ si ottiene tramite l'algoritmo di fattorizzazione di Gauss (vedi LAB05 e LAB06 per l'algoritmo). Si noti che per risparmiare spazio si può lavorare su una unica matrice (invece di fare le matrici separatamente, sfrutto il fatto che sull'altra diagonale hanno tutti zeri).

### Proprietà della fattorizzazione LU

Definiamo con $A_{kk}$ le sottomatrici principali di $A$. Con minori principali si intende il determinante delle sottomatrici principali di $A_{kk}$.

Se $\det A \neq 0$ e se $\det A_{kk} \neq 0$ per $k = 1, \dots, n-1$. Allora la fattorizzazione di $LU$ di $A$ esiste ed è unica.

Ci sono varie matrici che soddisfano questa proprietà (se godono di tale proprietà possiamo risparmiarci il controllo sui valori della diagonale). Ne elenchiamo alcune:

- **Matrice diagonale strettamente dominante**.
  Una matrice si dice a diagonale strettamente dominante se tutti gli elementi sulla diagonale sono maggiori della somma di tutti gli elementi successivi nella corrispettiva riga. In simboli:
  $$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \quad \text{per } i = 1, \dots, n$$
- **Matrice simmetrica definita positiva**.
  Una matrice $A$ è definita positiva se $A = A^T$ e $\underbrace{\boldsymbol{x}^T A \boldsymbol{x}}_{\substack{\text{il risultato è} \\ \text{uno scalare}}} > 0$ per ogni $x \neq 0$.

## Norme di vettori e di matrici

Sia $\|v\|$ la norma del vettore $v$ tale che $v \in \mathbb{R}^n \rightarrow \|v\| \in \mathbb{R}$. La norma gode delle seguenti proprietà:

1) $\|v\| \geq 0$ e $\|v\| = 0 \Leftrightarrow v = \underline{0}$
2) $\|\alpha v\| = |\alpha| \|v\|$ con $\alpha \in \mathbb{R}$
3) $\|v + w\| \leq \|v\| + \|w\|$ con $w \in \mathbb{R}^n$ (diseguaglianza triangolare)

Norme più conosciute:

- Norma infinito: $\|v\|_\infty = \max_{1 \leq i \leq n} |v_i|$ (prendo il massimo elemento del vettore)
- Norma due: $\|v\|_2 = \sqrt{\sum v_i}$
- Norma uno: $\|v\|_1 = \sum v_i$

$A \in \mathbb{R}^{n \times m} \rightarrow \|A\| \in \mathbb{R}$ La norma della matrice si basa sulla norma di vettore. Per definirla infatti si sceglie prima una norma di vettore$\|v\|$ e poi si calcola la norma $\|Av\|$ che sappiamo fare poiché anche $Av$ è un vettore. Infine, $\|A\| = \max_{\|v\|=1} \|Av\|$

Le proprietà sono analoghe ma ne ha una in più:

1) $\|A\| \geq 0$ e $\|A\| = 0 \Leftrightarrow A = 0$
2) $\|\alpha A\| = |\alpha| \|A\|$ con $\alpha \in \mathbb{R}$
3) $\|A + B\| \leq \|A\| + \|B\|$ con $B \in \mathbb{R}^{n \times m}$
4) $\|AB\| \leq \|A\| \|B\|$ (submoltiplicità)

Norme più conosciute:

- Norma infinito: $\|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^{n} |a_{ij}|$ (considero la somma degli elementi di ogni riga)
- Norma due: $\|v\|_2 = \sqrt{\rho(A^T A)}$ dove $\rho$ è il raggio spetrale, ossia l'autovalore di massimo modulo.
- Norma uno: $\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^{n} |a_{ij}|$ (considero la somma degli elementi di ogni colonna)

Nota: per le norme indotte vale la relazione di compatibilità $\|Av\| \leq \|A\| \|v\|$

## Condizionamento di un sistema lineare

Un sistema è ben condizionato se a piccoli errori sui dati corrispondono piccoli errori sui risultati. In caso contrario si dice mal condizionata. Sia $Ax = b$ il nostro sistema lineare con $A \in \mathbb{R}^{n\times m}$ una matrice non singolare, e consideriamo $\delta A$ la matrice di errori su $A$ e $\delta b$ il vettore di errori su $b$. Invece di $Ax = b$ sto in realtà risolvendo (considerando gli errori) il sistema lineare $(A + \delta A)(x + \delta x) = b + \delta b$, se il valore $\delta x$ è della stessa grandezza di $\delta A$ e $\delta b$ ($\delta x$ dipende solo da questi due valori, non consideriamo errori di calcolo) allora la matrice è ben condizionata.

Per vedere quanto è grave il condizionamento bisogna vedere, quindi, quanto dista il valore $\delta x$ dagli errori sui dati. Per fare ciò si usano le norme, in particolare la relazione di compatibilità $\|Av\| \leq \|A\|\|v\|$ dove la norma di matrice è indotta da quella di vettore: $\|A\| = \max_{\|v\|=1} \|Av\|$.

Consideriamo le perturbazioni relative sui dati: $\frac{\|\delta A\|}{\|A\|}$ e $\frac{\|\delta B\|}{\|B\|}$. Da questi posso ricavare gli errori relativi sui dati cercando una relazione che leghi

$$\frac{\|\delta A\|}{\|A\|} \text{ e } \frac{\|\delta B\|}{\|B\|} \text{ con } \frac{\|\delta x\|}{\|x\|}$$

## Dimostrazione: numero di condizionamento

Mettiamoci nell'ipotesi semplice $\delta A = 0$, e quindi risolviamo il sistema lineare

$$A(x + \delta x) = b + \delta b \Rightarrow Ax + A\delta x = b + \delta b \Rightarrow A\delta x = \delta b$$

Essendo $A$ invertibile:

$$\delta x = A^{-1}\delta b \Rightarrow \|\delta x\| = \|A^{-1}\delta b\| \leq \|A^{-1}\|\|\delta b\| \Rightarrow \frac{\|\delta x\|}{\|x\|} \leq \|A^{-1}\|\frac{\|\delta b\|}{\|x\|}$$

Tenendo conto del fatto che $\big||b|\big| \leq \big||A|\big|\big||x|\big| \Rightarrow \frac{1}{||x||} \leq \frac{||A||}{||b||}$ possiamo scrivere

$$\frac{\big||\delta x|\big|}{\big||x|\big|} \leq \big||A|\big| \cdot \big||A^{-1}|\big| \frac{\big||\delta b|\big|}{\big||b|\big|} \Rightarrow \frac{\big||\delta x|\big|}{\big||x|\big|} \leq K(A) \frac{\big||\delta b|\big|}{\big||b|\big|}$$

dove $K(A) = \big||A|\big| \cdot \big||A^{-1}|\big|$ è detto **numero di condizionamento** della matrice $A$.

