Appunti di Analisi Numerica

Fabio Biselli

Revision History

Revision	Date	Author(s)	Description
1.0	01/06/2011	Fabio Biselli	Creation
1.1	08/10/2011	Andra Maria Boscolo	Revision

Indice

Parte I

Analisi Numerica.
Prof. Diligenti.

Capitolo 1

Introduzione

1.1 Risoluzione di sistemi lineari

Come si risolve il generico problema $Ax = b$, $A \in \mathbb{R}^{n \times n}$ con A non singolare?

$$b \in \mathbb{R}^n, \qquad x \in \mathbb{R}^n,$$

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}.$$

Problema: trovare le x_i.

1.1.1 Algoritmo di Cramer

Utilizzando il metodo di Cramer sulla i-esima colonna:

$$x_i = \frac{\det A_i}{\det A},$$

(il costo consiste nel calcolare $n + 1$ determinanti distinti).

$$A_i = \begin{bmatrix} a_{1,1} & \cdots & b_1 & \cdots & a_{1,n} \\ \vdots & & \vdots & & \vdots \\ a_{n,1} & \cdots & b_n & \cdots & a_{n,n} \end{bmatrix}.$$

Non è un algoritmo sequenziale cioè non è necessario trovare x_1 per calcolare x_2 etc. Posso utilizzare n processori distinti, ognuno per calcolare x_i.

$\det A = \sum_{i=1}^{n} (-1)^{i+1} a_{1,i} \det A_i$ (elimino la prima riga e l'i-esima colonna).

Quanto costa il calcolo del determinante di una matrice?
Prodotto di $a_{1,i}\det A_i$.

$c_n := n$ numero dei prodotti per il calcolo del derminate $\det A$

$c_n = nc_{n-1} + n \geq nc_{n-1} \geq n(n-1)c_{n-2} \geq \cdots \geq n(n-1)(n-2)\cdots c_2 = n!$

Stima per difetto.

$c_2 = 2$ (2 operazioni come prodotto).

$c_{n+1} = (n-1)c_{n-2} + (n-1) \geq (n-1)c_{n-2}$.

Per determinare $n+1$ determinanti: $(n+1)n! = (n+1)!$ Costo in termini di operazioni di prodotto.

1.1.2 Matrice inversa.

Vedremo che l'algoritmo di eliminazione di Gauss si baserà sulle traformazioni matriciali, intanto osserviamo un metodo alternativo all'algoritmo di Cramer: matrice inversa.

$$x = Ix = A^{-1}Ax = A^{-1}b \text{ (prodotto matrice vettore).}$$

Stesso problema di calcolare i determinanti.

$$AA^{-1} = A^{-1}A = I.$$

$$A^{-1} = \frac{[A']^t}{\det A}.$$

Ove, ricordiamo, $[A']^t$ è la matrice aggiunta, ovvero la matrice dei cofattori:

$$A = \begin{bmatrix} a_{1,1}(-1)^{1+1}\det A_{1,1} & \cdots & a_{1,n}(-1)^{1+n}\det A_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1}(-1)^{n+1}\det A_{n,1} & \cdots & a_{n,n}(-1)^{n+n}\det A_{n,n} \end{bmatrix},$$

Sia $\overline{a}_i$ la colonna della matrice di indice i di $A^{-1} \quad i = 1, 2, \ldots, n$

$$A[\overline{a}_1\, \overline{a}_2\, \ldots\, \overline{a}_i\, \ldots\, \overline{a}_n] = [e_1\, e_2\, \ldots\, e_i\, \ldots\, e_n] = I.$$

$$[A\overline{a}_1\, A\overline{a}_2\, \ldots\, A\overline{a}_i\, \ldots\, A\overline{a}_n] = [e_1\, e_2\, \ldots\, e_i\, \ldots\, e_n] = I.$$

Dovrei arrivare a trovare A^{-1} ricavando gli $\overline{a}_i$.
Ma il problema iniziale $Ax = b \quad x = ?$ Occorre molto tempo.

Vediamo quanto costa la singola operazione di prodotto di un calcolatore reale.

Esempio: 10^{-8}sec.
$t =$ tempo d'esecuzione.
$t(n+1)!$
es. $t = 10^{-6}$sec.

Ordine Matrice	Cramer	Eliminazione di Gauss
12	103 minuti	$7,110^{-8}$sec.
13	24 ore	$8,910^{-4}$sec.
14	15 gg.	$1,110^{-3}$sec.
30	$1,610^{20}$ anni	$9,910^{-3}$sec.

Meglio l'alternativa? NO.

L'analisi numerica permette di risolvere questi problemi in tempo reale (al più qualche minuto).

Ci permette di arrivare a soluzione di $f(x) = 0$ anche se non esistono forme chiuse per determinare le radici. Non esiste sempre un algoritmo di risoluzione in forma chiusa.

- $Ax = b$, l'algoritmo ci permette di avere tecniche in forma chiusa ma tempi troppo lunghi.
- $f(x) = 0$, l'analisi qui non ci fornisce la soluzione precisa ma un'approssimazione, una stima.

$$\int_a^b f(x)dx = ?$$

$f(x) \in C^0([a;b])$, $F(b) - F(a)$ so che esiste (Torricelli-Barrow) ma non so determinarla per ogni f, l'Analisi Numerica non passa attraverso la primitiva.

1.1.3 Sistemi lineari particolari.

Matrici diagonale.

Matrice diagonale.

$$A = \begin{bmatrix} a_{1,1} & 0 & 0 & \cdots & 0 \\ 0 & a_{2,2} & 0 & \cdots & \vdots \\ 0 & 0 & a_{3,3} & & 0 \\ \vdots & \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \cdots & a_{n,n} \end{bmatrix},$$

$$x_i = \frac{b_i}{a_{i,i}} \quad i = 1, \ldots, n \quad \text{(richiede n divisioni).}$$

Costo divisione tn.

Matrici triangolari.

Matrice triangolare (inferiore).

$$A = \begin{bmatrix} a_{1,1} & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & 0 & \cdots & 0 \\ a_{j,1} & \vdots & a_{i,i} & & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ a_{n,1} & \cdots & a_{n,i} & \cdots & a_{n,n} \end{bmatrix};$$

$$x_1 = \frac{b_1}{a_{1,1}};$$

$$x_2 = \frac{b_2 - a_{2,1}x_1}{a_{2,2}};$$

$$x_n = \frac{b_n - a_{n,1}x_1 - \cdots - a_{n,n-1}x_{n-1}}{a_{n,n}}.$$

Ovvero:

$$x_k = \frac{\left[b_k - \sum_{i=1}^{k-1} a_{k,i}x_i\right]}{a_{k,k}}.$$

Costo:

Divisioni: n;

Prodotti: $\frac{n(n-1)}{2}$;

Somme: $\frac{n(n-1)}{2}$.

Costo complessivo: $\frac{n^2}{2} - \frac{n}{2} = \mathrm{O}(\frac{n^2}{2})$.

Discorso analogo per le matrici triangolari superiori.

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & \cdots & a_{1,n} \\ 0 & a_{2,2} & \cdots & \cdots & \vdots \\ \vdots & 0 & a_{i,i} & \cdots & a_{i,n} \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & a_{n,n} \end{bmatrix}.$$

Il cui costo è identico e la cui formula all'indietro è la seguente:

$$x_k = \frac{\left[b_k - \sum_{i=k+1}^{n} a_{k,i} x_i\right]}{a_{k,k}}.$$

La risoluzione di una matrice triangolare quindi, ha costo pari a $\mathrm{O}(n^2)$. L'algoritmo di eliminazione di Gauss serve per riportare il caso generale al caso con A triangolare. Il costo più incidente sarà quello della trasformazione.

Capitolo 2

Algoritmo di Gauss.

Lo scopo dell'algoritmo di Gauss è quello di trasformare una matrice quadrata A in una equivalente A_n triangolare, nel nostro caso superiore.

$$A \equiv A_1, \qquad A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix}.$$

2.1 Primo passo dell'algoritmo.

Chiamiamo $a_{i,j}^{(1)} = a_{i,j}$. Poniamo quindi A_1 come segue:

$$A_1 = \begin{bmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} & \cdots & a_{1,n}^{(1)} \\ a_{2,1}^{(1)} & a_{2,2}^{(1)} & \cdots & {}^{(1)}_{2,n} \\ \vdots & & \ddots & \vdots \\ a_{n,1}^{(1)} & a_{n,2}^{(1)} & \cdots & a_{n,n}^{(1)} \end{bmatrix}.$$

$$b_1 \equiv b, \qquad b_1 = \begin{bmatrix} b_1^{(1)} \\ \vdots \\ b_n^{(1)} \end{bmatrix}.$$

Il primo passo consiste nell'eliminare (annullando) gli elementi della prima colonna al di sotto del primo elemento $a_{1,1}^{(1)}$. Ecco il nostro sistema nella forma

originale, a cui abbiamo solo applicato le etichette.

$$\begin{cases} a_{1,1}^{(1)}x_1 & a_{1,2}^{(1)}x_2 & \cdots & a_{1,n}^{(1)}x_n = b_1^{(1)} \\ a_{2,1}^{(1)}x_1 & a_{2,2}^{(1)}x_2 & \cdots & a_{2,n}^{(1)}x_n = b_2^{(1)} \\ \vdots & & \ddots & \vdots \\ a_{n,1}^{(1)}x_1 & a_{n,2}^{(1)}x_2 & \cdots & a_{n,n}^{(1)}x_n = b_n^{(1)} \end{cases}$$

Ipotesi: $a_{1,1}^{(1)} \neq 0$. Modifichiamo la seconda riga come segue:

$$r_1^{(1)}m + r_2^{(1)} \rightsquigarrow r_2^{(2)}.$$

Le $r_i^{(k)}$ rappresentano le righe della nostra matrice nella posizione i, k è il numero della trasformazione*, occorre calcolare le combinazioni lineari che consentono di annullare i coefficienti desiderati. Ovvero si deve trovare il valore di m adatto.

$$a_{1,1}^{(1)}m + a_{2,1}^{(1)} = 0, \qquad m = -\frac{a_{2,1}^{(1)}}{a_{1,1}^{(1)}},$$

Se, tramite operazioni elementari sulle righe, si moltiplica per questo m la prima riga e si somma il risultato alla seconda, si ottiene il coefficiente di x_1 uguale a 0.

$$a_{2,j}^{(2)} = a_{1,j}^{(1)}m + a_{2,j}^{(1)} \quad j = 2, \ldots, n.$$

$$b_2^{(2)} = b_1^{(1)}m + b_2^{(1)}$$

La seconda riga verrà così trasformata nella seguente:

$$\left[\begin{array}{cccc} 0 & a_{2,2}^{(2)}x_2 & a_{2,3}^{(2)}x_3 & \cdots & a_{2,n}^{(2)}x_n = b_2^{(2)} \end{array} \right].$$

Si itera il processo su tutte le rimanenti righe della matrice, vediamo il calcolo della terza riga: combinazione lineare tra la prima e la terza riga (con un nuovo coefficiente m).

$$r_1^{(1)}m + r_3^{(1)} \rightsquigarrow r_3^{(2)}.$$

$$a_{1,1}^{(1)}m + a_{3,1}^{(1)} = 0, \qquad m = -\frac{a_{3,1}^{(1)}}{a_{1,1}^{(1)}},$$

*Se una riga o un coefficiente ha apice (k) significa che ha subito $k-1$ trasformazioni.

Hp: $a_{1,1}^{(1)} \neq 0$.

$$m_{i,1} = -\frac{a_{i,1}^{(1)}}{a_{1,1}^{(1)}} \quad i = 2, \ldots, n.$$

$$a_{i,j}^{(2)} = a_{1,j}^{(1)} m_{i,1} + a_{i,j}^{(1)} \quad j = 2, \ldots, n.$$

$$b_i^{(2)} = b_1^{(1)} m_{i,1} + b_i^{(1)}.$$

Figura 2.1: Forma compatta.

$$a_{3,j}^{(2)} = a_{1,j}^{(1)} m + a_{3,j}^{(1)} \quad j = 2, \ldots, n.$$

$$b_3^{(2)} = b_1^{(1)} m + b_3^{(1)}.$$

Ottenendo così la terza riga:

$$\left[\begin{array}{cccc} 0 & a_{3,2}^{(2)} x_2 & a_{3,3}^{(2)} x_3 & \cdots \quad a_{3,n}^{(2)} x_n = b_3^{(2)} \end{array} \right].$$

Iterando quindi il procedimento alle rimanenti righe si ottiene dunque la matrice equivalente A_2 come segue:

$$A_2 = \left[\begin{array}{cccc} a_{1,1}^{(1)} & a_{1,2}^{(1)} & \cdots & a_{1,n}^{(1)} \\ 0 & a_{2,2}^{(2)} & \cdots & a_{2,n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n,2}^{(2)} & \cdots & a_{n,n}^{(2)} \end{array} \right], \qquad b_2 = \left[\begin{array}{c} b_1^{(1)} \\ b_2^{(2)} \\ b_3^{(2)} \\ \vdots \\ b_n^{(2)} \end{array} \right].$$

All'inizio si è ipotizzato che $a_{1,1}^{(1)}$ fosse diverso da 0, ma se ciò non fosse non sarebbe un problema. E' infatti sufficiente prendere la riga che abbia come primo elemento un coefficiente non nullo e scambiarla con la prima. Sappiamo che questa esiste poiché la matrice di partenza A è non singolare.

Quanto costa questo algoritmo?

Divisioni: $n - 1$.

Prodotti[†]: $(n-1)^2 + (n-1)$.

Somme[†]: $(n-1)^2 + (n-1)$.

Il costo è all'incirca $O(n^2)$, ovvero già a questo solo passaggio costa di più della risoluzione di una matrice già in forma triangolare.

[†] $(n-1)^2$ per la matrice e $(n-1)$ per il vettore dei termini noti.

2.2 Passi successivi dell'algoritmo.

In questo momento il sistema è nella seguente forma:

$$\begin{cases} a_{1,1}^{(1)}x_1 & a_{1,2}^{(1)}x_2 & \cdots & a_{1,n}^{(1)}x_n = b_1^{(1)} \\ 0 & a_{2,2}^{(2)}x_2 & \cdots & a_{2,n}^{(2)}x_n = b_2^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n,2}^{(2)}x_2 & \cdots & a_{n,n}^{(2)}x_n = b_n^{(2)} \end{cases}$$

Ora occorre eliminare i coefficienti di x_2 al di sotto di $a_{2,2}^{(2)}$. Possiamo ancora assumere l'ipotesi: $a_{2,2}^{(2)} \neq 0$ e iterare il procedimento precedente.

Hp: $a_{2,2}^{(2)} \neq 0$.

$$m_{i,2} = -\frac{a_{i,2}^{(2)}}{a_{2,2}^{(2)}} \quad i = 3, \ldots, n.$$

$$a_{i,j}^{(3)} = a_{2,j}^{(2)} m_{i,2} + a_{i,j}^{(2)} \quad j = 3, \ldots, n.$$

$$b_i^{(3)} = b_2^{(2)} m_{i,2} + b_i^{(2)}.$$

Figura 2.2: Forma compatta del secondo passo.

$$[A_3|b_3] = \left[\begin{array}{ccccc|c} a_{1,1}^{(1)} & a_{1,2}^{(1)} & a_{1,3}^{(1)} & \cdots & a_{1,n}^{(1)} & b_1^{(1)} \\ 0 & a_{2,2}^{(2)} & a_{2,3}^{(2)} & \cdots & a_{2,n}^{(2)} & b_2^{(2)} \\ 0 & 0 & a_{3,3}^{(3)} & \cdots & a_{3,n}^{(3)} & b_3^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & a_{n,3}^{(3)} & \cdots & a_{n,n}^{(3)} & b_n^{(3)} \end{array}\right].$$

Osservazione 2.1. Abbiamo potuto supporre come prima $a_{2,2}^{(2)} \neq 0$ poiché, almeno un elemento della seconda colonna è non nullo. Si noti che scambiando due righe la soluzione del sistema non cambia, invece lo scambio tra colonne cambia l'ordine delle soluzioni.

Quanto costa questo passo?

Divisioni: $n-2$.

Prodotti[‡]: $(n-2)^2 + (n-2)$.

Somme[‡]: $(n-2)^2 + (n-2)$.

[‡] $(n-2)^2$ per la matrice e $(n-2)$ per il vettore dei termini noti.

2.2.1 Passo k-esimo.

Vediamo la generalizzazione dell'algoritmo al passo $k-1$. Avremo la matrice $[A_k|b_k]$ così formata:

$$[A_k|b_k] = \left[\begin{array}{ccccccc|c} a_{1,1}^{(1)} & a_{1,2}^{(1)} & \cdots & a_{1,k}^{(1)} & \cdots & a_{1,n}^{(1)} & b_1^{(1)} \\ 0 & a_{2,2}^{(2)} & \cdots & a_{2,k}^{(2)} & \cdots & a_{2,n}^{(2)} & b_2^{(2)} \\ 0 & 0 & \ddots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & a_{k,k}^{(k)} & \cdots & a_{k,n}^{(k)} & b_k^{(k)} \\ 0 & 0 & 0 & a_{k+1,k}^{(k)} & \cdots & a_{k+1,n}^{(k)} & b_{k+1}^{(k)} \\ 0 & 0 & 0 & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & a_{n,k}^{(k)} & \cdots & a_{n,n}^{(k)} & b_n^{(k)} \end{array}\right].$$

Hp: $a_{k,k}^{(k)} \neq 0$.

$$m_{i,k} = -\frac{a_{i,k}^{(k)}}{a_{k,k}^{(k)}} \quad i = k+1, \ldots, n.$$

$$a_{i,j}^{(k+1)} = a_{k,j}^{(k)} m_{i,k} + a_{i,j}^{(k)} \quad j = k+1, \ldots, n.$$

$$b_i^{(k+1)} = b_k^{(k)} m_{i,k} + b_i^{(k)}.$$

Figura 2.3: Forma compatta del passo $k-1$.

Costo del passo $k+1$.

Divisioni: $n-k$.

Prodotti[§]: $(n-k)^2 + (n-k)$.

Somme[§]: $(n-k)^2 + (n-k)$.

[§] $(n-k)^2$ per la matrice e $(n-k)$ per il vettore dei termini noti.

2.2.2 Termine dell'algoritmo.

Dopo $n-1$ passi otteniamo la matrice $[A_n|b_n]$ così formata:

$$[A_n|b_n] = \left[\begin{array}{ccccc|c} a_{1,1}^{(1)} & a_{1,2}^{(1)} & \cdots & \cdots & a_{1,n}^{(1)} & b_1^{(1)} \\ & a_{2,2}^{(2)} & \cdots & \cdots & a_{2,n}^{(2)} & b_2^{(2)} \\ & & a_{3,3}^{(3)} & \cdots & a_{3,n}^{(3)} & b_3^{(3)} \\ & & & \ddots & \vdots & \vdots \\ & & & & a_{n,n}^{(n)} & b_n^{(n)} \end{array}\right].$$

Il vettore x si calcola dunque risolvendo il problema:

$$A_n x = b_n.$$

2.3 Costo totale dell'algoritmo.

Divisioni:

$$(n-1) + (n-2) + (n-3) + \cdots + 1 = \sum_{i=1}^{n-1} i.$$

Prodotti:

$$(n-1)^2 + (n-1) + (n-2)^2 + (n-2) + \cdots + 1^2 + 1 = \sum_{i=1}^{n-1} i^2 + \sum_{i=1}^{n-1} i.$$

Somme:

$$(n-1)^2 + (n-1) + (n-2)^2 + (n-2) + \cdots + 1^2 + 1 = \sum_{i=1}^{n-1} i^2 + \sum_{i=1}^{n-1} i.$$

$$\sum_{i=1}^{n-1} i^2 + \sum_{i=1}^{n-1} i = \frac{(n-1)^3}{3} + \frac{(n-1)^2}{2} + \frac{(n-1)}{6} + \frac{n(n-1)}{6} \simeq \mathrm{O}(\frac{n^3}{3}).$$

Costo di prodotti e somme: $\mathrm{O}(\frac{2}{3}n^3)$.

n^3 è il costo di trasformazione della matrice A di ordine n.

n^2 è il costo della soluzione di una matrice triangolare di ordine n.

$\mathrm{O}(n^3) + \mathrm{O}(n^2) = \mathrm{O}(n^3)$.

Osservazione 2.2. Questo algoritmo, se il determinante di A è diverso da zero, permette anche di calcolare il determinante stesso, a meno del segno:

$$\det A = \prod_{k=1}^{n} a_{k,k}^{(k)}.$$

Con Laplace il costo computazionale sarebbe dell'ordine di $n!$.

Osservazione 2.3. Dobbiamo risolvere due sistemi lineari: $Ax = b$ e $Ax = c$, con $A \in \mathbb{R}^{n \times n}$ non singolare. Possiamo risolvere il primo sistema e poi il secondo, ma è evidente che eseguiremmo le stesse operazioni per entrambi i sistemi.

Soluzione:

$$[A_1|b_1|c_1] \rightsquigarrow [A_n|b_n|c_n].$$

Utilizzando l'algoritmo di eliminazione di Gauss sulla matrice di n righe e $n+2$ colonne di cui sopra otteniamo i due sistemi $A_n x = b_n$ e $A_n x = c_n$.

Osservazione 2.4. Come calcolare la matrice inversa?

$$AA^{-1} = A^{-1}A = I.$$

Sia $\overline{a}_i$ la colonna della matrice di indice i di A^{-1}. $\qquad (i = 1, 2, \ldots, n)$

$$A[\overline{a}_1\, \overline{a}_2\, \ldots\, \overline{a_i}\, \ldots\, \overline{a}_n] = [e_1\, e_2\, \ldots\, e_i\, \ldots\, e_n] = I.$$

$$[A\overline{a}_1\, A\overline{a}_2\, \ldots\, A\overline{a}_i\, \ldots\, A\overline{a}_n] = [e_1\, e_2\, \ldots\, e_i\, \ldots\, e_n] = I.$$

$$\begin{array}{ccc} A\overline{a}_1 & = & e_1 \\ A\overline{a}_2 & = & e_2 \\ \vdots & \vdots & \vdots \\ A\overline{a}_n & = & e_n \end{array} \qquad [A|e_1\, e_2\, \ldots\, e_n] = [A|I].$$

$[A|I]$ risulta una matrice di n righe per $2n$ colonne. Applicando la trasformazione con l'algoritmo di Gauss otteniamo:

$$[A|I] \rightsquigarrow [A_n|\tilde{I}] = [A_n|\tilde{e}_1\, \tilde{e}_2\, \ldots\, \tilde{e}_n\,].$$

$$A_n\overline{a}_i = \tilde{e}_i \qquad i = 1, \ldots, n.$$

Ciascuna risoluzione del sistema lineare ha un costo di n^2. Per la trasformazione completa il costo non è più $\mathrm{O}(\frac{n^3}{3})$. Occorre trasformare n vettori pari a n^2 in termini di costo.

Capitolo 3

Dalle matrici elementari all'algoritmo di Gauss.

3.1 Matrici elementari.

Una matrice elementare è una matrice quadrata a coefficienti reali (o complessi) del tipo:

$$I + A,$$

dove I è la matrice quadrata identica di ordine n e A è una matrice quadrata a coefficienti reali di ordine n (assumiamo sia di tipo triangolare superiore).

Definizione 3.1. Siano $\alpha \in \mathbb{R}$, $u, v \in \mathbb{R}^n$, $v \neq 0$, si dice matrice elementare la matrice $E(\alpha, u, v)$ così definita:

$$E(\alpha, u, v) := I + \alpha u v^t.^*$$

Teorema 3.2. *Per ogni matrice elementare $E(\alpha, u, v)$ con $\alpha v^t u \neq -1$ esiste una matrice inversa e la sua inversa è $E(\beta, u, v)$.*

Dimostrazione.

Caso $\alpha = 0$:

$$E(\alpha, u, v) = I.$$

Caso $\alpha \neq 0$:

$$\begin{aligned} I &= E(\alpha, u, v)E(\beta, u, v) \\ &= (I + \alpha u v^t)(I + \beta u v^t) \\ &= (I + \alpha u v^t + \beta u v^t + \alpha u v^t \beta u v^t), \end{aligned}$$

$^*t$ indica la trasposta, in questo caso un vettore riga.

ovvero:

$$\begin{aligned}[0] &= \alpha uv^t + \beta uv^t + \alpha \underbrace{v^t u}_{scalare} \beta uv^t \\ &= (\alpha + \beta + \alpha\beta v^t u)uv^t.\end{aligned}$$

$$\Longrightarrow \alpha + \beta + \alpha\beta v^t u = 0.$$

Da $0 = \alpha + \beta + \alpha\beta v^t u$ segue che:

$$\beta = \frac{-\alpha}{1 + \alpha v^t u}.$$

Siano $x, y \in \mathbb{R}^n$, $x \neq y$.
Esiste una matrice elementare che trasforma x in y?

$$E(\alpha, u, v)x = y,$$

$$(I + \alpha uv^t)x = y,$$

$$x + \alpha uv^t x = y, \tag{3.1}$$

$v^t x \neq 0$ altrimenti $x = y$ che è contro le ipotesi[†].

$$\alpha u = \frac{y - x}{v^t x},$$

e moltiplicando la 3.1 per v^t si ha:

$$\begin{aligned}v^t x + v^t \alpha u v^t x &= v^t y \\ &= v^t x(1 + v^t \alpha u).\end{aligned}$$

Se $v^t y \neq 0$ allora $(1 + v^t \alpha u) \neq 0$, allora:

$$\alpha v^t u \neq -1.$$

■

Esercizio. Sia $x \in \mathbb{R}^n$, $x_1 \neq 0$, trovare una matrice elementare che trasformi x in un multiplo del versore e_1 in cui la costante sia data da x_1.

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad E_1 \equiv E(\alpha_1, u_1, e_1).$$

[†] v^t e x tra loro non ortogonali.

$$\begin{aligned} E(\alpha_1, u_1, e_1)x &= x_1 e_1 \\ &= (I + \alpha_1 u_1 e_1^t)x \\ &= x + \alpha_1 u_1 e_1^t x. \end{aligned}$$

$$\alpha_1 u_1 = \frac{x_1 e_1 - x}{e_1^t x}, \qquad \alpha_1 u_1 = \begin{bmatrix} 0 \\ -\frac{x_2}{x_1} \\ \vdots \\ -\frac{x_n}{x_1} \end{bmatrix}.$$

$$e_1^t x = [1, 0, \ldots, 0] \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = x_1,$$

$$\frac{x_1 e_1 - x}{e_1^t x} = \left(x_1 \cdot \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} - \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \right) \cdot \frac{1}{x_1} = \begin{bmatrix} 0 \\ -\frac{x_2}{x_1} \\ \vdots \\ -\frac{x_n}{x_1} \end{bmatrix}.$$

$$E_1 = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{x_2}{x_1} \\ \vdots \\ -\frac{x_n}{x_1} \end{bmatrix} [1, 0, \ldots, 0] = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -\frac{x_2}{x_1} & 1 & 0 & \cdots & 0 \\ -\frac{x_3}{x_1} & 0 & 1 & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ -\frac{x_n}{x_1} & 0 & \cdots & 0 & 1 \end{bmatrix}.$$

La prima colonna di E_1 è costituita dai termini del vettore $\alpha_1 u_1$ che abbiamo costruito.

$$\alpha_1 e_1^t u_1 = e_1^t(\alpha_1 u_1) = [1\, 0\, \cdots\, 0] \begin{bmatrix} 0 \\ -\frac{x_2}{x_1} \\ \vdots \\ -\frac{x_n}{x_1} \end{bmatrix} = 0.$$

Ne segue che $\alpha_1 e_1^t u_1 = 0 \neq -1$ ovvero E_1 è invertibile e $\beta = -\alpha_1$ da cui si ottiene:

$$E_1^{-1} = I - \alpha_1 u_1 e_1^t.$$

$$E_1^{-1} = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} - \begin{bmatrix} 0 \\ -\frac{x_2}{x_1} \\ \vdots \\ -\frac{x_n}{x_1} \end{bmatrix} [1, 0, \ldots, 0] = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \frac{x_2}{x_1} & 1 & 0 & \cdots & 0 \\ \frac{x_3}{x_1} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & & \vdots \\ \frac{x_n}{x_1} & 0 & \cdots & \cdots & 1 \end{bmatrix}.$$

Esercizio. Sia $x \in \mathbb{R}^n$, $x_1 = 0$, $x_2 \neq 0$, trovare una matrice elementare che trasformi x in un multiplo del versore e_2 in cui la costante sia data da x_2.

$$x = \begin{bmatrix} 0 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad E_2 \equiv E(\alpha_2, u_2, e_2).$$

$$\begin{aligned} E(\alpha_2, u_2, e_2)x &= x_2 e_2 \\ &= (I + \alpha_2 u_2 e_2^t)x \\ &= x + \alpha_2 u_2 e_2^t x. \end{aligned}$$

$$\alpha_2 u_2 = \frac{x_2 e_2 - x}{e_2^t x}, \qquad \alpha_2 u_2 = \frac{1}{x_2}\left(\begin{bmatrix} 0 \\ x_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \\ -\frac{x_3}{x_2} \\ \vdots \\ -\frac{x_n}{x_2} \end{bmatrix}.$$

Da calcoli analoghi al primo esercizio si ottiene:

$$\alpha_2 e_2^t u_2 = e_2^t(\alpha_2 u_2) = 0 \neq -1.$$

Risulta quindi E_2 invertibile e $\beta = -\alpha_2$.

$$E_2 = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & -\frac{x_3}{x_2} & 1 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & -\frac{x_n}{x_2} & 0 & \cdots & 1 \end{bmatrix}.$$

$$E_2^{-1} = I - \alpha_2 u_2 e_2^t.$$

$$E_2^{-1} = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ -\frac{x_3}{x_2} \\ \vdots \\ -\frac{x_n}{x_2} \end{bmatrix} [0\,1\,0\,\cdots\,0] = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & \frac{x_3}{x_2} & 1 & \cdots & 0 \\ \vdots & \vdots & & & \vdots \\ 0 & \frac{x_n}{x_2} & 0 & \cdots & 1 \end{bmatrix}.$$

Osservazione 3.3. Sia ora $x \in \mathbb{R}^n$, $x_1 \neq 0$, $x_i = 0$ per ogni i diverso da 1:

$$x = \begin{bmatrix} x_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

sia ha che:

$$E_2 x = x + \alpha_2 u_2 \underbrace{e_2^t x}_{=0} = x.$$

3.2 L'algoritmo di Gauss come prodotto di matrici elementari.

Rivediamo l'algoritmo di Gauss in termini matriciali.

$$A_1 = \begin{bmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} & \cdots & a_{1,n}^{(1)} \\ a_{2,1}^{(1)} & a_{2,2}^{(1)} & \cdots & {}^{(1)}_{2,n} \\ \vdots & & \ddots & \vdots \\ a_{n,1}^{(1)} & a_{n,2}^{(1)} & \cdots & a_{n,n}^{(1)} \end{bmatrix}, \quad b_1 = b = \begin{bmatrix} b_1^{(1)} \\ \vdots \\ b_n^{(1)} \end{bmatrix}.$$

Hp. $a_{1,1}^{(1)} \neq 0$ (ipotesi forte).
Per costruire E_1 abbiamo preso $x \in \mathbb{R}^n$ tale che $x_1 \neq 0$, chiamo x la prima colonna di A_1.

$$E_1 x = E_1 \begin{bmatrix} a_{1,1}^{(1)} \\ \vdots \\ a_{n,1}^{(1)} \end{bmatrix} = \begin{bmatrix} a_{1,1}^{(1)} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

$$E_1 A_1 \stackrel{?}{=} A_2 = \begin{bmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} & \cdots & a_{1,n}^{(1)} \\ 0 & a_{2,2}^{(2)} & \cdots & {}^{(2)}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n,2}^{(2)} & \cdots & a_{n,n}^{(2)} \end{bmatrix}.$$

$$E_1 A_1 = E_1 [\overline{a}_1 \, \overline{a}_2 \, \cdots \, \overline{a}_n] = [E_1 \overline{a}_1 \, E_1 \overline{a}_2 \, \cdots \, E_n \overline{a}_n].$$

Dimostrazione.

$$E_1 A_1 \stackrel{?}{=} A_2.$$

Vediamo nel dettaglio le operazioni sulle colonne.

$$E_1\overline{a}_1 = E(\alpha_1, u_1, e_1)\overline{a}_1 = (I + \alpha_1 u_1 e_1^t)\overline{a}_1 = \overline{a}_1 + \alpha_1 u_1 e_1^t \overline{a}_1.$$

$$\overline{a}_1 + \alpha_1 u_1 \cdot \underbrace{e_1^t \overline{a}_1}_{scalare:\, a_{1,1}^{(1)}} = \overline{a}_1 + \alpha_1 u_1 a_{1,1}^{(1)} = \begin{bmatrix} a_{1,1}^{(1)} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Questo risultato deriva dai seguenti passaggi:

$$\alpha_1 u_1 = \begin{bmatrix} 0 \\ -\frac{a_{2,1}^{(1)}}{a_{1,1}^{(1)}} \\ \vdots \\ -\frac{a_{n,1}^{(1)}}{a_{1,1}^{(1)}} \end{bmatrix}, \qquad e_1^t \overline{a}_1 = [1\, 0\, \cdots\, 0] \begin{bmatrix} a_{1,1}^{(1)} \\ \vdots \\ a_{n,1}^{(1)} \end{bmatrix} = a_{1,1}^{(1)}.$$

$$\alpha_1 u_1 a_{1,1}^{(1)} = \begin{bmatrix} 0 \\ -\frac{a_{2,1}^{(1)}}{\cancel{a_{1,1}^{(1)}}} \\ \vdots \\ -\frac{a_{n,1}^{(1)}}{\cancel{a_{1,1}^{(1)}}} \end{bmatrix} \cancel{a_{1,1}^{(1)}} = \begin{bmatrix} 0 \\ -a_{2,1}^{(1)} \\ \vdots \\ -a_{n,1}^{(1)} \end{bmatrix}.$$

$$\overline{a}_1 + \alpha_1 u_1 e_1^t \overline{a}_1 = \begin{bmatrix} a_{1,1}^{(1)} \\ \vdots \\ a_{n,1}^{(1)} \end{bmatrix} - \begin{bmatrix} 0 \\ a_{2,1}^{(1)} \\ \vdots \\ a_{n,1}^{(1)} \end{bmatrix} = \begin{bmatrix} a_{1,1}^{(1)} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Cosa succede alle altre colonne?

$$E_1\overline{a}_j = ? \qquad j = 2, \ldots, n.$$

$$E_1\overline{a}_j = E(\alpha_1, u_1, e_1)\overline{a}_j = (I + \alpha_1 u_1 e_1^t)\overline{a}_j = \overline{a}_j + \alpha_1 u_1 e_1^t \overline{a}_j.$$

$$E_1\overline{a}_j = \begin{bmatrix} a_{1,j}^{(1)} \\ a_{2,j}^{(1)} \\ \vdots \\ a_{n,j}^{(1)} \end{bmatrix} + \begin{bmatrix} 0 \\ m_{2,1} \\ \vdots \\ m_{n,1} \end{bmatrix} \overbrace{a_{1,j}^{(1)}}^{scalare} = \begin{bmatrix} a_{1,j}^{(1)} \\ a_{2,j}^{(1)} + m_{2,1} a_{1,j}^{(1)} \\ \vdots \qquad \vdots \\ a_{n,j}^{(1)} + m_{n,1} a_{1,j}^{(1)} \end{bmatrix} = \begin{bmatrix} a_{1,j}^{(1)} \\ a_{2,j}^{(2)} \\ \vdots \\ a_{n,j}^{(2)} \end{bmatrix}.$$

■

Osservazione 3.4. Tornando alle matrici si nota che:

$$E_1 = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -\frac{x_2}{x_1} & 1 & 0 & \cdots & 0 \\ -\frac{x_3}{x_1} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & & \vdots \\ -\frac{x_n}{x_1} & 0 & \cdots & \cdots & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -\frac{a^{(1)}_{2,1}}{a^{(1)}_{1,1}} & 1 & 0 & \cdots & 0 \\ -\frac{a^{(1)}_{3,1}}{a^{(1)}_{1,1}} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & & \vdots \\ -\frac{a^{(1)}_{n,1}}{a^{(1)}_{1,1}} & 0 & \cdots & \cdots & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ m_{2,1} & 1 & 0 & \cdots & 0 \\ m_{3,1} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & & \vdots \\ m_{n,1} & 0 & \cdots & \cdots & 1 \end{bmatrix}.$$

E_1 è la matrice identità in cui colloco sotto al primo elemento i termini di eliminazione $m_{i,1}$. Inoltre la matrice è non singolare, quindi ammette inversa che si ottiene eliminando il segno "$-$" dalla prima colonna, ed il calcolo dell'inversa in termini computazionali ha costo nullo.

Analogamente il secondo passo dell'algoritmo di Gauss si ottiene come segue:
Hp. $a^{(2)}_{2,2} \neq 0$ (ipotesi forte).

$$E_2 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & m_{3,2} & 1 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & m_{n,2} & \cdots & \cdots & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} a^{(1)}_{1,1} & a^{(1)}_{1,2} & \cdots & a^{(1)}_{1,n} \\ 0 & a^{(2)}_{2,2} & \cdots & a^{(2)}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a^{(2)}_{n,2} & \cdots & a^{(2)}_{n,n} \end{bmatrix}.$$

$$E_2(E_1A_1) = E_2A_2 \stackrel{?}{=} A_3.$$

$$E_2A_2 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & m_{3,2} & 1 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & m_{n,2} & \cdots & \cdots & 1 \end{bmatrix} \begin{bmatrix} a^{(1)}_{1,1} & a^{(1)}_{1,2} & \cdots & a^{(1)}_{1,n} \\ 0 & a^{(2)}_{2,2} & \cdots & a^{(2)}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a^{(2)}_{n,2} & \cdots & a^{(2)}_{n,n} \end{bmatrix}.$$

Osservazione 3.5. Si potrebbe anche lavorare sulla sottomatrice di A_2 di ordine $n-1$ (anche nei passi successivi, con ordine decrescente) della forma:

$$\begin{bmatrix} a_{2,2}^{(2)} & \cdots & {}_{2,n}^{(2)} \\ \vdots & \ddots & \vdots \\ a_{n,2}^{(2)} & \cdots & a_{n,n}^{(2)} \end{bmatrix}$$

ma, per i nostri scopi, è utile utilizzare la matrice intera.

Dimostrazione.

$$E_2 A_2 \stackrel{?}{=} A_3.$$

Vediamo nel dettaglio le operazioni sulle colonne.