Tale valore è sicuramente $\geq 1$ essendo $\big||A|\big| \cdot \big||A^{-1}|\big| \geq \big||AA^{-1}|\big| = \big||I|\big| = 1$. Quindi per quanto possa essere piccolo tale valore sicuramente amplifica l'errore, e in base alla grandezza di tale valore la matrice sarà ben o mal condizionata.

$K(A)$ rappresenta un fattore di amplificazione dell'errore. Praticamente il numero di condizionamento ci da la misura di quanto l'errore sui dati venga amplificato sul risultato.

Un esempio di matrice mal condizionata è la matrice di Hilbert $H$ dove $h_{ij} = \frac{1}{i+j.1}$ e quindi ha la seguente forma:

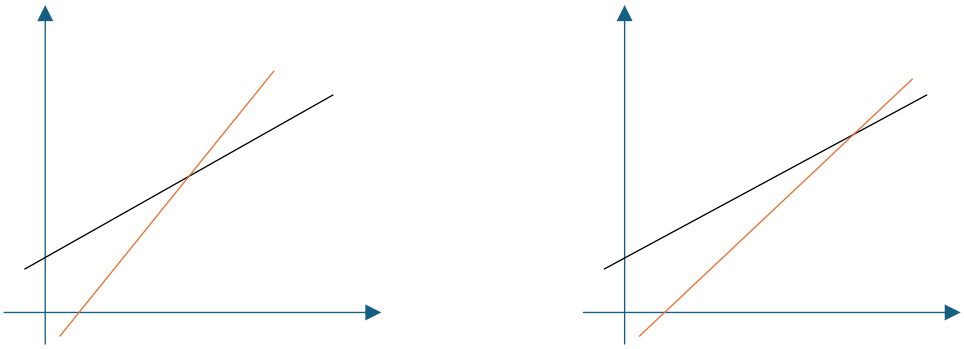
$$H_n = \begin{pmatrix} 1 & 1/2 & 1/3 & \cdots & 1/n \\ 1/2 & 1/3 & 1/4 & \cdots & 1/n+1 \\ 1/3 & 1/4 & 1/5 & \cdots & 1/n+2 \\ \vdots & \vdots & \vdots & & \vdots \\ 1/n & 1/n+1 & 1/n+2 & \cdots & 1/2n-1 \end{pmatrix}$$

$K(A)$ agisce sempre sull'amplificazione dell'errore, infatti:

$$\frac{||\delta x||}{||x||} \leq \frac{K(A)}{1 - K(A)\frac{||\delta A||}{||A||}} \left( \frac{||\delta A||}{||A||} + \frac{||\delta v||}{||b||} \right)$$

Prendiamo un esempio di una matrice $2x2$ che descrive due rette incidenti in un piano:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 = b_1 \\ a_{21}x_1 + a_{22}x_2 = b_2 \end{cases}$$



E quindi una piccola perturbazione di un dato fa si che il coefficiente angolare delle rette si muova; quindi, supponiamo di cambiare di poco questo coefficiente angolare risulterà un grande spostamento del punto di intersezione.

## Algoritmo di Thomas

Sia $A$ una matrice sparsa: ovvero sia $n_{nz}$ il numero di elementi diversi da zero, allora $n_{nz} \ll n^2$ (il numero di zeri è piccolo rispetto alla dimensione totale della matrice).

La quantità

$$s_p = \frac{n - n_{nz}}{n^2}$$

si chiama coefficiente di sparsità, cioè il rapporto tra il numero di elementi nulli e il numero di elementi totali. Tanto più il coefficiente è vicino a 1 tanto più è sparsa la matrice.

Prendiamo ad esempio la seguente eliminazione di gauss:

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & -1 \\ 1 & -1 & -2 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 1 \\ 1 & -2 & -1 \\ 1 & -1 & -3 \end{pmatrix}$$

Si noti che il triangolo superiore non conserva la natura sparsa della matrice in precedenza. Generalizzando se la matrice è sparsa dopo la riduzione il triangolo superiore non è più sparso.

Abbiamo delle matrici sparse che hanno una struttura precisa, come ad esempio la matrice tridiagonale:

$$A = \begin{pmatrix} a_{11} & a_{12} & 0 & \dots & \dots & 0 \\ a_{21} & a_{22} & a_{23} & 0 & & \vdots \\ 0 & a_{32} & \ddots & \ddots & 0 & \vdots \\ \vdots & 0 & \ddots & \ddots & a_{n-1,n-2} & 0 \\ \vdots & & \ddots & \ddots & a_{n-1,n-1} & a_{n-1,n} \\ 0 & \dots & \dots & 0 & a_{n,n-1} & a_{nn} \end{pmatrix}$$

Che sono un caso particolare di matrice a banda, ovvero una matrice che ha tutti gli elementi nulli all'esterno di una striscia diagonale.

Per queste matrici è possibile dimostrare che non si verifica il fenomeno del filling, e quindi viene conservata la struttura della matrice.

Sulle matrici sparse possiamo permetterci di eliminare delle operazioni nella fattorizzazione di LU, così da ridurne il tempo di esecuzione, che ricordiamo essere per LU di $T(n) = \frac{2}{3}n^3 + O(n^2)$.

Per la matrice tridiagonale il numero di zeri è $n_{nz} = n + 2n - 2 = 3n - 2$.

Riprendiamo l'algoritmo della fattorizzazione di LU:

```
for k = 1,...,n
    for i = k+1,...,n
        A(i,k) = A(i,k)/A(k,k)
        for j = k+1,...,n
            A(i,j) = A(i,j) - A(i,k)*A(k,j)
```

Si noti che quando si arriva a ciclare sulle righe, al passo $k = 1$ questo elemento viene ricoperto per un elemento diverso da zero. Ma dal passo successivo fino ad $n$ avrò zero; quindi, il ciclo sulle righe $i$ tecnicamente si risolve con una sola iterazione. Il ciclo sulle colonne $j$ segue lo stesso ragionamento: quando si raggiungono gli zeri i valori non cambiano; quindi, non c'è modifica dell'elemento e il ciclo si risolve in due iterazioni $k + 1$ e $k + 2$ (quello che cambiano sono solo gli elemnti diagonali)

Dunque, l'algoritmo si riduce a:

```
for k = 1,...,n
    A(k+1,k) = A(k+1,k)/A(k,k)
    A(k+1,k+1) = A(k+1,k+1) - A(k+1,k)*A(k,k+1)
```

Si noti che il costo computazionale di questo algoritmo è $T(n) = 3n$ che è un notevole risparmio computazionale.

Quindi, riconoscendo la struttura della matrice si può ridurre notevolmente il costo computazionale (si noti che per risparmiare spazio si può considerare invece della matrice solo i tre vettori diagonale).

Anche l'algoritmo di FW e BW si possono semplificare (lo si lascia come esercizio).

## Metodi indiretti o iterativi

Nel caso in cui la matrice è sparsa ma non ha una struttura conosciuta i metodi diretti non sono più efficienti. I metodi iterativi non trasformano la matrice iniziale ma sostituiscono il problema $Ax = b$ con $\det A \neq 0$ in $x = Tx + c$; vediamo dunque come sono legate le varie misure:

$$Ax = b \Rightarrow x = A^{-1}b \Rightarrow A^{-1}b = TA^{-1}b + c \Rightarrow \underbrace{(\boldsymbol{I - T})\boldsymbol{A^{-1}b = c}}_{\text{relazione di consistenza}}$$

Quindi i due problemi devono essere consistenti, ovvero avere la stessa soluzione.
La forma $x = Tx + c$ permette di partire da un vettore $x^{(0)}$ chiamato **vettore stima iniziale**, partendo da questo vettore si arriva alla seguente relazione:

$$x^{(k+1)} = Tx^{(k)} + c$$

Alla fine, ottengo una successione di vettori $\{x^{(k)}\}_{k=0,1,\dots}$. Si noti che essendo un prodotto tra matrice e vettore questa successione si può ottenere con un costo quadratico (fatto per $k$ volte).

## Tale successione è convergente? Se sì, converge alla soluzione di $Ax = b$?

Supponiamo che sia convergente quindi (il $k$ si riferisce all'iterata $k$-esima):

$$\lim_{k\to\infty} x^{(k)} = x^*$$

Dove $x^*$ è soluzione di $x = Tx + c$, ma poiché il metodo è consistente la soluzione di $x = Tx + c$ è $x^* = A^{-1}b$; dunque $x^*$ è soluzione di $Ax = b$.

Dimostriamo adesso che è convergente o perlomeno in che ipotesi è convergente:

Sia $e^{(k)} = x^{(k)} - x^*$ con $e^{(k)}$ errore all'iterata $k$ e $x^*$ soluzione del sistema; inoltre, ricordiamo che $x^* = Tx + c$ e $x^{(k+1)} = Tx^{(k)} + c$; sostituendo membro a membro:

$$x^{(k+1)} - x^* = T\left(x^{(k)} - x^*\right) \Rightarrow e^{(k+1)} = Te^{(k)} \quad \forall k = 0,1,\dots$$

Dunque, $e^{(k)} = Te^{(k-1)}, e^{(k-1)} = Te^{(k-2)}, \dots$ ma questo significa anche che

$$e^{(k)} = Te^{(k-1)} = T^2e^{(k-2)} = \cdots = T^ke^{(1)} = T^{k+1}e^{(0)}$$

Ciò significa che $\lim_{k\to\infty} x^{(k)} = x^*$ se $\lim_{k\to\infty} e^{(k)} = 0$; essendo $e^{(k)} = T^{k+1}e^{(0)}$ con $e^{(0)}$ che non dipende da $k$ abbiamo

$$\lim_{k\to\infty} e^{(k)} = e^{(0)} \cdot \lim_{k\to\infty} T^k$$

Quindi $e^{(k)} \to 0$ se e solo se $T^K \to 0$ e ciò è vero solo nel caso in cui $T$ sia una matrice convergente; ed una matrice si definisce convergente se, e solo se, il suo raggio spettrale è minore di uno: $\rho(T) < 1$.

Si ricorda che il raggio spettrale è l'autovalore di massimo modulo: $\rho(T) = \max_{1\le i\le n} |\lambda_i|$

Questo dimostra il seguente **teorema**:
Condizione necessaria e sufficiente affinché la successione $\{x^{(k)}\}$ generata dalla formula iterativa $x^{(k+1)} = Tx^{(k)} + x$ converga $\forall x^{(0)} \in \mathbb{R}^n$ è che il raggio spettrale $\rho(T) < 1$.

Si noti che se $\rho(T) \ge 1$ tale condizione non è verificate ma potrebbe comunque succedere che per un qualche $x^{(0)}$ ci sia convergenza ma non per tutti gli $x^{(0)}$.

Poiché questa condizione è un po' complicata da verificare possiamo rinunciare alla condizione necessaria e sufficiente per una condizione solo sufficiente ma di più facile verifica.
Poiché $\rho(T) < \|T\|$ allora se $\|T\| < 1$ sicuramente lo sarà anche il raggio spettrale e quindi la successione è convergente. Ma se $\|T\| \ge 1$ non è detto che il raggio spettrale non sia comunque minore di uno (la successione potrebbe lo stesso essere convergete).