Prima colonna:

$$E_2\overline{a}_1^{(2)} = E(\alpha_2, u_2, e_2)\overline{a}_1^{(2)} = (I + \alpha_2 u_2 e_2^t)\overline{a}_1^{(2)} = \overline{a}_1^{(2)} + \alpha_2 u_2 e_2^t \overline{a}_1^{(2)}.$$

$$\overline{a}_1^{(2)} + \alpha_2 u_2 \underbrace{e_2^t \overline{a}_1^{(2)}}_{=0} = \overline{a}_1^{(2)} = \begin{bmatrix} a_{1,1} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Seconda colonna:

$$E_2\overline{a}_2^{(2)} = E(\alpha_2, u_2, e_2)\overline{a}_2^{(2)} = (I + \alpha_2 u_2 e_2^t)\overline{a}_2^{(2)} = \overline{a}_2^{(2)} + \alpha_2 u_2 e_2^t \overline{a}_2^{(2)}.$$

$$\overline{a}_2^{(2)} + \alpha_2 u_2 \cdot \underbrace{e_2^t \overline{a}_2^{(2)}}_{scalare:\, a_{2,2}^{(2)}} = \overline{a}_2^{(2)} + \alpha_2 u_2 a_{2,2}^{(2)} = \begin{bmatrix} a_{2,1}^{(1)} \\ a_{2,2}^{(2)} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Questo risultato deriva dai seguenti passaggi:

$$\alpha_2 u_2 = \begin{bmatrix} 0 \\ 0 \\ -\frac{a_{3,2}^{(2)}}{a_{2,2}^{(2)}} \\ \vdots \\ -\frac{a_{n,2}^{(2)}}{a_{2,2}^{(2)}} \end{bmatrix}, \qquad e_2^t \overline{a}_2^{(2)} = [0\,1\,0\,\cdots\,0] \begin{bmatrix} a_{1,2}^{(1)} \\ a_{2,2}^{(2)} \\ \vdots \\ a_{n,2}^{(2)} \end{bmatrix} = a_{2,2}^{(2)}.$$

$$\alpha_2 u_2 a_{2,2}^{(2)} = \begin{bmatrix} 0 \\ 0 \\ -\frac{a_{3,2}^{(2)}}{\cancel{a_{2,2}^{(2)}}} \\ \vdots \\ -\frac{a_{n,2}^{(2)}}{\cancel{a_{2,2}^{(2)}}} \end{bmatrix} \cancel{a_{2,2}^{(2)}} = \begin{bmatrix} 0 \\ 0 \\ -a_{3,2}^{(2)} \\ \vdots \\ -a_{n,2}^{(2)} \end{bmatrix}.$$

$$\overline{a}_2^{(2)} + \alpha_2 u_2 e_2^t \overline{a}_2^{(2)} = \begin{bmatrix} a_{1,2}^{(1)} \\ a_{2,2}^{(2)} \\ \vdots \\ a_{n,2}^{(2)} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ a_{3,2}^{(2)} \\ \vdots \\ a_{n,2}^{(2)} \end{bmatrix} = \begin{bmatrix} a_{1,2}^{(1)} \\ a_{2,2}^{(2)} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Cosa succede alle altre colonne?

$$E_2 \overline{a}_j^{(2)} = ? \qquad j = 3, \ldots, n.$$

$$E_2 \overline{a}_j^{(2)} = E(\alpha_2, u_2, e_2)\overline{a}_j^{(2)} = (I + \alpha_2 u_2 e_2^t)\overline{a}_j^{(2)} = \overline{a}_j^{(2)} + \alpha_2 u_2 e_2^t \overline{a}_j^{(2)}.$$

$$E_2 \overline{a}_j^{(2)} = \begin{bmatrix} a_{1,j}^{(1)} \\ a_{2,j}^{(2)} \\ \vdots \\ a_{n,j}^{(2)} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ m_{3,2} \\ \vdots \\ m_{n,2} \end{bmatrix} \overbrace{a_{2,j}^{(2)}}^{scalare} = \begin{bmatrix} a_{1,j}^{(1)} \\ a_{2,j}^{(2)} \\ a_{3,j}^{(2)} + m_{3,2} a_{2,j}^{(2)} \\ \vdots \qquad \vdots \\ a_{n,j}^{(2)} + m_{n,2} a_{2,j}^{(2)} \end{bmatrix} = \begin{bmatrix} a_{1,j}^{(1)} \\ a_{2,j}^{(2)} \\ a_{3,j}^{(3)} \\ \vdots \\ a_{n,j}^{(3)} \end{bmatrix}.$$

Ne segue che:

$$\left[E_2 \overline{a}_1^{(2)} \; E_2 \overline{a}_2^{(2)} \; \cdots \; E_2 \overline{a}_n^{(2)} \right] = \begin{bmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} & a_{1,3}^{(1)} & \cdots & a_{1,n}^{(1)} \\ 0 & a_{2,2}^{(2)} & a_{2,3}^{(2)} & \cdots & a_{2,n}^{(2)} \\ 0 & 0 & a_{3,3}^{(3)} & \cdots & a_{3,n}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & a_{n,3}^{(3)} & \cdots & a_{n,n}^{(3)} \end{bmatrix}.$$

■

3.3 Prodotto di matrici elementari inverse.

Domanda: cosa fa il prodotto di due matrici inverse elementari?

$$E_1^{-1}E_2^{-1} = \quad ?$$

$$E_1 \equiv E(\alpha_1, u_1, e_1) = (I + \alpha_1 u_1 e_1^t).$$

$$E_2 \equiv E(\alpha_2, u_2, e_2) = (I + \alpha_2 u_2 e_2^t).$$

$$\begin{aligned} E_1^{-1}E_2^{-1} &= (I - \alpha_1 u_1 e_1^t)(I - \alpha_2 u_2 e_2^t) \\ &= I - \alpha_1 u_1 e_1^t - \alpha_2 u_2 e_2^t + \alpha_1\alpha_2 u_1 e_1^t u_2 e_2^t \\ &= I - \alpha_1 u_1 e_1^t - \alpha_2 u_2 e_2^t + \alpha_1 \cdot \underbrace{\alpha_2 e_1^t u_2}_{= e_1^t(\alpha_2 u_2) = 0} u_1 e_2^t \\ &= I - \alpha_1 u_1 e_1^t - \alpha_2 u_2 e_2^t. \end{aligned}$$

Tornando alla forma matriciale:

$$E_1^{-1}E_2^{-1} = I - \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ m_{2,1} & 0 & 0 & \cdots & 0 \\ m_{3,1} & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & & \vdots \\ m_{n,1} & 0 & \cdots & \cdots & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & m_{3,2} & 0 & \cdots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & m_{n,2} & \cdots & \cdots & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -m_{2,1} & 1 & 0 & \cdots & 0 \\ -m_{3,1} & -m_{3,2} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -m_{n,1} & -m_{n,2} & 0 & \cdots & 1 \end{bmatrix}.$$

Praticamente vengono "copiati" gli elementi della prima e della seconda colonna, cambiati di segno.

Osservazione 3.6. $E_2^{-1}E_1^{-1}$ non porta allo stesso risultato.

Esercizio. Sia $x \in \mathbb{R}^n$, $x_1 = x_2 = \cdots = x_{k-1} = 0$, $x_k \neq 0$ si vuole costruire una matrice elementare $E_k(\alpha_k, u_k, e_k)$ tale che:

$$E_k(\alpha_k, u_k, e_k)x = x_k e_k.$$

- Interpretare nell'ambito dell'algoritmo delle trasformazioni di Gauss.

- Costruire l'inversa.

- Costruire e verificare $E_1^{-1}E_2^{-1}\cdots E_{k-1}^{-1}E_k^{-1}$.

- Costo necessario per costruire l'inversa e il prodotto di:

$$E_1^{-1}E_2^{-1}\cdots E_{k-1}^{-1}E_k^{-1}.$$

Osservazione 3.7. ***Algoritmo di Gauss mediante moltiplicazione di matrici elementari***: Data la matrice $A = A_1$ con l'ipotesi fondamentale che $a_{k,k}^{(k)} \neq 0 \qquad k = 1, \ldots, n-1$ e date le matrici elementari E_k costruite come sopra, allora:

$$A_n = E_{n-1}E_{n-2}\cdots(E_2(E_1(A)))$$

Da cui:

$$E_1^{-1}E_2^{-1}\cdots E_{n-2}^{-1}E_{n-1}^{-1}A_n = A_1$$

Con $E_1^{-1}E_2^{-1}\cdots E_{n-2}^{-1}E_{n-1}^{-1}$ matrice triangolare inferiore e A_n matrice triangolare superiore. Ovvero A è *fattorizzabile* [‡] nel prodotto di due matrici: una triangolare superiore ed una triangolare inferiore.

3.4 Matrici di Permutazione.

Come si è visto nella sezione 3.2, da un problema $Ax = b$, mediante l'algoritmo di Gauss sotto forma di "pre-moltiplicazioni" di matrici elementari si può giungere alla definizione di un problema equivalente nella forma $A_n x = b_n$, con A_n di tipo triangolare superiore. Ovvero:

$$E_{n-1}E_{n-2}\cdots E_2E_1A_1 = A_n,$$

$$E_{n-1}E_{n-2}\cdots E_2E_1b_1 = b_n.$$

Ogni passo di questo algoritmo prevede un'ipotesi forte sugli elementi della diagonale della matrice di partenza: $a_{j,j}^{(j)} \neq 0$. Questo discorso cade quando troviamo un elemento diagonale nullo, ma come osservato nel secondo capitolo (2.1), è possibile effettuare scambi di righe per ottenere un elemento diagonale non nullo. Si utilizzano quindi le *matrici di permutazione.*

[‡] O *decomponibile.*

$$I = \begin{bmatrix} 1 & 0 & & \cdots & & & 0 \\ 0 & \ddots & & & & & \\ & 0_i & 1_i & 0_i & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & \\ & & & 0_j & 1_j & 0_j & \\ & & & & & & 0 \\ 0 & & \cdots & & & 0 & 1 \end{bmatrix} \quad P_{ij} = \begin{bmatrix} 1 & 0 & & \cdots & & & 0 \\ 0 & \ddots & & & & & \\ & & & 0_j & 1_j & 0_j & \\ \vdots & & & & & & \vdots \\ & 0_i & 1_i & 0_i & & & \\ & & & & & \ddots & 0 \\ 0 & & \cdots & & & 0 & 1 \end{bmatrix}.$$

Esempio. Matrice di ordine 4 che scambia la prima con la terza riga.

$$I_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad P_{1,3} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Per scambiare due righe in una matrice A è sufficiente applicare la relativa matrice di permutazione, ovvero pre-moltiplicare A per $P_{i,j}$.

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,i} & \cdots & a_{1,j} & \cdots & a_{1,n} \\ \vdots & & & & & & \\ a_{i,1} & \cdots & a_{i,i} & \cdots & a_{i,j} & \cdots & a_{i,n} \\ \vdots & & & & & & \\ a_{j,1} & \cdots & a_{j,i} & \cdots & a_{j,j} & \cdots & a_{j,n} \\ \vdots & & & & & & \\ a_{n,1} & \cdots & a_{n,i} & \cdots & a_{n,j} & \cdots & a_{n,n} \end{bmatrix},$$

$$P_{i,j}A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,i} & \cdots & a_{1,j} & \cdots & a_{1,n} \\ \vdots & & & & & & \\ a_{j,1} & \cdots & a_{j,i} & \cdots & a_{j,j} & \cdots & a_{j,n} \\ \vdots & & & & & & \\ a_{i,1} & \cdots & a_{i,i} & \cdots & a_{i,j} & \cdots & a_{i,n} \\ \vdots & & & & & & \\ a_{n,1} & \cdots & a_{n,i} & \cdots & a_{n,j} & \cdots & a_{n,n} \end{bmatrix}.$$

$P_{i,j}$ applicata alla matrice A scambia la riga i-esima con la riga j-esima.

Osservazione 3.8. Un applicazione di $P_{i,j}$ del tipo $AP_{i,j}$ invece scambia le colonne.

i

3.5 L'algoritmo di Gauss con matrici elementari e di permutazione.

Come si è visto nella sezione 3.2, è possibile ottenere l'algoritmo di Gauss mediante l'applicazione di matrici elementari. Nel caso in cui, ad un passo (j-esimo) dell'algoritmo, si abbia che $a_{j,j} = 0$ occorre quindi applicare anche una matrice di permutazione. Si può quindi riscrivere l'algoritmo come segue:

$$A_1 x = b_1.$$

Caso $a_{1,1} \neq 0$: $P_{1,1}A_1$, $P_{1,1}b_1$.

Caso $a_{1,1} = 0$: $P_{1,j}A_1$, $P_{1,j}b_1$.

Interpretando $P_{i,i}$ come la matrice identica I.

Si applicano quindi le matrici elementari:

Caso $a_{1,1} \neq 0$: $E_1P_{1,1}A_1$, $E_1P_{1,1}b_1$.

Caso $a_{1,1} = 0$: $E_1P_{1,j}A_1$, $E_1P_{1,j}b_1$.

$$E_1P_{1,j}A_1 = A_2.$$

Analogamente per il secondo passo:

Caso $a_{2,2} \neq 0$: $E_2P_{2,2}A_2$, $E_2P_{2,2}b_2^{(2)}$.

Caso $a_{2,2} = 0$: $E_2P_{2,j}A_2$, $E_2P_{2,j}b_2^{(2)}$.

Alla fine dell'algoritmo si otterrà la seguente situazione:

$$E_{n-1}P_{n-1,j} \cdots E_2P_{2,j}E_1P_{1,j}A_1 = A_n.$$
$$E_{n-1}P_{n-1,j} \cdots E_2P_{2,j}E_1P_{1,j}b_1 = b_n.$$

Posto il caso in cui $a_{j,j} = 0$ quale riga conviene scegliere per sostituire la j-esima? Si ha un'amplificazione dell'errore minore quando $m_{i,j}$ è il più piccolo possibile. Questo si ottiene scegliendo il massimo in modulo degli elementi al di sotto dell'elemento $a_{j,j}$ pivotale. In questo modo per ogni i, $m_{i,j} \leq 1$ minimizzando quindi l'errore. Tale operazione viene definita *pivoting parziale di righe* ed è sempre applicabile, non necessariamente se l'elemento $a_{j,j}$ è uguale a zero.

Capitolo 4

Fattorizzazione LU.

Definizione 4.1. Sia $A_1 \equiv A$, $Ax = b$, con $A \in \mathbb{R}^{n\times n}$, il solito problema lineare, dal capitolo precedente sappiamo esistere un algoritmo che trasforma la matrice quadrata di ordine n, A in una equivalente di tipo triangolare superiore:

$$E_{n-1}P_{n-1,j}\cdots E_2P_{2,j}E_1P_{1,j}A_1 = A_n.$$

Allora $A_n \equiv U$.

Osservazione 4.2. Se $P_{j,j} \equiv I$ (ovvero $a_{j,j} \neq 0$) per $j = 1, \ldots, n-1$ allora si ha:

$$E_{n-1}\cdots E_2E_1A_1 = U.$$

4.1 Fattorizzazione LU senza permutazioni.

Come si è dedotto dal teorema 3.2, per ogni matrice elementare E_i è possibile calcolarne l'inversa, per cui "lavorando" sulle matrici tramite moltiplicazioni riga per colonna da $E_{n-1}\cdots E_2E_1A_1 = U$ si ottiene:

$$\begin{aligned}
&\cancel{E_{n-1}^{-1}}\cancel{E_{n-1}}E_{n-2}\cdots E_2E_1A_1 = E_{n-1}^{-1}U,\\
&\cancel{E_{n-2}^{-1}}\cancel{E_{n-2}}\cdots E_2E_1A_1 = E_{n-2}^{-1}E_{n-1}^{-1}U,\\
&\qquad\vdots\\
&\cancel{E_1^{-1}}\cancel{E_1}A_1 = E_1^{-1}E_2^{-1}\cdots E_{n-2}^{-1}E_{n-1}^{-1}U,\\
&A_1 = \underbrace{\left(E_1^{-1}E_2^{-1}\cdots E_{n-2}^{-1}E_{n-1}^{-1}\right)}_{*}U.
\end{aligned}$$

Espandendo $E_1^{-1}E_2^{-1}\cdots E_{n-2}^{-1}E_{n-1}^{-1}$ $(*)$ si ottiene:

$$
\begin{aligned}
* &= (I-\alpha_1u_1e_1^t)(I-\alpha_2u_2e_2^t)\cdots(I-\alpha_{n-1}u_{n-1}e_{n-1}^t)\\
&= (I-\alpha_1u_1e_1^t-\alpha_2u_2e_2^t+\underbrace{\alpha_1u_1e_1^t\alpha_2u_2e_2^t}_{e_1^t(\alpha_2u_2)=\,0})(I-\alpha_3u_3e_3^t)\cdots(I-\alpha_{n-1}u_{n-1}e_{n-1}^t)\\
&= (I-\alpha_1u_1e_1^t-\alpha_2u_2e_2^t)(I-\alpha_3u_3e_3^t)\cdots(I-\alpha_{n-1}u_{n-1}e_{n-1}^t)\\
&= (I-\alpha_1u_1e_1^t-\alpha_2u_2e_2^t-\alpha_3u_3e_3^t+\underbrace{\cancel{\alpha_2u_2e_2^t\alpha_3u_3e_3^t}}_{e_2^t(\alpha_3u_3)=\,0}+\underbrace{\cancel{\alpha_1u_1e_1^t\alpha_2u_2e_2^t}}_{e_1^t(\alpha_2u_2)=\,0}\cdots(I-\alpha_{n-1}u_{n-1}e_{n-1}^t\\
&= (I-\alpha_1u_1e_1^t-\alpha_2u_2e_2^t-\alpha_3u_3e_3^t)\cdots(I-\alpha_{n-1}u_{n-1}e_{n-1}^t)\\
&= I-\alpha_1u_1e_1^t-\alpha_2u_2e_2^t-\alpha_3u_3e_3^t-\cdots-\alpha_{n-1}u_{n-1}e_{n-1}^t.
\end{aligned}
$$

Passando alla notazione matriciale:

$$
* = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} - \begin{bmatrix} 0 & \cdots & 0 \\ m_{2,1} & & 0 \\ \vdots & & \vdots \\ m_{n,1} & & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & m_{3,2} & & \vdots \\ \vdots & \vdots & & \\ 0 & m_{n,2} & & 0 \end{bmatrix} - \cdots
$$

$$
\cdots - \begin{bmatrix} 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ & & 0 & \vdots \\ 0 & & m_{n,n-1} & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -m_{2,1} & 1 & 0 & \cdots & 0 \\ -m_{3,1} & -m_{3,2} & 1 & & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ -m_{n,1} & -m_{n,2} & \cdots & -m_{n,n-1} & 1 \end{bmatrix} = L.
$$

L è quindi una matrice triangolare inferiore. Si può dunque vedere la matrice A come il prodotto di L ed U.

$$A = LU.$$

Osservazione 4.3. Dato il problema $Ax = b$, per quanto si è visto, è possibile riscriverlo come:

$$LUx = b.$$

Sia ora $Ay = c(x)$ un sistema lineare i cui termini noti siano in funzione delle soluzione del sistema precedente. Come visto nell'osservazione 2.3, vorremmo

risolvere entrambi i sistemi lineari in un colpo solo, risparmiando conti. Dato il legame che unisce la soluzione del primo sistema con la soluzione del secondo, questo non è immediatamente realizzabile. Tuttavia se sappiamo esistere una fattorizzazione LU della nostra matrice A, allora possiamo scrivere:

$$A = LU, \quad L\underbrace{Ux}_{z} = b.$$

Che equivale a risolvere il seguente sistema:

$$\begin{cases} Lz = b \\ Ux = z \end{cases}.$$

Una volta ricavata la soluzione di questo sistema (tramite sostituzioni in avanti e indietro) è sufficiente risolvere il sistema seguente:

$$L\underbrace{Uy}_{t} = c(x),$$

$$\begin{cases} Lt = c(x) \\ Uy = t \end{cases}.$$

Il costo della risoluzione del problema $Ay = c(x)$ è quindi $\mathrm{O}(n^2)$. In questo modo abbiamo utilizzato la fattorizzazione LU una sola volta, risolvendo poi sistemi lineari a costo inferiore.

4.2 Fattorizzazione LU con permutazioni.

Partiamo dall'ipotesi che, utilizzando l'algoritmo di Gauss per ottenere la matrice triangolare superiore U, si "incontrino" elementi diagonali nulli. Come visto in precedenza è possibile procedere con l'algoritmo invertendo l'ordine delle righe mediante l'utilizzo di matrici di permutazione. Ma cosa accade alla matrice $E_{n-1}P_{n-1,} \cdots E_2P_{2,}E_1P_1$? E' ancora una matrice triangolare inferiore?

Esempio. Caso semplice:

$$A_1 = A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix}.$$

$$E_3P_3E_2P_2E_1P_1A_1 = U. \tag{4.1}$$

$$E_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ m_{2,1} & 1 & 0 & 0 \\ m_{3,1} & 0 & 1 & 0 \\ m_{4,1} & 0 & 0 & 1 \end{bmatrix},$$

$$P_2E_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ m_{4,1} & 0 & 0 & 1 \\ m_{3,1} & 0 & 1 & 0 \\ m_{2,1} & 1 & 0 & 0 \end{bmatrix}.$$

In questo esempio P_2 inverte la seconda con la quarta riga. Come si vede non risulta più una matrice triangolare inferiore.

Utilizzando la seguente proprietà delle matrici elementari:

$$P_{i,j}P_{i,j} = I.$$

E' Possibile quindi rivedere la (4.1) come:

$$E_3P_3E_2P_2E_1IP_1A_1 = U, \quad I = P_2P_2.$$

Allora: $E_3P_3E_2P_2E_1P_2P_2P_1A_1 = U$.

$$E_3P_3E_2(P_2E_1P_2)P_2P_1A_1 = U.$$

$$E_1^{(1)} = (P_2E_1)P_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ m_{4,1} & 1 & 0 & 0 \\ m_{3,1} & 0 & 1 & 0 \\ m_{2,1} & 0 & 0 & 1 \end{bmatrix}.$$

Tale matrice è nuovamente di tipo triangolare inferiore, poichè P_2E_1 scambia la seconda con la quarta riga, mentre $(P_2E_1)P_2$ scambia la seconda con la quarta colonna.

Riscriviamo la (4.1):

$$E_3P_3E_2E_1^{(1)}P_2P_1A_1 = U.$$

$$E_3P_3E_2IE_1^{(1)}IP_2P_1A_1 = U.$$

$$E_3(P_3E_2P_3)(P_3E_1^{(1)}P_3)P_3P_2P_1A_1 = U.$$

Poste:

$$E_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & m_{3,2} & 1 & 0 \\ 0 & m_{4,2} & 0 & 1 \end{bmatrix}, \quad P_3E_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & m_{4,2} & 0 & 1 \\ 0 & m_{3,2} & 1 & 0 \end{bmatrix}.$$

Ora P_3 inverte la terza con la quarta riga. Come si vede non risulta più una matrice triangolare inferiore.

$$(P_3E_2)P_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & m_{4,2} & 1 & 0 \\ 0 & m_{3,2} & 0 & 1 \end{bmatrix} = E_2^{(1)}.$$

$$P_3E_1^{(1)}P_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ m_{4,1} & 1 & 0 & 0 \\ m_{3,1} & 0 & 1 & 0 \\ m_{2,1} & 0 & 0 & 1 \end{bmatrix} = E_1^{(2)}.$$

Ne segue che l'equazione si può riscrivere come:

$$\underbrace{E_3E_2^{(1)}E_1^{(2)}}_{mat.ele.}P_3P_2P_1A_1 = U.$$

$$P_3P_2P_1A_1 = E_1^{(2)^{-1}}E_2^{(1)^{-1}}E_3^{-1}U,$$

ovvero: $PA = LU$.

Possiamo generalizzare questi conti nella seguente:

Definizione 4.4. Data una matrice A non singolare, *esiste sempre* una matrice di permutazione P tale che il prodotto PA è fattorizzabile da due matrici, una triangolare inferiore ed una triangolare superiore (U data dall'algoritmo di Gauss e L a diagonale unitaria avente elementi opposti agli elementi $m_{i,j}$ di eliminazione).

4.3 Fattorizzazione generalizzata.

Siano P_i matrici generiche di permutazione, per $i = 1, \ldots, n-1$ (può ivi comparire anche la matrice identica I), possiamo quindi generalizzare la fattorizzazione LU, a partire dall'algoritmo di Gauss, come segue:

$$E_{n-1}P_{n-1,}\cdots E_2P_{2,}E_1P_{1,}A_1 = U.$$

$PA = LU \longrightarrow P = P_{n-1}P_{n-2}\cdots P_2P_1$.

Primo passo:

$$E_{n-1}P_{n-1,}\cdots E_2\underbrace{P_{2,}E_1P_{2,}}_{E_1^{(1)}}P_{2,}P_{1,}A_1 = U.$$

Secondo passo:

$$E_{n-1}P_{n-1,}\cdots E_3\underbrace{P_3E_2P_3}_{E_2^{(1)}}\underbrace{P_3E_1^{(1)}P_3}_{E_1^{(2)}}P_3P_2P_1A_1=U.$$

Terzo passo:

$$E_{n-1}P_{n-1,}\cdots E_4\underbrace{P_4E_3P_4}_{E_3^{(1)}}\underbrace{P_4E_2^{(1)}P_4}_{E_2^{(2)}}\underbrace{P_4E_1^{(3)}P_4}_{E_1^{(3)}}P_4P_3P_2P_1A_1=U.$$

k-esimo passo

$$E_{n-1}P_{n-1,}\cdots E_{k+1}\underbrace{P_{k+1}E_kP_{k+1}}_{E_k^{(1)}}E_{k-1}^{(2)}E_{k-2}^{(3)}\cdots E_2^{(k-1)}E_1^{(k)}P_{k+1}\cdots P_4P_3P_2P_1A_1=U.$$

Abbiamo visto che a meno di una matrice P di permutazione, la fattorizzazione LU è sempre possibile. E' chiaro che se durante l'algoritmo di Gauss non si fa mai uso di scambi di righe* o di colonne, non si utilizzano matrici di permutazione, quindi $P=I$. Quindi per sapere se per qualche j (che si può vedere come passo dell'algoritmo di Gauss) $P_j=I$, occorre eseguire ed analizzare tutti i passi? Cosa è possibile analizzare su A? In questo senso ci viene in aiuto il seguente:

Teorema 4.5. *Sia A una matrice di ordine n non singolare tale che tutti i minori principali Δ_k $(k=1,\ldots,n)$ siano invertibili.*

$$\Delta_k=\begin{bmatrix} a_{1,1} & \cdots & a_{1,k} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{k,k} \end{bmatrix}.$$

Allora esiste una matrice L triangolare inferiore con $l_{i,i}=1,\ (i=1,\ldots,n)$ e una matrice triangolare superiore U tale che:

$$A=LU.$$

Inoltre tale fattorizzazione è unica.

*Ad esempio: l'ipotesi $a_{k,k}^{(}k)\neq 0$ è sempre verificata, e in generale non mi interessa fare pivoting

Dimostrazione. Sia $\Delta_k \subseteq A_1$ non singolare, per $k = 1, \dots, n$.

$$A_1 = \begin{bmatrix} a_{1,1}^{(1)} & \cdots & a_{1,k}^{(1)} & \cdots & a_{1,n}^{(1)} \\ \vdots & \ddots & \vdots & & \vdots \\ a_{k,1}^{(1)} & \cdots & a_{k,k}^{(1)} & \cdots & a_{k,n}^{(1)} \\ \vdots & & \vdots & \ddots & \vdots \\ a_{n,1}^{(1)} & \cdots & a_{n,k}^{(1)} & \cdots & a_{n,n}^{(1)} \end{bmatrix}.$$

$\det\Delta_k \neq 0$.

$$\Delta_1 = [a_{1,1}^{(1)}], \quad \Delta_2 = \begin{bmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} \\ a_{2,1}^{(1)} & a_{2,2}^{(1)} \end{bmatrix}.$$

Si dimostra per induzione su k.

- Caso base: $k = 1$.

  $\det\Delta_1 \neq 0 \Rightarrow a_{1,1}^{(1)} \neq 0$, dunque $P_1 \equiv I$.

- Passo induttivo:
  Ipotesi induttiva: $P_{k-1} = P_{k-2} \cdots P_2 = P_1 = I$.

  Tesi: $P_k \equiv I$.

  Tramite il passo induttivo si è costruito $A_k = (E_{k-1} \cdots E_1)A_1$:

$$A_k = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ * & 1 & 0 & \cdots & 0 \\ * & * & 1 & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ * & * & \cdots & * & 1 \end{bmatrix} \begin{bmatrix} a_{1,1}^{(1)} & \cdots & a_{1,k}^{(1)} & \cdots & a_{1,n}^{(1)} \\ \vdots & \ddots & \vdots & & \vdots \\ a_{k,1}^{(1)} & \cdots & a_{k,k}^{(1)} & \cdots & a_{k,n}^{(1)} \\ \vdots & & \vdots & \ddots & \vdots \\ a_{n,1}^{(1)} & \cdots & a_{n,k}^{(1)} & \cdots & a_{n,n}^{(1)} \end{bmatrix} =$$

$$= \begin{bmatrix} a_{1,1}^{(1)} & \cdots & a_{1,k-1}^{(1)} & a_{1,k}^{(1)} & \cdots & a_{1,n}^{(1)} \\ 0 & a_{2,2}^{(2)} & \cdots & a_{2,k}^{(2)} & \cdots & a_{2,n}^{(2)} \\ \vdots & & \ddots & \vdots & & \vdots \\ 0 & & & a_{k,k}^{(k)} & \cdots & a_{k,n}^{(k)} \\ \vdots & & & \vdots & & \vdots \\ 0 & & & a_{n,k}^{(k)} & \cdots & a_{n,n}^{(k)} \end{bmatrix}.$$

  Possiamo vedere A_k come una matrice a blocchi:

$$\left[\begin{array}{c|c} A_{1,1}^{(k)} & A_{1,2}^{(k)} \\ \hline A_{2,1}^{(k)} & A_{2,2}^{(k)} \end{array}\right] = \left[\begin{array}{c|c} B_{1,1} & B_{1,2} \\ \hline B_{2,1} & B_{2,2} \end{array}\right] \left[\begin{array}{c|c} A_{1,1}^{(1)} & A_{1,2}^{(1)} \\ \hline A_{2,1}^{(1)} & A_{2,2}^{(1)} \end{array}\right].$$

Come si ottiene $A_{1,1}^{(k)}$? Prodotto righe per colonne.

$$\left[\begin{array}{c|c} B_{1,1}A_{1,1}^{(1)} + \cancel{B_{1,2}A_{2,1}^{(1)}} & B_{1,1}A_{1,2}^{(1)} + B_{1,2}A_{2,2}^{(1)} \\ \hline B_{2,1}A_{1,1}^{(1)} + B_{2,2}A_{2,1}^{(1)} & B_{2,1}A_{1,2}^{(1)} + B_{2,2}A_{2,2}^{(1)} \end{array}\right].$$

Si noti che $B_{1,2}$ è la matrice nulla.

Si ha dunque che $A_{1,1}^{(k)} = B_{1,1}A_{1,1}^{(1)}$, i rimanenti blocchi non sono utili, poiché occorrono solo elementi che stanno sulla diagonale principale.

$$\det\left(A_{1,1}^{(k)}\right) = \det(B_{1,1})\det\left(A_{1,1}^{(1)}\right). \qquad \text{(Binet)}$$

$$\prod_{i=1}^{k} a_{i,i}^{(i)} = 1 \cdot \det(\Delta_k) \neq 0.$$

Allora:

$$a_{i,i}^{(i)} \neq 0 \qquad i = 1, \ldots, k.$$

Ovvero per ogni i si ha quindi che $P_i = I$. Questo dimostra l'esistenza di due matrici di tipo triangolare inferiore e triangolare superiore che fattorizzano A. Non resta che dimostrarne l'unicità.

Sia A una matrice di ordine n non singolare. Siano L una matrice triangolare inferiore ed U una matrice triangolare superiore di ordine n tali che:

$$A = LU.$$

Per quanto dimostrato sopra sappiamo che esistono.

Ipotesi: $(L)_{i,i} = 1, \quad 1 \leq i \leq n.$

Siano ora L_1, L_2 e U_1, U_2 tali che:

$$A = L_1U_1,$$

$$A = L_2U_2,$$

$$(L_1)_{i,i} = (L_2)_{i,i} = 1, \quad i = 1, \ldots, n.$$

Tesi: $L_1U_1 = L_2U_2$.

$$L_2^{-1}L_1U_1 = L_2^{-1}L_2U_2 = IU_2.$$
$$L_2^{-1}L_1 = U_2U_1^{-1}.$$

$$L_2^{-1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ * & 1 & 0 & \cdots & 0 \\ * & * & 1 & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ * & * & \cdots & * & 1 \end{bmatrix} \xrightarrow{\text{infatti}} \begin{cases} L_2\overline{a}_1 = e_1 \\ \vdots \\ L_2\overline{a}_n = e_n \end{cases}$$ Ma L_2 è già triangolare.

$$L_2^{-1}L_1 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ *' & 1 & 0 & \cdots & 0 \\ *' & *' & 1 & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ *' & *' & \cdots & *' & 1 \end{bmatrix}.$$

$$(L_2^{-1}L_1)_{i,j} = \begin{cases} 0 & j > i \\ 1 & j = i \\ *' & \text{altrimenti.} \end{cases}$$

Come risulta $U_2U_1^{-1}$?

Come si calcola U_1^{-1}?
$U_1x = e_1$, $U_2x = e_2, \ldots U_nx = e_n$ come nell'esercizio 3.1.

$$U_1^{-1} = \begin{bmatrix} * & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ 0 & 0 & * & \cdots & * \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & * \end{bmatrix}.$$

$$L_2^{-1}L_1 = U_2U_1^{-1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ *' & 1 & 0 & \cdots & 0 \\ *' & *' & 1 & & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ *' & *' & \cdots & *' & 1 \end{bmatrix} = \begin{bmatrix} * & * & * & \cdots & * \\ 0 & * & * & \cdots & * \\ 0 & 0 & * & \cdots & * \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & * \end{bmatrix}.$$

Per ottenere l'uguaglianza occorre che gli elementi $*$ al di sopra della diagonale principale di $U_2U_1^{-1}$ siano tutti nulli come gli elementi $*'$ al di sotto della

diagonale principale di $L_2^{-1}L_1$. Inoltre gli elementi della diagonale di $U_2U_1^{-1}$ devono essere tutti uguali a 1. Da questo segue che:

$$L_2^{-1}L_1 = I \quad \wedge \quad U_2U_1^{-1} = I.$$

Ovvero $L_2 = L_1$ e $U_2 = U_1$.

■

Si analizzi ora l'ipotesi che $(L)_{i,i} = c$ con $i = 1, \ldots, n$ e $c \neq 0$. Si ottiene così una differente fattorizzazione, se ne ha una unica fissando c. Se dovesse cadere questa ipotesi?

$$A = LU \text{ è ancora unica?}$$

Vediamo un esempio semplice:

$$L = \begin{bmatrix} l_{1,1} & 0 \\ l_{2,1} & l_{2,2} \end{bmatrix} \qquad U = \begin{bmatrix} u_{1,1} & u_{1,2} \\ 0 & u_{2,2} \end{bmatrix}$$

In questo esempio abbiamo 4 equazioni in 6 incognite. Ovvero ∞^2 possibili soluzioni. Se fissiamo $l_{1,1} = l_{2,2} = 1$ otteniamo l'unicità.

Generalizzando:

$$L = \begin{bmatrix} l_{1,1} & 0 & \cdots & 0 \\ l_{2,1} & l_{2,2} & & \vdots \\ \vdots & \vdots & \ddots & 0 \\ l_{n,1} & l_{n,2} & \cdots & l_{n,n} \end{bmatrix} \qquad U = \begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n} \\ 0 & u_{2,2} & \cdots & u_{2,n} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & u_{n,n} \end{bmatrix}$$

Numero di equazioni: n^2.
Numero di incognite: $n + (n-1) + \cdots + 2 + 1 = \frac{n(n+1)}{\cancel{2}} \cdot \cancel{2} = n^2 + n$.
Occorre, quindi, scegliere in modo opportuno i parametri o porre uguale a 1 gli elementi diagonali.

Abbiamo quindi visto che nella fattorizzazione $PA = LU$, $P = I$ se e solo se la matrice A e tutti i suoi minori risultano non singolari. Esiste un'altra proprietà verificabile che ci consente di fattorizzare $A = LU$ direttamente?

Teorema 4.6. *Sia $A \in \mathbb{R}^{n\times n}$ non singolare diagonal dominante in senso stretto, ovvero*

$$|a_{i,i}| > \sum_{\substack{j=1 \\ i\neq j}}^{n} |a_{i,j}|, \qquad i = 1, \ldots, n.$$

allora la sua decomposizione LU è ottenibile senza ricorrere a tecniche di pivoting.

4.4 Matrici Tridiagonali

Vediamo ora qualche risultato analogo utilizzando matrici tridiagonali.

Definizione 4.7. Sia $A \in \mathbb{R}^{n\times n}$ una matrice tridiagonale:

$$A = \begin{bmatrix} a_1 & c_1 & 0 & \cdots & 0 \\ b_2 & a_2 & c_2 & \ddots & \vdots \\ 0 & b_3 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & a_{n-1} & c_{n-1} \\ 0 & \cdots & 0 & b_n & a_n \end{bmatrix}.$$

Si definiscono:
$d_0 := 1;$
$d_1 := a_1;$
$\vdots$
$d_k := a_k d_{k-1} - b_k c_{k-1} d_{k-2}. \qquad (k = 2, \ldots, n)$

Proposizione 4.8. *Sia $A \in \mathbb{R}^{n\times n}$ una matrice tridiagonale definita come sopra, allora:*

$$(\forall k)(1 \leq k \leq n) \Rightarrow (d_k = det\Delta_k \wedge detA = d_n).$$

Dimostrazione. $d_n \stackrel{?}{=} \det A$.
Si dimostra per induzione su n.

Caso base: $n = 1$.

$d_1 = a_1 = \det\Delta_1$.

Ipotesi induttiva: $\Delta_{n-1} = d_{n-1}$.

Tesi: $\det A = d_n$.

$$\begin{array}{rcl} \det A & = & a_n \det\Delta_{n-1} + b_n(-1)\det[*] \\ & \stackrel{hp.ind}{=} & a_n d_{n-1} - b_n c_{n-1} d_{n-1} \\ & = & d_n. \end{array}$$

Su A abbiamo applicato lo sviluppo di Laplace, $[*]$ è la matrice su cui abbiamo applicato lo sviluppo sull'ultima colonna.

■

Osservazione 4.9. Tale proposizione, se vista come algoritmo di risoluzione del determinante ha costo pari a $O(3(n-1))$. Ovvero ha costo lineare (strettamente minore a $n^{3\dagger}$). La struttura della matrice quindi, cambia pesantemente il costo computazionale.

Come si calcola il polinomio caratteristico di una matrice tridiagonale?

$$A - \lambda I = \begin{bmatrix} a_1 - \lambda & c_1 & 0 & \cdots & 0 \\ b_2 & a_2 - \lambda & c_2 & \ddots & \vdots \\ 0 & b_3 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & a_{n-1} - \lambda & c_{n-1} \\ 0 & \cdots & 0 & b_n & a_n - \lambda \end{bmatrix}.$$

$$\det(A - \lambda I)?$$

$d_0(\lambda) := 1;$
$d_1(\lambda) := a_1 - \lambda;$
$\vdots$
$d_k(\lambda) := (a_k - \lambda)(d_{k-1}(\lambda) - b_k c_{k-1} d_{k-2}(\lambda)). \qquad (k = 2, \ldots, n)$

$d_n(\lambda)$ è il polinomio caratteristico di A.