**N.B.:** tale risultato è sulla matrice di iterazione $T$ e non sulla matrice di origine $A$

## Costruzione della matrice T

Scriviamo $A$ come somma di due matricei $A = M - N$ in cui $M$ è invertibile; quindi $Ax = b$ diventa $(M - N)x = b \Rightarrow Mx = Nx + b \Rightarrow x = \underbrace{M^{-1}N}_{T} x + \underbrace{M^{-1}b}_{c} = Tx + c$. Dove

$$T = M^{-1}N = M^{-1}(M - A) = \underbrace{I - M^{-1}A}_{\text{matrice di iterazione}}$$

Non c'è bisogno di verificare la consistenza poiché questa è garantita per costruzione.

La scelta di $M$ e $N$ dipende dal metodo usato, ne vediamo di seguito alcuni.

## Metodo di Jacobi

$$A = D - (L + U)$$

Nota: la matrice $L$ ed $U$ non sono le stesse della fattorizzazione LU. (ovviamente $\det A \neq 0$).

Alla matrice $D$ (che precedentemente abbiamo chiamato $M$) chiediamo di non essere invertibile, ossia che sulla sua diagonale non ci siano elementi nulli: $a_{ii} \neq 0 \ \forall i = 1, \dots, n$ (ovviamente se la matrice $A$ ha degli elementi nulli sulla diagonale posso effettuare scambi di riga per non escludere a priori l'utilizzo del metodo di Jacobi).

A questo punto la matrice di iterazione diventa $T = I - D^{-1}A$ mentre il vettore $c = D^{-1}b$

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases} \Rightarrow \begin{cases} x_1 = \dfrac{b_1 - (a_{12}x_2 + \cdots + a_{1n}x_n)}{a_{11}} \\ x_2 = \dfrac{b_2 - (a_{21}x_1 + \cdots + a_{2n}x_n)}{a_{22}} \\ \vdots \end{cases}$$

Generalizzando, ad ogni iterata sfrutto i risultati dell'iterazione precedente (parto con un vettore iniziale $x^{(0)}$:

$$(*)\ x_i^{(k+1)} = \left( b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} \right) \Big/ a_{ii} \quad \text{con } i = 1, \dots, n \text{ e } k = 1, \dots, kmax$$

Potenzialmente questo passo iterativo è infinito ma voglio fermarmi dopo un certo numero di iterazioni. Vediamo come scegliere $kmax$ al fine di ridurre il costo computazionale ($kmax$ dipende dalla velocità di convergenza e quindi devo darmi un criterio su quanto grande scegliere il $kmax$); in genere si tenta di stimare l'errore attraverso la norma di due iterazioni successive:

$$\left\| x^{(k+1)} - x^{(k)} \right\| < tol$$

Quindi daremo in input all'algoritmo una tolleranza, a tale valore si dà un numero di sicurezza $\overline{k}$ dove arresto il procedimento iterativo se $\left\| x^{(k+1)} - x^{(k)} \right\| < tol$ oppure se $k > \overline{k}$. Questo criterio di arresto è necessario per non ciclare all'infinito nel caso ci si imbatta in una successione non convergente.

In $(*)$ ho tanti passi quanti sono gli elementi diversi da zero $n_{nz}$, quindi il costo computazionale è $T(n) = 2n_{nz} * kmax$; e se le matrici sono molto sparse $n_{nz} \ll n^2$.

### *Verifica della convergenza*

Il metodo di Jacobi risolve il sistema lineare $Ax = b$ partendo da: $A = D - (L + U)$. Con $T = I - D^{-1}A$; ma vediamo in dettaglio come è composta la matrice $T$:

$$T = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix} - \begin{pmatrix} \frac{1}{a_{11}} & & \\ & \ddots & \\ & & \frac{1}{a_{nn}} \end{pmatrix} * \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} 0 & -\frac{a_{12}}{a_{11}} & \cdots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & \ddots & & \\ \vdots & & & \\ -\frac{a_{n1}}{a_{nn}} & & & 0 \end{pmatrix}$$

La matrice $T$ è dunque composta da tutti $-a_{ij}/a_{ii}$ tranne che sulla diagonale principale dove tutti gli elementi sono zeri.

A questo punto calcoliamo la norma infinito della matrice $T$ e poniamola minore di 1:

$$\|T\|_\infty = \max_{1\le i\le n} \sum_{j=1}^{n} |t_{ij}| = \max_{1\le i\le n} \sum_{\substack{j=1 \\ j\neq i}}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| < 1$$

A questo punto possiamo usare un metodo più semplice per verificare la condizione sufficiente di convergenza (si noti che gli elementi di $T$ sono combinazione degli elementi della matrice di origine $A$):

$$\max_{1\le i\le n} \sum_{\substack{j=1 \\ j\neq i}}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| < 1 \Leftrightarrow \sum_{\substack{j=1 \\ j\neq i}}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| < 1 \Leftrightarrow \frac{1}{|a_{ii}|} \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| < 1$$

Ma ciò significa che

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| < |a_{ii}| \;\; \forall i = 1, \dots, n$$

Cioè per verificare che $\|T\|_\infty < 1$ si può verificare direttamente che l'elemento sulla diagonale principale sia maggiore della somma di tutti gli altri elementi su quella riga.

**Osservazione**: Se la matrice $A$ è una matrice a diagonale strettamente dominante per righe allora il metodo di Jacobi è convergente.

### *Stima dell'errore*

Sia $x^{(k+1)} = Tx^{(k)} + c$ e supponiamo $\|T\| < 1$ (quindi il metodo è convergente).

La relazione tra gli errori è la seguente: $x^{(k+1)} - x = T\left(x^{(k)} - x\right)$. Sommando e sottraendo $x^{(k)}$ e $x^{(k+1)}$, e distribuendo la somma abbiamo:

$$x^{(k+1)} - x = T\left(x^{(k)} - x^{(k+1)}\right) + T(x^{(k+1)} - x)$$

Sfruttando le proprietà delle norme:

$$\left\|x^{(k+1)} - x\right\| = \underbrace{\left\|T\left(x^{(k)} - x^{(k+1)}\right) + T\left(x^{(k+1)} - x\right)\right\|}_{\text{norma della somma di due vettori}} \Rightarrow$$

$$\left\|x^{(k+1)} - x\right\| \le \left\|T\left(x^{(k)} - x^{(k+1)}\right)\right\| + \left\|T\left(x^{(k+1)} - x\right)\right\| \Rightarrow$$

$$(1 - \|T\|)\left\|x^{(k+1)} - x\right\| \le \|T\|\left\|x^{(k)} - x^{(k+1)}\right\| \Rightarrow$$

$$\left\|x^{(k+1)} - x\right\| \le \frac{\|T\|}{1 - \|T\|} \underbrace{\left\|x^{(k)} - x^{(k+1)}\right\|}_{\text{criterio di arresto}}$$

Vediamo anche la velocità di convergenza, che dipende dal raggio spettrale; infatti, più $\rho(T)$ è piccolo tanto più il metodo converge velocemente: $\rho(T)$ è indice della velocità di convergenza.

Questo significa che se ho due metodi convergenti con matrici di iterazioni $T_1$ e $T_2$, rispettivamente metodo 1 e metodo 2; allora, se $\rho(T_1) < \rho(T_2)$ il metodo 1 converge più velocemente del metodo 2 (praticamente il metodo 1 esce con un $k$ inferiore rispetto al metodo 2).

## Metodo di Gauss-Seidel

Ricordiamo che per il metodo di Jacobi calcoliamo le incognite nel seguente modo:

$$x_i^{(k+1)} = \left( b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} \right) \Big/ a_{ii}$$

Il metodo di Gauss-Seidel invece cerca di convergere più velocemente del metodo di Jacobi sfruttando i valori precedentemente calcolati:

$$\begin{cases} x_1^{(k+1)} = \dfrac{b_1 - \left( a_{12} x_2^{(k)} + \cdots + a_{1n} x_n^{(k)} \right)}{a_{11}} \\ x_2 = \dfrac{b_2 - \left( a_{21} x_1^{(k)} + \cdots + a_{2n} x_n^{(k)} \right)}{a_{22}} \\ \vdots \end{cases} \Rightarrow x_i^{(k+1)} = \left( b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right) \Big/ a_{ii}$$

Questo significa che invece di $A = D - (L + U)$ considero $A = (D - L) - U$ ovvero

$$Ax = b \Leftrightarrow (D - L)x^{(k+1)} = Ux^{(k)} + b$$

Invertendo $D - L$:

$$x^{(k+1)} = (D - L)^{-1} U x^{(k)} + (D - L)^{-1} b \Rightarrow x^{(k+1)} = \underbrace{(I - (D - L)^{-1} A)}_{T} x^{(k)} + \underbrace{(D - L)^{-1} b}_{c}$$

Dunque, le matrici di iterazioni sono diverse tra il metodo di Jacobi e Gauss-Seidel. Anche per quest'ultimo si dimostra che se $A$ è diagonale strettamente dominante allora il metodo di Gauss-Seidel converge (non lo dimostreremo perché è leggermente più complicato di quello di Jacobi).

Generalmente Gauss-Seidel è più veloce rispetto Jacobi ma non in tutti i casi. Infatti, se $\rho(T_{GS}) < \rho(T_J) < 1$ non è detto che Gauss-Seidel sia il metodo più veloce.

Ci sono dei casi particolari da considerare per Gauss-Seidel e Jacobi:

- Se A è simmetrica definita positiva allora $GS$ è convergente (ma non sappiamo nulla di Jacobi)
- **Teorema di Stein Rosenberg**: Se $A$ è tale che $a_{ii} > 0, i = 1, \dots, n$ e $a_{ij} \leq 0 \; \forall i, j = 1, \dots, n$ con $i \neq j$ (quindi gi elementi non diagonali) allora si verifica una delle seguenti:
  - $\rho(T_{GS}) \leq \rho(T_J) < 1$
  - $\rho(T_{GS}) = \rho(T_J) = 0$
  - $1 < \rho(T_J) < \rho(T_{GS})$
  - $1 = \rho(T_J) = \rho(T_{GS})$

  Questo teorema da una relazione tra i due metodi dicendo che non è detto che ci sia convergenza ma in questi casi se uno dei due converge allora converge anche l'altro.
  E, o hanno tutti e due la stessa velocità oppure Gauss-Seidel converge più velocemente.

## Rappresentazione di matrici sparse

Matlab permette di rappresentare matrici sparse con la funzione sparse; tale funzione costruisce una matrice passando gli elementi rappresentati in formato coordinate.

Dobbiamo dare in input tre vettori:

- $a$ contiene gli elementi di $A$ che sono diversi da $0$
- $r$ contiene gli indici di righe degli elementi di $a$
- $c$ contiene gli indici di colonne degli elementi di $a$

sparse restituisce poi una matrice in formato CSR (*Compressed Sparse Row*); tale formato utilizza solo il vettore $a$ e $c$. Tutti e tre sono vettori di lunghezza $n_{nz}$ che è molto minore di $n^2$.

Prendiamo ad esempio la seguente matrice $5 \times 5$:

$$A = \begin{pmatrix} 3 & 0 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 5 & 1 & 0 \\ 0 & 0 & 0 & 7 & 1 \\ 0 & 4 & 1 & 0 & 2 \end{pmatrix}$$

Per semplicità memorizziamo gli elementi in $a$ in ordine di riga (ma non è necessario):

- $a = (3\ 1\ 1\ 2\ 5\ 1\ 7\ 1\ 4\ 1\ 2)$ Vettore dell'elemento
- $r = (1\ 1\ 2\ 2\ 3\ 3\ 4\ 4\ 5\ 5\ 5)$ Indice di riga
- $c = (1\ 3\ 1\ 2\ 3\ 4\ 4\ 5\ 2\ 3\ 5)$ Indice di colonna

La sintassi da usare è $A = \text{sparse}(r, c, a, n_{row}, n_{col})$.