Osservazione 4.10. Non si calcola $d_n(\lambda)$ (nella sua espressione generica), serve solo il valore di $d_n(\lambda)$ dato un certo $\overline{\lambda}$.

Proposizione 4.11. *Sia A una matrice tridiagonale:*

$$A = \begin{bmatrix} a_1 & c_1 & 0 & \cdots & 0 \\ b_2 & a_2 & c_2 & \ddots & \vdots \\ 0 & b_3 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & a_{n-1} & c_{n-1} \\ 0 & \cdots & 0 & b_n & a_n \end{bmatrix},$$

$d_0 := 1;$
$d_1 := a_1;$
$\vdots$
$d_k := a_k d_{k-1} - b_k c_{k-1} d_{k-2}. \qquad (k = 2, \ldots, n),$

†Sarebbe il costo del calcolo del determinante in caso si sia fatto uso dell'algoritmo di Gauss. Si veda l'osservazione [**?**]

tale che $d_k \neq 0$ *per* $k = 1, \dots, n$*, allora* A *è direttamente fattorizzabile in* $A = LU$*.*

$$L = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ b_2\frac{d_0}{d_1} & 1 & 0 & \ddots & \vdots \\ 0 & b_3\frac{d_1}{d_2} & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 1 & 0 \\ 0 & \cdots & 0 & b_n\frac{d_{n-2}}{d_{n-1}} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} \frac{d_1}{d_0} & c_1 & 0 & \cdots & 0 \\ 0 & \frac{d_2}{d_1} & c_2 & \ddots & \vdots \\ 0 & 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \frac{d_{n-1}}{d_{n-2}} & c_{n-1} \\ 0 & \cdots & 0 & 0 & \frac{d_n}{d_{n-1}} \end{bmatrix}.$$

Sia L *che* U *sono due matrici bidiagonali, il costo di costruzione è "quasi" gratuito poiché è dato dal solo colacolo dei* d_i*.*

Dimostrazione. $A \stackrel{?}{=} LU$.
E' sufficiente dimostrare che per ogni $k = 1, \dots, n$:

- $(LU)_{k,k} = a_k$.

  $k = 1$:

  $$1 \cdot \frac{d_1}{d_0} = 1 \cdot \frac{a_1}{1} = a_1.$$

  $k = 2, \dots, n$:

  $$b_k \frac{d_{k-1}}{d_{k-2}} \cdot c_{k-1} + 1 \cdot \frac{d_k}{d_{k-1}} = \frac{\cancel{b_k c_{k-1} d_{k-2}} + a_k \cancel{d_{k-1}} \cancel{- b_k c_{k-1} d_{k-2}}}{\cancel{d_{k-1}}} = a_k.$$

- $(LU)_{k,k-1} = b_k$.

  $k = 2$:

  $$b_2 \frac{d_0}{d_1} \cdot \frac{d_1}{d_0} = b_2 \cdot 1 = b_2.$$

  $k = 3, \dots, n$:

- $(LU)_{k,k+1} = c_k$.

  $k = 1$:

  $k = 2, \dots, n$:

- $(LU)_{i,j} = 0, \quad \text{se } |i - j| \geq 2.$

$k = 1$:

$k = 2, \ldots, n$:

■

In questo caso la soluzione di $Ax = t$ è data da:

$$(LU)x = t \equiv \begin{cases} Ly = t \\ Ux = y. \end{cases}$$

$y_1 = t_1$;

$y_k = t_k - b_k \frac{d_{k-2}}{d_{k-1}} y_{k-1}, \qquad (k = 2, \ldots, n)$

$x_n = y_n \frac{d_{n-1}}{d_n}$,

$x_k = (y_k - c_k x_{k+1}) \frac{d_{k-1}}{d_k}. \qquad (k = n-1, \ldots, 1)$

4.5 Fattorizzazione LU diretta.

Abbiamo visto dal teorema [**?**] l'esistenza (ed unicità), sotto opportune ipotesi, di una fattorizzazione LU diretta di una matrice A. Ci preoccupiamo ora di costruire tale fattorizzazione senza far uso dell'algoritmo di Gauss.

Siano A, L, U del teorema precedente:

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{bmatrix}, \qquad \Delta_k \neq 0, \quad k = 1, \ldots, n.$$

$$L = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{2,1} & 1 & & \\ \vdots & & \ddots & \\ l_{n,1} & \cdots & l_{n,n-1} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n} \\ 0 & u_{2,2} & & u_{2,n} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & u_{n,n} \end{bmatrix}.$$

Si noti che la prima riga di L è nota, da questo punto possiamo quindi calcolare la prima riga di U. In questo modo abbiamo trovato la prima colonna di U e si può quindi calcolare la prima colonna di L.

Iterando il discorso è quindi possibile trovare la seconda riga di U e quindi la seconda colonna di L e così via. Vediamo in concreto.

- La prima riga di L è nota: $[1\,0\,\cdots\,0]$, si può calcolare la prima riga di U:

$$1 \cdot u_{1,i} = a_{1,i}. \quad (i = 1, \ldots, n)$$

$$[a_{1,1}\, a_{1,2}\, \cdots\, a_{1,n}] = [u_{1,1}\, u_{1,2}\, \cdots\, u_{1,n}]\,.$$

- Ora, dalla prima colonna di U calcoliamo la prima colonna di L:

$$l_{i,1} \cdot u_{1,1} = a_{i,1}. \quad (i = 1, \ldots, n)$$

$$\overline{a}_1 = \begin{bmatrix} a_{1,1} \\ \vdots \\ a_{n,1} \end{bmatrix}, \quad \overline{u}_1 = \begin{bmatrix} u_{1,1} \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \overline{l}_1 = \begin{bmatrix} \frac{a_{1,1}}{u_{1,1}} \\ \vdots \\ \frac{a_{n,1}}{u_{1,1}} \end{bmatrix}.$$

- Possiamo quindi fissare la seconda riga di L:

$$l_2 = \left[\frac{a_{2,1}}{u_{1,1}}\ 1\ 0\ \cdots\ 0\right].$$

e calcolare la seconda riga di U:

$$l_{2,i} \cdot u_{1,i} + 1 \cdot u_{2,i} = a_{2,i}. \quad (i = 1, \ldots, n)$$

$$[l_{2,1}\ \cdots\ l_{2,n}]\,[u_{1,1}\ \cdots\ u_{1,n}] + 1 \cdot [u_{2,1}\ \cdots\ u_{2,n}] = [a_{2,1}\ \cdots\ a_{2,n}]\,.$$

$$u_{2,i} = a_{2,i} - l_{2,i}u_{1,i}. \quad (i = 1, \ldots, n)$$

$$u_2 = [(a_{2,1} - l_{2,1}u_{1,1})\ (a_{2,2} - 1 \cdot u_{1,2})\ (a_{2,3} - 0 \cdot u_{1,2}) \cdots\ (a_{2,n} - 0 \cdot u_{1,i})]\,.$$

$$u_2 = [(a_{2,1} - l_{2,1}u_{1,1})\ (a_{2,2} - 1 \cdot u_{1,2})\ (a_{2,3}) \cdots\ (a_{2,n})]\,.$$

- E' ora possibile fissare la seconda colonna di U e dedurre quindi gli elementi della seconda colonna di L:

$$\overline{u}_2 = \begin{bmatrix} u_{1,2} \\ a_{2,2} - 1 \cdot u_{1,2} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

$$l_{i,1}u_{1,2} + \underbrace{l_{i,2}}_{\text{variabile}} \cdot u_{2,2} = a_{i,2}.$$

$$l_{i,2} = \frac{a_{i,2} - l_{i,1}u_{1,2}}{u_{2,2}}.$$

Generalizzando l'algoritmo si ottiene:

- $k = 1, \ldots, n$;
  - $j = k, \ldots, n$;

$$u_{k,j} = a_{k,j} - \sum_{p=1}^{k-1} l_{k,p} u_{p,j}.$$

  - $i = k+1, \ldots, n$;

$$l_{i,k} = \frac{\left[a_{i,k} - \sum_{p=1}^{k-1} l_{i,p} u_{p,k}\right]}{u_{k,k}}.$$

Tale algoritmo, nella pratica, viene anche chiamato algoritmo ***riga-colonna***.

Osservazione 4.12. E' possibile salvare le matrici L ed U direttamente entro la matrice A. Basta procedere per righe se si considerano le righe di U, e per colonne se si considerano le colonne di L.

Osservazione 4.13. Il costo dell'algoritmo *riga-colonna* è pari a $O(\frac{n}{3})$.

Ricapitolando: se sappiamo che A è non singolare allora sappiamo che esiste P tale che $PA = LU$. Ci chiediamo: quando $P \equiv I$? Nell'algoritmo di Gauss ad ogni passo in cui non servono permutazioni, in quest'altro caso ogni volta che il minore è diverso da 0.

Capitolo 5

Altre fattorizzazioni.

5.1 Verso la fattorizzazione di Cholesky.

Proposizione 5.1. *Sia $A \in \mathbb{R}^{n\times n}$, simmetrica ($A = A^t$) e definita positiva ($\forall x \in \mathbb{R}^n : x \neq 0, \quad x^tAx > 0$), allora A è non singolare.*

Dimostrazione. Sia A singolare, allora esiste $x \in \mathbb{R}^n$ con $x \neq 0$ tale che $Ax = 0$.

$$x^t \underbrace{Ax}_{=0} \overset{?}{>} 0. \qquad 0 \not> 0. \qquad \text{Assurdo.}$$

■

Proposizione 5.2. *I minori Δ_k della matrice A di cui sopra sono definiti positivi, non singolari.*

Dimostrazione. Posto Δ_k minore principale per $k = 1, \ldots, n$.

Δ_k è simmetrico, sia $v_k \in \mathbb{R}^k$ con $v_k \neq 0$, allora esiste $w \in \mathbb{R}^n$ tale che:

$$w = \begin{bmatrix} v_k \\ 0_{k+1} \\ \vdots \\ 0_n \end{bmatrix}, \quad w \neq 0.$$

$$0 < w^tAw = \begin{bmatrix} v_k^t & 0_{k+1} & \cdots & 0_n \end{bmatrix} \left[\begin{array}{c|c} \Delta_k & A_{1,2} \\ \hline A_{2,1} & A_{2,2} \end{array}\right] \begin{bmatrix} v_k \\ 0_{k+1} \\ \vdots \\ v_n \end{bmatrix} = v_k^t \Delta_k v_k.$$

Pertanto $v_k^t \Delta_k v_k > 0$ per ogni $v_k \in \mathbb{R}^k$ con $v_k \neq 0$. Allora Δ_k è definito postivo, non singolare.

■

Osservazione 5.3. A è non singolare e tutti i minori principali sono non nulli per il teorema 4.5.

Osservazione 5.4. Anche la matrice A^{-1} è definita positiva.

$$\forall y \in \mathbb{R}^n, \quad y \neq 0, \quad A^{-1}y = x, \quad x \neq 0.$$

Analizziamo $y^t A y$.

Se $y^t A y > 0$ allora A^{-1} è positiva.

$$(A^{-1}y)^t = x^t \implies y^t A^{-1} = x^t \implies y^t = x^t A.$$

Osserviamo che $A^{-1} = (A^{-1})^t$ poichè A è simmetrica.

$$y^t A^{-1} y = x^t A A^{-1} y = x^t \underbrace{A A^{-1}}_{=\, I} A x = x^t A x > 0.$$

Com'è il determinante della matrice A di cui sopra? Tale determinante è strettamente maggiore di zero ($\mathrm{Det}A > 0$) perchè è definita positiva. Ovvero:

$$A^{-1} = \begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,n} \\ \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{n,1} & \alpha_{n,2} & \cdots & \alpha_{n,n} \end{bmatrix}.$$

$\alpha_{1,1} \neq 0$ e $\Delta_k \neq 0$ definito positivo per ogni $k = 1, \ldots, n$. Allora $\alpha_{1,1} > 0$ (come minore principale estratto da A) e

$$0 < \alpha_{1,1} = \frac{\det \begin{bmatrix} a_{2,2} & \cdots & a_{2,n} \\ \vdots & & \vdots \\ a_{n,2} & \cdots & a_{n,n} \end{bmatrix}}{\det A} \quad \Rightarrow \ \det A > 0.$$

Sfruttando queste condizioni e tenendo in considerazione l'osservazione 5.3 si può parlare di fattorizzazione diretta. Abbiamo già visto che esiste una fattorizzazione $A = LU$ con L matrice triangolare inferiore con elementi diagonali uguali a 1 ed U matrice triangolare superiore.

$$\det A = \prod_{i=1}^{n} u_{i,i}.$$

Come sono gli elementi $u_{i,i}$?

Intanto sappiamo che $u_{1,1}$ non può essere negativo:

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{bmatrix}, \quad \Delta_1 = a_{1,1}$$

Poichè A la possiamo vedere come matrice a blocchi, allora si ha:

$$A = \left[\begin{array}{c|c} \Delta_1 & A_{1,2} \\ \hline A_{2,1} & A_{2,2} \end{array}\right] = LU.$$

$$\left[\begin{array}{c|c} \Delta_1 & A_{1,2} \\ \hline A_{2,1} & A_{2,2} \end{array}\right] = \left[\begin{array}{c|c} 1 & L_{1,2} \\ \hline L_{2,1} & L_{2,2} \end{array}\right] \left[\begin{array}{c|c} u_{1,1} & U_{1,2} \\ \hline U_{2,1} & U_{2,2} \end{array}\right] \Rightarrow \Delta_1 = u_{1,1}$$

Si ha quindi che $\Delta_1 > 0$.

Si può dimostrare per induzione che $\Delta_k > 0$ per ogni $k = 1, \ldots, n$.

$$A = \left[\begin{array}{c|c} \Delta_2 & A_{1,2} \\ \hline A_{2,1} & A_{2,2} \end{array}\right] = \left[\begin{array}{cc|c} 1 & 0 & \\ * & 1 & L_{1,2} \\ \hline \multicolumn{2}{c|}{L_{2,1}} & L_{2,2} \end{array}\right] \left[\begin{array}{cc|c} u_{1,1} & * & \\ 0 & u_{2,2} & U_{1,2} \\ \hline \multicolumn{2}{c|}{U_{2,1}} & U_{2,2} \end{array}\right] \Rightarrow \det\Delta_2 = u_{1,1} \cdot u_{2,2}$$

Si ha quindi che $0 < \det\Delta_2 = u_{1,1} \cdot u_{2,2} \wedge u_{1,1} > 0 \Rightarrow u_{2,2} > 0$.

5.2 Fattorizzazione di Cholesky.

Sia $A \in \mathbb{R}^{n \times n}$, simmetrica ($A = A^t$) e definita positiva, sappiamo dal capitolo precedente che A è non singolare e che ogni suo minore è definito positivo.

Costruiamo ora una matrice diagonale D tale che:

$$D = \begin{bmatrix} \sqrt{u_{1,1}} & 0 & \cdots & 0 \\ 0 & \sqrt{u_{2,2}} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \sqrt{u_{n,n}} \end{bmatrix}.$$

$$A = LU = \left(\underbrace{LD}_{B} \underbrace{D^{-1}U}_{C} \right) = BC.$$

Essendo $A = A^t$ si ha che $(BC)^t = BC$, allora $BC = C^t B^t$.

$$B = LD = \begin{bmatrix} \sqrt{u_{1,1}} & 0 & \cdots & 0 \\ * & \sqrt{u_{2,2}} & & \vdots \\ \vdots & & \ddots & 0 \\ * & \cdots & * & \sqrt{u_{n,n}} \end{bmatrix}$$

Allora:

$$\det B = \prod_{k=1}^{n} \sqrt{u_{k,k}} \neq 0.$$

Ovvero B è invertibile, inoltre anche B^t è non singolare ed invertibile.

$$\begin{aligned} B^{-1}BC &= B^{-1}C^tB^t \\ C &= B^{-1}C^tB^t \\ C\left[B^t\right]^{-1} &= B^{-1}C^tB^t\left[B^t\right]^{-1}. \end{aligned}$$

$$\Rightarrow\ C\left[B^t\right]^{-1} = B^{-1}C^t.$$

Vediamo com'è fatta la matrice $B^{-1}C^t$:

$$B^{-1}C^t = \begin{bmatrix} \frac{1}{\sqrt{u_{1,1}}} & 0 & \cdots & 0 \\ * & \frac{1}{\sqrt{u_{2,2}}} & & \vdots \\ \vdots & & \ddots & 0 \\ * & \cdots & * & \frac{1}{\sqrt{u_{n,n}}} \end{bmatrix} \begin{bmatrix} \sqrt{u_{1,1}} & 0 & \cdots & 0 \\ *' & \sqrt{u_{2,2}} & & \vdots \\ \vdots & & \ddots & 0 \\ *' & \cdots & *' & \sqrt{u_{n,n}} \end{bmatrix}.$$

Il prodotto risulta sempre una matrice triangolare inferiore.

$$B^{-1}C^t = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ *'' & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ *'' & \cdots & *'' & 1 \end{bmatrix}.$$

Per determinare B^{-1} risolvo i sistemi $Bx = e_k$, per $k = 1, \ldots, n$.

$$C = D^{-1}U = \begin{bmatrix} \frac{1}{\sqrt{u_{1,1}}} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sqrt{u_{2,2}}} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \frac{1}{\sqrt{u_{n,n}}} \end{bmatrix} \begin{bmatrix} u_{1,1} & * & \cdots & * \\ 0 & u_{2,2} & & \vdots \\ \vdots & & \ddots & * \\ 0 & \cdots & 0 & u_{n,n} \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{u_{1,1}}{\sqrt{u_{1,1}}} & * & \cdots & * \\ 0 & \frac{u_{2,2}}{\sqrt{u_{2,2}}} & & \vdots \\ \vdots & & \ddots & * \\ 0 & \cdots & 0 & \frac{u_{n,n}}{\sqrt{u_{n,n}}} \end{bmatrix} = \begin{bmatrix} \sqrt{u_{1,1}} & * & \cdots & * \\ 0 & \sqrt{u_{2,2}} & & \vdots \\ \vdots & & \ddots & * \\ 0 & \cdots & 0 & \sqrt{u_{n,n}} \end{bmatrix}.$$

$$B^t = \begin{bmatrix} \sqrt{u_{1,1}} & * & \cdots & * \\ 0 & \sqrt{u_{2,2}} & & \vdots \\ \vdots & & \ddots & * \\ 0 & \cdots & 0 & \sqrt{u_{n,n}} \end{bmatrix} \Rightarrow \left[B^t\right]^{-1} = \begin{bmatrix} \frac{1}{\sqrt{u_{1,1}}} & * & \cdots & * \\ 0 & \frac{1}{\sqrt{u_{2,2}}} & & \vdots \\ \vdots & & \ddots & * \\ 0 & \cdots & 0 & \frac{1}{\sqrt{u_{n,n}}} \end{bmatrix}.$$

$$C\left[B^t\right]^{-1} = \begin{bmatrix} 1 & * & \cdots & * \\ 0 & 1 & & \vdots \\ \vdots & & \ddots & * \\ 0 & \cdots & 0 & 1 \end{bmatrix}.$$

$$C\left[B^t\right]^{-1} = \begin{bmatrix} 1 & * & \cdots & * \\ 0 & 1 & & \vdots \\ \vdots & & \ddots & * \\ 0 & \cdots & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ *'' & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ *'' & \cdots & *'' & 1 \end{bmatrix} = B^{-1}C^t.$$

Quindi si ha che se $C\left[B^t\right]^{-1} = I$ allora $C = B^t$ e da $B^{-1}C = I$ segue che $C^t = B$, allora:

$$A = BB^t.$$

Questa fattorizzazione si realizza tramite il prodotto di una matrice B con la sua trasposta. Se A è una matrice simmetrica e definita positiva, allora esiste una matrice B non singolare che fattorizza A.

5.2.1 Unicità della fattorizzazione.

Sotto quali condizioni si ha l'unicità della fattorizzazione di Cholesky?

Teorema 5.5. *Sia A una matrice quadrata di ordine n simmetrica e definita positiva, sia B la matrice tale che il prodotto con la sua trasposta fattorizzi A, ovvero $A = BB^t$, allora tale fattorizzazione è unica.*

Dimostrazione. Siano B_1 e B_2 tali che:

$$A = B_1B_1^t \quad \wedge \quad A = B_2B_2^t.$$

Ipotesi: $(B_1)_{i,i} > 0 \ \wedge \ (B_2)_{i,i} > 0, \quad i = 1, \ldots, n.$

$$A = B_1D_1^{-1}D_1B_1^t.$$

$$A = B_2 D_2^{-1} D_2 B_2^t.$$

Ricordiamo che D_1 e D_2 sono matrici diagonali aventi come elementi diagonali i medesimi delle matrici B_1 e B_2.

$$B_1 D_1^{-1} = \begin{bmatrix} b_{1,1}^{(1)} & 0 & \cdots & 0 \\ * & b_{2,2}^{(1)} & & \vdots \\ \vdots & & \ddots & 0 \\ * & \cdots & * & b_{n,n}^{(1)} \end{bmatrix} \begin{bmatrix} \frac{1}{b_{1,1}^{(1)}} & 0 & \cdots & 0 \\ 0 & \frac{1}{b_{2,2}^{(1)}} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \frac{1}{b_{n,n}^{(1)}} \end{bmatrix}.$$

$$B_1 D_1^{-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ * & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ * & \cdots & * & 1 \end{bmatrix} = L_1.$$

$$B_2 D_2^{-1} = \begin{bmatrix} b_{1,1}^{(2)} & 0 & \cdots & 0 \\ * & b_{2,2}^{(2)} & & \vdots \\ \vdots & & \ddots & 0 \\ * & \cdots & * & b_{n,n}^{(2)} \end{bmatrix} \begin{bmatrix} \frac{1}{b_{1,1}^{(2)}} & 0 & \cdots & 0 \\ 0 & \frac{1}{b_{2,2}^{(2)}} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \frac{1}{b_{n,n}^{(2)}} \end{bmatrix}.$$

$$B_2 D_2^{-1} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ * & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ * & \cdots & * & 1 \end{bmatrix} = L_2.$$

Allora si ha che $A = L_1 U_1$ e $A = L_2 U_2$, poiché tali fattorizzazioni sono uniche segue che:

$$L_1 = L_2 \;\Rightarrow\; B_1 D_1^{-1} = B_2 D_2^{-1},$$
$$U_1 = U_2 \;\Rightarrow\; B_1 D_1^t = B_2 D_2^t.$$

$$\text{Ovvero:} \quad \left(B_1 D_1^t\right)_{i,i} = \left(B_2 D_2^t\right)_{i,i}, \quad i = 1, \ldots, n.$$
$$\left[b_{i,i}^{(1)}\right]^2 = \left[b_{i,i}^{(2)}\right]^2, \quad i = 1, \ldots, n.$$

Si ha quindi, per ogni i che: $b_{i,i}^{(1)} = b_{i,i}^{(2)}$, quindi $D_1 = D_2$ per la genericità di i.

$$D_1 = D_2 \;\Rightarrow\; B_1 = B_2.$$

■

5.2.2 Costruzione della matrice B.

Vediamo come costruire la matrice B della fattorizzazione di Cholesky. Sappiamo che B è una matrice triangolare inferiore.

$$B = \begin{bmatrix} b_{1,1} & 0 & \cdots & 0 \\ b_{2,1} & b_{2,2} & & \vdots \\ \vdots & & \ddots & 0 \\ b_{n,1} & b_{n,2} & \cdots & b_{n,n} \end{bmatrix}, \quad B^t = \begin{bmatrix} b_{1,1} & b_{2,1} & \cdots & b_{n,1} \\ 0 & b_{2,2} & & b_{n,2} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & b_{n,n} \end{bmatrix}.$$

Poiché $A = BB^t$ si ha che:

$$a_{i,j} = \sum_{k=1}^{n} b_{i,k} b_{j,k}. \qquad i = 1, \ldots, n; \quad j = 1, \ldots, n. \tag{5.1}$$

Nota Bene. Dato che gli elementi $b_{p,q}$, con $1 \leq p \leq q \leq n$ sono nulli, non è necessario scorrere totalmente gli indici i e j dell'equazione 5.1, è sufficiente prendere k fino l'indice minore dei due, ovvero:

$$a_{i,j} = \sum_{k=1}^{\min(i,j)} b_{i,k} b_{j,k}. \qquad i = 1, \ldots, n; \quad j = 1, \ldots, n.$$

Inoltre per simmetria possiamo esprimere tale equazione come:

$$a_{i,j} = \sum_{k=1}^{i} b_{i,k} b_{j,k}. \quad i = 1, \ldots, n, \ i \leq j. \tag{5.2}$$

Costruiamo la matrice B.

$i = 1$.

- $j = 1$.
  $a_{1,1} = (b_{1,1})^2 \implies b_{1,1} = \pm\sqrt{a_{1,1}}$.
- $j = 2$.
  $a_{1,2} = b_{1,1} \cdot b_{2,1} \implies b_{2,1} = \frac{a_{1,2}}{b_{1,1}}$.

  $\vdots$
- $j = n$.
  $a_{1,n} = b_{1,1} \cdot b_{n,1} \implies b_{n,1} = \frac{a_{1,n}}{b_{1,1}}$.

i generico.

- $j = i$.
  $a_{i,i} = \sum_{k=1}^{i} b_{i,k} \cdot b_{i,k} = \sum_{k=1}^{i-1} (b_{i,k})^2 + (b_{i,i})^2$.

$$b_{i,i} = \left[a_{i,i} - \sum_{k=1}^{i-1} (b_{i,k})^2 \right]^{\frac{1}{2}}.$$

- $j = i + 1$.
  $a_{i,i+1} = \sum_{k=1}^{n} b_{i,k} \cdot b_{i+1,k} = \sum_{k=1}^{i-1} b_{i,k} \cdot b_{i+1,k} + b_{i,i} \cdot b_{i+1,i}$.

$$b_{i+1,i} = \frac{\left[a_{i,i+1} - \sum_{k=1}^{i-1} b_{i,k} \cdot b_{i+1,k} \right]}{b_{i,i}}.$$

$\vdots$

- $j = n$.
  $a_{i,n} = \sum_{k=1}^{n} b_{i,k} \cdot b_{n,k} = \sum_{k=1}^{i-1} b_{i,k} \cdot b_{n,k} + b_{i,i} \cdot b_{n,i}$.

$$b_{n,i} = \frac{\left[a_{i,n} - \sum_{k=1}^{i-1} b_{i,k} \cdot b_{n,k} \right]}{b_{i,i}}.$$

Osservazione 5.6. Come si può notare, nella *fattorizzazione di Cholesky* si costruisce una sola matrice (B), il costo è quindi dimezzato rispetto alla LU e risulta essere pari a $O(\frac{n}{6})$.

5.3 Fattorizzazione QR.

5.3.1 Matrici di Householder e proprietà.

Definizione 5.7. Sia $u \in \mathbb{R}^n$ generico, $u \neq 0$, $\beta = -\frac{2}{\|u\|_2^2} \in \mathbb{R}$. Possiamo definire una matrice elementare U tale che:

$$U = E(\beta, u, u) = I - \frac{2uu^t}{\|u\|_2^2},$$

si dice che U è una matrice di *Householder*, dato $x \in \mathbb{R}^n$, Ux una riflessione e

$$U = I - \frac{2uu^t}{\|u\|_2^2}$$

un *riflettore*.

Proposizione 5.8. *Sia U una matrice di Housholder, allora valgono le seguenti proprietà:*

1. $U = U^t$. *(simmetrica)*
2. $UU^t = U^tU = I$. *(ortogonale)*
3. $U^2 = I$. *(idempotente)*

Dimostrazione.

1. $U \stackrel{?}{=} U^t$.
$$U^t = \left(I - \frac{2uu^t}{\|u\|_2^2}\right)^t = I - \frac{2uu^t}{\|u\|_2^2} = U.$$

2. $UU^t \stackrel{?}{=} U^tU = I$.
Poiché $U = U^t$ si ha che:
$$\left(I - \frac{2uu^t}{\|u\|_2^2}\right) \cdot \left(I - \frac{2uu^t}{\|u\|_2^2}\right) = I - \cancel{\frac{2uu^t}{\|u\|_2^2}} - \cancel{\frac{2uu^t}{\|u\|_2^2}} + \cancel{\frac{4u(u^tu)u^t}{\|u\|_2^2\|u\|_2^2}} = I.$$

3. $U^2 \stackrel{?}{=} I$.
Immediato da 1 e 2.

■

Osservazione 5.9. Sia $u \in \mathbb{R}^n$, $u \neq 0$, posto $\|u\|_2^2 = 1$ si ha che $U = I - 2uu^t$. Ovvero, passando al generico problema, posto $x = v + w$ e $v = \alpha u$:

$$\begin{aligned} Ux &= (I - 2uu^t)\,x \\ &= x - 2uu^tx = x - 2uu^t(v + w) \\ &= x - 2uu^tv - 2u \underbrace{u^tw}_{=0\ (u\perp w)} \\ &= x - 2uu^t\alpha u \\ &= x - 2\alpha u \underbrace{u^tu}_{=1\ (\|u\|_2^2=1)} \\ &= x - 2\alpha u = v + w - 2v \\ &= -v + w. \end{aligned}$$

Il problema ora è che da un singolo sistema lineare siamo passati a due.

Si dice che Ux ha fatto una *riflessione*. La norma euclidea di x è quindi uguale alla norma euclidea di Ux o, detto in altre parole, una matrice di Householder *non cambia* la norma (euclidea) del vettore a cui è applicata.

$$\|x\|_2 = \|Ux\|_2.$$

La figura 5.1 mostra vettorialmente l'applicazione di una matrice di Householder ad un generico vettore $x = v + w$ nella forma di cui sopra.

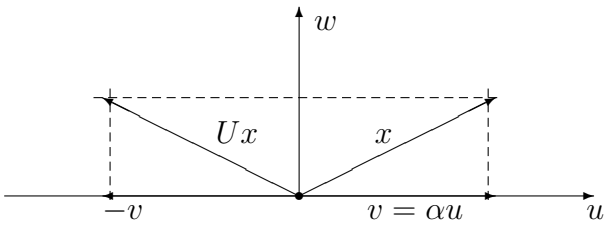


Figura 5.1: L'applicazione grafica di una matrice di Householder

5.3.2 Costruzione di Q.

Usiamo nuovamente le matrici elementari.

$$U = E(\beta, u, u) = I - \frac{2uu^t}{\|u\|_2^2},$$

Sia $x \in \mathbb{R}^n,\ x \neq 0,\ s_1 = \pm\|x\|_2$.

$$u = x + s_1 \cdot e_1, \qquad u = \begin{bmatrix} x_1 + s_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

$$\begin{aligned} Ux &= \left(I - \tfrac{2uu^t}{\|u\|_2^2}\right) x \\ &= x - \tfrac{2uu^t}{\|u\|_2^2} x \\ &= x - u\tfrac{2u^t x}{\|u\|_2^2} \\ &= x - u. \end{aligned}$$

L'ultima uguaglianza deriva dai seguenti calcoli:

$$\begin{aligned} 2u^t x &= 2[(x_1 + s_1)x_1 + x_2^2 + \cdots + x_n^2] \\ &= 2(x_1^2 + \cdots + x_n^2 + s_1 x_1) \\ &= 2(s_1^2 + s_1 x_1) \\ &= 2s_1(s_1 + x_1), \end{aligned}$$

$$\begin{aligned} \|u\|_2^2 &= (x_1 + s_1)^2 + x_2^2 + \cdots + x_n^2 \\ &= s_1^2 + s_1^2 + 2s_1 x_1 \\ &= 2s_1(s_1 + x_1). \end{aligned}$$

Possiamo quindi esprimere Ux come $x - u$, ovvero:

$$Ux = x - u = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} - \begin{bmatrix} x_1 + s_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} -s_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Osservazione 5.10. U non è univocamente determinata, si può notare che è possibile scegliere come s_1 sia $+\|x\|_2$ che $-\|x\|_2$. La scelta più opportuna per determinare s_1 è tale che $x_1 + s_1$ sia di segno concorde in modo che la somma non venga 0 (cancellazione numerica). Quindi

$$x_1 > 0 \longrightarrow s_1 = +\|x\|_2, \quad x_1 < 0 \longrightarrow s_1 = -\|x\|_2.$$

Sia $x \in \mathbb{R}^n,\ x \neq 0$, estraiamo $x' \in \mathbb{R}^{n-1}$, ovvero:

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \qquad x' = \begin{bmatrix} x_2 \\ \vdots \\ x_n \end{bmatrix}.$$

In pratica abbiamo eliminato x_1. Si ha dunque $s_2 = \pm\|x'\|_2$.

$$u_2' = x' + s_2 e_1 \in \mathbb{R}^{n-1}, \quad u_2 = \begin{bmatrix} 0 \\ u_2' \end{bmatrix}.$$

Costruiamo il riflettore elementare U_2:

$$U_2 = I - \frac{2u_2u_2^t}{\|u_2\|_2^2}.$$

$$\begin{aligned} U_2x &= \left(I - \tfrac{2u_2u_2^t}{\|u_2\|_2^2}\right)x \\ &= x - \tfrac{2u_2u_2^t}{\|u_2\|_2^2}x \\ &= x - u_2\tfrac{2u_2^tx}{\|u_2\|_2^2} \\ &= x - u_2. \end{aligned}$$

L'ultima uguaglianza deriva dai seguenti calcoli:

$$\begin{aligned} 2u_2^tx &= 2[0\ u_2']x \\ &= 2(x_2^2 + \cdots + x_n^2 + s_2x_2) \\ &= 2(s_2^2 + s_2x_2) \\ &= 2s_2(s_2 + x_2), \end{aligned}$$

$$\begin{aligned} \|u_2\|_2^2 &= 0 + \underbrace{x_2^2 + 2x_2s_2 + s_2^2}_{(x_2+s)^2} + x_3^2 + \cdots + x_n^2 \\ &= s_2^2 + s_2^2 + 2s_2x_2 \\ &= 2s_2(s_2 + x_2). \end{aligned}$$

Possiamo quindi esprimere $U_2 x$ come $x - u_2$, ovvero:

$$U_2 x = x - u_2 = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} - \begin{bmatrix} 0 \\ x_2 + s_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 \\ -s_2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Analizziamo U_2 in termini matriciali.

$$\begin{aligned} U_2 &= \left[\begin{array}{c|c} I_1 & 0 \\ \hline 0 & I_{n-1} \end{array}\right] - \tfrac{2}{\|u_2\|_2^2} \begin{bmatrix} 0 \\ u_2' \end{bmatrix} [0 \; u_2'^t] \\ &= \left[\begin{array}{c|c} I_1 & 0 \\ \hline 0 & I_{n-1} \end{array}\right] - \tfrac{2}{\|u_2\|_2^2} \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & u_2' u_2'^t \end{array}\right] \\ &= \left[\begin{array}{c|c} I_1 & 0 \\ \hline 0 & I_{n-1} \end{array}\right] - \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & \frac{2}{\|u_2\|_2^2} u_2' u_2'^t \end{array}\right] \\ &= \left[\begin{array}{c|c} I_1 & 0 \\ \hline 0 & I_{n-1} - \frac{2}{\|u_2\|_2^2} u_2' u_2'^t \end{array}\right] \\ &= \left[\begin{array}{c|c} I_1 & 0 \\ \hline 0 & U_2^{n-1} \end{array}\right]. \end{aligned}$$

Sia $x \in \mathbb{R}^n,\ x \neq 0$, estraiamo $x' \in \mathbb{R}^{n-k+1}$, ovvero:

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \qquad x' = \begin{bmatrix} x_k \\ x_{k+1} \\ \vdots \\ x_n \end{bmatrix}.$$

In pratica abbiamo eliminato $x_1, x_2, \ldots x_{k-1}$. Si ha dunque $s_k = \pm\|x'\|_2$.

$$u_k' = x' + s_k e_1 \in \mathbb{R}^{n-k+1}, \quad u_k = \begin{bmatrix} 0 \\ u_k' \end{bmatrix}.$$

Costruiamo il riflettore elementare U_k:

$$U_k = I - \frac{2 u_k u_k^t}{\|u_k\|_2^2}.$$

$$\begin{aligned} U_k x &= \left(I - \tfrac{2u_k u_k^t}{\|u_k\|_2^2}\right) x \\ &= x - \tfrac{2u_k u_k^t}{\|u_k\|_2^2} x \\ &= x - u_k \tfrac{2u_k^t x}{\|u_k\|_2^2} \\ &= x - u_k. \end{aligned}$$

Possiamo quindi esprimere $U_k x$ come $x - u_k$, ovvero:

$$U_k x = x - u_k = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} - \begin{bmatrix} 0_1 \\ 0_2 \\ \vdots \\ 0_{k-1} \\ x_k + s_k \\ x_{k+1} \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_{k-1} \\ -s_k \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Analizziamo U_k in termini matriciali.

$$\begin{aligned} U_k &= \left[\begin{array}{c|c} I_{k-1} & 0 \\ \hline 0 & I_{n-k+1} \end{array}\right] - \tfrac{2}{\|u_k\|_2^2} \begin{bmatrix} 0 \\ u'_k \end{bmatrix} [0 \; u_k'^t] \\ &= \left[\begin{array}{c|c} I_{k-1} & 0 \\ \hline 0 & I_{n-k+1} \end{array}\right] - \tfrac{2}{\|u_k\|_2^2} \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & u'_k u_k'^t \end{array}\right] \\ &= \left[\begin{array}{c|c} I_{k-1} & 0 \\ \hline 0 & I_{n-k+1} \end{array}\right] - \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & \frac{2}{\|u_k\|_k^2} u'_k u_k'^t \end{array}\right] \\ &= \left[\begin{array}{c|c} I_{k-1} & 0 \\ \hline 0 & I_{n-k+1} - \frac{2}{\|u_k\|_2^2} u'_k u_k'^t \end{array}\right] \\ &= \left[\begin{array}{c|c} I_{k-1} & 0 \\ \hline 0 & U_k^{n-(k-1)} \end{array}\right]. \end{aligned}$$

Dalle matrici U_i, con $i = 1, \ldots, n$, vedremo che si potrà costruire la matrice Q, partendo dal generico problema $Ax = b$.

5.3.3 Unicità della fattorizzazione QR

Dall'osservazione 5.10, si deduce che la fattorizzazione QR mediante riflettori elementari di Householder *non è* quindi unica! La scelta in segno della norma del sottovettore estratto è arbitraria.