Si noti che il vettore $r$ presenta delle ridondanze; infatti, potrei usare un vettore $rowptr$ costruito nel seguente modo: ogni elemento $rowptr(i)$ è il numero di elementi diversi da zero necessari per arrivare al primo elemento diverso da zero per la riga di indice $i$ partendo dalla cella $A(1,1)$. Per la matrice $A$ dell'esempio precedente avremo $rowptr = (1\ 3\ 5\ 7\ 9\ (n+1))$ di dimensione $n+1$; in questo modo ricavo che nella prima riga ci sono $3 - 1 = 2$ elementi non nulli; quindi, prendo i primi due elementi dal vettore $a$ e recupero l'indice di colonna da $c$. Allo stesso modo, nella seconda rica ci sono $5 - 3 = 2$ elementi non nulli che ricavo prendendo i successivi due elementi di $a$ e gli indici di colonna da $c$. Dato che l'ultima riga non ha una riga successiva con cui confrontare, come ultimo elemento di $rowptr$ metto la dimensione di $a$, ossia $|a| + 1$, dato che $|a|$ è proprio il numero di elementi non nulli in $A$, nel nostro caso $|a| + 1 = 11 + 1 = 12$ e quindi per l'ultima riga avrò un numero di elementi non nulli pari a $12 - 9 = 3$.

## Jacob per matrici sparse

Vogliamo calcolare un algoritmo che prende in input i vettori $a, c, rowptr$ e dia in output $y = Ax$ conoscendo $A_{n\times n}, x_n$ e $n_{nz}$.

Riportiamo di seguito lo pseudocodice dell'algoritmo classico di Jacob:

```
for i = 1,...,n
    somma = 0
    for j = 1,..,n
        somma = somma + A(i,j)x(j)
    end
    y(i) = somma
end
```

$$T(n) = 2n^2$$

Possiamo sfruttare la memorizzazione CSR per abbattere il costo computazionale, così da non lavorare sugli elementi di riga nulli

```
for i = 1,...,n
    somma = 0
    for j = rowptr(i),..,rowptr(i+1)-1
        somma = somma + a(j)x(c(j))
    end
    y(i) = somma
end
```

In questo caso il costo computazionale è $T(n) = 2n_{nz}$

# 4. Risoluzione numerica di equazioni differenziali ordinarie

Sia $y'(t) = f\big(t, y(t)\big)$ la forma della nostra derivata, con $t \in [t_0, T]$ e $y$ incognita. Se $f$ è continua in $[t_0, T] \times \mathbb{R}$ (quindi $y$ è variabile scalare) allora esiste soluzione di $y'(t) = f\big(t, y(t)\big)$ (questa soluzione non è unica, ne esistono infinite).

Per garantire l'unicità del sistema oltre alla relazione differenziale $y'(t) = f\big(t, y(t)\big)$ abbiamo la condizione $y(t_0) = y_0$ con $t_0$ e $y_0$ noti. Inoltre, devono valere le seguenti condizioni:

1) $f$ è continua in $[t_0, T] \times \mathbb{R}$
2) $f$ è Lipschitziana rispetto a $y$, cioè $\exists L > 0 : |f(t, y) - f(t, z)| \leq L|y - z| \; \forall (t, y), (t, z) \in [0, T]$

Queste due condizioni implicano che esiste un'unica soluzione del problema e tale soluzione è derivabile con $y' \in [t_0, T]$ (quindi è continua).

Tutti i problemi in cui ci poniamo sono continui e Lipschitziani (queste condizioni possono estendersi ai sistemi lineari, basta considerare le norme al posto dei valori assoluti).

Per risolvere numericamente il problema prendiamo l'intervallo $t \in [t_0, T]$ e sostituiamo a tale intervallo un insieme di punti $I_N$ chiamato insieme di rete (per semplicità equispaziati):

$$I_N = \left\{ t_n = t_0 + nh : \; n = 0, \dots, N, h = \frac{t - t_0}{N} \right\}$$

Con $h$ chiamato passo di discretizzazione e i punti sono definiti con il precedente $t$.

## Metodo di Eulero

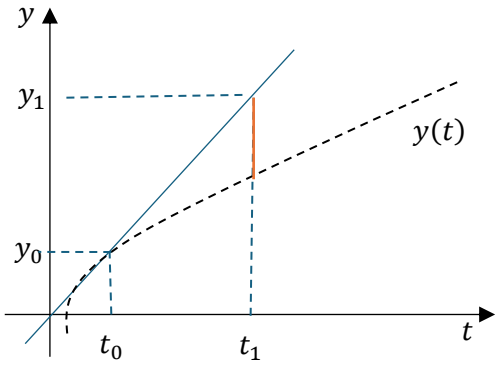
Sia un sistema di assi cartesiani con il punto iniziale noto:

$$\begin{cases} y'(t) = 2t - y \\ y(0) = 1 \end{cases}$$

$f$ ci dice il valore della derivata nella soluzione del punto $T$. La derivata nel punto $0$ è

$$y'(0) = 2(0) - y(0) = 0 - 1 = -1$$

Ma questo è possibile solo perché conosciamo $y(0)$; quindi di questo problema ho solo queste informazioni.




Supponiamo di conoscere la curva di soluzione chiamata curva fantasma come in figura (linea tratteggiata nera). Poiché conosco $y(t_0)$ conosco anche il coefficiente della retta e quindi la tangente della curva fantasma (in blu):

$$y - y_0 = y'(t_0)(t - t_0) \Rightarrow y = y_0 + f(t_0, y_0)(t - t_0)$$

Ora per avere la soluzione nel punto $t_1$, l'unica cosa che posso fare è prendere la soluzione nella retta tangente, quindi sia $y(t_1)$ la soluzione vera, io trovo la sua approssimazione $y_1 \approx y(t_0)$

Valutiamo l'equazione della tangente in $t_2$:

$$y = y_0 + f(t_0, y_0) \cdot (t - t_0) \Rightarrow y_1 = y_0 + f(t_0, y_0) \underbrace{(t_1 - t_0)}_{h} \Rightarrow y = y_0 + h\, f(t_0, y_0)$$

L'errore dell'approssimazione al tempo $t$ è pari a $y(t_1) - y_1$ con $y(t_1)$ soluzione vera.

Costruiamo adesso la tangente in $y_1$: $y = y_1 + f(t_1, y_1)\,(t - t_1) \Rightarrow y_2 = y_1 + h\,f(t_1, y_1)$ con errore $e(t_2) = y(t_2) - y_2$; ma questo dipende anche da $e(t_1) = y(t_1) - y_1$, poiché l'approssimazione di $y_1$ è dipesa da $y_0$.

Quindi oltre all'errore locale di troncamento (segnato in rosso nella figura precedente) commesso al singolo passo abbiamo anche l'errore globale dovuto alle approssimazioni precedenti.

- Errore locale di troncamento: errore commesso sul singolo passo, cioè errore commesso supponendo di partire da una soluzione esatta
- Errore globale: contributo di tutti gli errori locali (più difficile da calcolare)

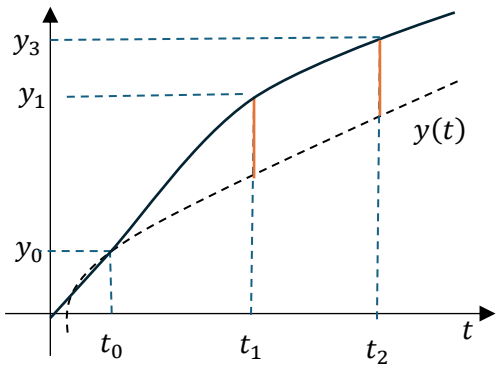
Generalizzando le iterazioni svolte con la tangente, in un generico punto $t_n$ con soluzione $y_n$:

1) Costruiamo la retta tangente alla curva soluzione che passa su $(t_n, y_n)$ con equazione
$$y = y_n + f(t_n, y_n)\,(t - t_n)$$
2) $y_{n+1}$ è l'ordinata della retta tangente corrispondente a $t_{n+1}$ con $y_{n+1} = y_n + f(t_n, y_n)\underbrace{h}_{(t_{n+1} - t_n)}$

Questo metodo prende il nome di metodo di **Eulero**.

## Calcolo dell'errore

Calcoliamo la velocità di convergenza del metodo di Eulero. L'errore locale di troncamento è $y(y_{n+1}) = y(t_{n+1}) - \tilde{y}_{n+1}$ dove $\tilde{y}_{n+1} = y(t_n) + h\,f\big(t_n, y(t_n)\big)$ è soluzione numerica a partire da valori esatti.




$$\Upsilon(t_{n+1}) = y(t_{n+1}) - \tilde{y}_{n+1} =$$
$$= y(t_{n+1}) - y(t_n) - h\,f\big(t_n, y(t_n)\big)$$

Ma $f\big(t_n, y(t_n)\big) = y'(t_n)$ quindi

$$(*)\ \ \Upsilon(t_{n+1}) = y(t_{n+1}) - y(t_n) - h y'(t_n)$$

Supponiamo che $y \in C^2[t_0, T]$ (ipotesi di localizzazione), possiamo scrivere $y(t_{n+1})$ come la somma tra il polinomio di Taylor e il resto di Lagrange:

$$y(t_{n+1}) = \underbrace{y(t_n) + y'(t_n)\,(t_{n+1} - t_n)}_{\text{Polinomio di Taylor di grado 1}} + \underbrace{y''(\xi_n)\frac{(t_{n+1} - t_n)^2}{2}}_{\text{Resto di Lagrange}} \quad \text{con} \quad \xi_n \in [t_n, t_{n+1}]$$

Usando l'ipotesi di localizzazione insieme alla $(*)$ posso scrivere:

$$y(t_{n+1}) = y(t_n) + h\,y'(t_n) + y''(\xi_n)\frac{h^2}{2}$$

Dunque, l'errore locale è pari a

$$\Upsilon(t_{n+1}) = \frac{y''(\xi_n)}{2}\,h^2 = O(h^2)$$

Per quanto riguarda l'errore globale, esso è semplicemente la differenza tra la soluzione esatta e quella approssimata: $y(t_{n+1}) - y_{n+1}$. Se calcolassi l'errore globale con

$$N\frac{y''(\xi)}{2}h^2 = \underbrace{Nh}_{T-t_0}\frac{y''(\xi)}{2}h$$

perderei una potenza di $h$ dunque $y(t_{n+1}) - y_{n+1} = O(h)$.

## Numerical ODEs

La crescita di popolazioni di organismi ha molte applicazioni nelle scienze e nell'ingegneria. Uno dei modelli più semplici prevede che la velocità di crescita della popolazione sia proporzionale, ad ogni istante $t$ di tempo, alla popolazione esistente.

Il modello esponenziale è il seguente: $\frac{dp(t)}{dt} = k_g p(t)$, dove $p(t)$ popolazione al tempo $t$ e $k_g$ la velocità di crescita.

Se la crescita non è illimitata (fattori come disponibilità di cibo, inquinamento e ambiente inibiscono la crescita), la velocità di crescita non è costante, ma è data da

$$k_g = g_g\left(\frac{1-p}{p_{\max}}\right)$$

- $k_g$ velocità di crescita massima in condizioni non limitate
- $p$ popolazione
- $p_{\max}$ popolazione massima (capacità portante)

Il modello diventa il seguente, conosciuto come modello logistico:

$$\frac{dp(t)}{dt} = k_g\left(1 - \frac{p(t)}{p_{\max}}\right)p(t)$$

ODE nella forma generica: $\frac{dy(t)}{dt} = f\big(t, y(t)\big)$.
Osservazioni:

- $f(t,y)$ è definito dal modello
- $y$ è la grandezza che si vuole rappresentare in funzione di $t$
- $t$ gioca, nella maggior parte dei casi, il ruolo del tempo
- l'equazione generale ammette una famiglia di soluzioni che dipende da una costante.