5.3.4 Algoritmo di fattorizzazione QR.

Applichiamo i risultati di cui sopra alla risoluzione dei sistemi lineari.

$$Ax = b, \qquad A \in \mathbb{R}^{n\times n}, \qquad x, b \in \mathbb{R}^n, \quad x = ?$$

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{bmatrix}.$$

Passo 1: poniamo x equivalente alla prima colonna di A e costruiamo U_1 come visto nella sezione precedente.

$$x \equiv \overline{a}_1, \quad U_1 A, \quad U_1 b.$$

Al termine noto b penseremo dopo.

$$U_1 A = U_1 [\overline{a}_1\ \overline{a}_2\ \cdots\ \overline{a}_n] = [U_1\overline{a}_1\ U_1\overline{a}_2\ \cdots\ U_1\overline{a}_n]$$

$$= \left[\begin{array}{c|c|c|c|c} -s_1 & & & & \\ 0 & & & & \\ \vdots & U_1\overline{a}_2 & U_1\overline{a}_3 & \cdots & U_1\overline{a}_n \\ 0 & & & & \end{array}\right].$$

Come si nota occorre calcolare $n-1$ prodotti matrice vettore, con n righe per n prodotti, il costo di un'operazione simile è un $\mathrm{O}(n^2)$. Consideriamo la forma algebrica piuttosto che quella matriciale.

$$U_1\overline{a}_k = \overline{a}_k - \underbrace{\frac{2}{\|u_1\|_2^2}}_{n \text{ prod.}} u_1 \underbrace{u_1^t \overline{a}_k}_{n \text{ prod.}}.$$

Il costo computazionale di questa operazione è circa $2n$, ovvero un costo lineare nel numero delle colonne, molto minore di n^2.

Osservazione 5.11. E' sufficiente calcolare $\|u_1\|_2^2$ una sola volta si avrà quindi che per calcolare $U_1\overline{a}_2$ il costo è $2n$, mentre per tutte le altre $n-2$ colonne questo si riduce a $\mathrm{O}(n)$.

A questo punto dell'algoritmo abbiamo, data $A_1 \equiv A$:

$$U_1 A_1 = A_2 = \begin{bmatrix} -s_1 & a_{1,2}^{(2)} & \cdots & a_{1,n}^{(2)} \\ 0 & a_{2,2}^{(2)} & \cdots & a_{2,n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n,2}^{(2)} & \cdots & a_{n,n}^{(2)} \end{bmatrix}.$$

Passo 2: poniamo x' equivalente alla seconda colonna di A ristretto alla seconda, terza, ..., ultima componente e costruiamo U_2 come visto nella sezione precedente.

$$x' \equiv \begin{bmatrix} a_{2,2}^{(2)} \\ \vdots \\ a_{n,2}^{(2)} \end{bmatrix}.$$

Si ha dunque $s_2 = \pm\|x'\|_2$.

$$u_2' = x' + s_2 e_1 \in \mathbb{R}^{n-1}, \quad u_2 = \begin{bmatrix} 0 \\ u_2' \end{bmatrix}.$$

Ottenuto U_2^{n-1} costruiamo U_2'.

$$U_2 = \left[\begin{array}{c|c} I_1 & 0 \\ \hline 0 & U_2^{n-1} \end{array}\right].$$

$$U_2 A_2 = U_2 \left[U_2 \overline{a}_1^{(2)} \; U_2 \overline{a}_2^{(2)} \; \cdots \; U_2 \overline{a}_n^{(2)} \right].$$

$$U_2 \overline{a}_1^{(2)} = \left[\begin{array}{c|c} I_1 & 0 \\ \hline 0 & U_2^{n-1} \end{array}\right] \begin{bmatrix} -s_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} -s_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Quindi non altera la prima colonna della matrice.

$$U_2 A_2 = A_3 = \begin{bmatrix} -s_1 & a_{1,2}^{(2)} & a_{1,3}^{(2)} & \cdots & a_{1,n}^{(2)} \\ 0 & -s_2 & a_{2,3}^{(3)} & \cdots & a_{2,n}^{(3)} \\ 0 & 0 & a_{3,3}^{(3)} & \cdots & a_{3,n}^{(3)} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & a_{n,3}^{(3)} & \cdots & a_{n,n}^{(3)} \end{bmatrix}.$$

Passo k: poniamo x' equivalente alla k-esima colonna di A ristretto alla $k+1$-esima, $k+2$-esima, ..., ultima componente e costruiamo U_k come visto nella sezione precedente.

$$x' \equiv \begin{bmatrix} a_{k,k}^{(k)} \\ a_{k+1,k}^{(k)} \\ \vdots \\ a_{n,k}^{(k)} \end{bmatrix}.$$

Si ha dunque $s_k = \pm\|x'\|_2$, $x' \in \mathbb{R}^{n-k+1}$.

$$u'_k = x' + s_k e_1 \in \mathbb{R}^{n-k+1}, \quad u_k = \begin{bmatrix} 0 \\ u'_k \end{bmatrix}.$$

Ottenuto U_k^{n-k+1} costruiamo U'_k.

$$U_k = \left[\begin{array}{c|c} I_k & 0 \\ \hline 0 & U_k^{n-k+1} \end{array}\right].$$

$$U_k A_k = U_k \left[U_k \overline{a}_1^{(k)} \; U_2 \overline{a}_2^{(k)} \; \cdots \; U_k \overline{a}_n^{(k)} \right].$$

$$U_k \overline{a}_1^{(k)} = \left[\begin{array}{c|c} I_k & 0 \\ \hline 0 & U_k^{n-k+1} \end{array}\right] \begin{bmatrix} -s_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} -s_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Quindi non altera la prima colonna della matrice. Allo stesso modo non altera le restanti $k-1$ colonne.

$$U_k A_k = A_{k+1} = \begin{bmatrix} -s_1 & a_{1,2}^{(2)} & a_{1,3}^{(3)} & \cdots & a_{1,k}^{(k)} & a_{1,k+1}^{(k)} & \cdots & a_{1,n}^{(k)} \\ 0 & -s_2 & a_{2,3}^{(3)} & \cdots & a_{2,k}^{(k)} & a_{2,k+1}^{(k)} & \cdots & a_{2,n}^{(k)} \\ 0 & 0 & -s_3 & \cdots & a_{3,k}^{(k)} & a_{3,k+1}^{(k)} & \cdots & a_{3,n}^{(k)} \\ 0 & 0 & 0 & \ddots & & & & \vdots \\ & & & & -s_k & a_{k,k+1}^{(k)} & \cdots & a_{k,n}^{(k)} \\ \vdots & \vdots & \vdots & & 0 & a_{k+1,k+1}^{(k)} & \cdots & a_{k+1,n}^{(k)} \\ & & & & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & a_{n,k+1}^{(k)} & \cdots & a_{n,n}^{(k)} \end{bmatrix}.$$

Riassumendo, alla fine dell'algoritmo, si ottiene una matrice triangolare superiore:

$$U_{n-1}U_{n-2}\cdots U_2U_1A_1 = R.$$

Poiché per ogni k compreso tra 1 ed n si ha U_K ortogonale (ovvero invertibile), possiamo ottenere la seguente:

$$\begin{aligned} A \equiv A_1 &= U_1^{-1}U_2^{-1}\cdots U_{n-1}^{-1}R \\ &= U_1^tU_2^t\cdots U_{n-1}^tR \\ &= U_1U_2\cdots U_{n-1}R \\ &= QR. \end{aligned}$$

Il problema originale $Ax = b$ si trasforma dunque in $QRx = b$, occorre ora risolvere il sistema:

$$\begin{cases} Rx = y \\ Qy = b. \end{cases}$$

Molto semplice, poiché si riduce con un passo ad un unico problema:

$$Rx = Q^{-1}b \implies Rx = Q^tb.$$

Analizziamo il costo dell'algoritmo.

Per calcolare $t = Qb$ si hanno n^2 operazioni di prodotto, per calcolare $Rx = t$ ancora n^2 prodotti, ovvero un costo totale di $2n^2$.

Tale costo è equivalente a quello della fattorizzazione LU, nonostante si risolva un singolo sistema lineare. Allora perchè utilizzare la fattorizzazione QR?

Osservazione 5.12. Sia $Ax = b$ un generico problema lineare con $A \in \mathbb{R}^{n\times n}$ non singolare, $x, b \in \mathbb{R}^n$, si ha:

$$Ax - b = 0.$$

Il problema, visto in questi termini, è quello di trovare un vettore $x \in \mathbb{R}^n$ tale che $\min_{x\in\mathbb{R}^n} \|Ax - b\|_2 = 0$. Ovvero cercare il vettore x che minimizza la norma euclidea. Il problema originario e quest'ultimo trovato sono dunque equivalenti!

Non solo, ma considerando una fattorizzazione QR della matrice A, è possibile applicare Q internamente, poichè esso non cambia la norma del vettore b. *

*Si veda l'osservazione 5.9

$$\begin{aligned}\min_{x\in\mathbb{R}^n}\|Ax-b\|_2 &= \min_{x\in\mathbb{R}^n}\|Q(Ax-b)\|_2\\ &= \min_{x\in\mathbb{R}^n}\|QAx-Qb\|_2\\ &= \min_{x\in\mathbb{R}^n}\|Rx-Qb\|_2.\end{aligned}$$

Parliamo ora del caso in cui la matrice A non sia necessariamente quadrata.

Sia $A\in\mathbb{R}^{n\times m}$, con $n\geq m,\ \text{rang}(A)=m,\ \rho(A)=m,\ x\in\mathbb{R}^m,\ b\in\mathbb{R}^n$, si ha che il generico sistema lineare $Ax=b$ per l'osservazione 5.12 si può scrivere come:

$$\min_{x\in\mathbb{R}^m}\|Ax-b\|_2.$$

Costruiamo ora una fattorizzazione QR passo per passo della matrice A.

$$A=\begin{bmatrix} a_{1,1} & \cdots & a_{1,m}\\ \vdots & & \vdots\\ a_{m,1} & \cdots & a_{m,m}\\ \vdots & & \vdots\\ a_{n,1} & \cdots & a_{n,m}\end{bmatrix}.$$

Primo passo:

$$A_2=\begin{bmatrix} -s_1 & a_{1,2}^{(2)} & \cdots & a_{1,m}^{(2)}\\ 0 & a_{2,2}^{(2)} & \cdots & a_{2,m}^{(2)}\\ \vdots & \vdots & & \vdots\\ 0 & a_{n,2}^{(2)} & \cdots & a_{n,m}^{(2)}\end{bmatrix}.$$

Secondo passo:

$$A_3=\begin{bmatrix} -s_1 & a_{1,2}^{(2)} & a_{1,3}^{(3)} & \cdots\\ 0 & -s_2 & a_{2,3}^{(3)} & \cdots\\ 0 & 0 & a_{3,3}^{(3)} & \\ \vdots & \vdots & \vdots & \\ 0 & 0 & a_{n,3}^{(3)} & \cdots\end{bmatrix}.$$

Dopo m passi (ovvero per tutte le m colonne) si arriva ad una situazione del tipo:

$$A_{m+1} = \underbrace{U_m ... U_3 U_2 U_1}_{Q} A = \begin{bmatrix} -s_1 & a_{1,2}^{(2)} & a_{1,3}^{(3)} & \cdots & a_{1,m}^{(m)} \\ 0 & -s_2 & a_{2,3}^{(3)} & \cdots & a_{2,m}^{(m)} \\ 0 & 0 & -s_3 & & a_{3,m}^{(m)} \\ \vdots & \vdots & & \ddots & \\ 0 & 0 & 0 & \cdots & -s_m \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Otteniamo così una matrice QA (con Q ortogonale) che può essere espressa come:

$$QA = \begin{bmatrix} R \\ 0 \end{bmatrix}.$$

Avente R matrice quadrata triangolare superiore di ordine m.

Tornando al problema originale si ha quindi:

$$\begin{aligned} Ax = b \Longleftrightarrow \min_{x \in \mathbb{R}^m} \|Ax - b\|_2 &= \min_{x \in \mathbb{R}^m} \|Q(Ax - b)\|_2 \\ &= \min_{x \in \mathbb{R}^m} \|QAx - Qb\|_2 \\ &= \min_{x \in \mathbb{R}^m} \left\| \begin{bmatrix} R \\ 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} - \begin{bmatrix} b_1 \\ \hline b_2 \end{bmatrix} \right\|_2 \\ &= \min_{x \in \mathbb{R}^m} \left\| \begin{array}{c} Rx - b_1 \\ \hline -b_2 \end{array} \right\|_2, \end{aligned}$$

avente $b_1 \in \mathbb{R}^m$ e $b_2 \in \mathbb{R}^{n-m}$. Risolvendo il problema $Rx = b_1$, si trova il vettore del minimo. In $\|b_2\|_2$ è presente il resto (o residuo).

Osservazione 5.13. Se si applica questo procedimento ad una matrice A quadrata, risolvere $Rx = b_1$ significa fondamentalmente trovare il vettore x tale che la norma da minimizzare sia zero (avente quindi vettore b_2 nullo).

Ma perchè utilizzare matrici rettangolari aventi più righe che colonne? Che idea di fondo modellano? Vedremo nella sezione 9.4 la formalizzazione precisa del problema ai minimi quadrati, però idealmente si pensi ad un insieme di punti nel piano. Ognuno di questi punti è univocamente determinato dalle coppie (x_i, y_i). Ipotizziamo di voler andare a creare un polinomio che

non passi necessariamente per tutti i punti ma che, essendo di grado basso, cerchi di avvicinarsi il più possibile ad essi.[†]. Tale tecnica viene in gergo chiamata *curve fitting* e la sua formalizzazione passa tramite la definizione di matrici rettangolari. In questo senso quindi, la fattorizzazione QR è uno strumento utile alla risoluzione di questo tipo di problemi.

[†]Vedremo nel capitolo 7 il concetto di interpolazione puntuale e della relazione tra numero di punti e grado del polinomio interpolante

Capitolo 6

Analisi di stabilità.

Definizione 6.1. Sia A una matrice, $\lambda_1, \ldots, \lambda_s$ gli autovalori di A, si dice *raggio spettrale* di A:

$$\rho(A) := \max_{1 \leq i \leq s} (|\lambda_i|).$$

Consideriamo $\|\cdot\|$ l'operatore di norma matriciale naturale (o indotta) e sia λ un generico autovalore di A, sia x un autovettore corrispondente normalizzato ($\|x\| = 1$), quindi

$$Ax = \lambda x.$$

$$\|Ax\| = \|\lambda x\| = |\lambda| \cdot \|x\| = |\lambda|.$$

$$\max \|Aw\| \geq \|w\| = 1,$$

$$\Rightarrow \|A\| \geq |\lambda| \quad \forall \lambda \text{ autovalore di } A.$$

$$\Rightarrow \|A\| \geq \rho(A).$$

Osservazione 6.2. I problemi visti fin'ora sono problemi teorici, quando ci si mette nei termini di modellare il problema al fine di utilizzare uno strumento automatico per il calcolo della soluzione, causa approssimazione di macchina si risolve un sistema differente, in cui abbiamo:

$$(A + \delta A)(x + \delta x) = b + \delta b,$$

con δA e δb matrice e vettore delle perturbazioni del sistema lineare originario.

Vediamo come un problema di questo tipo possa comunque essere manipolato per i nostri fini, nonostante l'approssimazione numerica.

6.1 Condizionamento del problema.

Teorema 6.3. *Siano $\delta A \in \mathbb{R}^{n\times n}$ e $\delta b \in \mathbb{R}^n$ rispettivamente la matrice ed il vettore delle perturbazioni del sistema lineare $Ax = b$ con $b \neq 0$.*

Sia $\|\cdot\|$ la norma matriciale indotta e $\|\delta A\| \cdot \|A^{-1}\| < 1$. Allora la matrice $A + \delta A$ è non singolare.

Indicata con $x+\delta x$ la soluzione del sistema perturbato $(A+\delta A)(x+\delta x) = b + \delta b$, si ha che:

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{cond(A)}{1 - cond(A) \cdot \frac{\|\delta A\|}{\|A\|}} \left[\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|}\right].$$

Dove $cond(A) := \|A\| \cdot \|A^{-1}\|$ è l'indice di condizionamento.

Osservazione 6.4.

$$\frac{\|\delta b\|}{\|b\|} \quad \text{e} \quad \frac{\|\delta A\|}{\|A\|},$$

sono gli errori relativi sui dati, mentre

$$\frac{\|\delta x\|}{\|x\|}$$

è l'errore relativo sul risultato.

Osservazione 6.5. Se $\|A\| \cdot \|A^{-1}\| < \frac{1}{2}$ allora:

$$\frac{\|\delta x\|}{\|x\|} \leq 2\text{cond}(A) \left[\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|}\right].$$

Osservazione 6.6. Un numero di condizionamento approssimativamente vicino a 1, è *indice* del buon condizionamento del problema. Sicuramente un numero di condizionamento alto *garantisce* il mal condizionamento del problema.

Osservazione 6.7. Come si vede dalla definizione, l'errore relativo dipende dal numero di condizionamento, quanto dal grando di perturbazione che hanno la matrice e il vettore dei termini noti. Quindi anche avendo un numero di condizionamento considerato buono, non possiamo essere sicuri che l'errore relativo sul risultato sia accettabile! Il numero di condizionamento è legato alla matrice A, il vettore b potrebbe essere assai perturbato.

Osservazione 6.8. Il numero di condizionamento inficia fondamentalmente due cose:

- il residuo $|Ax - b|$ con x soluzione approssimata;
- la precisione numerica $\|x - y\|_\infty$ con x soluzione approssimata ed y soluzione esatta del problema.

Vediamo ora un primo risultato per dimostrare il teorema 6.3.

Teorema 6.9. *Sia $A \in \mathbb{R}^{n\times n}$, $\|\cdot\|$ la norma matriciale indotta, se $\|A\| < 1$ allora $I + A$ è non singolare e risulta:*

$$\|(I + A)^{-1}\| \leq \frac{1}{1 - \|A\|}.$$

Dimostrazione. $\|A\| < 1 \Rightarrow \rho(A) \leq \|A\| < 1$. Allora tutti gli autovalori sono contenuti nel cerchio di raggio minore di 1, in particolare $\lambda \neq \pm 1$.

$I + A$ ha autovalori dati da $1 + \lambda \neq 0$, ovvero non ha autovalore nullo quindi è non singolare.

$$\begin{aligned}(I + A)(I + A)^{-1} &= I,\\ (I + A)^{-1} + A(I + A)^{-1} &= I,\\ (I + A)^{-1} &= I - A(I + A)^{-1}.\end{aligned}$$

$$\begin{aligned}\|(I + A)^{-1}\| &= \|I - A(I + A)^{-1}\|\\ &\leq \|I\| + \|A(I + A)^{-1}\|\\ &\leq 1 + \|A\| \cdot \|(I + A)^{-1}\|.\end{aligned}$$

$\Rightarrow \|(I + A)^{-1}\|(1 - \|A\|) \leq 1.$

$\Rightarrow \|(I + A)^{-1}\| \leq \frac{1}{1-\|A\|}.$

■

Dimostrazione. Teorema 6.3. $A + \delta A$ è non singolare.

$$A + \delta A = A(I + A^{-1}\delta A).$$

$I + A^{-1}\delta A$ è non singolare poiché $\|A^{-1}\delta A\| \leq \|\delta A\| \cdot \|A^{-1} < 1\|$ per il teorema precedente. Allora $A + \delta A$ è non singolare essendo prodotto di matrici non singolari. Inoltre:

$$\|(I + A^{-1}\delta A)^{-1}\| \leq \frac{1}{1 - \|A^{-1}\delta A\|} \leq \frac{1}{1 - \|A^{-1}\| \cdot \|\delta A\|}.$$

$$(A + \delta A)(A + \delta A)^{-1} = I.$$

$$(A+\delta A)(x+\delta x) = (b+\delta b); \qquad Ax = b.$$

$$(A+\delta A)(x+\delta x) - Ax = (b+\delta b) - b.$$

$$\cancel{Ax} + A\delta x + \delta A x + \delta A \delta x \cancel{- Ax} = \cancel{b} + \delta b \cancel{- b}.$$

$$(A+\delta A)\delta x = \delta b - \delta A x.$$

$$\begin{aligned} \delta x &= (A+\delta A)^{-1}\delta b - (A+\delta A)^{-1}\delta A x \\ &= [A(I+A^{-1}\delta A)]^{-1}\delta b - [A(I+A^{-1}\delta A)]^{-1}\delta A x \\ &= (I+A^{-1}\delta A)^{-1}A^{-1}\delta b - (I+A^{-1}\delta A)^{-1}A^{-1}\delta A x. \end{aligned}$$

$$\begin{aligned} \|\delta x\| &\leq \|(I+A^{-1}\delta A)^{-1}\| \cdot \|A^{-1}\| \cdot \|\delta b\| + \|(I+A^{-1}\delta A)^{-1}\| \cdot \|A^{-1}\| \cdot \|\delta A\| \cdot \|x\| \\ &\leq \tfrac{1}{1-\|A^{-1}\|\cdot\|\delta A\|}\|A^{-1}\|(\|\delta b\| + \|\delta A\| \cdot \|x\|). \end{aligned}$$

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\| \cdot \|A\| \cdot \frac{\|\delta A\|}{\|A\|}} \cdot \left[\frac{\|\delta b\|}{\|x\|} + \frac{\|\delta A\|}{\|A\|}\|A\|\right].$$

Da $Ax = b$ segue che:

$$\|A\| \cdot \|x\| \geq \|Ax\| = \|b\| \;\Rightarrow\; \|x\| \geq \frac{\|b\|}{\|A\|}.$$

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{\mathrm{cond}(A)}{1 - \mathrm{cond}(A) \cdot \frac{\|\delta A\|}{\|A\|}} \left[\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|}\right].$$

■

Capitolo 7

Interpolazione.

Dati alcuni valori numerici consideriamo i seguenti problemi:

- vogliamo ricostruire una traiettoria (grafico), ad esempio lo spostamento di un braccio meccanico;
- vogliamo ricostruire una funzione complessa mediante approssimazione numerica;
- vogliamo calcolare l'integrale di una funzione la cui primitiva è ignota (ad esempio approssimandola con un polinomio).

Analogamente si possono estendere i citati problemi alla terza dimensione, ovvero ricostruire superfici, ad esempio per creare nuovi modelli di macchine.

Teorema 7.1. *Dati $n+1$ valori $x_0, x_1, \ldots, x_n$ distinti ($x_i \neq x_j, i \neq j$) e $n+1$ numeri naturali $\alpha_0, \alpha_1, \ldots, \alpha_n$, sia $m = n + \sum_{i=0}^{n} \alpha_i$. Presi comunque i numeri $y_i^{(l)}$: $i = 0, 1, \ldots, n$; $l = 0, \ldots, \alpha_i$, esiste al più un polinomio $p(x) \in \mathcal{P}_m$ tale che:*

$$p^{(l)}(x_i) = y_i^{(l)}, \quad i = 0, \ldots, n; \ l = 0, \ldots, \alpha_i;$$

$$p^{(l)}(x_i) := \left. \frac{d^{(l)} p(x)}{dx^l} \right|_{x=x_i}.$$

Esempio. Primo caso:

$$\begin{array}{cccc} x_0 & x_1 & \cdots & x_n \\ y_0 & y_1 & \cdots & y_n \end{array}$$

Qui, molto semplicemente, ad ogni punto del dominio abbiamo il punto dell'immagine, $p(x) \in \mathcal{P}_n$ per cui $p(x_i) = y_i$. Per ogni i si ha quindi $\alpha_i = 0$. Si traccia quindi la funzione per quei punti.

Esempio. Secondo caso:

$$\begin{array}{cccc} x_0 & x_1 & \cdots & x_n \\ y_0 & y_1 & \cdots & y_n \\ y_0' & y_1' & & \\ & y_1'' & & \end{array}$$

Ora sono aumentate le informazioni sui punti, ovvero su y_0 e y_1 conosco la velocità e su y_1 anche la concavità.

I numeri naturali α_i indicano il numero di vincoli sul punto x_i, ovvero:

$$\begin{array}{rcll} \alpha_0 & = & 0 & \\ \alpha_1 & = & 0 & \\ & \vdots & & \text{un solo passaggio!} \\ \alpha_n & = & 0 & \end{array}$$

Lo schema di cui sopra modella la situazione del primo caso.

$$\begin{array}{rcl} \alpha_0 & = & 1 \\ \alpha_1 & = & 2 \\ \alpha_2 & = & 0 \\ & \vdots & \\ \alpha_n & = & 0 \end{array}$$

Questo è il secondo caso, nel primo punto abbiamo passaggio e derivata prima, nel secondo passaggio, derivata prima e derivata seconda, in tutti gli altri punti solo il passaggio.

Osservazione 7.2. Il teorema non dice nulla su una situazione del tipo:

$$\begin{array}{cccc} x_0 & x_1 & x_2 & \cdots \\ y_0 & y_1 & y_2 & \cdots \\ y_0' & & y_2' & \cdots \\ & y_1'' & y_2'' & \cdots \end{array}$$

C'è un vuoto, ovvero manca la derivata prima di y_1, per poter applicare il teorema non ci devono essere buchi.

In questo problema, inoltre, devo dimostrare anche l'esistenza della derivata prima, invece se sono verificate le ipotesi del teorema sappiamo già che esiste la soluzione del problema ed è unica, quindi ci possiamo preoccupare solo di trovare l'algoritmo più efficiente.

Dimostrazione. (Teorema 7.1)
Interpretiamo i punti $x_0, \ldots, x_n$ ordinati rispetto agli indici.

Siano $\varphi_0(x), \varphi_1(x), \ldots, \varphi_m(x)$ una base dello *spazio vettoriale* $\mathcal{P}_m$ dei polinomi di grado al più m (esempio: $\varphi_0(x) = c,\ \varphi_1(x) = x,\ \ldots,\ \varphi_m(x) = x^m$).

$$p(x) = \sum_{i=0}^{m} a_i \varphi_i(x), \qquad \text{(polinomio interpolatore)}$$

Problema: determinare i coefficienti a_i del polinomio.

$$\begin{cases} y_0 = p(x_0) = a_0\varphi_0(x_0) + a_1\varphi_1(x_0) + \cdots + a_m\varphi_m(x_0) \\ y_0' = p'(x_0) = a_0\varphi_0'(x_0) + a_1\varphi_1'(x_0) + \cdots + a_m\varphi_m'(x_0) \\ \vdots \qquad\quad = \\ y_0^{\alpha_0} = p^{(\alpha_0)}(x_0) = a_0\varphi_0^{(\alpha_0)}(x_0) + a_1\varphi_1^{(\alpha_0)}(x_0) + \cdots + a_m\varphi_m^{(\alpha_0)}(x_0). \end{cases}$$

Abbiamo ottenuto $\alpha_0 + 1$ equazioni relative al punto x_0, possiamo fare la costruzione di un pacchetto di equazioni analogo per ogni punto x_i con $i = 0, \ldots, n$, fino ad ottenere:

$$\begin{cases} y_n = p(x_n) = a_0\varphi_0(x_n) + a_1\varphi_1(x_n) + \cdots + a_m\varphi_m(x_n) \\ \vdots \\ y_n^{\alpha_n} = p^{(\alpha_n)}(x_n) = a_0\varphi_0^{(\alpha_n)}(x_n) + a_1\varphi_1^{(\alpha_n)}(x_n) + \cdots + a_m\varphi_m^{(\alpha_n)}(x_n). \end{cases}$$

Mettendo assieme tutti questi vincoli otteniamo la seguente matrice:

$$G = \begin{bmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \cdots & \varphi_m(x_0) \\ \varphi_0'(x_0) & \varphi_1'(x_0) & \cdots & \varphi_m'(x_0) \\ \vdots & & & \\ \varphi_0^{(\alpha_0)}(x_0) & \varphi_1^{(\alpha_0)}(x_0) & \cdots & \varphi_m^{(\alpha_0)}(x_0) \\ \varphi_0(x_1) & \varphi_1(x_1) & \cdots & \varphi_m(x_1) \\ \vdots & & & \\ \varphi_0^{(\alpha_1)}(x_1) & \varphi_1^{(\alpha_1)}(x_1) & \cdots & \varphi_m^{(\alpha_1)}(x_1) \\ \vdots & & & \\ \varphi_0(x_n) & \varphi_1(x_n) & \cdots & \varphi_m(x_n) \\ \vdots & & & \\ \varphi_0^{(\alpha_n)}(x_n) & \varphi_1^{(\alpha_n)}(x_n) & \cdots & \varphi_m^{(\alpha_n)}(x_n) \end{bmatrix}.$$

Numero totale di colonne: $m + 1$.

Numero di righe totali:

$$\sum_{i=0}^{n}(\alpha_i+1)=\sum_{i=0}^{n}\alpha_i+n+1=m+1.$$

$$a=\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{m-1} \\ a_m \end{bmatrix}, \quad b=\begin{bmatrix} y_0 \\ \vdots \\ y_0^{(\alpha_0)} \\ \vdots \\ y_n \\ \vdots \\ y_n^{(\alpha_n)} \end{bmatrix}.$$

Ottengo il polinomio risolvendo il sistema lineare:

$$Ga=b.$$

(Un sistema lineare nasce sempre dalla traduzione di un problema di costruzione.)

Proposizione 7.3. *Se G è non singolare allora esiste un'unica soluzione del sistema lineare.*

Dimostrazione. G è non singolare.
Supponiamo per assurdo che G sia una matrice singolare, allora esiste $\overline{a} \in \mathbb{R}^{m+1}$ con $\overline{a} \neq 0$ tale che $Ga=0$.

$$\overline{a}=\begin{bmatrix} \overline{a_1} \\ \vdots \\ \overline{a_{m+1}} \end{bmatrix}.$$

Prodotti riga per colonna:

$$\sum_{i=0}^{m}\overline{a}_i\varphi_i(x_0)=0$$

$$\sum_{i=0}^{m}\overline{a}_i\varphi_i'(x_0)=0$$

$$\vdots$$

$$\sum_{i=0}^{m}\overline{a}_i\varphi_i^{(\alpha_0)}(x_0)=0$$

$$\vdots$$

$$\sum_{i=0}^{m} \overline{a}_i \varphi_i(x_n) = 0$$

$$\vdots$$

$$\sum_{i=0}^{m} \overline{a}_i \varphi_i^{(\alpha_n)}(x_n) = 0$$

Costruiamo un polinomio $q(x)$ come combinazione lineare utilizzando come coefficienti le componenti di questo vettore:

$$q(x) = \sum_{i=0}^{m} \overline{a}_i \varphi_i(x) \qquad q(x) \in \mathcal{P}_m.$$

$q(x)$ deve avere m zeri. Ovvero:

$$q(x_0) = G^{(1)*} a = 0 \;\longrightarrow\; x_0 \text{ è radice del polinomio.}$$

$$q'(x_0) = 0 \;\longrightarrow\; x_0 \text{ è radice della derivata.}$$

x_0 è radice di molteplicità due.

Nota Bene. α radice di molteplicità m per $q(x)$:

$$\longrightarrow\; q(x) = (x-\alpha)^m S(x),$$

$$\begin{aligned} q'(x) &= m(x-\alpha)^{m-1} S(x) + (x-\alpha)^m S'(x) \\ &= (x-\alpha)^{m-1}[mS(x) + (x-\alpha)S(x)]. \end{aligned}$$

α è radice di $q'(x)$ di molteplicità $m-1$.

$$q^{(m-1)}(x) = D^{m-1}[(x-\alpha)^m S(x)] = \sum_{k=0}^{m-1} \binom{m-1}{k} D^{m-1-k}(x-\alpha)^m D^k S(x).$$

Quindi sappiamo che α è soluzione di molteplicità m, allora è soluzione fino alla derivata $m-1$-esima.

Problema contrario:
so solo che α è radice di $q(x)$.

$$q(x) = (x-\alpha)S(x),$$

*Prima riga di G.

e so che α è anche radice della derivata prima:

$$q'(x) = S(x) + (x-\alpha)S'(x) = 0 \Longleftrightarrow S(\alpha) = 0.$$

$$\longrightarrow \ S(x) = (x-\alpha)R(x).$$

$$q'(x) = (x-\alpha)R(x) + (x-\alpha)S'(x).$$

$$\longrightarrow \ q(x) = (x-\alpha)^2 R(x).$$

Allora α è radice di molteplicità almeno 2.

Nel punto x_0 ho $\alpha_0 + 1$ zeri dato che $q(x)$ annulla fino alla derivata α_0. Analogamente fino a x_n avremo:

$$(\alpha_0 + 1) + (\alpha_1 + 1) + \cdots + (\alpha_n + 1) = m + 1 \text{ zeri.}$$

$$\longrightarrow \ q(x) = 0 \ \longrightarrow \overline{a} = 0.$$

Assurdo.

Quindi G risulta non singolare.

■

G si dice matrice di Gram.

(Teorema 7.1)

■

Osservazione 7.4. (Unicità del polinomio interpolatore.)
Siano $q(x)$, $p(x) \in \mathcal{P}_m$ tali che:

$$p^{(l)}(x_i) = y_i^{(l)},$$
$$q^{(l)}(x_i) = y_i^{(l)}.$$
$$i = 0, \ldots, n; \ l = 0, \ldots, \alpha_i.$$

Definiamo $r(x) \in \mathcal{P}_m$ tale che:

$$r(x) = p(x) - q(x).$$

Si ha dunque $r^{(l)}(x_i) = p^{(l)}(x_i) - q^{(l)}(x_i) = y_i^{(l)} - y_i^{(l)} = 0$. Un polinomio di grado m con $m+1$ zeri. Il polinomio r è nullo, quindi i polinomi p e q sono uguali ($p = q$).

7.1 Algoritmi di costruzione del polinomio interpolatore.

Siano $x_0, x_1, \ldots, x_n$ punti distinti ($x_i \neq x_j,\ i \neq j$) e $y_0, y_1, \ldots, y_n$ [$\alpha_0 = \alpha_1 = \cdots = \alpha_n = 0$]. Costruiamo il polinomio, che per il teorema 7.1 sappiamo esistere ed essere unico.

$$\exists!\ p(x) \in \mathcal{P}_n \colon p(x_i) = y_i, \quad i = 0, 1, \ldots, n.$$

Supponiamo di prendere $\varphi_0(x), \ldots, \varphi_n(x)$ base di $\mathcal{P}_n$.

$$p(x) = \sum_{i=0}^{n} a_i \varphi_i(x).$$

$$p(x_j) = \sum_{i=0}^{n} a_i \varphi_i(x_j), \quad j = 0, \ldots, n.$$

Vediamo la corrispondente matrice di Gram:

$$G = \begin{bmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \cdots & \varphi_m(x_0) \\ \varphi_0(x_1) & \varphi_1(x_1) & \cdots & \varphi_m(x_1) \\ & & n+1 \text{ righe} & \\ \vdots & & & \vdots \\ & & n+1 \text{ colonne} & \\ \varphi_0(x_n) & \varphi_1(x_n) & \cdots & \varphi_m(x_n) \end{bmatrix}.$$

$$a = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix}, \quad b = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

G è non singolare, possiamo quindi risolvere il problema $Ga = b$ ad esempio con l'algoritmo di Gauss.

Osservazione 7.5. Se le x_i calcolate differiscono anche solo di quantità infinitesime dalle x_i originali (ad esempio una linea retta), costruiamo proprio un modello diverso (nel nostro esempio una funzione ondulatoria, di grado maggiore quindi). Un braccio meccanico sbaglia traiettoria, con conseguenze non desiderabili.

Con il metodo della risoluzione del sistema lineare abbiamo quindi un rischio di errore elevato ed un costo computazionale alto. Vediamo alcune semplificazioni.

7.1.1 Matrice di Vandermonde.

Segliamo una base più semplice della precedente:

$$1,\ x,\ x^2,\ \ldots,\ x^n.$$

$$p(x) = \sum_{i=0}^{n} a_i x^i.$$

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ & & & & \\ & & n+1 \text{ righe} & & \\ \vdots & & & & \vdots \\ & & n+1 \text{ colonne} & & \\ & & & & \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix}.$$

$$a = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix}, \quad b = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

La matrice V si dice matrice di Vandermonde, ora il sistema lineare da risolvere è $Va = b$.

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

Segue che il determinante di V è diverso da zero (dimostrazione per esercizio).

Con il cambio di base abbiamo ottenuto un problema più semplice, ma occorre risolvere sempre un sistema lineare. Esiste una base che mi permette di non risolvere alcun sistema lineare? Ciò accade se troviamo una base tale che:

$$G \equiv I.$$

Esercizio. Esercizio preliminare.

Siano x_0, x_1, x_2 distinti, $l_0, l_1, l_2 \in \mathcal{P}_2$.

$$l_0(x_i) = \delta_{0,i} = \begin{cases} 1 & \text{se } i = 0 \\ 0 & \text{se } i \neq 0 \end{cases} \qquad i = 0, 1, 2.$$

$$l_1(x_i) = \delta_{1,i} = \begin{cases} 1 & \text{se } i = 1 \\ 0 & \text{se } i \neq 1 \end{cases} \qquad i = 0, 1, 2.$$

$$l_2(x_i) = \delta_{2,i} = \begin{cases} 1 & \text{se } i = 2 \\ 0 & \text{se } i \neq 2 \end{cases} \qquad i = 0, 1, 2.$$

$l_0(x) = \alpha_0(x - x_1)(x - x_2)$,
$l_0(x_0) = \alpha_0(x_0 - x_1)(x_0 - x_2) = 1 \longrightarrow \alpha_0 = \frac{1}{(x_0 - x_1)(x_0 - x_2)}$,

$$l_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}.$$

$l_1(x) = \alpha_1(x - x_0)(x - x_2)$,
$l_1(x_1) = \alpha_1(x_1 - x_0)(x_1 - x_2) = 1 \longrightarrow \alpha_1 = \frac{1}{(x_1 - x_0)(x_1 - x_2)}$,

$$l_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}.$$

$l_2(x) = \alpha_2(x - x_1)(x - x_0)$,
$l_2(x_2) = \alpha_2(x_2 - x_1)(x_2 - x_0) = 1 \longrightarrow \alpha_2 = \frac{1}{(x_2 - x_1)(x_2 - x_0)}$,

$$l_2(x) = \frac{(x - x_1)(x - x_0)}{(x_2 - x_1)(x_2 - x_0)}.$$

Proposizione 7.6. *I polinomi l_0, l_1 e l_2 sono linearmente indipendenti.*

Dimostrazione. Consideriamo la seguente combinazione lineare:

$$l(x) := \; c_0 l_0(x) + c_1 l_1(x) + c_2 l_2(x) = 0.$$

Tesi: $l(x) = 0 \Longleftrightarrow c_i = 0, \quad \forall i = 0, 1, 2.$

$$c_0 \underbrace{l_0(x_0)}_{1} = 0 \longrightarrow c_0 = 0.$$

$$c_1 \underbrace{l_1(x_1)}_{1} = 0 \longrightarrow c_1 = 0.$$

$$c_2 \underbrace{l_2(x_2)}_{1} = 0 \longrightarrow c_2 = 0.$$

■

Vediamo la matrice di Gram.

$$G = \begin{bmatrix} l_0(x_0) & l_1(x_0) & l_2(x_0) \\ l_0(x_1) & l_1(x_1) & l_2(x_1) \\ l_0(x_2) & l_1(x_2) & l_2(x_2) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I.$$

Vuol dire che:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}.$$

$p(x) = a_0 l_0(x) + a_1 l_1(x) + a_2 l_2(x) = y_0 l_0(x) + y_1 l_1(x) + y_2 l_2(x)$. Questa è la generica equazione della parabola.

Osservazione 7.7. Ciascun polinomio l_i è di secondo grado.

7.1.2 Il polinomio interpolatore di Lagrange.

Osserviamo questa situazione: $p(x) \in \mathcal{P}_5$

x_0	x_1	x_2	x_3	x_4	x_5
y_0	y_1	y_2	y_3	y_4	y_5

Abbiamo sei vincoli di passaggio, allora il polinomio $p(x)$ è unico ed il polinomio inperpolatore coincide con $p(x)$.

Se invece avessimo solo cinque vincoli di passaggio?

x_0	x_1	x_2	x_3	x_4
y_0	y_1	y_2	y_3	y_4

Abbiamo troppi pochi vincoli, potremmo avere il polinomio preciso solo di quarto grado al massimo. Il polinomio di interpolazione rappresenterà quindi solo un'approssimazione di $p(x)$.

Nota Bene. Esiste una soluzione unica solamente quando il numero dei nodi coincide con il numero dei gradi di libertà del polinomio.

Definizione 7.8. Siano $x_0, x_1, \ldots, x_n$ distinti , $l_0, \ldots, l_j, \ldots, l_n \in \mathcal{P}_n$ tali che:

$$l_0(x_i) = \delta_{0,i} = \begin{cases} 1 & \text{se } i = 0 \\ 0 & \text{se } i \neq 0 \end{cases} \qquad i = 0, \ldots n.$$

$$\vdots$$

$$l_j(x_i) = \delta_{j,i} = \begin{cases} 1 & \text{se } i = j \\ 0 & \text{se } i \neq j \end{cases} \qquad i = 0, \ldots n.$$

$$\vdots$$

$$l_n(x_i) = \delta_{n,i} = \begin{cases} 1 & \text{se } i = n \\ 0 & \text{se } i \neq n \end{cases} \qquad i = 0, \ldots n.$$

$$l_0(x) = \alpha_0(x - x_1)(x - x_2)\cdots(x - x_n),$$
$$l_0(x_0) = \alpha_0(x_0 - x_1)(x_0 - x_2)\cdots(x_0 - x_n) = 1.$$
$$\longrightarrow \ \alpha_0 = \frac{1}{(x_0 - x_1)(x_0 - x_2)\cdots(x_0 - x_n)}.$$
$$l_0(x) = \frac{(x - x_1)(x - x_2)\cdots(x - x_n)}{(x_0 - x_1)(x_0 - x_2)\cdots(x_0 - x_n)}.$$
$$\vdots$$
$$l_j(x) = \alpha_j(x - x_0)\cdots(x - x_{j-1})(x - x_{j+1})\cdots(x - x_n),$$
$$l_j(x_j) = \alpha_j(x_j - x_0)\cdots(x_j - x_{j-1})(x_j - x_{j+1})\cdots(x_j - x_n) = 1.$$
$$\longrightarrow \ \alpha_J = \frac{1}{(x_j - x_0)\cdots(x_j - x_{j-1})(x_j - x_{j+1})\cdots(x_j - x_n)}.$$
$$l_j(x) = \frac{(x - x_0)\cdots(x - x_{j-1})(x - x_{j+1})\cdots(x - x_n)}{(x_j - x_0)\cdots(x_j - x_{j-1})(x_j - x_{j+1})\cdots(x_j - x_n)}.$$
$$\vdots$$
$$l_n(x) = \prod_{\substack{k=0 \\ k\neq n}}^{n} \frac{x - x_k}{x_n - x_k}.$$

I polinomi l_i sono detti *polinomi lagrangiani.*

Partendo dalla base $l_0(x), l_1(x), \ldots, l_n(x)$ di $\mathcal{P}_n$, con $l_i \in \mathcal{P}_n$ per ogni i si ha:

$$G = I, \qquad \overline{a} = b,$$

quindi il polinomio $p(x)$ che ne risulta:

$$\begin{aligned} p(x) &= y_0 l_0(x) + y_1 l_1(x) + \cdots + y_n l_n(x) \\ &= \sum_{i=0}^{n} y_i l_i(x) \\ &= \sum_{i=0}^{n} y_i \prod_{\substack{k=0 \\ k\neq i}}^{n} \frac{x - x_k}{x_i - x_k}. \end{aligned}$$

I coefficienti del polinomio sono le valutazioni y_i del polinomio nelle ascisse x_i assegnati come dati.
Il polinomio $p(x)$ si dice *polinomio interpolatore di Lagrange.*

$$p(x) = \sum_{i=0}^{n} y_i \prod_{\substack{k=0 \\ k\neq i}}^{n} \frac{x - x_k}{x_i - x_k}.$$

Osservazione 7.9. Se risolvessimo un sistema lineare per trovare il polinomio interpolatore dovremmo studiare il condizionamento, comunque avremmo una potenzialità d'errore. Quindi è inutile e sbagliato voler usare la base canonica $(1, x, x^2, \ldots)$ perchè dipende dalla sensibilità della matrice.

Invece con i polinomi di Lagrange la matrice non esiste, quindi non occorre studiare il condizionamento. Vedremo comunque che non si userà neanche questo sistema poiché abbiamo troppe valutazioni.

Valutiamo ora il costo del polinomio lagrangiano. Consideriamo $l_0(x)$:

$$l_0(x) = \prod_{\substack{k=0 \\ k\neq 0}}^{n} \frac{x - x_k}{x_0 - x_k},$$

abbiamo n somme al numeratore ed altrettante al denominatore, 1 divisione (se cosideriamo un'unica linea di frazione) e $2(n-1)$ prodotti. Consideriamo ora $l_1(x)$:

$$l_1(x) = \prod_{\substack{k=0 \\ k\neq 1}}^{n} \frac{x - x_k}{x_1 - x_k},$$

qui abbiamo sempre 1 divisione e $2(n-1)$ prodotti, ma il numero di somme è n poichè le altre n le ho già calcolate al passo precedente!

Iterando il discorso si evince che il costo preponderante (che non cambia) è quello dei prodotti, mentre le somme diminuiscono sempre più. Pertanto il costo totale del polinomio $p(x)$ in termini di prodotti è $2(n-1)(n+1)$ che equivale a circa $\mathrm{O}(2n^2)$.

Osservazione 7.10. Siano $x_0, x_1, \ldots, x_n$ distinti , $l_0, \ldots, l_j, \ldots, l_n \in \mathcal{P}_n$ tali che:

$$\begin{array}{cccc} x_0 & x_1 & \cdots & x_n \\ y_0 & y_1 & \cdots & y_n \end{array}$$

$$p(x) = \sum_{i=0}^{n} y_i l_i(x).$$

Vogliamo aggiungere un punto x_{n+1} e un passaggio y_{n+1}, tipo se ci interessasse un istante potremmo prenderne uno vicino per fare uno zoom.

Come cambia il fenomeno aggiungendo l'informazione?

Vogliamo vedere se il modello è sensibile all'inserimento di questo nuovo dato o no. Costruiamo $\overline{p}(x)$;

$$\overline{p}(x) = \sum_{i=0}^{n+1} y_i l_i(x).$$

Possiamo costruire $\overline{p}$ a partire da p o dobbiamo buttarlo via? Ovvero possiamo risparmiare conti?

$$\overline{p}(x) = \sum_{i=0}^{n} y_i l_i(x) + y_{n+1} l_{n+1}(x) \stackrel{?}{=} p(x) + y_{n+1} l_{n+1}(x).$$

Vorrei, ma non è così, infatti:

$$l_i = \prod_{\substack{k=0 \\ k \neq i}}^{n+1} \neq \prod_{\substack{k=0 \\ k \neq i}}^{n},$$

da cui segue che $\overline{p}(x^*) \neq p(x^*) + y_{n+1} l_{n+1}(x^*)$.

Vorrei comunque trovare un legame tra p e $\overline{p}$ per risparmiare i conti.

Osservazione 7.11. Il polinomio interpolatore di Lagrange ha notevoli vantaggi:

- La matrice G di Gram associata alla base dei polinomi elementari coincide con la matrice identica I;
- i coefficienti del polinomio sono proprio i dati y_i;
- i polinomi $l_n(x)$ dipendono solo dalle x. Ovvero nel caso in cui si avesse:

  $$\begin{array}{cccc} x_0 & x_1 & \cdots & x_n \\ y'_0 & y'_1 & \cdots & y'_n \end{array}$$

  allora il polinomio $p_n(x)$ sarebbe semplicemente:

  $$p(x) = \sum_{i=0}^{n} y'_i l_i(x).$$

Abbiamo però visto che tale polinomio è sensibile all'inserimento di un punto. Ovvero dobbiamo rifare tutti i conti.

Esercizio. Che risposta diamo ad un problema del tipo:

$$\begin{array}{ccc} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ y'_0 & & \end{array}$$

con quattro gradi di libertà?

Costruiamo "a mano" la matrice associata al polinomio di terzo grado e discutiamo su di esso.

7.2 Applicazioni dei polinomi interpolatori.

Calcolo integrale.

Sia $f(x) \in \mathcal{C}([a,b])$ e

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

come si risolve non conoscendo F? Oppure

$$(b-a) \cdot f(\lambda),$$

come si calcola non conoscendo λ?

Se esistesse $g(x)$ facilmente integrabile tale che $f(x) \cong g(x)$ allora:

$$\int_a^b f(x)dx \cong \int_a^b g(x)dx.$$

Come g prendiamo un polinomio che approssima f (i polinomi sono facilmente integrabili):

$$g(x) = \sum_{i=0}^{n} f(x_i) l_i(x).$$

$$\begin{aligned} \int_a^b g(x)dx &= \int_a^b \sum_{i=0}^{n} f(x_i) l_i(x)dx \\ &= \sum_{i=0}^{n} f(x_i) \underbrace{\int_a^b l_i(x)dx}_{\text{n° calcolabile}=A_i} \\ &= \underbrace{\sum_{i=0}^{n} f(x_i) A_i}_{\text{somma pesata degli } A_i}. \end{aligned}$$

La funzione f è continua su $[a,b]$, dal teorema di Weierstrass si ha che:

$$\forall \varepsilon > 0 \ \exists n_\varepsilon : \|f(x) - g_{n_\varepsilon}(x)\|_\infty < \varepsilon, \quad \text{ovvero:}$$

$$\max_{a \le x \le b} |f(x) - g_{n_\varepsilon}(x)| < \varepsilon.$$

Quindi ogni funzione continua può essere approssimata da un polinomio. L'ambiente polinomiale è buono per approssimare una f continua, ma non si parla di polinomi interpolatori. Il polinomio di Weierstrass *non* richiede il passaggio per certi punti (per costruirlo si usano i polinomi di Bernulli).

Che errore commettiamo nel calcolare l'integrale di f approssimandolo con la sommatoria?

$$\text{errore} = I(f) - \sum_{i=0}^{n} f(x_i)A_i.$$

Per trovarlo dobbiamo conoscere $f(\tilde{x}) - g(\tilde{x})$.

$$\text{errore} := f(\tilde{x}) - g(\tilde{x}).$$

Non si può conoscere il valore esatto dell'errore, per conoscerlo occorre conoscere anche f, ma a quel punto g è inutile.

Teorema 7.12. *Siano $f(x) \in \mathcal{C}^{n+1}([a,b])$, $x_0, x_1, \ldots, x_n \in [a,b]$ valori distinti e $g(x) \in \mathcal{P}_n$ il suo polinomio interpolatore ($g(x_i) = f(x_i),\ i = 0, \ldots, n$). Allora preso un arbitrario $x \in [a,b]$ con $x \neq x_i$ esiste almeno un punto $\xi_x \in I_x = [\min\{x_0, \ldots, x_n\}, \max\{x_0, \ldots, x_n\}]$ tale che:*

$$e(x) := f(x) - g(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \cdot \omega(x).$$

$$\omega(x) = \prod_{k=0}^{n}(x - x_k).$$

Dimostrazione. Sia I_x l'intervallo che contiene sia x che x_i.

Se $x \equiv x_i,\ i = 0, 1, \ldots, n,\ e(x_i) = f(x_i) - g(x_i) = 0$, quindi si può esprimere l'errore sotto questa espressione:

$$e(x) := k_x^n \cdot \omega_n(x).$$

Così, $\omega(x)$ è uguale a zero per $x = x_i$.

$$f(x) = g(x) + e(x),$$

$$f(x) = g(x) + k_x^n \cdot \omega_n(x).$$

Definiamo una nuova funzione G come segue:

$$G(t) := f(t) - g(t) - k_x^n \cdot \omega_n(t).$$

Con $f(t) \in \mathcal{C}^{n+1}$, $g(t) \in \mathcal{C}^{\infty}$, $\omega_n(t) \in \mathcal{C}^{\infty}$ e k_x^n una costante rispetto a t.

$$G(t) \in \mathcal{C}^{n+1}([a,b]).$$

$$G(x_i) = \cancelto{0}{f(x_i) - g(x_i)} - \cancelto{0}{k_x^n \cdot \omega_n(x_i)} = 0, \quad i = 0, \ldots, n.$$

G ha almeno $n+1$ zeri.

$$G(x) = \underbrace{f(x) - g(x)}_{=e(x)} - \underbrace{k_x^n \cdot \omega_n(x)}_{=e(x)} = 0, \quad \forall x \in [a,b], x \neq x_i.$$

G ha almeno $n+2$ zeri. G' si annulla in almeno un punto tra x_2 e x_3 (teorema di Rolle), il discorso è uguale per $x_0, x_1, \ldots, x_n$.

$G'(t)$ ha almeno $n+1$ zeri.
$G''(t)$ ha almeno n zeri.
$\vdots$
$G^{(n+1)}(t)$ ha almeno uno zero.

$$G^{(n+1)}(t) = f^{(n+1)}(t) - 0 - k_x^n (n+1)!$$
$$G^{(n+1)}(\xi_x) = f^{(n+1)}(\xi_x) - k_x^n (n+1)! = 0 \quad \xi_x \text{ è lo zero.}$$
$$\longrightarrow k_x^n = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}.$$
$$\longrightarrow e(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \cdot \omega(x).$$

■

Esempio. Vediamo un'applicazione:

$$I(f) = \int_a^b f(x)dx = \int_a^b (g(x) + e(x))\, dx.$$

$$I(f) = \int_a^b \sum_{i=0}^{n} f(x_i) l_i(x) dx + \int_a^b e(x)dx$$
$$= \sum_{i=0}^{n} A_i f(x_i) + \int_a^b \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \omega_n(x) dx.$$

E' necessario che f sia almeno appartenente a $\mathcal{C}^{n+1}([a,b])$ per dire questo.

Quantifichiamo l'errore $e(x)$.

$$|e(x)| = \frac{|f^{(n+1)}(\xi_x)|}{(n+1)!} |\omega_n(x)| \leq \frac{M}{(n+1)!} |\omega_n(x)| \leq \frac{M}{(n+1)!} (b-a)^{n+1}.$$
$$M := \max_{a \leq x \leq b} \left| f^{(n+1)}(\xi_x) \right|.$$

$$\omega_n(x) = (x - x_0)(x - x_1) \cdots (x - x_n), \quad \forall i \ (x - x_i) \leq b - a.$$

La disuguaglianza è interessante se $(b - a)$ è una quantità piccola, ovvero $(b - a)^{n+1}$ è ancora più piccola al crescere di n. M è sconosciuto perchè non conosciamo la f, ma a volte possiamo avere un maggiorante di $|f^{(n+1)}(\xi_x)|$ come dato del problema.

Esempio. Dati tre punti $x_0 \neq x_1 \neq x_2$ e tre passaggi relativi y_0, y_1, y_2 otteniamo i seguenti polinomi:

$$\begin{array}{cccc} x_0 & y_0 & & \\ & & g_{01}(x) & \\ x_1 & y_1 & & g_{012}(x) \\ & & g_{12}(x) & \\ x_2 & y_2 & & \end{array}$$

$$g_{01}(x) = \frac{(x - x_0)y_1 + (x_1 - x)y_0}{x_1 - x_0}, \quad \text{retta interpolatrice.}$$

$$g_{12}(x) = \frac{(x - x_1)y_2 + (x_2 - x)y_1}{x_2 - x_1}.$$

$$g_{012}(x) = \frac{(x - x_0)g_{12}(x) + (x_2 - x)g_{01}(x)}{x_2 - x_0}.$$

Mediante $g_{01}(\tilde{x})$ e $g_{12}(\tilde{x})$ calcoliamo $g_{012}(\tilde{x})$.

$$g_{012}(x_0) \stackrel{?}{=} y_0, \quad g_{012}(x_1) \stackrel{?}{=} y_1, \quad g_{012}(x_2) \stackrel{?}{=} y_2.$$

$$g_{012}(x_0) = \frac{(x_2 - x_0)g_{01}(x_0)}{(x_2 - x_0)} = g_{01}(x_0) = y_0.$$

Analogamente per gli altri dati x_1 e x_2 i risultati sono quelli attesi, quindi g_{012} è il polinomio interpolatore di grado due. E' unico ed è costruito da due polinomi di grado inferiore.

Com'è fatto il coefficiente di grado massimo? Dati A_{01} e A_{12} i coefficienti di grado massimo dei rispettivi polinomi g_{01} e g_{12} si ha:

$$\frac{A_{12} - A_{01}}{x_2 - x_0}.$$

Cosa accade se introducessimo un quarto punto x_3 ed il relativo passaggio?

$$\begin{array}{llll}
x_0 & y_0 & & \\
 & & g_{01}(x) & \\
x_1 & y_1 & & g_{012}(x) \\
 & & g_{12}(x) & \\
x_2 & y_2 & & g_{123}(x) \\
 & & g_{23}(x) & \\
x_3 & y_3 & &
\end{array} \qquad g_{0123}(x)$$

Come prima viene costruito il polinomio $g_{0123}(x)$ utilizzando i dati ottenuti in precedenza.

$$g_{0123}(x) = \frac{(x - x_0)g_{123}(x) + (x_3 - x)g_{012}(x)}{x_3 - x_0}.$$

Questo procedimento è utile quando vogliamo calcolare il valore di $g_{0123}(\tilde{x})$.

Generalizziamo il discorso. Siano:

$$\begin{array}{ccccc}
x_0 & x_1 & \cdots & x_{n-1} & x_n \\
y_0 & y_1 & \cdots & x_{n-1} & y_n
\end{array}$$

il nostro insieme di dati (punti e passaggio relativo).

Ipotesi:
$g_{01\ldots n-1}(x)$ polinomio interpolatore sui nodi $x_0, x_1 \ldots, x_{n-1}$.
$g_{12\ldots n}(x)$ polinomio interpolatore sui nodi $x_1, x_2 \ldots, x_n$.

Tesi:

$$g_{01\ldots n}(x) = \frac{(x - x_0)g_{12\ldots n}(x) + (x_n - x)g_{01\ldots n-1}(x)}{x_n - x_0},$$

polinomio interpolatore sui nodi $x_0, x_1, \ldots, x_n$.

Verifica:

$$g_{01\ldots n}(x_0) = \frac{x_n - x_0}{x_n - x_0}^{1} g_{01\ldots n-1}(x_0) = y_0.$$

$$g_{01\ldots n}(x_n) = \frac{x_n - x_0}{x_n - x_0}^{1} g_{1\ldots n}(x_n) = y_n.$$

$$\begin{aligned}
g_{01\ldots n}(x_i) &= \frac{(x_i - x_0)y_i + (x_n - x_i)y_i}{x_n - x_0} \\
&= \frac{y_i(\cancel{x_i} - x_0 + x_n - \cancel{x_i})}{x_n - x_0} \\
&= y_i \frac{\cancel{x_n - x_0}}{\cancel{x_n - x_0}} \qquad (1 \le i \le n-1). \\
&= y_i.
\end{aligned}$$

Se abbiamo un numero elevato di punti (ad esempio 54) è sbagliato calcolare il polinomio (di grado 53) direttamente, bisogna usare la tecnica sopra descritta.

$$g_{01\ldots n}(x) = \frac{(x - x_0)g_{12\ldots n} + (x_n - x)g_{01\ldots n-1}(x)}{x_n - x_0}.$$

7.3 Polinomio interpolatore con differenze divise.

Un'idea alternativa alla costruzione del polinomio interpolatore sfrutta le differenze divise.

- x_0, y_0, $p_0(x) \in \mathcal{P}_0$ tale che:

$$p_0(x_0) = y_0, \quad p_0(x) = A_0.$$

  $p_0(x_0) = A_0 = y_0$ è la differenza divisa di ordine zero.

- x_0, y_0, x_1, y_1, $p_1(x) \in \mathcal{P}_1$ tale che:

$$p_1(x_0) = y_0, \quad p_1(x_1) = y_1,$$

$$p_1(x) = p_0(x) + \boxed{\ ?\ }$$

  $\boxed{\ ?\ }$ dovrà essere un termine lineare in x (nullo nel punto x_0). In altre parole costruiremo p_1 sulla base del polinomio $p_0(x)$ già calcolato.

$$p_1(x) = p_0(x) + A_1(x - x_0) \rightarrow y_1 = p_1(x_1) = p_0(x_1) + A_1(x_1 - x_0).$$

  $A_1 \in \mathcal{P}_1$, $p_0(x_1) = y_0$ perchè p_0 è il polinomio costante, da cui:

$$A_1 = \frac{y_1 - y_0}{x_1 - x_0}, \quad \text{è la differenza divisa del prim'ordine.}$$

  Un modo alternativo per scriverla è: $A_1 = f[x_0, x_1]$.

$$p_1(x) = p_0(x) + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0).$$

(rapporto incrementale della funzione)

$$\Longrightarrow p_1(x) = f[x_0] + f[x_0, x_1](x - x_0).$$

- x_0, y_0, x_1, y_1, x_2, y_2, $p_2(x) \in \mathcal{P}_2$ tale che:

$$p_2(x_0) = y_0, \quad p_2(x_1) = y_1, \quad p_2(x_2) = y_2,$$

$$p_2(x) = p_1(x) + \boxed{\ ?\ }$$

Analogamente $\boxed{\ ?\ }$ dovrà essere un termine lineare in x (nullo nei punti x_0 e x_1), di secondo grado (poiché $\mathcal{P}_1$ è di primo grado).

$$p_2(x) = p_1(x) + A_2(x - x_0)(x - x_1).$$

Occorre determinare A_2.

$$p_2(x_2) = y_2 = p_1(x_2) + A_2(x_2 - x_0)(x_2 - x_1).$$

$$y_2 = p_0(x_2) + \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_0) + A_2(x_2 - x_0)(x_2 - x_1).$$

$$A_2 = \frac{y_2 - p_1(x_2)}{(x_2 - x_0)(x_2 - x_1)}, \quad \text{è la differenza divisa del second'ordine.}$$

Un modo alternativo per scriverla è: $A_2 = f[x_0, x_1, x_2]$. A_2 si realizza se conosciamo i rapporti incrementali sui tre punti, sarà quindi sufficiente generarlo in modo più semplice con i rapporti incrementali opportuni.

$$A_2 = \frac{y_2 - p_1(x_2)}{(x_2 - x_0)(x_2 - x_1)}$$

$$= \frac{y_2 - y_0 - \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)}$$

$$= \frac{\frac{y_2 - y_1 + y_1 - y_0}{x_2 - x_1} - \frac{y_1 - y_0}{(x_1 - x_0)(x_2 - x_1)}(x_2 - x_1 + x_1 - x_0)}{x_2 - x_0}$$

$$= \frac{\frac{y_2 - y_1}{x_2 - x_1} + \cancel{\frac{y_1 - y_0}{x_2 - x_1}} - \frac{y_1 - y_0}{x_1 - x_0} - \cancel{\frac{y_1 - y_0}{x_2 - x_1}}}{x_2 - x_0}$$

$$= \frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0} = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = f[x_0, x_1, x_2].$$

Infatti:

$$\begin{array}{cc} x_0 & x_1 \\ y_0 & y_1 \end{array}$$

$$p_{12}(x) = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) = y_1 + f[x_1, x_2](x - x_1).$$

Possiamo costruire le differenze divise così:

$$\begin{array}{llll} x_0 & y_0 & & \\ & & f[x_0,x_1] & \\ x_1 & y_1 & & f[x_0,x_1,x_2] \\ & & f[x_1,x_2] & \\ x_2 & y_2 & & \end{array}$$

$$p_0(x) = y_0 = f[x_0]; \quad p_1(x) = f[x_0] + f[x_0,x_1](x-x_0).$$
$$p_2(x) = f[x_0] + f[x_0,x_1](x-x_0) + f[x_0,x_1,x_2](x-x_0)(x-x_1).$$

I polinomi successivi verranno costruiti ricavando opportunamente le differenze divise. Il polinomio così ottenuto si chiama *polinomio interpolatore alle differenze divise* o ***polinomio di Newton***.

7.3.1 Generalizzazione del polinomio di Newton.

Siano $x_0, x_1, \ldots, x_n$ e $y_0, y_1, \ldots, y_n$, $n+1$ punti distinti con i relativi passaggi:

$$\begin{array}{cccc} x_0 & x_1 & \cdots & x_n \\ y_0 & y_1 & \cdots & y_n \end{array}$$

tali che: $p_n \in \mathcal{P}_n$, $p_i(x_i) = y_i,\ i = 0, \ldots, n$, allora:

$$p_n(x) = p_{n-1}(x) + A_n(x-x_0)(x-x_1)\cdots(x-x_{n-1})^{\dagger}.$$

$p_{n-1}(x)$ interpola i seguenti punti: $\begin{array}{cccc} x_0 & x_1 & \cdots & x_{n-1} \\ y_0 & y_1 & \cdots & y_{n-1} \end{array}$

$$p_n(x_n) = y_n = p_{n-1}(x_n) + A_n(x-x_0)(x-x_1)\cdots(x-x_{n-1}).$$

$$A_n = \frac{y_n - p_{n-1}(x_n)}{(x-x_0)(x-x_1)\cdots(x-x_{n-1})} = f[x_0,x_1,\ldots,x_n].$$

A_n si può costruire anche con la tabella dei coefficienti come visto prima:

$$\begin{array}{llllll} x_0 & y_0 & & & & \\ & & f[x_0,x_1] & & & \\ x_1 & y_1 & & f[x_0,x_1,x_2] & & \\ & & f[x_1,x_2] & & \ddots & \\ x_2 & y_2 & & & & f[x_0,x_1,\ldots,x_n] \\ & & & & & \\ \vdots & & & f[x_{n-2},x_{n-1},x_n] & & \\ & & f[x_{n-1},x_n] & & & \\ x_n & y_n & & & & \end{array}$$

[†]Il vettore A_n è fatto come in precedenza, è il coefficiente del grado massimo.

Si può dimostrare che $A_n = f[x_0, x_1, \ldots, x_n]$ per induzione.
Dimostrazione.

Base: $A_0 = f[x_0]$.

Caso induttivo su n punti.

Ipotesi induttiva:

$$\begin{array}{cccc} x_0 & x_1 & \cdots & x_{n-1} \\ y_0 & y_1 & \cdots & y_{n-1} \end{array} \qquad f[x_0, x_1, \ldots, x_{n-1}].$$

$$\begin{array}{cccc} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{array} \qquad f[x_1, x_2, \ldots, x_n].$$

A_n è il coefficiente del termine di grado massimo del polinomio p_n:

$$\frac{f[x_1, x_2, \ldots, x_n] - f[x_0, x_1, \ldots, x_{n-1}]}{x_n - x_0} = A_n.$$

$$\begin{array}{rl} p_n(x) = & f[x_0] \\ + & f[x_0, x_1](x - x_0) \\ + & f[x_0, x_1, x_2](x - x_0)(x - x_1) \\ & \vdots \\ + & f[x_0, x_1, \ldots, x_n](x - x_0)(x - x_1) \cdots (x - x_{n-1}). \end{array}$$

$$A_n = f[x_0, x_1, \ldots, x_n].$$

$p_n(x)$ è un polinomio di grado al più n ($\leq n$) che interpola tutti i punti a partire da polinomi di ordine inferiore.

■

Se volessimo costruire un polinomio "vicino" a $\overline{x}$ dovremmo costruire i polinomi interpolatori usando la costruzione delle differnze divise, vicino a $\overline{x}$.

$$\begin{array}{ccc} & x_0 & y_0 \\ & \vdots & \\ & x_k & y_k \\ \overline{x} \quad \rightarrow & \vdots & \\ & x_n & y_n \end{array}$$

Se abbiamo un fenomeno oscillante nella parte iniziale è inutile prendere un valore in cui il fenomeno è diventato piatto. Quindi è utile prendere dei dati vicini a questo punto $\overline{x}$. (Aggiungere un punto non cambia nulla, anzi significa aggiungere un termine alla tabella e ricavarne il valore).

7.3.2 Costo computazionale.

Per analizzare il costo computazionale delle differenze divise vediamo le operazioni che servono per la costruzione delle colonne:

- Differenza divisa di primo grado:

  Somme: $2(n)$;

  Divisioni: n.

- Differenza divisa di secondo grado:

  Somme: $2(n-1)$;

  Divisioni: $n-1$.

- Differenza divisa di terzo grado:

  Somme: $2(n-2)$;

  Divisioni: $n-2$.

  $\vdots$

- Differenza divisa di grado $n-2$:

  Somme: $2 \cdot 2$;

  Divisioni: 2.

- Differenza divisa di grado $n-1$:

  Somme: 2;

  Divisioni: 1.

Somme totali:
$2[n + (n-1) + (n-2) + \cdots + 2 + 1] = \frac{2(n)(n+1)}{2}$.

Divisioni totali:
$n + (n-1) + (n-2) + \cdots + 2 + 1 = \frac{(n)(n+1)}{2}$.

$$\simeq \mathrm{O}\left(\frac{n^2}{2}\right).$$

$$
\begin{aligned}
p_0(x) &= y_0 = f[x_0] \\
p_1(x) &= p_0(x) + f[x_0, x_1](x - x_0) \\
p_2(x) &= p_1(x) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\
&\vdots \\
p_n(x) &= p_{n-1}(x) + f[x_0, x_1, \ldots, x_n](x - x_0)(x - x_1)\cdots(x - x_{n-1}).
\end{aligned}
$$

Per conoscere il valore del modello in un punto $\overline{x}$ assegnato *non nodale* è sufficiente usare lo schema di cui sopra. Usando sempre il valore del polinomio precedente.

Partendo invece direttamente dal polinomio nella seguente forma:

$$p_n(x) = f[x_0]+f[x_0, x_1](x-x_0)+\cdots+f[x_0, x_1, \ldots, x_n](x-x_0)(x-x_1)\cdots(x-x_{n-1}).$$

quanto costa valutare il polinomio in un punto $\overline{x}$?

Prima di rispondere mettiamoci in una situazione più familiare:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1}x^{n-1} + a_n x^n, \quad a_n \neq 0.$$

$$p(\overline{x}) = a_0 + a_1 \overline{x} + a_2 \overline{x}^2 + \cdots + a_{n-1}\overline{x}^{n-1} + a_n \overline{x}^n, \quad a_n \neq 0.$$

Somme totali: n.

Prodotti totali:

$$1 + 2 + 3 + \cdots + \overbrace{\underbrace{(n-1)}_{\text{potenza di } n} + \underbrace{1}_{a_n \cdot x^n}}^{n} \simeq \mathrm{O}\left(\tfrac{n^2}{2}\right).$$

Un altro modo per valutare i prodotti è mantenendo in memoria le potenze precedenti, ovvero si hanno sempre due prodotti, uno per il coefficiente ed uno per la nuova potenza:
$1 + 2 + 2 + \cdots + 2 = 2n - 1 \ \simeq \mathrm{O}\,(2n)$.

Un'altra possibilità per ridurre il numero dei prodotti è la seguente:

Esempio. $n = 3$

$$
\begin{aligned}
p(\overline{x}) &= a_0 + a_1\overline{x} + a_2\overline{x}^2 + a_3\overline{x}^3 \\
&= ((a_3\overline{x} + a_2)\overline{x} + a_1)\overline{x} + a_0.
\end{aligned}
$$

Somme totali: n.
Prodotti totali: n.

$$p(x) = (\cdots(((a_n x + a_{n-1})\, x + a_{n-2})\, x + a_{n-3})\cdots)\, x + a_0.$$

Sia $d_n := a_n$, (coeffciente di grado massimo).

- $k = n-1, \ldots, 0$:

  $d_k = d_{k+1}x + a_k$;

$d_0 = p(x)$.

Consideriamo $\overline{x}$.

Sia $d_n := a_n$, (coeffciente di grado massimo).

- $k = n-1, \ldots, 0$:

  $d_k = d_{k+1}\overline{x} + a_k$;

$d_0 = p(\overline{x})$.

Questo algoritmo (ottimale) si chiama *Algoritmo di Hörmer*.

Osservazione 7.13. Con i moderni calcolatori non si coglie la differenza tra i tre algoritmi su pochi dati ($n = 10$), aumentando sensibilmente questo numero però le differenze si notano.

Tornando al seguente polinomio:

$$p_n(x) = f[x_0]+f[x_0,x_1](x-x_0)+\cdots+f[x_0,x_1,\ldots,x_n](x-x_0)(x-x_1)\cdots(x-x_{n-1})$$

possiamo sfruttare Hörmer?

$$\begin{array}{rcl} p_n(x) & = & (\cdots(((f[x_0,\ldots,x_n](x-x_{n-1})+f[x_0,\ldots,x_{n-1}])(x-x_{n-2}) \\ & + & \quad f[x_0,\ldots,x_{n-2}])(x-x_{n-3})+\cdots)\cdots)(x-x_0)+f[x_0]. \end{array}$$

$d_n := f[x_0,\ldots,x_n]$, (coeffciente di grado massimo).

$\square$ $k = n-1, \ldots, 0$:

$$d_k = d_{k+1}(x-x_k) + f[x_0,\ldots,x_k];$$

$d_0 = p_n(x)$.

Questo algoritmo nella tecnica lagrangiana non è utilizzabile.

7.3.3 Calcolo dell'errore.

Presi $n+1$ punti distinti e $n+1$ passaggi:

$$\begin{array}{cccc} x_0 & x_1 & \cdots & x_n \\ y_0 & y_1 & \cdots & y_n \end{array}$$

tali che: $p_n \in \mathcal{P}_n$, $p_i(x_1) = y_i$, $i = 0, \ldots, n$; $f(x) \in \mathcal{C}^{n+1}([a,b])$ e $x_i \in (a,b)$. Per il teorema 7.12:

$\forall x \in [a,b], x \neq x_i,\ (i = 0, \ldots n)$

$$e(x) = f(x) - p(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}\omega_n(x),$$

$$\text{con } \ \omega_n(x) = (x - x_0)\cdots(x - x_n) = \prod_{k=0}^{n}(x - x_k).$$

Vogliamo una forma alternativa dell'errore.

Esempio. Dati due punti e due passaggi:

$$\begin{array}{cc} x_0 & x_1 \\ f(x_0) & f(x_1) \end{array}$$

$p_1(x) \in \mathcal{P}_1 \colon p_i(x_i) = f(x_i), \quad i = 0, 1.$
$p_1(x) = f[x_0] + f[x_0, x_1](x - x_0).$

Sia $x \neq x_0, x_1$,
$p_2(x) \in \mathcal{P}_2$ interpolante in x_0, x_1, x, cioè tale che:

$$p_2(x) = f(x),\ p_2(x_i) = f(x_i) \quad i = 0, 1.$$

$$p_2(t) = p_1(t) + f[x_0, x_1, x](t - x_0)(t - x_1).$$
$$p_2(x) = p_1(x) + f[x_0, x_1, x](x - x_0)(x - x_1) = f(x).$$

$$\longrightarrow\ f(x) - p_1(x) = f[x_0, x_1, x](x - x_0)(x - x_1) =: e(x). \qquad (7.1)$$

$$f(x) - p_1(x) = \frac{f''(\xi_x)}{2!}(x - x_0)(x - x_1) \quad \text{(errore formale)} \qquad (7.2)$$

In 7.1 abbiamo $f[x_0, x_1, x]$ al posto di $\frac{f''(\xi_x)}{2!}$, però se risolvessimo $f[x_0, x_1, x]$ otterremmo un $f(x)$ non conosciuto, quindi non abbiamo ancora un risultato computazionale. Abbiamo ottenuto il seguente risultato:

$$f[x_0, x_1, x] = \frac{f''(\xi_x)}{2!}$$

in quanto $e(x) = (7.1) = (7.2)$, ovvero l'errore è lo stesso.

Il collegamento tra differenza divisa e derivata è possibile quando la funzione è sufficientemente regolare. Il vantaggio è che in $f[x_0, \dots, x_n]$ serve solo f e non devo calcolarne la derivata. *Questo sarà importantissimo quando parleremo di equazioni differenziali.*

Esempio. Supponiamo di prendere due punti molto vicini:

$$\begin{array}{cc} x_0 & x_1 \\ f(x_0) & f(x_1) \end{array}$$

tali che $x_1 = x_0 + \varepsilon$.

$$f[x_0, x_1] = f[x_0, x_0 + \varepsilon] = \frac{f(x_0 + \varepsilon) - f(x_0)}{\cancel{x_0} + \varepsilon - \cancel{x_0}}.$$

$$\underbrace{\lim_{\varepsilon \to 0} f[x_0, x_0 + \varepsilon]}_{= f[x_0, x_0]} = \lim_{\varepsilon \to 0} \frac{f(x_0 + \varepsilon) - f(x_0)}{\varepsilon} = f'(x_0).$$

Se consideriamo $f'(x_0)$, lo possiamo associare a $f[x_0, x_0] \neq \frac{y_0 - y_0}{x_0 - x_0}$, che è la differenza divisa su due nodi coincidenti.