La relazione $y' = f(t,y)$ stabilisce, tramite la $f$ data, un legame tra un punto del piano $(t,y)$ e $y'(t)$ cioè il coefficiente angolare della retta tangente alla soluzione in quel punto.

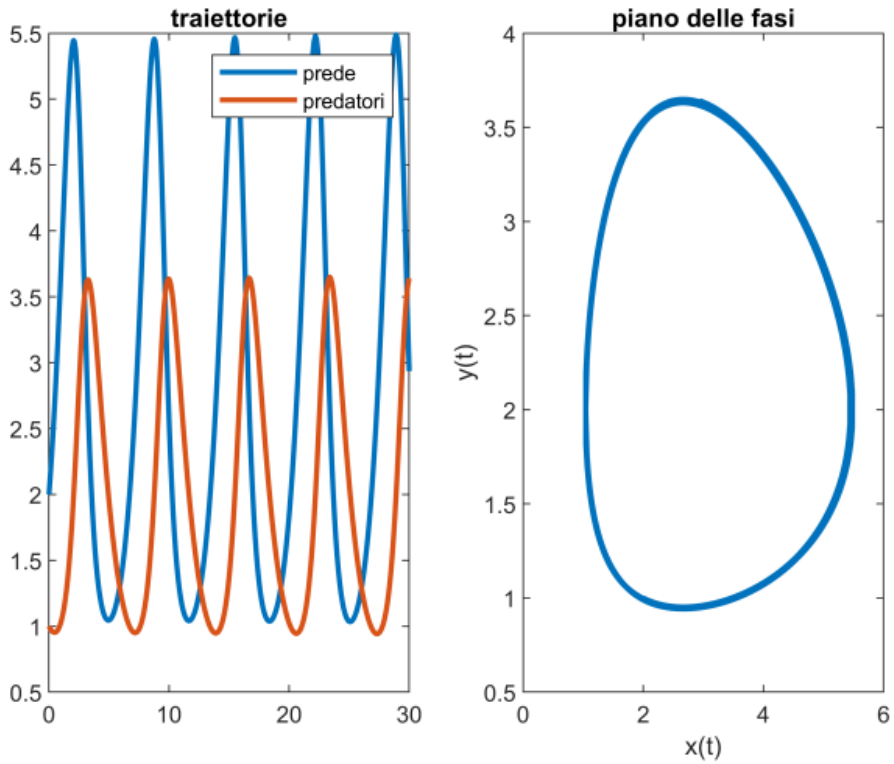
### Sistemi "famosi"

#### *Preda-predatore (equazioni di Lotka-Volterra)*

Abbiamo $x = x(t)$ prede e $y = y(t)$ predatori ed un sistema non lineare (per la presenza di $xy$) di due equazioni:

$$\begin{cases}\frac{dx}{dt} = \alpha x - \beta xy \\ \frac{dy}{dt} = \delta xy - \gamma y\end{cases}$$

- $\alpha =$ tasso di crescita delle prede
- $\gamma =$ tasso di morte dei predatori
- $\beta, \delta =$ costanti che rappresentano come gli incontri preda-predatore influenzano la morte della preda o la crescita del predatore

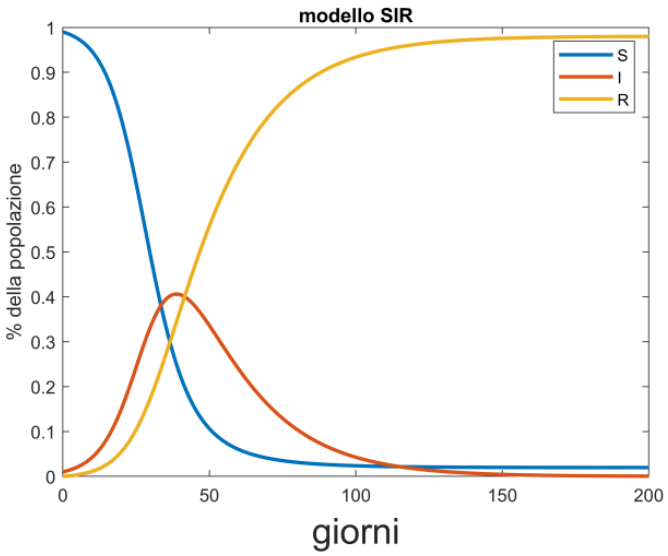



*Il modello SIR*

Classifica una popolazione, con un numero costante di individui, in tre classi:

1) $S(t)$: numero di individui che sono suscettibili ad essere infettati al tempo $t$
2) $I(t)$: numero di individui infettivi al tempo $t$
3) $R(t)$: numero di individui rimossi, cioè non più infettivi al tempo $t$, perché sono stati curati o perché sono deceduti.

$$S(t) \to I(t) \to R(t)$$




Il problema matematico è il seguente:

$$\begin{cases} \dfrac{dS(t)}{dt} = -\beta \dfrac{S(t)I(t)}{N} \\ \dfrac{dI(t)}{dt} = \beta \dfrac{S(t)I(t)}{N} - \gamma I(t) \\ \dfrac{dR(t)}{dt} = \gamma I(t) \end{cases}$$
$$S(t_0) = S_0, I(t_0) = I_0, R(t_0) = R_0$$

Dove $\beta$ è il numero di persone infettate da un infettivo (che ha incontrato solo suscettibili), e $\gamma$ è la probabilità che una persona guarisca al tempo $t$.

Si osservi che

- $R_0 = \beta/\gamma$ è il numero riproduttivo di base, cioè il numero di casi che un singolo individuo infetto può generare in una popolazione in cui tutti sono suscettibili all'infezione
- La popolazione è fissa $S(t) + I(t) + R(t) = N$
- Se $R_0 > 1$ la malattia infettiva evolve in epidemia.