Consideriamo ora la seguente situazione:

$$\begin{array}{cc} x_0 & y_0 \\ & f[x_0, x_0] = f'(x_0) \\ x_0 & y_0 \\ & f[x_0, x_1] \\ x_1 & y_1 \end{array}$$

$\xi_x \in [x_0, x_1], \quad x_0 = x + \varepsilon_0, \quad x_1 = x + \varepsilon_1.$

$$\underbrace{\lim_{\varepsilon_0, \varepsilon_1 \to 0} f[x_0, x_1, x]}_{= f[x, x, x]} = \lim_{\substack{\varepsilon_0 \to 0 \\ \varepsilon_1 \to 0}} \frac{f''(\xi_x)}{2!} = \frac{f''(\xi_x)}{2!}.$$

$$f[x_0, x_1, x] = \frac{f''(\xi_x)}{2!}.$$

Anche una differenza divisa del secondo ordine con tre punti coincidenti è interpolabile formalmente in funzione della derivata seconda, coincidono a meno di una costante.

$$
\begin{array}{llll}
x_0 & y_0 & & \\
 & & f[x_0,x_0]=f'(x_0) & \\
x_0 & y_0 & & f[x_0,x_0,x_0]=\frac{f''(x)}{2!} \\
 & & f[x_0,x_0]=f'(x_0) & \\
x_0 & y_0 & & f[x_0,x_0,x_1]=\frac{f[x_0,x_1]-f'(x_0)}{x_1-x_0} \\
 & & f[x_0,x_1] & \\
x_1 & y_1 & &
\end{array}
\qquad f[x_0,x_0,x_0,x_1]
$$

Tornando all'errore, generalizzando il discorso precedente, dati $n+1$ punti distinti e $n+1$ passaggi:

$$
\begin{array}{cccc}
x_0 & x_1 & \cdots & x_n \\
f(x_0) & f(x_1) & \cdots & f(x_n)
\end{array}
$$

$$
f(x)\in\mathcal{C}^{(n+1)}([a,b]),\quad e(x)=f(x)-p(x)=\frac{f^{(n+1)}(\xi_x)}{(n+1)!}\omega_n(x).
$$

$p(x_i)=f(x_i),\quad i=0,\dots,n.$

Sia $x\neq x_i\quad i=0,\dots,n$ e p_{n+1} interpolante in $x_0,\dots x_n,x$ tale che:

$$
p_{n+1}(x)=f(x),\quad p_{n+1}(x_i)=f(x_i)\quad i=0,\dots,n.
$$

$$
p_{n+1}(t)=p_n(t)+f[x_0,x_1,\dots,x_n,x](t-x_0)(t-x_1)\cdots(t-x_n).
$$

$$
p_{n+1}(x)=p_n(x)+f[x_0,x_1,\dots,x_n,x](x-x_0)(x-x_1)\cdots(x-x_n)=f(x).
$$

$$
f(x)-p_n(x)=f[x_0,x_1,\dots,x_n,x]\underbrace{(x-x_0)(x-x_1)\cdots(x-x_n)}_{\omega_n(x)}.
$$

$$
f(x)-p(x)=\frac{f^{(n+1)}(\xi_x)}{(n+1)!}\omega_n(x).
$$

$$
\longrightarrow\ f[x_0,x_1,\dots,x_n,x]=\frac{f^{(n+1)}(\xi_x)}{(n+1)!}.
$$

$f[x_0,x_1,\dots,x_n,x]$ è la differenza divisa di ordine $n+1$ (poiché abbiamo $n+2$ punti). Formalmente abbiamo l'uguaglianza di cui sopra, poi buttando via qualcosa avremo l'errore; l'incertezza deriva dal fatto che non si conosce il valore di $f(x)$ fuori dai nodi. Si noti che il fattoriale al denominatore ha l'ordine della derivata.

7.4 Nodi di Chebyshev.

Riprendiamo dai polinomi di Lagrange.

Siano:

$$\begin{array}{cccc} x_0 & x_1 & \cdots & x_n \\ y_0 & y_1 & \cdots & y_n \end{array}$$

il nostro insieme di dati (punti distinti e passaggio relativo),

$$p(x) \in \mathcal{P}_n, \quad p(x_i) = y_i, \quad i = 0, \ldots, n.$$

$$p(x) = \sum_{i=0}^{n} y_i l_i(x).$$

Assegnamo anche $\overline{y}_0, \overline{y}_1, \ldots, \overline{y}_n$, con $\overline{y}_i = y_i + \varepsilon_i$, con ε_i perturbazioni.

$$\longrightarrow \ \varepsilon_i = \overline{y}_i - y_i.$$

Costruiamo ora il polinomio perturbato $\overline{p}(x) \in \mathcal{P}_n$:

$$\overline{p}(x) = \sum_{i=0}^{n} \overline{y}_i l_i(x).$$

$$l_i(x) = \prod_{\substack{k=0 \\ k \neq i}}^{n} \frac{x - x_k}{x_i - x_k}.$$

$$p(x) - \overline{p}(x) = \sum_{i=0}^{n} y_i l_i(x) - \sum_{i=0}^{n} \overline{y}_i l_i(x) = \sum_{i=0}^{n} (y_i - \overline{y}_i) l_i(x).$$

$$\begin{aligned} |p(x) - \overline{p}(x)| &= \left| \sum_{i=0}^{n} (y_i - \overline{y}_i) l_i(x) \right| \\ &\leq \sum_{i=0}^{n} |y_i - \overline{y}_i| \, |l_i(x)| \\ &= \sum_{i=0}^{n} |\varepsilon_i| \, |l_i(x)| \\ &\leq \sum_{i=0}^{n} \|\varepsilon_i\|_\infty \, |l_i(x)| \\ &= \|\varepsilon\|_\infty \sum_{i=0}^{n} |l_i(x)| \, . \end{aligned}$$

$$\text{Considerando il vettore } \varepsilon = \begin{bmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{bmatrix}.$$

$$\max_{x_0 \le x \le x_n} |p(x) - \overline{p}(x)| \le \max_{x_0 \le x \le x_n} \|\varepsilon\|_\infty \sum_{i=0}^{n} |l_i(x)| = \|\varepsilon\|_\infty \underbrace{\max_{x_0 \le x \le x_n} \sum_{i=0}^{n} |l_i(x)|}_{\Lambda_n \text{ costante di Lebesque}}.$$

$$\longrightarrow \|p(x) - \overline{p}(x)\|_\infty \le \|\varepsilon\|_\infty \Lambda_n.$$

Osservazione 7.14. Λ_n dipende solo dai nodi.

$$\|p(x) - \overline{p}(x)\|_\infty \le \|\varepsilon\|_\infty \Lambda_n.$$

Questo è interpretabile come la stima dell'errore sul polinomio $\overline{p}(x)$ rispetto a $p(x)$. Più Λ_n è piccolo e più è piccolo l'errore relativo al calcolo di $p(x)$. E' importante minimizzare l'errore di propagazione.

Sia $f(x) \in C^0([a,b])$ tale che:

$$\begin{array}{cccc} x_0 & x_1 & \cdots & x_n \\ y_0 & y_1 & \cdots & y_n \end{array} \quad \in [a,b]$$

$y_i = f(x_i), \quad i = 0, \ldots, n.$

In realtà non avremo valori y_i esatti, ma approssimazioni dovuti all'*approssimazione di macchina*.

7.4.1 Minimizzazione di Λ_n.

Qual'è la scelta dei punti nodali che rende la costante di Lebesque (Λ_n) minima? A questa domanda risponde *Chebyshev*:

- si prenda come intervallo $[-1,1] = [a,b]$;
- si costruisca una sequenza di polinomi, il minimo Λ_n si ottiene prendendo come nodi gli $n+1$ zeri del polinomio ortogonale di Chebishev.

  (Quindi se possiamo, utilizziamo i nodi di Chebishev).

Definizione 7.15. Sia $x \in [a,b]$, $c(x)$ il polinomio di Chebyshev di grado $n+1$, gli $n+1$ zeri di $c(x)$ sono gli x_i tali che:

$$x_i = \frac{b-a}{2} \cos\left(\frac{2i-1}{n} \cdot \frac{\pi}{2}\right) + \frac{a+b}{2}, \qquad i = 0, \ldots, n.$$

$x_i \in [a,b]$.

Sia $x_i \in [-1,1]$ un nodo di Chebyshev:

$$t = \alpha x_i + \beta, \quad t_i = \frac{b-a}{2} x_i + \frac{b-a}{2}.$$

t, t_i sono nodi nell'intervallo $[a,b]$.

$$a = \alpha(-1) + \beta \longrightarrow 2\beta = a + b;$$
$$b = \alpha(1) + \beta \longrightarrow 2\alpha = b - a.$$

Il procedimento è un po' lungo, quindi utilizziamo punti simili (che vengono sempre chiamati in gergo "punti di Chebyshev").

1. $x_i = \cos \frac{2i-1}{2n}\pi, \quad i = 0, \ldots, n.$
2. $x_i = \cos \frac{i}{n}\pi, \quad i = 0, \ldots, n.$

$$\longrightarrow \Lambda_n \simeq e^{\frac{n}{2}}, \quad x_i \text{ punti equispaziati.}$$

$$h = \frac{b-a}{n}, \quad x_i = a + ih, \quad i = 0, \ldots, n.$$

$$\Lambda_n \simeq e^{\frac{n}{2}}, : \quad \text{comportamento esplosivo.}$$

In conclusione il polinomio che costruiamo può discostarsi molto da quello teorico senza errori nei dati. Se invece utilizziamo i nodi di Chebyshev:

$$\longrightarrow \Lambda_n \simeq \frac{2}{\pi} \log n,$$

ovvero la divergenza è *molto* più lenta.

$$\lim_{n\to\infty} \frac{\Lambda_n}{\log n} = \frac{2}{\pi}.$$

Abbiamo una riduzione di Λ_n, questo garantisce una maggiore precisione del polinomio approssimante $\overline{p}$. In sostanza la costante di Lebesque è l'*indice di stabilità* del polinomio interpolatore.

7.4.2 Relazione tra errori relativi.

$\|p(x)\|_\infty = \max\limits_{x_0 \le x \le x_n} |p(x)| \ge \|p(x_i)\|_\infty = \max\limits_{0 \le i \le n} |p(x_i)| = \|y\|_\infty.$

$$y = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

Osservazione 7.16.

$$\underbrace{\frac{\|p(x) - \overline{p}(x)\|_\infty}{\|p(x)\|_\infty}}_{\text{errore relativo sul polinomio}} \le \frac{\|\varepsilon\|_\infty}{\|y\|_\infty} \Lambda_n \Bigg\} \text{errore relativo ai dati iniziali.}$$

Teorema 7.17. *Sia $f(x) \in \mathcal{C}^0([a,b])$, siano $x_0, x_1, \ldots, x_n$ punti distinti in $[a,b]$ e $p(x)$ il polinomio interpolatore di f:*

$$p(x) \in \mathcal{P}_n \colon p(x_i) = f(x_i), \quad i = 0, \ldots, n.$$

Allora:

$$\max_{a \le x \le b} |f(x) - \overline{p}(x)| = \|f(x) - p(x)\|_\infty \le (1 + \Lambda_n)\, E_n(t).$$

$$E_n(t) := \inf_{q(x) \in \mathcal{P}_n} \|f(x) - q(x)\|_\infty.$$

$E_n(t)$ è il grado di approssimazione della funzione f in norma infinito.

Osservazione 7.18. q è un polinomio qualsiasi che varia nella classe dei polinomi di grado minore o uguale ad n.

Un'idea è quella di costruire una successione di polinomi interpolanti che tenda ad f.

$$\begin{array}{llllllll}
x_0 & & p_0(x) & & & & & \\
x_0 & x_1 & & p_1(x) & & & & \\
x_0 & x_1 & x_2 & & p_2(x) & & & \\
x_0 & x_1 & x_2 & x_3 & & p_3(x) & & \\
\vdots & & & & & & & \\
x_0 & & & \cdots & x_n & & p_n(x) & \\
\vdots & & & & & & & \\
x_0 & & & \cdots & x_n & \cdots & x_m & p_m(x)
\end{array}$$

Otteniamo così una matrice infinita costituita da elementi riga diversi.
Il risultato che vogliamo é:

$$\lim_{n\to+\infty} p_n(x) \stackrel{?}{=} f(x).$$

Questo risultato *non c'è*! Ovvero aggiungere dei punti non sempre migliora il modello.

Esempio. Funzione di Runge.

$$f(x) = \frac{1}{1+x^2}, \quad [a,b] = [-5,5].$$

$$x_i = -5 + ih, \quad h = \frac{10}{n}, \quad i = 0, \ldots, n.$$

Costruiamo i polinomi interpolatori (con la tecnica che vogliamo), disegnandoli al variare di n, vediamo come migliora o peggiora l'approssimazione del polinomio. Nell'avvicinarsi ai bordi dell'intervallo (-5 e 5) abbiamo errori grandi.

Nota Bene. Il problema è nei nodi scelti. Se avessimo altre famiglie di nodi potremmo avere *convergenza*. In sostanza occorre scegliere opportunamente i nodi.

Dimostrazione. (Teorema 7.17)

$$\begin{aligned} f(x) - p(x) &= f(x) - q(x) + q(x) - p(x) \\ &= f(x) - q(x) - (p(x) - q(x)). \end{aligned}$$

$$\begin{aligned} |f(x) - p(x)| &= |f(x) - q(x) - (p(x) - q(x))| \\ &\leq |f(x) - q(x)| - |p(x) - q(x)| \\ &= |f(x) - q(x)| + |\textstyle\sum_{i=0}^n f(x_i)l_i(x) - \sum_{i=0}^n q(x_i)l_i(x)| \\ &= |f(x) - q(x)| + |\textstyle\sum_{i=0}^n (f(x_i) - q(x_i))l_i(x)|\,. \end{aligned}$$

Abbiamo potuto esprimere $q(x)$ in questa forma:

$$q(x) = \sum_{i=0}^{n} q(x_i)l_i(x)$$

poiché q è un polinomio equivalente al suo polinomio interpolatore.

$$\begin{aligned} \longrightarrow \ \|f(x) - p(x)\|_\infty &\leq \|f(x) - q(x)\|_\infty + \|f(x) - q(x)\|_\infty \| \underbrace{\sum_{i=0}^{n} l_i(x)}_{\Lambda_n} \|_\infty \\ &= \|f(x) + q(x)\|_\infty (1 + \Lambda_n). \end{aligned}$$

■

Perchè concentrarsi sull'interpolazione se non porta a convergenza? Ovviamente perchè ci sono condizioni che la permettono. Cerchiamo uno spazio di polinomi denso nello spazio di funzioni.

Al posto di utilizzare il $\lim_{n\to+\infty}$ fissiamo n e facciamo tendere l'ampiezza dell'intervallo $[a,b]$ a zero. In pratica "tagliuzziamo" il dominio in tanti sottointervalli, così otteniamo una buona approssimazione di f per ogni intervallino.

$$\omega_n(x) = (x-x_0)(x-x_1)\cdots(x-x_n) \leq (b-a)^n.$$
$$b-a \longrightarrow 0 \implies \omega_n(x) \longrightarrow 0.$$

Consideriamo una nuova quantità di dati, introducendo il dato "pendenza".

$$\begin{array}{cccc} x_0 & x_1 & \cdots & x_n \\ y_0 & y_1 & \cdots & y_n \\ y_0' & y_1' & \cdots & y_n' \end{array}$$

$p(x) \in \mathcal{P}_{2n+1}$ tale che $p(x_i) = y_i$, $p'(x_i) = y_i'$, per ogni $i = 0, \ldots, n$.

Il polinomio ha $2n+2$ gradi di libertà, quindi esiste un unico polinomio $p(x) \in \mathcal{P}_{2n+1}$ in cui imponiamo $2n+2$ vincoli.

Una prima formula possibile di soluzione è la seguente:

$$p(x) = \sum_{i=0}^{n} \left(A_i(x)y_i + B_i(x)y_i'\right). \tag{7.3}$$

Dove, per ogni $i = 0, \ldots, n$:

$$A_i = [1 - 2(x-x_i)l_i'(x_i)] \cdot l_i^2(x),$$
$$B_i = (x-x_i)l_i^2(x),$$
$$l_i(x) = \prod_{\substack{k=0 \\ k\neq i}} \frac{x-x_k}{x_i-x_k}.$$

Definizione 7.19. Un polinomio $p \in \mathcal{P}_{2n+1}$ della forma di 7.3 si dice *polinomio interpolatore secondo Hermite.*

Proposizione 7.20. *Il polinomio $p \in \mathcal{P}_{2n+1}$ costruito con 7.3 interpola i punti dati:*

$$\begin{array}{cccc} x_0 & x_1 & \cdots & x_n \\ y_0 & y_1 & \cdots & y_n \\ y_0' & y_1' & \cdots & y_n' \end{array}$$

Ovvero $p(x) \in \mathcal{P}_{2n+1}$, $p(x_i) = y_i$, $p'(x_i) = y_i'$, per ogni $i = 0, \ldots, n$.

Dimostrazione.

– $p(x) \in \mathcal{P}_{2n+1}$:

A_i ha grado $2n+1$, B_i ha grado $2n+1$, ne segue che p è combinazione lineare di due polinomi di grado $2n+1$.

– $p(x_i) = y_i, \qquad \forall i = 0, \dots, n$:

$$p(x_k) = \sum_{i=0}^{n} \left(A_i(x_k)y_i + B_i(x_k)y_i'\right) \stackrel{?}{=} y_k,$$

$$B_i(x_k) = (x_k - x_i)l_i^2(x_k) = \begin{cases} 0 & i = k, \\ \\ 0 & i \neq k. \end{cases}$$

$$A_i(x_k) = [1 - 2(x_k - x_i) \cdot l_i'(x_k)] \cdot l_i^2(x_k) = \begin{cases} 1 \cdot l_i^2(x_i) = 1 & i = k, \\ \\ 0 & i \neq k. \end{cases}$$

$$\longrightarrow \; p(x_k) = A_k(x_k)y_k = y_k.$$

– $p'(x_k) = y_k'$:

$$p'(x) = \sum_{i=0}^{n} A_i'(x)y_i + B_i'(x)y_i'.$$

$$A_i'(x) = -2l_i'(x_i)l_i^2(x_i) + [1 - 2(x - x_i)l_i'(x_i)] \cdot 2l_i(x)l_i'(x).$$

$$B_i'(x) = l_i^2(x) + (x - x_i)2l_i(x)l_i'(x).$$

$p'(x) = ?$

- $A_i'(x_k) = -2l_i'(x_i)l_i^2(x_k) + [1 - 2(x_k - x_i)l_i'(x_i)] \cdot 2l_i(x_k)l_i'(x_k).$

$$A_i'(x_k) = \begin{cases} 0 & \text{se } i = k, \\ \\ 0 & \text{se } i \neq k. \end{cases}$$

- $B_i'(x_k) = l_i^2(x_k) + (x_k - x_i)2l_i(x_k)l_i'(x_k).$

$$B_i'(x_k) = \begin{cases} 1 & \text{se } i = k, \\ \\ 0 & \text{se } i \neq k. \end{cases}$$

$\longrightarrow \; p'(x_k) = y_k'.$

■

7.5 Polinomio di Hermite e differenze divise.

Ricordiamo come si definiscono le derivate mediante differenze divise:

$$f[x_i, x_i] = f'(x_i), \qquad f[\underbrace{x_i, \ldots, x_i}_{n}] = \frac{f^{(n)}(x_i)}{n!}.$$

- Partiamo da x_0, y_0 e costruiamo $p_0(x) \in \mathcal{P}_0$ tale che $p_0(x_0) = y_0$:

$$\longrightarrow \; p_0(x) = A_0 = y_0 = f[x_0].$$

Vogliamo costruire un polinomio $p_1(x) \in \mathcal{P}_1$ tale che $p_1(x_0) = y_0$ e $p_1'(x_0) = y_0'$ della forma:

$$p_1(x) = p_0(x) + \boxed{\;?\;} = p_0(x) + A_1(x - x_0).$$

$\boxed{\;?\;}$ si deve annullare in x_0 da cui $A_1(x - x_0)$.

$$p_1'(x) = A_1, \quad y_0' = p_1'(x_0) = A_1 = f[x_0, x_0].$$

Vediamo cosa abbiamo fatto:

$$\begin{array}{ll} x_0 & y_0 \equiv f[x_0] \\ & \qquad\qquad f[x_0, x_0] = y_0' \\ x_0 & y_0 \end{array}$$

$$p_1(x) = f[x_0] + f[x_0, x_0](x - x_0).$$

- Dati il primo punto (x_0) ed il relativo passaggio abbiamo quindi ottenuto la pendenza, ora aggiungiamo il dato successivo x_1 ed il relativo passaggio y_1, abbiamo la seguente situazione:

$$\begin{array}{lll} x_0 & y_0 & y_0' \\ x_1 & y_1 & \end{array}$$

e costruiamo quindi p_2 (avendo tre vincoli sarà di grado ≤ 2), $p_2(x) \in \mathcal{P}_2$ tale che $p_2(x_0) = y_0$, $p_2'(x_0) = y_0'$ e $p_2(x_1) = y_1$ della seguente forma:

$$p_2(x) = p_1(x) + \boxed{\;?\;} = p_1(x) + A_2(x - x_0)^2.$$

$\boxed{\;?\;}$ deve annullarsi in x_0 e in $p'(x_0)$ da cui otteniamo $A_2(x - x_0)^2$.

$$y_1 = p_2(x_1) = p_1(x_1) + A_2(x - x_0)^2.$$

$$\begin{aligned} A_2 &= \frac{y_1-y_0-f[x_0,x_0](x_1-x_0)}{(x_1-x_0)^2} \\ &= \frac{\frac{y_1-y_0}{(x_1-x_0)}-\frac{f[x_0,x_0](x_1-x_0)}{(x_1-x_0)}}{x_1-x_0} \\ &= \frac{f[x_0,x_1]-f[x_0,x_0]}{(x_1-x_0)} = f[x_0,x_0,x_1]. \end{aligned}$$

Abbiamo raggiunto la seguente situazione:

$$\begin{array}{llll} x_0 & y_0 \equiv f[x_0] & & \\ & & f[x_0,x_0] & \\ x_0 & y_0 & & f[x_0,x_0,x_1] \\ & & f[x_0,x_1] & \\ x_1 & y_1 & & \end{array}$$

$$p_2(x) = f[x_0] + f[x_0,x_0](x-x_0) + f[x_0,x_0,x_1](x-x_0)^2.$$

- A questo punto cerchiamo la pendenza nel punto y_1 e calcoliamo il polinomio p_3 con quattro vincoli.

$$\begin{array}{lll} x_0 & y_0 & y_0' \\ x_1 & y_1 & y_1' \end{array}$$

Vogliamo ottenere $p_3 \in \mathcal{P}_3$ tale che:

$$p_3(x_i) = y_i,\ p_3'(x_i) = y_i', \qquad i = 0, 1.$$

$$\begin{aligned} p_3(x) &= p_2(x) + A_3(x-x_0)^2(x-x_1). \\ p_3'(x) &= p_2'(x) + 2\cdot A_3(x-x_0)(x-x_1) + A_3(x-x_0)^2. \\ y_1' &= p_3'(x_1) = p_2'(x) + A_3(x_1-x_0)^2. \end{aligned}$$

$$p_2'(x) = f[x_0,x_0] + 2\cdot f[x_0,x_0,x_1](x-x_0).$$

$$\begin{aligned} A_3 &= \frac{f[x_1,x_1]-f[x_0,x_0]-2f[x_0,x_0,x_1](x_1-x_0)}{(x_1-x_0)^2} \\ &= \frac{\frac{f[x_1,x_1]-f[x_0,x_1]}{(x_1-x_0)}+\frac{f[x_0,x_1]-f[x_0,x_0]}{(x_1-x_0)}-\frac{2\cdot f[x_0,x_0x_1](x_1-x_0)}{(x_1-x_0)}}{x_1-x_0} \\ &= \frac{f[x_0,x_1,x_1]+f[x_0,x_0,x_1]-2f[x_0,x_0,x_1]}{(x_1-x_0)} \\ &= f[x_0,x_0,x_1,x_1]. \end{aligned}$$

Abbiamo raggiunto la seguente situazione:

$$
\begin{array}{llllll}
x_0 & y_0 \equiv f[x_0] & & & & \\
 & & f[x_0,x_0] & & & \\
x_0 & y_0 & & f[x_0,x_0,x_1] & & \\
 & & f[x_0,x_1] & & f[x_0,x_0,x_1,x_1] & \\
x_1 & y_1 & & f[x_0,x_1,x_1] & & \\
 & & f[x_1,x_1] & & & \\
x_1 & y_1 & & & &
\end{array}
$$

$$
\begin{aligned}
p_3(x) &= p_2(x) + f[x_0,x_0,x_1,x_1](x-x_0)^2(x-x_1) \\
&= f[x_0] + f[x_0,x_0](x-x_0) + f[x_0,x_0,x_1](x-x_0)^2 + \\
&\quad + f[x_0,x_0,x_1,x_1](x-x_0)^2(x-x_1).
\end{aligned}
$$

Osservazione 7.21. Nella tabellina si ripetono x_i e y_i tante volte quanti sono i vincoli assegnati a quei punti, nel nostro esempio abbiamo due vincoli, il passaggio e la pendenza.

$$
\begin{array}{llllll}
x_0 & y_0 & & & & \\
 & & f[x_0,x_0] & & & \\
x_0 & y_0 & & f[x_0,x_0,x_1] & & \\
 & & f[x_0,x_1] & & f[x_0,x_0,x_1,x_1] & \\
x_1 & y_1 & & f[x_0,x_1,x_1] & & \ddots \\
 & & f[x_1,x_1] & & & \\
x_1 & y_1 & & & & f[x_0,x_0,x_1,x_1,x_2,\ldots,x_n,x_n] \\
 & & & & & \\
x_2 & y_2 & & & & \\
 & & & & & \\
\vdots & & & f[x_{n-1},x_n,x_n] & & \\
 & & f[x_n,x_n] & & & \\
x_n & y_n & & & &
\end{array}
$$

Il termine finale ottenuto $f[x_0,x_0,x_1,x_1,x_2,\ldots,x_n,x_n]$ ha $2n$ numeri ed è quindi un polinomio di ordine $2n-1$.

$$
\begin{aligned}
p(x) &= f[x_0] + f[x_0,x_0] + f[x_0,x_0,x_1](x-x_0)^2 + \\
&\quad + f[x_0,x_0,x_1,x_1](x-x_0)^2(x-x_1) + \\
&\quad \vdots \\
&\quad + f[x_0,x_0,\ldots,x_n,x_n](x-x_0)^2(x-x_1)^2\cdots(x-x_{n-1})^2(x-x_n).
\end{aligned}
$$

Nota Bene. L'ultimo termine $(x-x_n)$ non è elevato al quadrato.

Osservazione 7.22. Non è necessario sviluppare sempre la tabella completa, solitamente lo faremo con i dati vicino al punto che ci interesserà.

Proposizione 7.23. *Siano $x_0, x_1, \ldots, x_n$ punti distinti in $[a, b]$, siano $\alpha_0, \alpha_1, \ldots, \alpha_n$ numeri naturali, e sia:*

$$m = n + \sum_{i=0}^{n} \alpha_i.$$

Allora, assegnati i numeri reali $y_i^{(j)}$ con $i = 0, \ldots, n$ e $j = 0, \ldots, \alpha_i$; esiste un unico polinomio $p(x) \in \mathcal{P}_m$ tale che:

$$p^{(j)}(x_i) = y_i^{(j)}, \quad i = 0, \ldots, n; \ j = 0, \ldots, \alpha_i.$$

Esempio. Se per x_0 abbiamo tre vincoli: y_0, y_0', y_0'', dobbiamo ripetere tre volte la coppia x_0 y_0 nella tabella:

$$\begin{array}{ll} x_0 & y_0 \\ & \\ & f[x_0, x_0] = y_0' \\ x_0 & y_0 \\ & f[x_0, x_0] = y_0' \\ x_0 & y_0 \end{array} \qquad f[x_0, x_0, x_0] = \frac{f''(x_0)}{2!} = y_0''$$

$$p(x) = f[x_0] + \underbrace{f[x_0, x_0]}_{=f'(x_0)}(x - x_0) + \underbrace{f[x_0, x_0, x_0]}_{=\frac{f''(x)}{2}}(x - x_0)^2.$$

p è il polinomio di Taylor.

$$\alpha_0 + 1 \begin{cases} x_0 & y_0 \\ & \\ & f[x_0, x_0] = y_0' \\ x_0 & y_0 \\ & \\ & \\ \vdots & \\ x_0 & y_0 \end{cases} \qquad f[x_0, x_0, x_0] = \frac{y_0''}{2} \qquad f[x_0, x_0, x_0, x_0] = \frac{y_0'''}{3!}$$

$$\alpha_1 + 1 \begin{cases} x_1 & y_1 \\ x_1 & y_1 \\ \vdots & \\ x_1 & y_1 \end{cases}$$

$$\alpha_2 + 1 \begin{cases} x_2 & y_2 \\ x_2 & y_2 \\ \vdots & \\ x_2 & y_2 \end{cases}$$

$$\longrightarrow \sum_{i=0}^{n} \alpha_i + 1 = t\text{: il polinomio interpolatore ha grado } t-1.$$

Esempio. Consideriamo il caso in cui non si abbiano dei dati completi:

$$\begin{array}{ccc} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ y_0' & & y_2' \\ y_0'' & & y_2'' \\ & & y_2''' \end{array}$$

$$\begin{array}{cccccccc}
x_0 & y_0 & & & & & & \\
 & & f[x_0,x_0] & & & & & \\
x_0 & y_0 & & f[x_0,x_0,x_0] & & & & \\
 & & f[x_0,x_0] & & f[x_0,x_0,x_0,x_1] & & & \\
x_0 & y_0 & & f[x_0,x_0,x_1] & & \ddots & & \\
 & & f[x_0,x_1] & & f[x_0,x_0,x_1,x_2] & & \ddots & \\
x_1 & y_1 & & f[x_0,x_1,x_2] & & & & (*) \\
 & & f[x_1,x_2] & & f[x_0,x_1,x_2,x_2] & & & \\
x_2 & x_2 & & f[x_1,x_2,x_2] & & & & \\
 & & f[x_2,x_2] & & f[x_1,x_2,x_2,x_2] & & & \\
x_2 & x_2 & & f[x_2,x_2,x_2] & & & & \\
 & & f[x_2,x_2] & & & & & \\
x_2 & x_2 & & & & & &
\end{array}$$

$$(*) = f[x_0,x_0,x_0,x_1,x_2,x_2,x_2,x_2].$$

Teorema 7.24. *Siano $x_0, x_1, \ldots, x_n$ punti distinti in $[a,b]$, siano $\alpha_0, \alpha_1, \ldots, \alpha_n$ numeri naturali, e sia:*

$$m = n + \sum_{i=0}^{n} \alpha_i.$$

Sia $f(x) \in \mathcal{C}^{m+1}([a,b])$ e $p(x) \in \mathcal{P}_m$ il polinomio interpolatore generalizzante la funzione tale che:

$$p^{(j)}(x_i) = f^{(j)}(x_i), \quad i = 0, \ldots, n;\ j = 0, \ldots, \alpha_i.$$

Allora per ogni $x \in [a,b]$ esiste un punto $\xi_x \in I_x$ tale che:

$$e(x) := f(x) - p(x) = \frac{f^{(m+1)}(\xi_x)}{(m+1)!} \underbrace{(x-x_0)^{\alpha_0+1} \cdot (x-x_1)^{\alpha_1+1} \cdots (x-x_n)^{\alpha_n+1}}_{\omega_m(x)}.$$

$$I_x := [\min(x, x_0, x_1, \ldots, x_n), \max(x, x_0, x_1, \ldots, x_n)].$$

Osservazione 7.25. Se per ogni $i = 0, \ldots, n,\ \alpha_i = 0,$ allora vuol dire che per ogni punto abbiamo solo il passaggio.

Dimostrazione. Sia $q \in \mathcal{P}_m$ così definito:

$$q(t) = p(t) + \underbrace{\frac{f(x) - p(x)}{\omega_m(x)} \omega_m(t)}_{\text{costante rispetto a } t}.$$

$$q(x) = \cancel{p(x)} + \frac{f(x) - \cancel{p(x)}}{\cancel{\omega_m(x)}} \cdot \cancel{\omega_m(x)} = f(x) \longrightarrow q(t) \text{ interpola } f \text{ in } x.$$

$$q(x_i) = p(x_i) + \cancel{\frac{f(x) - p(x)}{\omega_m(x)}}^{\to 0} \cdot \omega_m(x_i) = f(x_i),$$

$$\longrightarrow q(t) \text{ interpola } f \text{ nei punti } x_i \ \forall i = 0, \ldots, n.$$

$$q^{(j)}(x_i) = p^{(j)}(x_i) + \cancel{\frac{f(x) - p(x)}{\omega_m(x)}}^{\to 0} \cdot \omega_m^{(j)}(x_i) = p^{(j)}(x_i) = f^{(j)}(x_i),$$

$$\forall i = 0, \ldots, n;\ \forall j = 0, \ldots, \alpha_i.$$

$q(t)$ interpola la f nel punto x e nei punti x_i con relative condizioni.

$p(t) \in \mathcal{P}_m$,

$$\text{il grado di } \omega_m = \sum_{i=0}^{n} (\alpha_i + 1) = \sum_{i=0}^{n} \alpha_i + n + 1 = m + 1.$$

Definiamo ora una funzione G tale che:

$$G(t) = f(t) - q(t),$$

quanti zeri ha la funzione G? Intanto sappiamo che $G \in \mathcal{C}^{m+1}$ poiché $f \in \mathcal{C}^{m+1}$ e $q \in \mathcal{C}^{\infty}$.

$$t = x \longrightarrow G(x) = f(x) - q(x) = 0.$$
$$t = x_i \longrightarrow G(x_i) = f(x_i) - q(x_i) = 0, \quad \forall i = 0, \ldots, n.$$

Inoltre fissato i:

$$G^j(x_i) = f^j(x_i) - q^j(x_i) = 0, \quad \forall j = 0, \ldots, n.$$

$i = 0 \longrightarrow x_0$ è radice di molteplicità $\alpha_0 \neq 1$ almeno.
$\forall i,\ x_i$ è radice di molteplicità $\alpha_i \neq 1$ almeno.

G ha almeno $(\sum_{i=0}^{n} \alpha_i + 1) + 1$ zeri, cioé ha almeno $m + 2$ zeri.

G' ha almeno $m + 1$ zeri per il teorema di Rolle: presi due zeri consecutivi la derivata prima si annulla in un punto in mezzo.
$\vdots$
$G^{(m+1)}$ ha almeno uno uno zero.

$$\Longrightarrow \exists\, \xi_x :\ G(\xi_x) = 0.$$

$$\begin{aligned} G^{(m+1)}(t) &= f^{(m+1)}(t) - q^{(m+1)}(t) = f^{(m+1)}(t) - \\ &\quad - \cancelto{0}{p^{(m+1)}(t)} + \tfrac{f(x)-p(x)}{\omega_m(x)} \cdot \underbrace{\omega_m^{(m+1)}(t)}_{(n+1)!} \\ &= f^{(m+1)}(t) - \tfrac{f(x)-p(x)}{\omega_m(x)} \cdot (n+1)! \end{aligned}$$

$$\longrightarrow 0 = G^{(m+1)}(\xi_x) = f^{(m+1)}(\xi_x) + \frac{f(x) - p(x)}{\omega_m(x)} \cdot (n+1)!$$

$$\longrightarrow e(x) = f(x) - p(x) = \frac{f^{(m+1)}(\xi_x)\omega_m(x)}{(n+1)!}.$$

Caso: $x_i\ y_i\ y_i' \quad i = 0, \ldots, n.$
$p(x)$ polinomio interpolatore.

$$e(x) = f(x) - p(x) = \frac{f^{(2n+2)}(\xi_x)}{(2n+2)!}(x - x_0)^2 \cdots (x - x_n)^2.$$

■

Capitolo 8

Interpolazione a tratti.

Sia $[a, b]$ un intervallo *chiuso* e *limitato*, e Δ la decomposizione così definita:

$$\Delta = \{a = x_0 < x_1 < \cdots < x_i < \cdots < x_n = b\}.$$

Vogliamo costruire su ciascun tratto $[x_{i-1}, x_i]$ della decomposizione Δ per ogni $i = 1, \ldots n$, una funzione lineare che interpola i dati y_{i-1} e y_i.

$$\begin{aligned} S_1^{(i)} &= y_{i-1} + f[x_{i-1}, x_i](x - x_{i-1}) \\ &= \tfrac{(x-x_{i-1})y_i + (x_i - x)y_{i-1}}{x_i - x_{i-1}}, \qquad x_{i-1} < x < x_i. \end{aligned}$$

$$S_1(x) = \begin{cases} S_1^{(1)}(x) & x_0 \le x \le x_1 \quad S_1^{(1)}(x) = y_0 + f[x_0, x_1](x - x_0) \\ S_1^{(2)}(x) & x_1 \le x \le x_2 \quad S_1^{(2)}(x) = y_1 + f[x_1, x_2](x - x_1) \\ S_1^{(3)}(x) & x_2 \le x \le x_3 \quad S_1^{(3)}(x) = y_2 + f[x_2, x_3](x - x_2) \\ \vdots & \qquad\qquad\quad \vdots \\ S_1^{(n)}(x) & x_{n-1} \le x \le x_n \quad S_1^{(n)}(x) = y_2 + f[x_{n-1}, x_n](x - x_n). \end{cases}$$

$S_1(x) \in \mathcal{C}^0([a, b])$.

Osservazione 8.1. $S_1^{(i)}(x) \in \mathcal{C}^\infty([x_{i-1}, x_i])$. Ma la rispettiva funzione $S_1(x)$ è solo $\mathcal{C}^0$, ovvero perdiamo di regolarità nei *nodi* ($\notin \mathcal{C}^1$).

Prendendo un x interno su un generico tratto, $x \in [x_{i-1}, x_i]$ che errorre commettiamo?

$$e(x) = f(x) - S_1^{(j)}(x).$$

Se $f(x) \in \mathcal{C}^2([a, b]) \longrightarrow f \in \mathcal{C}^2([[x_{i-1}, x_i])$.

$$\longrightarrow e(x) = f(x) - S_1^{(i)}(x) = \frac{f''(\xi_x)}{2}(x - x_{i-1})(x - x_i).$$

$$
\begin{aligned}
|e(x)| &= |f(x) - S_1^{(i)}(x)| \\
&= \tfrac{|f''(\xi_x)|}{2} \cdot |(x - x_{i-1})(x - x_i)| \\
&\leq \frac{1}{2} \max_{t \in [x_{i-1}, x_i]} |f''(t)| \cdot |(x - x_{i-1})(x - x_i)| \\
&\leq \frac{1}{2} \max_{t \in [x_{i-1}, x_i]} |f''(t)| \cdot \left| \left(\frac{x_{i-1} - x_i}{2} - x_{i-1} \right) \left(\frac{x_{i-1}}{2} - x_i \right) \right| \\
&= \max_{t \in [x_{i-1}, x_i]} |f''(t)| \frac{1}{8} h_i^2.
\end{aligned}
$$

$$
\longrightarrow \left| f(x) - S_1^{(i)}(x) \right| \leq \max_{t \in [x_{i-1}, x_i]} |f''(t)| \, \frac{1}{8} h^2.
$$

$h = \max\limits_{1 \leq i \leq n} h_i$ è la norma della decomposizione. Ovvero: $\max\limits_{1 \leq i \leq n} (x_{i-1} - x_i)$

$$
\forall x \in [a, b], \ |f(x) - S_1(x)| \leq \max_{t \in [a,b]} |f''(t)| \frac{1}{8} h^2 = \|f''(x)\|_\infty \frac{1}{8} h^2 \underset{h \to 0}{\longrightarrow} 0.
$$

La logica di questa procedura è quella di prendere un intervallo, tagliuzzarlo e costruire il polinomio di primo grado. Aumentare i punti non implica aumentare il grado del polinomio ma costruiamo in questo caso altri polinomi di primo grado.

Il fatto che l'errore tende a zero per h che tende a zero ci dice che "raffinando" la partizione i due grafici vanno a coincidere. In questo caso si parla di *convergenza*.

Questo è un ottimo risultato, l'unico inconveniente è il numero dei polinomi di primo grado da calcolare, siamo tuttavia nell'ipotesi $f \in \mathcal{C}^2$.

Vale un risultato analogo per $f \in \mathcal{C}^0$? L'oggetto su cui discutere è $\|f''(x)\|_\infty$ (non l'abbiamo).

Definizione 8.2. (Modulo di continuità) Sia $f(x) \in \mathcal{C}^0([a, b])$, si dice *modulo di continuità* della funzione f di parametro δ una funzione così definita:

$$
\omega(f, \delta) := \max |f(x_1) - f(x_2)|, \quad \forall x_1, x_2 \in [a, b] \colon |x_1 - x_2| < \delta.
$$

Osservazione 8.3.

$$
\lim_{\delta \to 0} \omega(f, \delta) = 0. \qquad 1 = \frac{(x - x_{i-1}) + (x_i - x)}{h_i}.
$$

Tesi:

$$|f(x) - S_1(x)| \leq \omega(f,h) \underset{h\to 0}{\longrightarrow} 0.$$

$$\begin{aligned} f(x) - S_1^{(i)}(x) &= 1 \cdot f(x) - \tfrac{(x-x_{i-1})f(x_i)+(x_i-x)f(x_{i-1})}{h_i} \\ &= \tfrac{f(x)(x-x_{i-1})+(x_i-x)}{h_i} - \tfrac{(x-x_{i-1})f(x_i)+(x_i-x)f(x_{i-1})}{h_i} \\ &= \tfrac{(x-x_{i-1})[f(x)-f(x_i)]-(x_i-x)[f(x)-f(x_{i-1})]}{h_i} \quad . \end{aligned}$$

$$\begin{aligned} \left|f(x) - S_1^{(i)}(x)\right| &\leq \tfrac{(x-x_{i-1})|f(x)-f(x_i)|-(x_i-x)|f(x)-f(x_{i-1})|}{h_i} \\ &\leq \tfrac{(x-x_{i-1})\max\{|f(x)-f(x_i)|,|f(x)-f(x_{i-1})|\}+(x_i-x)\max\{|f(x)-f(x_i)|,|f(x)-f(x_{i-1})|\}}{h_i} \\ &= \tfrac{(x-x_{i-1})+(x_i-x)}{h_i}{}^{\nearrow 1} \max\{|f(x) - f(x_i)|, |f(x) - f(x_{i-1})|\} \quad . \end{aligned}$$

$$\left|f(x) - S_1^{(i)}(x)\right| \leq \max_{x_i \leq x \leq x_{i-1}} \{|f(x) - f(x_i)|,\ |f(x) - f(x_{i-1})|\} \leq \omega(f, h_i).$$

$$\left|f(x) - S_1^{(i)}(x)\right| \leq \omega(f, h_i).$$

$$\longrightarrow \ \forall x \in [a,b] \quad |e(x)| = |f(x) - S_1(x)| \leq \omega(f,h) \underset{h\to 0}{\longrightarrow} 0.$$

$$\text{con } h = \max_{1\leq i\leq n} h_i.$$

Quindi abbiamo ottenuto un risultato analogo per le funzioni continue.

Osservazione 8.4. Queste tecniche hanno senso se usiamo tabelle con dati precisi, se i dati fossero affetti da errori sarebbe meglio usare altre tecniche (minimi quadrati).

Lo scopo è di semplificare l'oggetto funzione con cui lavoriamo. Abbiamo scoperto che possiamo farlo con l'unione dei polinomi di primo grado calcolati sulla partizione dell'intervallo. Il grafico dell'unione dei polinomi si sovrappone ad f al tendere a zero della finezza della partizione.

Esempio. Sia $[a,b]$ il nostro intervallo, Δ la seguente decomposizione:

$$\Delta = \{a = x_0 < x_1 < \cdots < x_n = b\}$$

$$\begin{array}{ccccccc} y_0 & y_1 & \cdots & y_i & \cdots & y_n \\ y_0' & y_1' & \cdots & y_i' & \cdots & y_n' \end{array}$$

Posto $[x_{i-1}, x_i] \subset \Delta$ per ogni $i = 1, \ldots, n$, vogliamo calcolare $p(x) \in \mathcal{P}_3$ con $x_{i-1} \leq x \leq x_i$ tale che:

$$p(x_{i-1}) = y_{i-1}, \quad p(x_i) = y_i,$$

$$p'(x_{i-1}) = y'_{i-1}, \quad p'(x_i) = y'_i.$$

$$p(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^2(x - x_i).$$

Usiamo le quattro condizioni.

$$p'(x) = b_i + 2c_i(x - x_{i-1}) + 2d_i(x - x_{i-1})(x - x_i) + d_i(x - x_{i-1})^2.$$

1. $y_{i-1} = p(x_{i-1}) = a_i$.
2. $y'_{i-1} = p'(x_{i-1}) = b_i$.
3. $y_i = a_i + b_i(x_i - x_{i-1}) + c_i(x_i - x_{i-1})^2$:

$$\longrightarrow \; c_i = \frac{y'_i - b_i(x_i - x_{i-1})}{(x_i - x_{i-1})^2} = \frac{y_i - a_1 - b_i h_i}{h_i^2}.$$

4. $y'_1 = p(x_i) = b_i + 2c_i h_i + d_i h_1^2$:

$$\longrightarrow \; d_i = \frac{y'_i - b_i - 2c_i h_i}{h_i^2}.$$

$$P(x) = \begin{cases} p_3^{(1)}(x) = \ldots & x_0 \leq x \leq x_1, \\ p_3^{(2)}(x) = \ldots & x_1 \leq x \leq x_2, \\ \vdots & \\ p_3^{(n)}(x) = \ldots & x_{n-1} \leq x \leq x_n. \end{cases}$$

$p_3^{(i)}(x) \in \mathcal{C}^\infty([x_{i-1}, x_i]), \quad \forall i = 1, \ldots, n.$

$$P(x) \in \mathcal{C}^1([a, b]).$$

Sono i polinomi di Hermite generalizzati.

8.1 Spline cubica.

Vorremmo ottenere una regolarità maggiore di P dell'esempio precedente.

Definizione 8.5. Sia $[a, b]$ un intervallo, Δ la seguente decomposizione:

$$\Delta = \{a = x_0 < x_1 < \cdots < x_n = b\}$$

$$\begin{array}{ccccccc} y_0 & y_1 & \cdots & y_i & \cdots & y_n \\ y'_0 & y'_1 & \cdots & y'_i & \cdots & y'_n \end{array}$$

Posto $[x_{i-1}, x_i] \subset \Delta$ per ogni $i = 1, \ldots, n$, $h_i = x_i - x_{i-1}$,

$$h = \max_{1 \le i \le n} h_i \text{ la norma della decomposizione;}$$

si dice *spline cubica interpolante* relativa alla decomposizione Δ la funzione $S_{3,\Delta}(x)$ tale che:

1. $S_{3,\Delta}(x)$ è una funzione polinomiale definita a tratti $[x_{i-1}, x_i]$, $i = 1, \ldots, n$, e su ciascun tratto è di terzo grado.
2. $S_{3,\Delta}(x) \in \mathcal{C}^2([a, b])$.
3. $S_{3,\Delta}(x_i) = y_i, \quad i = 1, \ldots, n.$

$$S_{3,\Delta}(x) = \begin{cases} S^{(1)}_{3,\Delta}(x) = a^{(1)}_0 + a^{(1)}_1(x - x_0) + a^{(1)}_2(x - x_0)^2 + a^{(1)}_3(x - x_0)^3 \\ S^{(2)}_{3,\Delta}(x) = a^{(2)}_0 + a^{(2)}_1(x - x_1) + a^{(2)}_2(x - x_1)^2 + a^{(2)}_3(x - x_1)^3 \\ \vdots \\ S^{(n)}_{3,\Delta}(x) = a^{(n)}_0 + a^{(n)}_1(x - x_{n-1}) + a^{(n)}_2(x - x_{n-1})^2 + a^{(n)}_3(x - x_{n-1})^3. \end{cases}$$

Nota Bene. $S^{(1)}_{3,\Delta}(x)$ vale per gli $x \in [x_0, x_1]$, $S^{(2)}_{3,\Delta}(x)$ vale per gli $x \in [x_1, x_2]$, etc.

Il punto 2 della definizione allarga la *regolarità* e, chiedendo meno vincoli, non richiede di conoscere il valore delle derivate prime.

Senza il punto 3 abbiamo la definizione di *spline* (non interpolante).

Gradi di libertà: $4n$.
Vincoli:

$3(n-1)$: imporre che i due polinomi contigui si raccordino in modo continuo.
$+n-1$: imporre a quale quota avviene il raccordo: $S_{3,\Delta}(x_i) = y_i$.
$+2$: posto $S_{3,\Delta}(a) = y_0$ e $S_{3,\Delta}(b) = y_n$.
$= 4n - 2$.

Mancano due vincoli, quindi abbiamo ∞^2 spline cubiche interpolanti.

Aggiungiamo quindi i due vincoli noi: otteniamo spline differenti a seconda dei vincoli aggiunti. Ad esempio aggiungendo le derivate seconde dei due estremi otteniamo la *spline cubica naturale*.

Definizione 8.6. (Spline lineare)
$S_{1,\Delta}(x)$:

1. $S_{1,\Delta}(x)$ è una funzione polinomiale definita a tratti $[x_{i-1}, x_i]$, $i = 1, \dots, n$, e su ciascun tratto è di primo grado.
2. $S_{1,\Delta}(x) \in \mathcal{C}^0([a,b])$.
3. $S_{1,\Delta}(x_i) = y_i, \quad i = 1, \dots, n.$

Al tendere di h a zero, la spline lineare tende ad f (convergenza), sostanzialmente questa spline unisce i punti con delle rette.

Definizione 8.7. (Spline di ordine k)
$S_{k,\Delta}(x)$:

1. $S_{k,\Delta}(x)$ è una funzione polinomiale definita a tratti $[x_{i-1}, x_i]$, $i = 1, \dots, n$, e su ciascun tratto è di grado k.
2. $S_{k,\Delta}(x) \in \mathcal{C}^{k-1}([a,b])$.
3. $S_{k,\Delta}(x_i) = y_i, \quad i = 1, \dots, n.$

$$S_{k,\Delta}(x) = \begin{cases} S^{(1)}_{k,\Delta}(x) = a_0^{(1)} + a_1^{(1)}(x - x_0) + \cdots + a_k^{(1)}(x - x_0)^k \\ \vdots \\ S^{(i)}_{k,\Delta}(x) = a_0^{(i)} + a_1^{(i)}(x - x_{i-1}) + \cdots + a_k^{(i)}(x - x_{i-1})^k \\ \vdots \\ S^{(n)}_{k,\Delta}(x) = a_0^{(n)} + a_1^{(n)}(x - x_{n-1}) + \cdots + a_k^{(n)}(x - x_{n-1})^k. \end{cases}$$

Nota Bene. $S^{(1)}_{k,\Delta}(x)$ vale per gli $x \in [x_0, x_1]$, $S^{(2)}_{k,\Delta}(x)$ vale per gli $x \in [x_1, x_2]$, etc.

Gradi di libertà: $(k+1)n$.
Vincoli:

$$\begin{array}{rl} k(n-1) & + \\ n-1 & + \\ n+1 & = \\ \hline (k+1)n - (k-1) & \end{array}$$

Mancano $k-1$ vincoli, quindi abbiamo ∞^{k-1} spline cubiche interpolanti possibili. Quindi avremo bisogno di aggiungere condizioni pari al grado della spline meno uno $(k-1)$.

Momento della spline.

Costruiamo una spline cubica interpolante con un *metodo* che da come vantaggio la possibilità di risolvere un sistema di ordine inferiore a $4n$. Avremo una matrice *tridiagonale*.

- Sostituiamo il problema del calcolo dei coefficienti con il calcolo di parametri che permetteranno di ricostruirli.
- Computazionalmente la spline cubica equivale ad una matrice di n righe e 4 colonne.

I parametri che calcoliamo saranno il *momento della spline*.

Definizione 8.8. (Momento della spline)
Il *momento della spline* di ordine i è definito come segue:

$$M_i := \Big[S_{3,\Delta}(x)\Big]''_{x=x_i} = \Big[S^{(i)}_{3,\Delta}(x)\Big]''_{x=x_i}.$$

In altri termini è una riduzione delle incoglite del sistema lineare.

Com'è, nel tratto $[x_{i-1}, x_i]$, la derivata seconda della spline? E' un polinomio di primo grado. Se conoscessimo i due valori M_i e M_{i-1} avremmo la seguente forma:

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

Integrando $[S^{(i)}_{3,\Delta}(x)]''$ otteniamo:

$$\Big[S^{(i)}_{3,\Delta}(x)\Big]' = \frac{(x - x_{i-1})^2}{2h_i}M_i - \frac{(x_i - x)^2}{2h_i}M_{i-1} + A_1.$$

Integrando ulteriormente:

$$S^{(i)}_{3,\Delta}(x) = \frac{(x - x_{i-1})^3}{6h_i}M_i + \frac{(x_i - x)^3}{6h_i}M_{i-1} + A_1(x - x_{i-1}) + B_i.$$

Come otteniamo A_i e B_i? Imponiamo l'interpolazione.

$$y_{i-1} = S^{(i)}_{3,\Delta}(x_{i-1}) = \frac{(x_i - x_{i-1})^{\cancel{2}}}{6\cancel{h_i}}M_{i-1} + B_i = \frac{h_i^2 M_{i-1}}{6} + B_i.$$

$$\longrightarrow \; B_i = y_{i-1} - \frac{h_i^2 M_{i-1}}{6}.$$

$$y_i = S^{(i)}_{3,\Delta}(x_i) = h_i^3 \frac{M_i}{6h_i} + A_i h_i + B_i.$$

$$A_i = \frac{y_i - \frac{h_i^2 M_i}{6} - B_i}{h_i}.$$

$$A_i = \frac{y_i - y_{i-1} + \frac{h_i^2 M_{i-1}}{6} - \frac{h_i^2 M_i}{6}}{h_i}.$$

$$\longrightarrow \; A_i = \frac{y_i - y_{i-1}}{h_i} + \frac{h_i}{6}\left(M_{i-1} - M_i\right).$$

Problema 1: costruire il sistema che calcoli i momenti M_i.

Problema 2: calcolare i coefficienti $a_k^{(i)} \quad k = 0, \ldots, 4n$ nota questa forma alternativa (i momenti) del polinomio.

Iniziamo a risolvere il problema 2.

- $a_0^{(i)} = S^{(i)}_{3,\Delta}(x_i) = y_i.$
- $a_1^{(i)} = ?$

$$a_1^{(i)} = \left[S^{(i)}_{3,\Delta}(x)\right]'_{x=x_{i-1}} = \left[a_1^{(i)} + 2a_2^{(i)}(x - x_{i-1}) + 3a_3^{(i)}(x - x_{i-1})^2\right]_{x=x_{i-1}}.$$

$$a_1^{(i)} = -\frac{h_i}{2} M_{i-1} + A_i.$$

- $a_2^{(i)} = ?$

$$2a_2^{(i)} = \left[S^{(i)}_{3,\Delta}(x)\right]''_{x=x_{i-1}} = \left[2a_2^{(i)} + 6a_3^{(i)}(x - x_{i-1})\right]_{x=x_{i-1}}$$

$$2a_2^{(i)} = M_{i-1}.$$

$$a_2^{(i)} = \frac{M_{i-1}}{2}.$$

- $a_3^{(i)} = ?$

$$6a_3^{(i)} = \left[S^{(i)}_{3,\Delta}(x)\right]''_{x=x_{i-1}} = \frac{M_i - M_{i-1}}{h_i}.$$

$$a_3^{(i)} = \frac{M_i - M_{i-1}}{6h_i}.$$

I nodi da calcolare sono $n+1$. Dobbiamo risolvere il problema 1, ovvero costruire un sistema che abbia come incognite i momenti $M_i, \quad i = 0, \ldots, n$ (gli $n+1$ nodi). Abbiamo già usato:

- l'interpolazione, ovvero abbiamo usato $\mathcal{C}^0$ perchè il momento sinistro è uguale al momento destro della spline.
- $\mathcal{C}^2$, sempre per l'uguaglianza tra momento sinistro e destro.

Manca da utilizzare la continuità della derivata prima nei nodi interni, avendo $n-1$ vincoli restano sempre due gradi di libertà.

$$\left[S_{3,\Delta}(x)\right]'' = \left[S^{(i)}_{3,\Delta}(x)\right]'' = \frac{(x-x_{i-1})}{h_i}M_i + \frac{(x_i-x)}{h_i}M_{i-1}.$$

$$\left[S^{(i)}_{3,\Delta}(x)\right]' = \frac{(x-x_{i-1})^2}{2h_i}M_i - \frac{(x_i-x)^2}{2h_i}M_{i-1} + A_i.$$

$$\left[S^{(i)}_{3,\Delta}(x)\right]' = \frac{(x-x_{i-1})^2}{2h_i}M_i - \frac{(x_i-x)^2}{2h_i}M_{i-1} + \frac{y_i-y_{i-1}}{h_i} + \frac{h_i}{6}\left(M_{i-1}-M_i\right).$$

$$S_{3,\Delta}(x) = \frac{(x-x_{i-1})^3}{6h_i}M_i + \frac{(x_i-x)^3}{6h_i}M_{i-1} + A_i(x-x_{i-1}) + B_i.$$

Usiamo ora l'unica condizione che manca: la continuità della derivata prima dei nodi interni $x_1, \ldots, x_{n-1}$.

$$\underbrace{\left[S^{(i)}_{3,\Delta}(x)\right]'}_{\substack{x_i \le x \le x_{i+1} \\ h_{i+1} = x_{i+1} - x_i}} = \frac{(x-x_i)^2}{2h_{i+1}}M_{i+1} - \frac{(x_{i+1}-x)^2}{2h_{i+1}}M_i + \frac{y_{i+1}-y_i}{h_{i+1}} + \frac{h_{i+1}}{6}\left(M_i - M_{i+1}\right).$$

Dobbiamo imporre la continuità della derivata prima nel nodo x_i, cioé:

$$\left[S^{(i)}_{3,\Delta}(x)\right]'_{x=x_i} = \left[S^{(i+1)}_{3,\Delta}(x)\right]'_{x=x_i},$$

in altri termini:

$$\lim_{x\to x_i^-}\left[S^{(i)}_{3,\Delta}(x)\right]' = \lim_{x\to x_i^+}\left[S^{(i+1)}_{3,\Delta}(x)\right]'.$$

E' sufficiente calcolare i limiti nel punto:

$$\frac{h_i M_i}{2} + \frac{h_i}{6}M_{i-1} - \frac{h_i}{6}M_i + \frac{y_i - y_{i-1}}{h_i} = -\frac{h_{i+1}M_i}{2} + \frac{h_{i+1}}{6}M_i - \frac{h_{i+1}}{6}M_{i+1} + \frac{y_{i+1}-y_i}{h_{i+1}}.$$

$$\Updownarrow$$

$$\frac{h_i}{6}M_{i-1} + M_i\left[\frac{h_i}{2} - \frac{h_i}{6} + \frac{h_{i+1}}{2} - \frac{h_{i+1}}{6}\right] + \frac{h_{i+1}}{6}M_{i+1} = \frac{y_{i+1}-y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}.$$

$$\Updownarrow$$

$$\frac{h_i}{6}M_{i-1} + \frac{2}{6}M_i(h_i + h_{i+1}) + \frac{h_{i+1}}{6}M_{i+1} = \frac{y_{i+1}-y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}.$$

$$\Updownarrow$$

$$\frac{h_i}{h_i + h_{i+1}}M_{i-1} + 2M_i + \frac{h_{i+1}}{h_i + h_{i+1}}M_{i+1} = \frac{6}{h_i + h_{i+1}}\left(\frac{y_{i+1}-y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}\right).$$

Posti α_i, β_i e d_i tali che:

$$\alpha_i = \frac{h_i}{h_i + h_{i+1}}, \quad \beta_i = \frac{h_{i+1}}{h_i + h_{i+1}}, \quad d_i = \frac{6}{h_i + h_{i+1}}\left(\frac{y_{i+1}-y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}\right),$$

possiamo esprimere il limite di cui sopra come segue:

$$\alpha_i M_{i-1} + 2M_i + \beta_i M_{i+1} = d_i.$$

$i = 1$:
$\alpha_1 M_0 + 2M_1 + \beta_1 M_2 = d_1.$

$i = 2$:
$\alpha_2 M_1 + 2M_2 + \beta_2 M_3 = d_2.$

$\vdots$

$i = n - 1$:
$\alpha_{n-1} M_{n-2} + 2M_{n-1} + \beta_{n-1} M_n = d_{n-1}.$

Risulta quindi un sistema di $n - 1$ equazioni in $n + 1$ incognite, d_i è il vettore dei termini noti, possiamo scrivere il sistema in forma matriciale (A):

$$A = \begin{pmatrix} \alpha_1 & 2 & \beta_1 & 0 & & \cdots & 0 \\ 0 & \alpha_2 & 2 & \beta_2 & 0 & \cdots & \vdots \\ \vdots & & & & & & 0 \\ 0 & \cdots & & 0 & \alpha_{n-1} & 2 & \beta_{n-1} \end{pmatrix}, \quad d_i = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \end{pmatrix}.$$

Occorre introdurre delle condizioni ulteriori poiché abbiamo un numero di incognite maggiore del numero di equazioni.

8.1.1 Spline cubica naturale.

Una soluzione è la spline cubica naturale: $M_0 = M_n = 0$.

$$\longrightarrow \quad \Big[S_{3,\Delta}(x)\Big]''_{x=x_0} = \Big[S_{3,\Delta}(x)\Big]''_{x=x_n} = 0.$$

Ora abbiamo due incognite in meno, quelle date dai momenti M_0 e M_n. Abbiamo ora un sistema in cui spariscono α_1 e β_{n-1}.

$$\begin{cases} 2M_1 + \beta_1 M_2 = d_1 \\ \alpha_2 M_1 + 2M_2 + \beta_2 M_3 = d_2 \\ \vdots \\ \alpha_{n-1} M_{n-2} + 2M_{n-1} = d_{n-1} \end{cases}$$

$$A = \begin{bmatrix} 2 & \beta_1 & 0 & \cdots & 0 \\ \alpha_2 & 2 & \beta_2 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \alpha_{n-2} & 2 & \beta_{n-2} \\ 0 & \cdots & 0 & \alpha_{n-1} & 2 \end{bmatrix}.$$

$A \in \mathbb{R}^{(n-1)\times(n-1)}$ matrice tridiagonale irriducibile.

$\alpha_i + \beta_i = 1, \quad i = 2, \dots n-1.$

$0 < \beta_1 \leq 1.$

$0 < \alpha_{n-1} \leq 1.$

Osservazione 8.9. La matrice A è una matrice diagonal dominante:

$$\begin{cases} \alpha_i + \beta_i = 1, \\ \beta_1 < 1 < 2, \\ \alpha_{n-1} < 1 < 2. \end{cases}$$

Abbiamo quindi il seguente sistema lineare:

$$Ax = d, \quad d = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-1} \end{bmatrix}, \quad x = \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-1} \end{bmatrix}.$$

La proprietà diagonal dominante è utile perchè permette di affermare che la matrice non ha autovalore nullo. Ovvero è non singolare.

8.1.2 Spline cubica a valori assegnati.

Posti $M_0 = \overline{M}_0$ e $M_n = \overline{M}_n$ il nuovo vettore dei termini noti d risulta:

$$d = \begin{bmatrix} d_1 - \alpha_1 \overline{M}_0 \\ d_2 \\ \vdots \\ d_{n-2} \\ d_{n-1} - \beta n - 1\overline{M}_n \end{bmatrix}$$

La matrice A invece non cambia.

8.1.3 Spline cubica vincolata.

$$\Big[S_{3,\Delta}(x)\Big]'_{x=x_0} = y'_0 \quad \wedge \quad \Big[S_{3,\Delta}(x)\Big]'_{x=x_n} = y'_n.$$

Sfruttando una logica diversa da prima otteniamo due ulteriori vincoli (equazioni), la nostra matrice A rappresenterà dunque un sistema lineare a $n+1$ equazioni in $n+1$ incognite.

$$\underbrace{\Big[S_{3,\Delta}(x)\Big]'}_{x_0 \le x \le x_1} = \Big[S^{(1)}_{3,\Delta}(x)\Big]'_{x=x_0} =$$

$$= \left[\cancelto{0}{\frac{(x-x_0)^2}{2h_1}} M_1 - \frac{(x_1-x)^2}{2h_1} M_0 + \frac{y_1-y_0}{h_1} + \frac{h_1}{6}(M_0 - M_1) \right]_{x=x_0} = y'_0$$

$$\Updownarrow$$

$$-\frac{h_1}{2}M_0 + \frac{h_1}{6}M_0 - \frac{h_1}{6}M_1 + \frac{y_1-y_0}{h_1} = y'_0$$

$$\Updownarrow$$

$$-M_0\left(\frac{h_1}{2} - \frac{h_1}{6}\right) - \frac{h_1}{6}M_1 + \frac{y_1-y_0}{h_1} = y'_0$$

$$\Updownarrow$$

$$\frac{2}{6}h_1 M_0 + \frac{h_1}{6}M_1 = \frac{y_1-y_0}{h_1} - y'_0$$

$$\Updownarrow$$

$$2M_0 + M_1 = 6y_1 - 6y_0 - \frac{6y'_0}{h_1}.$$

Dunque, posto $d_0 = 6y_1 - 6y_0 - \frac{6y'_0}{h_1}$, abbiamo il seguente risultato:

$$2M_0 + M_1 = d_0.$$

Analogamente, posto $d_n = 6y_n - 6y_{n-1} - \frac{6y'_{n-1}}{h_n}$, otteniamo:

$$2M_{n-1} + M_n = d_n.$$

Otteniamo quindi la seguente matrice A, tridiagonale, irriducibile, diagonal dominante:

$$A = \begin{bmatrix} 2 & 1 & 0 & \cdots & & 0 \\ \alpha_1 & 2 & \beta_1 & 0 & \cdots & 0 \\ 0 & \alpha_2 & 2 & \beta_2 & & \vdots \\ 0 & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \alpha_{n-1} & 2 & \beta_{n-1} \\ 0 & & \cdots & 0 & 1 & 2 \end{bmatrix},$$

ed il seguente sistema lineare:

$$Ax = d, \quad d = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}, \quad x = \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix}.$$

Consideriamo A sopra di ordine $n+1$, $x \in \mathbb{R}^{n+1}, x \neq 0, y = Ax$. Cosa si può dire di $\|x\|_\infty$ e di $\|y\|_\infty$?

Proposizione 8.10. *Sia r l'indice per cui:*

$$|x_r| = \max_{0 \leq i \leq n} |x_i| = \|x\|_\infty.$$

Siano $\alpha_0 = 0$, $\beta_r = 0$ (per operare sulla prima riga e sull'ultima colonna),

$$\begin{aligned} |y_r| &= |\alpha_r x_{r-1} + 2x_r + \beta_r x_{r+1}| \\ &\geq 2|x_r| - \alpha_r|x_{r-1}| - \beta_r|x_{r+1}| \\ &\geq 2|x_r| - \alpha_r|x_r| - \beta_r|x_r| \\ &\geq |x_r| = \|x\|_\infty. \end{aligned}$$

Ovvero che $\|y\|_\infty \geq |y_r|$ in quanto è il $\max |y_i|$.

$$\|y\|_\infty \geq |y_r| \geq \|x\|_\infty \longrightarrow \|y\|_\infty \geq \|x\|_\infty.$$

Il vettore x viene trasformato mediante A in un vettore di norma ∞ maggiore. Con questo risultato dimostreremo la non singolarità di A.

Dimostrazione. Sia A non singolare, allora $x \in \mathbb{R}^{n+1}$, $x \neq 0$, $Ax = 0$.

$$\longrightarrow \|Ax\| = \|0\| = 0 \geq \|x\|_\infty > 0. \quad \text{Assurdo.}$$

■

8.1.4 Spline cubica periodica.

$$\underbrace{\Big[S_{3,\Delta}(x)\Big]'_{x=x_0} = \Big[S_{3,\Delta}(x)\Big]'_{x=x_n}}_{\text{equazione aggiuntiva (*)}}; \quad \underbrace{\Big[S_{3,\Delta}(x)\Big]''_{x=x_0} = \Big[S_{3,\Delta}(x)\Big]''_{x=x_n}}_{M_0=M_n}.$$

Ponendo $M_0 = M_n$ riduciamo il sistema ad n incognite, con l'equazione aggiuntiva avremo un sistema $n \times n$.

$$\underbrace{\Big[S_{3,\Delta}(x)\Big]'}_{x_0 \leq x \leq x_1} = \Big[S^{(1)}_{3,\Delta}(x)\Big]'_{x=x_0} =$$

$$= \left[\frac{(x-x_0)^2}{2h_1}M_1 - \frac{(x_1-x)^2}{2h_1}M_0 + \frac{y_1-y_0}{h_1} + \frac{h_1}{6}(M_0 - M_1)\right]_{x=x_0}.$$

$$\underbrace{\Big[S_{3,\Delta}(x)\Big]'}_{x_0 \leq x \leq x_1} = \Big[S^{(n)}_{3,\Delta}(x)\Big]'_{x=x_n} =$$

$$= \left[\frac{(x-x_{n-1})^2}{2h_n}M_n - \frac{(x_n-x)^2}{2h_n}M_{n-1} + \frac{y_n-y_{n-1}}{h_n} + \frac{h_n}{6}(M_{n-1} - M_n)\right]_{x=x_n}.$$

$$\Updownarrow$$

$$\Big[S^{(1)}_{3,\Delta}(x)\Big]'_{x=x_0} = -\frac{h_1}{2}M_0 + \frac{h_1}{6}M_0 - \frac{h_1}{6}M_1 + \frac{y_1-y_0}{h_1}.$$

$$\Big[S^{(n)}_{3,\Delta}(x)\Big]'_{x=x_n} = \frac{h_n}{2}M_n + \frac{h_n}{6}M_{n-1} - \frac{h_n}{6}M_n + \frac{y_n-y_{n-1}}{h_n}.$$

Dalla relazione (*) otteniamo:

$$-\frac{h_1}{2}M_0 + \frac{h_1}{6}M_0 - \frac{h_1}{6}M_1 + \frac{y_1-y_0}{h_1} = \frac{h_n}{2}M_n + \frac{h_n}{6}M_{n-1} - \frac{h_n}{6}M_n + \frac{y_n-y_{n-1}}{h_n},$$

avendo imposto $M_0 = M_n$ otteniamo dunque:

$$-\frac{h_n}{6}M_{n-1} + M_n\left(-\frac{h_1}{2} + \frac{h_1}{6} + \frac{h_n}{6} - \frac{h_n}{2}\right) - \frac{h_1}{6}M_1 = -\frac{y_1-y_0}{h_1} + \frac{y_n-y_{n-1}}{h_n}.$$

$$\Updownarrow$$

$$\frac{h_1}{6}M_1 + \frac{2}{6}M_n\,(h_1 + h_n) + \frac{h_n}{6}M_{n-1} = \frac{y_1 - y_0}{h_1} - \frac{y_n - y_{n-1}}{h_n}.$$

$$\Updownarrow$$

$$\text{moltiplichiamo per } \tfrac{6}{h_1+h_n}$$

$$\Updownarrow$$

$$\underbrace{\frac{h_1}{h_1 + h_n}}_{\alpha_1} M_1 + 2M_n + \underbrace{\frac{h_n}{h_1 + h_n}}_{\beta_1} M_{n-1} = \frac{6}{h_1 + h_n} \cdot \frac{y_1 - y_0}{h_1} - \frac{y_n - y_{n-1}}{h_n}$$

Osservazione 8.11. Si noti che con questa procedura perdiamo la struttura tridiagonale della matrice poiché l'equazione aggiunta è del tipo: $\alpha_n M_1 + \cdots + \beta_n M_{n-1} + 2M_n$ e il vincolo $M_0 = M_n$ trasforma la prima equazione in: $2M_1 + \beta_1 M_2 + \cdots + \alpha_1 M_n$.

$$A = \begin{bmatrix} 2 & \beta_1 & 0 & \cdots & 0 & \alpha_1 \\ \alpha_2 & 2 & \beta_2 & 0 & \cdots & 0 \\ 0 & \alpha_3 & 2 & \beta_3 & & \vdots \\ 0 & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & \alpha_{n-1} & 2 & \beta_{n-1} \\ \alpha_n & & \cdots & 0 & \beta_n & 2 \end{bmatrix}.$$

In questo caso occorre usare la tecnica gaussiana o tecniche analoghe, però tale procedimento potrebbe "sporcare" gli zeri. La tecnica più adatta è di tipo iterativo —non fatta in questo corso— che sfrutta la forte sparsità della matrice e non sporca gli zeri.

8.2 Caratterizzazione delle spline cubiche.

Definizione 8.12. Sia $f \in \mathcal{C}^2([a,b])$, si dice *pseudo norma* di f il numero reale $\|f\|^2$ tale che:

$$\|f\|^2 := \int_a^b [f''(x)]\,dx.$$

Non è una norma perchè esiste una $f \neq 0$ tale che:

$$\int_a^b [f''(x)]\,dx = 0.$$

Sia $\Delta = \{a = x_0 < x_1 < \cdots < x_n = b\}$ una decomposizione arbitraria di $[a, b]$, $S_{3,\Delta}(x)$ una spline cubica relativa alla decomposizione —non è richiesto che sia interpolante—.

Teorema 8.13. *(Holiday) Sia $f(x) \in \mathcal{C}^0([a, b])$, Δ una decomposizione arbitraria di $[a, b]$ e $S_{3,\Delta}(x)$ una spline cubica relativa alla decomposizione allora:*

$$\|f(x) - S_{3,\Delta}(x)\|^2 =$$

$$\begin{aligned} = \quad & \|f(x)\|^2 \\ & -2\left[\left(f'(x) - S'_{3,\Delta}(x)\right) S''_{3,\Delta}(x)\big|_{x=a}^{x=b} + \textstyle\sum_{i=1}^{n}(f(x) - S_{3,\Delta}(x))S''_{3,\Delta}(x)\big|_{x=x_{i-1}^{+}}^{x=x_i^{-}}\right] \\ & -\|S_{3,\Delta}(x)\|^2. \end{aligned}$$

Sia ora $S_{3,\Delta}(x)$ una spline cubica relativa alla decomposizione Δ *interpolante* la funzione f.

- Per spline cubica interpolante naturale:

  $$\|f(x) - S_{3,\Delta}(x)\|^2 = \|f(x)\|^2 + 2[0] - \|S_{3,\Delta}(x)\|^2 > 0.$$

  $$\longrightarrow \|f(x)\|^2 \geq \|S_{3,\Delta}(x)\|^2.$$

  $f, S \in \mathcal{C}^2$, osserviamo che tra tutte le funzioni $\mathcal{C}^2$ che passano per quei punti, la spline cubica interpolante naturale è quella che minimizza il funzionale $\|\cdot\|^2$.

  Analogamente per le spline cubiche vincolate con f che soddisfa le relazioni di derivata prima agli estremi.

- Per la spline cubica interpolante vincolata:

  $$S'_{3,\Delta}(x)\big|_{x=x_0} = f'(x_0), \quad S'_{3,\Delta}(x)\big|_{x=x_n} = f'(x_n).$$

  Andando a calcolare la pseudo norma la quantità $\sum_{i=1}^{n}(f(x) - S_{3,\Delta}(x))$ si annulla poiché S è interpolante, la quantità $f'(x) - S'_{3,\Delta}(x)\big|_{x=a}^{x=b}$ si annulla perchè è differenza.

  $$\longrightarrow \|f(x)\|^2 \geq \|S_{3,\Delta}(x)\|^2.$$

  La funzione spline vincolata minimizza $\|\cdot\|^2$ nell'insieme delle funzioni $f \in \mathcal{C}^2$ che passano per quei punti e vincolate ad avere tali valori $f'(x_0)$ e $f'(x_n)$.

- Per spline cubica interpolante periodica:

$$\|f(x)\|^2 \geq \|S_{3,\Delta}(x)\|^2.$$

  Minimizzando $\|\cdot\|^2$ vuol dire che $S_{3,\Delta}(x)$ minimizza la curvatura nell'intervallo $[a,b]$ (come somme di curve locali).

Essendo $\|\cdot\|^2$ un funzionale di energia, la spline minimizza l'energia?

Definizione 8.14. Sia $y = g(x)$, $x \in [a,b]$, la curvatura c in x di $g(x)$ è definita come segue:

$$c(x) := \frac{g''(x)}{[1+(g'(x))^2]^{\frac{3}{2}}}.$$

Se $g'(x)$ è prossimo a 0 il denominatore è prossimo a 1.

La spline è quella funzione che passa nei punti segnati e che minimizza la curvatura sull'intervallo.

$$\|f(x) - S_{3,\Delta}(x)\|^2 = \|f(x)\|^2 - \|S_{3,\Delta}(x)\|^2. \tag{8.1}$$

Ultilizzeremo l'equazione 8.1 per dimostrare l'unicità di una spline cubica naturale, vincolante o periodica.

8.3 Unicità delle spline cubiche.

Proposizione 8.15. *Siano $S_{3,\Delta}(x) \in \mathcal{C}^2$ e $\overline{S}_{3,\Delta}(x) \in \mathcal{C}^2$, due spline cubiche interpolanti la funzione f nei medesimi punti. Allora:*

$$\overline{S}_{3,\Delta}(x) = S_{3,\Delta}(x).$$

Dimostrazione. Supponiamo esistano $S_{3,\Delta}(x)$ e $\overline{S}_{3,\Delta}(x)$, due spline cubiche interpolanti soddisfacenti le medesime condizioni.

Se $\overline{S}_{3,\Delta}(x)$ ha le stesse caratteristiche di S, cioé interpola nei medesimi punti f e appartiene alla classe $\mathcal{C}^2$ allora la possiamo "prendere" come f:

$$\|\overline{S}_{3,\Delta}(x) - S_{3,\Delta}(x)\|^2 = \|\overline{S}_{3,\Delta}(x)\|^2 - \|S_{3,\Delta}(x)\|^2 \geq 0. \tag{8.2}$$

Analogamente possiamo scambiare i ruoli di $\overline{S}_{3,\Delta}(x)$ e $S_{3,\Delta}(x)$:

$$\|S_{3,\Delta}(x) - \overline{S}_{3,\Delta}(x)\|^2 = \|S_{3,\Delta}(x)\|^2 - \|\overline{S}_{3,\Delta}(x)\|^2 \geq 0. \tag{8.3}$$

Da 8.2 e 8.3 otteniamo:

$$0 = \|S_{3,\Delta}(x)\|^2 - \|\overline{S}_{3,\Delta}(x)\|^2 = \int_a^b [S''_{3,\Delta}(x) - \overline{S}''_{3,\Delta}(x)]^2 dx.$$

Siccome la funzione integranda è sempre positiva (il quadrato) l'unica possibilità poiché l'integrale sia nullo è che la funzione integranda sia identicamente nulla, cioè:

$$\overline{S}''_{3,\Delta}(x) - S''_{3,\Delta}(x) = 0 \longrightarrow \overline{S}''_{3,\Delta}(x) = S''_{3,\Delta}(x).$$

$$\longrightarrow \overline{S}_{3,\Delta}(x) = S_{3,\Delta}(x) + cx + d.$$

Deve succedere che:

$$\begin{cases} ca + d = 0 \\ cb + d = 0 \end{cases} \quad \begin{bmatrix} a & 1 \\ b & 1 \end{bmatrix} = a - b \neq 0.$$

L'unica soluzione è quella nulla $(0, 0)$ ovvero $c = d = 0$.

$$\longrightarrow \overline{S}_{3,\Delta}(x) = S_{3,\Delta}(x).$$

■

8.4 Spline lineari.

Ricordiamo il *polinomio interpolatore di Lagrange*, costruito come combinazione lineare dei polinomi *lagrangiani* con coefficienti i valori che deve interpolare (ovvero i dati):

$$p(x) = \sum_{i=1}^{n} y_i l_i(x).$$

Possiamo recuperare questa idea nell'ambiente delle spline?

8.4.1 Funzioni polinomiali a cappuccio (Spline a cappuccio).

Costruiamo i seguenti polinomi lineari a cappuccio definiti a tratti:

$$B_i(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}} & x_{i-1} \leq x \leq x_i, \\ \\ \frac{x - x_i}{x_{i+1} - x_i} & x_i \leq x \leq x_{i+1}, \\ \\ 0 & \text{altrimenti.} \end{cases}$$

Possiamo costruirli su un dominio costituito interamente dal nostro intervallo (la decomposizione Δ), per $i = 0, \ldots, n-1$. Cioè costruiamo un polinomio

lineare con i polinomi lineari —e non con $1, \ldots, n$— usando i due tratti delle B_i e B_{i+1} tra $[x_i, x_{i+1}]$. Ovvero:

$$p(x) = \alpha_i B_i(x) + \alpha_{i+1} B_{i+1}(x), \quad x \in [x_i, x_{i+1}].$$

Questo si può fare in ogni tratto tranne che in $[x_0, x_1]$ e in $[x_{n-1}, x_n]$, poiché il polinomio relativo in questi tratti non sarebbe completo. Ad esempio nel tratto $[x_0, x_1]$ avremmo solo la componente $\alpha_1 B_1(x)$ del polinomio, quindi o costruiamo un mezzo cappuccio oppure aggiungiamo un punto (x_{-1} in questo caso).

$$B_0(x) = \begin{cases} \frac{x_1 - x}{x_1 - x_0} & x_0 \leq x \leq x_1, \\ 0 & \text{altrimenti.} \end{cases}$$

$$B_n(x) = \begin{cases} \frac{x_n - x}{x_n - x_{n-1}} & x_{n-1} \leq x \leq x_n, \\ 0 & \text{altrimenti.} \end{cases}$$

Proposizione 8.16. *I polinomi $B_i(x)$ definiti come sopra, con $i = 0, \ldots, n$ sono linearmente indipendenti.*

Dimostrazione. (traccia)

$$\sum_{i=1}^{n} c_i B_i(x) = 0 \iff c_i = 0 \ \forall i = 1, \ldots, n.$$

Per concludere la dimostrazione è sufficiente prendere i nodi e valutare la combinazione lineare, per ogni nodo si salva solo B_i allora se $c_i B_i = 0$ si ha che $c_i = 0$.

■

Ritornando alle spline, definiamo la seguente spline lineare:

$$S_{1,\Delta}(x) = \sum_{i=0}^{n} \alpha_i B_i(x).$$

$S_{1,\Delta}(x)$ è interpolante, $S_{1,\Delta}(x_k) = y_k$ per ogni $k = 0, \ldots, n$.

$$S_{1,\Delta}(x_k) = \sum_{i=0}^{n} \alpha_i B_i(x_k) = \alpha_0 \cancelto{0}{B_0(x_k)} + \cdots + \alpha_k B_k(x_k) + \cdots + \alpha_n \cancelto{0}{B_n(x_k)} = y_k.$$

$$\longrightarrow \ y_k = \alpha_k \cancelto{1}{B_k(x_k)} = \alpha_k.$$

$$\longrightarrow \; S_{1,\Delta}(x) = \sum_{i=0}^{n} y_i B_i(x).$$

Abbiamo quindi ottenuto una spline lineare come combinazione lineare i cui coefficienti sono i dati che già avevamo.

B_i interpola:

$$\begin{array}{ccc} x_0 & 0 & (x_0, 0) \\ x_1 & 0 & (x_1, 0) \\ \vdots & & \\ x_{i-1} & 0 & (x_{i-1}, 0) \\ x_i & 1 & (x_i, 1) \\ x_{i+1} & 0 & (x_{i+1}, 0) \\ \vdots & & \\ x_n & 0 & (x_n, 0) \end{array}$$

$$B_i(x_j) = \begin{cases} 1 & \text{se } j = i \\ 0 & \text{se } j \neq i. \end{cases}$$

Osservazione 8.17. B_i risente dell'errore solo nel tratto $[x_{i-1}, x_{i+1}]$, mentre il polinomio lagrangiano l_i lo distribuisce su tutto $[a, b]$, abbiamo il grande vantaggio che tale errore, dove c'è, resti concentrato localmente.

Se $f(x_i) = y_i$ allora:

$$S_{1,\Delta}(x) = \sum_{i=0}^{n} f(x_i) B_i(x).$$

Proprietà: supporto.

La proprietà che differenzia i polinomi lagrangiani dai polinomi a cappuccio B_i è quella data dall'osservazione 8.17, tale proprietà è chiamata *supporto*.

Proprietà: convergenza.

Le spline lineari convergono alle funzioni di classe $\mathcal{C}^2$. Invece non c'è convergenza con polinomi classici, neanche con i lagrangiani (es: Runghe).

Capitolo 9

Convergenza.

9.1 Teorema di Faber.

Sia $[a,b] \subset \mathbb{R}$, f una funzione continua su tale intervallo ($f \in \mathcal{C}^0([a,b])$), prendiamo $n+1$ punti e costruiamo i polinomi interpolatori:

$$\begin{array}{ll} x_0^1 & p_0(x) \\ x_0^2\ x_1^2 & p_1(x) \\ x_0^3\ x_1^3\ x_2^3 & p_2(x) \\ \vdots & \\ x_0^{n+1}\ x_1^{n+1}\ \cdots\ x_n^{n+1} & p_n(x) \end{array}$$

Gli x_i^{j+1} possono coincidere con gli x_i^j, ma in una data riga ogni x_i^k sono tutti distinti per ogni $i = 0, \ldots, k-1$.

Possiamo costruire una matrice infinita. Tuttavia Runghe ci porta a non essere sicuri di questa situazione.

$$f(x) = \frac{1}{1+x^2}, \quad x \in [-5,5]$$

al crescere di n abbiamo una divergenza sugli estremi dell'intervallo usando i nodi e i polinomi interpolatori.

Enunciamo un teorema che contiene questa idea.

Teorema 9.1. *Per ogni matrice interpolatoria esiste una funzione continua che* non *è il limite della successione dei polinomi* $\{p_n(x)\}$.

Osservazione 9.2. Con questo teorema non si afferma che non sia possibile costruire una successione che converga alla funzione. Nell'esempio di

Runghe si vedeva che, sostituendo a nodi equidistanti i nodi di Chebyshev, la divergenza agli estremi scompariva.

Per ogni funzione continua esiste una matrice interpolatoria che converge ad una funzione continua. Il problema è diverso.

9.2 Polinomi di Chebyshev.

Se $f \in \mathcal{C}^1$ è utile utilizzare i nodi di Chebyshev (riportando $f \in [-1,1]$) poiché otteniamo convergenza. Infatti:

$$f(x) \in \mathcal{C}^1([a,b]), \quad \|f(x) - p_n(x)\|_\infty = \mathrm{O}\left(\frac{1}{\sqrt{n}}\right).$$

In cui $p_n(x)$ interpola f sui nodi di Chebyshev.

$$f, g: \qquad \underbrace{(f,g)_\omega}_{\text{scalare}} = \int_{-1} \underbrace{\frac{1}{\sqrt{1-x^2}}}_{\text{f.ne peso}} f(x)g(x)dx, \qquad (f,g)_\omega = 0.$$

$$(p_n, p_m)_\omega = \int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} p_n(x)p_m(x)dx.$$

$(p_n, p_m)_\omega$ caratterizza l'ortogonalità nei polinomi di Chebyshev.

Ricordiamo che i nodi di Chebyshev sono gli *zeri* del polinomio di Chebyshev, quindi prima occorre costruire i polinomi, quindi calcolarne gli zeri.

9.2.1 Costruzione dei polinomi di Chebyshev.

Come si trovano i polinomi di Chebyshev? Sappiamo che sono definiti nell'intervallo $[-1,1]$.

$p_0(x) = 1.$
$p_1(x) = ax + b.$

$$\omega(x) = \frac{1}{\sqrt{1-x^2}}.$$

$$(p_0, p_1)_\omega = \int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} p_0(x)p_1(x)dx = 0.$$

Questa è la prima condizione da imporre, poiché vogliamo dei polinomi ortogonali, tuttavia non è sufficiente per avere entrambi i coefficienti a e b.

Se prendiamo un polinomio *monico*[*] (è sufficiente dividere per a) gli zeri non cambiano.

$$p_1(x) = x + b'.$$

Questo ha una condizione ed una variabile, quindi lo determiniamo.

$$p_2(x) = ax^2 + bx + c.$$

$$\begin{cases} (p_0, p_2)_\omega = 0 \\ (p_1, p_2)_\omega = 0 \end{cases} \longleftrightarrow \begin{cases} \int_{-1}^{1} \omega(x) p_0(x) p_2(x) dx = 0 \\ \int_{-1}^{1} \omega(x) p_1(x) p_2(x) dx = 0 \end{cases}$$

Anche qui possiamo determinare il polinomio monico, due condizioni e due incognite. E' sempre possibile costruire una famiglia di polinomi monici.

Osservazione 9.3. Cambiando la funzione *peso* cambia la famiglia che si costruisce.

Vediamo la seguente funzione peso:

$$\omega(x) = \log(x). \quad \in [0, 1]$$

Osservazione 9.4. La funzione peso può avere problemi di integrazione (singolarità deboli) solo in punti estremi e avere convergenza.

Possiamo utilizzare Newton-Cotes?

- Chiuse? No perchè il logaritmo non è definito in 0.
- Aperte? Dipende da quanto vicino allo zero prendiamo i nodi. Infatti nell'avvicinarsi a zero otteniamo un nodo del tutto sperimentale per calcolare l'integrale. L'errore dipende dalle derivate.

Se costruiamo una formula di quadratura che *assorba* la singolarità debole della funzione peso possiamo analizzare solo la funzione f.

9.3 Spline.

L'idea di utilizzare le spline lineare è estendibile alle spline di altri ordini?

[*]Un polinomio si dice monico se il coefficiente della variabile di grado maggiore è pari a 1 ($a_n = 1$).

9.3.1 Spline cardinali.

Sia $S_{3,\Delta}(x)$ una spline cubica interpolante vincolata (completa),

$$\left.\begin{array}{c} S'_{3,\Delta}(x_0) = y'_0 = f'(x_0) \\ S'_{3,\Delta}(x_n) = y'_n = f'(x_n) \end{array}\right\} \text{ assegnate.}$$

$$\begin{array}{c} S_{3,\Delta,-1}(x) \\ \vdots \\ S_{3,\Delta,i}(x) \\ \vdots \\ S_{3,\Delta,n+1}(x) \end{array}$$

$$\begin{array}{c} S_{3,\Delta,i}(x) \\ i = 0, \ldots, n. \end{array} = \begin{cases} S_{3,\Delta,i}(x_j) = \delta_{i,j} & j = 0, \ldots, n. \\ S'_{3,\Delta,i}(x_0) = 0 \\ S'_{3,\Delta,i}(x_n) = 0 \end{cases}$$

$$S_{3,\Delta,-1}(x) = \begin{cases} S_{3,\Delta,-1}(x_j) = 0 & j = 0, \ldots, n. \\ S'_{3,\Delta,-1}(x_0) = 1 \\ S'_{3,\Delta,-1}(x_n) = 0 \end{cases}$$

$$S_{3,\Delta,n+1}(x) = \begin{cases} S_{3,\Delta,n+1}(x_j) = 0 & j = 0, \ldots, n. \\ S'_{3,\Delta,n+1}(x_0) = 0 \\ S'_{3,\Delta,n+1}(x_n) = 1 \end{cases}$$

Le spline così definite si dicono *cardinali*.

$$S_{3,\Delta}(x) = \sum_{j=0}^{n} f(x_j) S_{3,\Delta,j}(x) + f'(x_0) S_{3,\Delta,-1}(x) + f'(x_n) S_{3,\Delta,n+1}(x).$$

$$S_{3,\Delta}(x_i) = \sum_{j=0}^{n} f(x_j) S_{3,\Delta,j}(x_i) = f(x_i).$$

$$S'_{3,\Delta}(x) = \sum_{j=0}^{n} f(x_j) S'_{3,\Delta,j}(x) + f'(x_0) S'_{3,\Delta,-1}(x) + f'(x_n) S'_{3,\Delta,n+1}(x).$$

$$S'_{3,\Delta}(x_0) = \cdots = f'(x_0).$$

Soddisfa il primo vincolo, analogamente

$$S'_{3,\Delta}(x_n) = \cdots = f'(x_n).$$

$\Longrightarrow$ $S_{3,\Delta}(x)$ è una spline cubica interpolante vincolata.

Questa nuova espressione di $S_{3,\Delta}(x)$ è simile all'interpolazione lagrangiana. Infatti $S_{3,\Delta,j}(x)$ sono definiti su tutto l'intervallo come i polinomi lagrangiani.

9.3.2 Spline cubiche naturali.

Come si costruisce una spline cubica naturale utilizzando le spline cardinali?

Innanzitutto non dobbiamo più avere dati aggiuntivi iniziali poiché la spline è caratterizzata da:

$$S''_{3,\Delta}(x_0) = S''_{3,\Delta}(x_n) = 0.$$

Quindi rimangono questi soli vincoli:

$$\begin{matrix} S_{3,\Delta,i}(x) \\ i = 0, \ldots, n. \end{matrix} = \begin{cases} S_{3,\Delta,i}(x_j) = \delta_{i,j} & j = 0, \ldots, n. \\ S'_{3,\Delta,i}(x_0) = 0 \\ S'_{3,\Delta,i}(x_n) = 0 \end{cases}$$

$S_{3,\Delta,-1}$ e $S_{3,\Delta,n+1}$ non servono più perchè non c'è più la condizione di vincolo.

$$S_{3,\Delta}(x) = \sum_{i=0}^{n} f(x_i) S_{3,\Delta,i}(x).$$

Cosa disturba? L'errore locale si propaga su tutto l'intervallo come in Lagrange, perchè la base stessa si espande su tutto l'intervallo.

Nuovo passo:

occorre costruire una base di funzioni (cubiche) che abbiano supporto minimo all'interno dell'intervallo.

Suggerimento:

si prendano come funzioni polinomi B_i simili a quelli del capitolo precedente ma di grado tre.

Osservazione 9.5. La dimensione dello spazio delle spline cubiche naturali è $n + 1$ mentre quello delle spline cubiche vincolate è $n + 3$.

9.4 Tecnica dei minimi quadrati.

Prendiamo $n+1$ punti non necessariamente distinti e $n+1$ passaggi distinti:

$$x_i \quad y_i \qquad i = 0, \ldots, n.$$

Dato $m \ll n$, come si ottiene il modello $f_m(x)$?

Sia $\varphi_0(x), \varphi_1(x), \ldots, \varphi_m(x)$ una base per lo spazio in cui stiamo costruendo l'approssimazione di x.

$$f(x) = c_0\varphi_0(x) + c_1\varphi_1(x) + \cdots + c_m\varphi_m(x).$$

$c_j = ?\ j = 0, \ldots, m.$

Determinati gli elementi c_i otteniamo il modello.

$$r_i = y_i - f_m(x_i) \quad i = 0, \ldots n.$$

r_i si dice *residuo*, ovvero quanto il modello differisce dal punto dato.

$$\sum_{i=0}^{n} r_i^2 = \sum_{i=0}^{n} (y_i - f_m(x_i))^2 = \sum_{i=0}^{n} \left(y_i - \sum_{j=0}^{m} c_j\varphi_j(x_i) \right)^2.$$

Occorre cercarne il minimo:

$$\min \sum_{i=0}^{n} r_i^2 = \min_{c\in\mathbb{R}^{m+1}} \sum_{i=0}^{n} \left(y_i - \sum_{j=0}^{m} c_j\varphi_j(x_i) \right)^2 \qquad c = \begin{bmatrix} 0 \\ c_1 \\ \vdots \\ c_m \end{bmatrix},$$

$y_i = f_m(x_i) + r_i = \sum_{j=0}^{m} c_j\varphi_j(x_i), \quad i = 0, \ldots, n.$

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix} = \underbrace{\begin{bmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \cdots & \varphi_m(x_0) \\ \varphi_0(x_1) & \varphi_1(x_1) & \cdots & \varphi_m(x_1) \\ \vdots & & & \\ \varphi_0(x_n) & \varphi_1(x_n) & \cdots & \varphi_m(x_n) \end{bmatrix}}_{A\in\mathbb{R}^{n+1\times m+1}:\ \text{rettangolare}} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} + \begin{bmatrix} r_0 \\ r_1 \\ \vdots \\ r_n \end{bmatrix}$$

$Ac = y$ (è il minimo).

Condizione necessaria per trovare il minimo:

$$\frac{\partial \sum_{i=0}^{n} r_i^2}{\partial c_k} = 0, \quad \frac{\partial \xi^2}{\partial c_k} = 0. \qquad k = 0, \ldots, m.$$

$$\text{se} \quad \sum_{i=0}^{n} r_i^2 = \xi^2.$$

$$\frac{\partial}{\partial c_k}\sum_{i=0}^{n}\left(y_i-\sum_{j=0}^{m}c_j\varphi_j(x_i)\right)^2=0. \qquad k=0,\dots,m.$$

$$2\sum_{i=0}^{n}\left(y_i-\sum_{j=0}^{m}c_j\varphi_j(x_i)\right)=0.$$

$$\sum_{i=0}^{n}y_i\varphi_k(x_i)-\sum_{i=0}^{n}\sum_{j=0}^{m}c_j\varphi_j(x_i)\varphi_k(x_i)=0.$$

$$\sum_{j=0}^{m}c_j\sum_{i=0}^{n}\varphi_j(x_i)\varphi_k(x_i)=\sum_{i=0}^{n}y_i\varphi_k(x_i). \qquad k=0,\dots,m.$$

Vediamo cosa significano:

$$k=0,\quad \sum_{j=0}^{m}c_j\sum_{i=0}^{n}\varphi_j(x_i)\varphi_0(x_i)=\sum_{i=0}^{n}y_i\varphi_0(x_i).$$

$k=0\ \wedge\ j=0\ \longrightarrow \sum_{i=0}^{n}\varphi_0(x_i)\varphi_0(x_i)$ etc.

$$\begin{array}{c} k=0 \\ k=1 \\ \vdots \\ k=m \end{array}\underbrace{\begin{bmatrix} \sum_{i=0}^{n}\varphi_0(x_i)\varphi_0(x_i) & \sum_{i=0}^{n}\varphi_1(x_i)\varphi_0(x_i) & \cdots & \sum_{i=0}^{n}\varphi_m(x_i)\varphi_0(x_i) \\ \sum_{i=0}^{n}\varphi_0(x_i)\varphi_1(x_i) & \sum_{i=0}^{n}\varphi_1(x_i)\varphi_1(x_i) & \cdots & \sum_{i=0}^{n}\varphi_m(x_i)\varphi_1(x_i) \\ \vdots & \vdots & \ddots & \\ \sum_{i=0}^{n}\varphi_0(x_i)\varphi_m(x_i) & \sum_{i=0}^{n}\varphi_1(x_i)\varphi_m(x_i) & \cdots & \sum_{i=0}^{n}\varphi_m(x_i)\varphi_m(x_i) \end{bmatrix}}_{B\in\mathbb{R}^{m+1\times m+1}\text{: quadrata}}$$

$$d=\begin{bmatrix}\sum_{i=0}^{n}y_i\varphi_0(x_i)\\ \vdots \\ \sum_{i=0}^{n}y_i\varphi_m(x_i)\end{bmatrix},\quad c=\begin{bmatrix}c_0\\ \vdots\\ c_m\end{bmatrix}.$$

$d\in\mathbb{R}^{m+1}$ è il vettore dei termini noti, mentre $c\in\mathbb{R}^{m+1}$ è il vettore delle incognite. Dobbiamo quindi risolvere il sistema $Bc=d$.

Osservazione 9.6. $B=A^tA$ e $d=A^ty$, da queste relazioni otteniamo:

$$Ac=y\ \longrightarrow\ \begin{matrix}A^tAc=A^ty\\ Bc=d\end{matrix}.$$

Poiché $\varphi_0, \varphi_1, \ldots, \varphi_n$ sono linearmente indipendenti la matrice A ha rango *massimo*, ovvero $n+1$. Non possiamo parlare di singolarità perchè la matrice è rettangolare.

Ne segue che B è *non singolare*.

Possiamo utilizzare la tecnica gaussiana per risolvere il sistema, ma essendo B simmetrica, se riuscissimo a dimostrare che è anche definita positiva potremmo applicare la fattorizzazione di Cholesky, riducendo il costo computazionale.

$(A^tA)^t = A^tA$.

$$\forall x \in \mathbb{R}^{m+1} \quad x^tA^tAx > 0,\ (Ax)^tAx > 0.$$

E' definita positiva essendo questo un prodotto scalare.

La tecnica dei minimi quadrati che minimizza quel funzionale al quadrato si riduce alla soluzione di un sistema lineare mediante Cholesky.

Esempio. Retta ai minimi quadrati.

$$x_i \quad y_i$$

$f(x) = ax + b \quad = c_0 + c_1x$
$a, b =?$
$\varphi_0(x) = 1.$
$\varphi_1(x) = x.$

$$A = \underbrace{\begin{bmatrix} 1 & x_0 \\ 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}}_{\substack{\uparrow \quad \uparrow \\ \varphi_0(x_i) \quad \varphi_0(x_i)}} \qquad B = \underbrace{\begin{bmatrix} \quad & \quad \\ \quad & \quad \end{bmatrix}}_{\substack{2\times 2 \\ \text{qualunque sia} \\ \text{il n° dei nodi}}} \qquad c = \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} \qquad d = A^t \cdot \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}.$$

Esiste un'alternativa a questo modo di procedere più corretta e più stabile?

$$\xi^2 = \sum_{i=0}^{n} r_i^2.$$

$$\min \xi^2 = \min_c \sum_{i=0}^{n} \left(y_i - \sum_{j=0}^{m} c_j \varphi_j(x_i) \right)^2 = \min_c \|y - Ac\|_2^2 .$$

Ma A non è una matrice quadrata, occorre quindi ripercorrere la stessa "strada" utilizzata per arrivare alla fattorizzazione delle matrici QR nel caso rettangolare. Quindi utilizziamo la fattorizzazione QR su A.

$$A = QR.$$

Domanda:

E' applicabile questa tecnica a $f(x) = c_0 + c_1 e^{c_3 x}$?

La risposta è no, perchè questo non è un problema lineare nelle incognite.

Invece in $f(x) = c_0 + c_1 e^x$ la tecnica è applicabile perchè avremmo $\varphi_0 = 1$ e $\varphi_1 = e^x$ ovvero un problema lineare nelle due incognite c_0 e c_1.

Teorema 9.7. *Sia $f(x) \in \mathcal{C}^4([a,b])$, $S_{3,\Delta}(x)$ una spline cubica vincolata (completa) relativa ad una decomposizione Δ di $[a,b]$ e interpolante la funzione $f(x)$. Indicato con M, H e h:*

$$M = \max_{a \le x \le b} \left| f^{(iv)}(x) \right|, \ H = \max_{1 \le i \le n} h_i, \ h = \min_{1 \le i \le n} h_i;$$

allora si ha:

1. $|f(x) - S_{3,\Delta}(x)| \le \frac{7}{8} M \frac{H^5}{h}$.

   Osservazione 9.8. Se il passo è costante si ha $H = h$ per cui:

   $$|f(x) - S_{3,\Delta}(x)| \le \frac{7}{8} M H^4,$$

   ovvero una convergenza dell'ordine di H^4.

   $$H \to 0 \implies f(x) \to S_{3,\Delta}(x).$$

2. $\left| f(x)' - S'_{3,\Delta}(x) \right| \le \frac{7}{4} M \frac{H^4}{h}$.
   Analogamente a prima se il passo è costante abbiamo una convergenza dell'ordine di H^3.

3. $\left| f''(x) - S''_{3,\Delta}(x) \right| \le \frac{7}{4} M \frac{H^3}{h}$.
   Quindi per approssimare le funzioni derivate della f basta derivare la spline che la approssima.

4. $\left| f'''(x) - S'''_{3,\Delta}(x) \right| \le \frac{7}{2} M \frac{H^2}{h}$.
   $x \in (x_{i-1}, x_i) \quad i = 1, \ldots, n$, *perchè $S \notin \mathcal{C}^\infty$ nei bordi.*

$$b_1 - a_1 = \frac{b_0 - a_0}{2}, \quad b_2 - a_2 = \frac{b_1 - a_1}{2} = \frac{b_0 - a_0}{2^2}.$$

Capitolo 10

Metodi iterativi.

I metodi di risoluzione di sistemi lineari visti fin'ora sono metodi diretti, esiste un'altra categoria di metodi per tale obiettivo: i *metodi iterativi*.

I metodi iterativi costituiscono una successione di vettori che converge al valore soluzione. Dato un problema $Ax = b$,

$$x^{(0)} \quad \text{vettore dato.}$$

$$x^{(k+1)} = Bx^{(k)} + q. \quad k = 0, 1, \ldots$$

con ipotesi varie abbiamo che:

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

$$\lim_{k \to +\infty} \|x^{(k)} - x\| = 0.$$

$$\|x^{(k)} - x\| \leq \varepsilon. \qquad (\varepsilon \text{ tolleranza})$$

Perchè usarlo se dovremo "troncare" le iterazioni ad un numero finito? Semplicemente perchè l'Analisi Numerica contempla e accetta l'esistenza di un errore.

10.1 Differenza dai metodi diretti.

I metodi diretti sono caratterizzati dal fatto di sapere già inizialmente quale sarà il costo computazionale della risoluzione di un dato sistema.

Nei metodi iterativi, fissata una precisione, quante operazioni servono per ottenerla? Applicheremo il metodo iterativo se avrà un costo minore.

Operazione cardine:
prodotto matrice-vettore $Bx^{(k)}$, ha costo n^2 se n è l'ordine della matrice. Quindi, se riusciamo ad ottenere la stessa approssimazione con *meno* di n iterazioni, abbiamo *risparmiato*. Inoltre se B fosse tridiagonale il prodotto matrice vettore costerebbe n (costo lineare), allora avremmo a disposizione fino a n^2 iterazioni.

Nell'interpolazione lo scopo è di ricostruire l'andamento di un fenomeno avendo a disposizione dei dati discreti.

10.2 Risoluzione di sistemi non lineari.

Data una funzione $f\colon [a,b] \to \mathbb{R}$, trovare $\alpha \in [a,b]$ tale che $f(\alpha) = 0$.

Non avendo a disposizione un algoritmo come nel caso dei sistemi lineari, poiché l'Analisi non offre l'algoritmo costruttivo, occorrerà procedere con un *metodo iterativo*, troncando il procedimento infinito in modo tale che il risultato sia sufficientemente vicino. Partendo da:

$$x^{(0)} \quad \text{dato},$$

costruiamo una successione $\{x^{(k)}\}$ di punti sperando che:

$$\lim_{k \to +\infty} x^{(k)} = \alpha.$$

Bloccheremo il procedimento in corrispondenza di una tolleranza ε tale che:

$$\left| x^{(k)} - \alpha \right| < \varepsilon.$$

Osservazione 10.1. Tale disuguaglianza non può essere testata non conoscendo ε, dovremo trovare una disuguaglianza alternativa.

10.2.1 Metodo di bisezione (dicotomica).

E' un metodo dell'Analisi.

Teorema 10.2. *Sia* $f(x) \in \mathcal{C}^0([a,b])$ *tale che* $f(a) \cdot f(b) < 0$, *allora*

$$\exists \alpha \in [a,b]\colon f(\alpha) = 0.$$

Nota Bene. Una funzione può avere uno zero anche in punto molto "avanti" nel dominio, anche in corrispondenza di un numero non esprimibile dalla macchina. Come si opera in questi casi? Occorrerà scalare l'intervallo in modo da saper esprimere la radice.

Sia $[a_0, b_0] := [a, b]$, poniamo:

$$x^{(0)} = \frac{a_0 + b_0}{2}.$$

Se $f(x^{(0)}) = 0$ allora $\alpha = x^{(0)}$. STOP.

Osservazione 10.3. In Matlab $f(x^{(0)}) = 0$ non ci assicura che $x^{(0)}$ sia radice a causa dell'approssimazione di macchina. Sarà qualcosa di leggermente diverso, quindi occorre fare attenzione a dire che siamo vicini ad α.

- Se $f(x^{(0)})f(a_0) < 0$ allora $a_1 = a_0$ e $b_1 = x^{(0)}$.
- Se $f(x^{(0)})f(b_0) < 0$ allora $a_1 = x^{(0)}$ e $b_1 = b_0$.

Otteniamo:

$$f : [a_1, b_1] \to \mathbb{R} \colon f(a_1) \cdot f(b_1) < 0.$$

Osservazione 10.4. Se valesse il primo caso *non procederemmo* ad analizzare il secondo, questo perchè (parlando in termini informatici) essendo vero il primo "if" non occorre calcolare le radici del secondo tratto, risparmiando così calcoli.

$$x^{(1)} = \frac{a_1 + b_1}{2}.$$

- Se $f(x^{(1)}) = 0$ allora $\alpha = x^{(1)}$. STOP.
- Se $f(x^{(1)})f(a_1) < 0$ allora $a_2 = a_1$ e $b_2 = x^{(1)}$.
- Se $f(x^{(1)})f(b_1) < 0$ allora $a_2 = x^{(1)}$ e $b_2 = b_1$.

Sostituiremo il test $\left|x^{(k)} - \alpha\right| < \varepsilon$ con:

$$|\,\text{ampiezza intervallo}\,| < \varepsilon.$$

Porremo la radice approssimata nel punto medio di quell'intervallo.

$$b_1 - a_1 = \frac{b_0 - a_0}{2}, \quad b_2 - a_2 = \frac{b_1 - a_1}{2} = \frac{b_0 - a_0}{2^2}.$$

Dopo k iterazioni arriveremo ad avere il seguente intervallo:

$$I_{k-1} = [a_{k-1}, b_{k-1}],$$

tale che $f(a_{k-1}) \cdot f(b_{k-1}) < 0$.

$$x^{(k-1)} = \frac{a_{k-1} + b_{k-1}}{2}.$$

- Se $f(x^{(k-1)}) = 0$ allora $\alpha = x^{(k-1)}$. STOP.
- Se $f(x^{(k-1)})f(a_{k-1}) < 0$ allora $a_k = a_{k-1}$ e $b_k = x^{(k-1)}$.
- Se $f(x^{(k-1)})f(b_{k-1}) < 0$ allora $a_k = x^{(k-1)}$ e $b_k = b_{k-1}$.

Ottenendo un nuovo intervallo $I_k = [a_k, b_k]$ di ampiezza:

$$b_k - a_k = \frac{b_{k-1} - a_{k-1}}{2} = \frac{b_0 - a_0}{2^k}.$$

Osservazione 10.5. Alla fine del procedimento otterremo un'unica radice di f, se ve ne fossero altre sarebbero state perse.

$$|\text{errore}| = \left|x^{(k)} - \alpha\right| \leq b_k - a_k = \frac{b_0 - a_0}{2^k}.$$

$$\longrightarrow \ |\text{errore}| \leq \lim_{k\to+\infty} \frac{b_0 - a_0}{2^k} = 0.$$

$$\longrightarrow \ \left\{x^{(k)}\right\}, \quad \lim_{k\to+\infty} x^{(k)} = \alpha.$$

Abbiamo costruito un processo che genera una successione che converge ad α.

A priori, fissata una tolleranza, possiamo avere una stima del numero di iterazioni necessarie.

$$|\text{errore}| \leq \frac{b_0 - a_0}{2^k}.$$

Vorremmo ottenere $|\text{errore}| < \varepsilon$.

$$\longrightarrow \ \frac{b_0 - a_0}{2^k} < \varepsilon \implies \frac{b_0 - a_0}{\varepsilon} < 2^k.$$

$$\longrightarrow \ k > \log_2\left(\frac{b_0 - a_0}{\varepsilon}\right).$$

$$\longrightarrow \ k > \frac{\log_{10}\left(\frac{b_0-a_0}{\varepsilon}\right)}{\log_2} \simeq \frac{\log_{10}\left(\frac{b_0-a_0}{\varepsilon}\right)}{0.6971}.$$

Costo complessivo:

è la somma del costo delle valutazioni di f.

Osservazione 10.6. Il metodo dicotomico si dice *globalmente* convergente, cioè non importa quanto sia ampio l'intervallo, né quale sia il punto iniziale da cui partiamo.

Con il metodo dicotomico troviamo una radice, siamo sicuri che converga, ma è una procedura "lenta". Sostituiamola con una più veloce: semplifichiamo il problema.

Supponiamo di voler conoscere la radice α_3. Partiamo da un $x^{(0)}$ arbitrario e controlliamo $f(x^{(0)})$. Sostituiamo localmente la funzione con una retta passante per $f(x^{(0)})$ e con una pendenza scelta da noi. Costruiamo il punto $x^{(1)}$ come intersezione della retta con l'asse delle x e calcoliamo $f(x^{(1)})$, quindi scegliamo nuovamente la pendenza e ricominciamo il procedimento.

Se lo facciamo sensatamente, non è certo, ma probabile che ci stiamo avvicinando ad α_3

Osservazione 10.7. La scelta di $x^{(0)}$ deve essere opportuna, infatti se non dovesse appartenere all'intervallo il processo si bloccherà poiché non sarà possibile calcolare $f(x^{(0)})$.

Questo non si può quindi definire un metodo *globale*.

Metodo di Newton: o delle rette tangenti, è un metodo di approssimazione che utilizza le tangenti, che vedremo più avanti.

Come potremmo prendere $x^{(0)}$ in modo da utilizzare il metodo delle tangenti?

f' di segno fissato.

f'' di segno fissato.

Se $f'(x_0) \cdot f''(x_0) > 0$ allora il processo converge.

Tuttavia abbiamo convergenze locali.

Osservazione 10.8. (Sul metodo dicotomico)

Dato un punto medio (j-esimo passo), che legame c'è tra $|x_k - \alpha|$ e $\frac{|x_j - \alpha|}{10}$?

$$\frac{b_0 - a_0}{2^k} = \frac{b_0 - a_0}{2^j} \cdot \frac{1}{10}.$$

$$\longrightarrow\ 10 \cdot 2^{j-k} = 1 \implies \log_2 \frac{1}{10} = j - k.$$
$$\longrightarrow\ k - j = \log_2 10 \simeq 3.32.$$

Ovvero tra 3 e 4 iterazioni sistemiamo una cifra decimale.

$$f(\alpha) = f(x) + f'(\xi)(\alpha - x), \quad |\alpha - \xi| \leq |\alpha - x|.$$

$$0 = f(x_n) + f'(\xi_n)(\alpha - x_n).$$
$$0 = f(x_n) + (x_{n+1} - x_n)q_n.$$

$$x_{n+1} = x_n - \frac{f(x_n)}{q_n}.$$

q_n approssima $f'(\xi_n)$.

10.2.2 Metodo delle corde.

Il metodo consiste nel costruire una retta dal coefficiente angolare q (indipendente dal numero di intervalli di cui è composta la decomposizione), se a e b sono gli estremi dell'intervallo su cui stiamo cercando la soluzione, q viene definito come:

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

La retta di coefficiente angolare q e passante per $f(a)$ e $f(b)$ si dice *corda*.

Per trovare x_1 a questo punto si deve fare l'intersezione con l'asse delle x, si calcola ora $f(x_1)$ per costruire la nuova corda passante per $f(x_1)$ e di coefficiente angolare sempre q (le corde sono tutte parallele). Si itera il procedimento per trovare $x_2, \ldots, x_n$.

$$x_{n+1} = x_n - f(x_n)\frac{b - a}{f(b) - f(a)}.$$

Osservazione 10.9. Anche questo procedimento ha una "falla", se nel calcolo di x_i questo esce dal nostro intervallo di partenza non riusciremmo a calcolarne l'immagine attraverso f, interrompendo così la procedura.

10.2.3 Metodo delle tangenti o di Newton.

$$q_n = f'(x_n).$$

$$\begin{cases} x_0 \quad \text{dato.} \\ x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad n = 0, \dots \end{cases}$$

Anche con questo metodo possiamo avere fallimenti. Il solito problema è quello di aver scelto un x_0 che non appartenga all'intervallo oppure di ottenere un x_k non appartenente all'intervallo ad un k-esimo passo. Questa situazione non è neanche prevedibile.

Quindi la convergenza è assicurata *se e solo se* x_0 sarà sufficientemente vicino alla radice.

Teorema 10.10. *(di convergenza di Newton)*
Sia $f(x) \in \mathcal{C}^2(I)$, dove I è un intervallo contenente una radice semplice α dell'equazione $f(x) = 0$. Allora preso un punto x_0 "sufficientemente" vicino ad α la successione generata dal metodo di Newton (delle tangenti) converge ad α, cioè:

$$\lim_{n \to +\infty} x_n = \alpha,$$

e inoltre:

$$\lim_{n \to +\infty} \frac{\alpha - x_{n+1}}{(\alpha - x_n)^2} = -\frac{f''(\alpha)}{2f'(\alpha)}$$

Ovvero α è semplice se e solo se $f(\alpha) = 0 \wedge f'(\alpha) \neq 0$.

Definizione 10.11. Sia $\{x_n\}$ una successione generata da un metodo iterativo, $\lim_{n \to +\infty} x_n = \alpha$ tale che:

$$\exists\, c > 0 \colon \forall n \ \lim_{n \to +\infty} \frac{|\alpha - x_{n+1}|}{|\alpha - x_n|} = c,$$

cioé:

$$\lim_{n \to +\infty} \frac{|e_{n+1}|}{|e_n|} = c,$$

un metodo iterativo si dice di ordine p se la successione $\{x_n\}$ generata ha convergenza di ordine p, ovvero:

$$\exists\, c > 0, \exists\, p \geq 1 \colon \forall n \ \lim_{n \to +\infty} \frac{|e_{n+1}|}{|e_n|^p} = c,$$

Proposizione 10.12. *Se $f''(\alpha) \neq 0$ allora,*

$$\lim_{n \to +\infty} \frac{|e_{n+1}|}{|e_n|} = \frac{|f''(\alpha)|}{2|f'(\alpha)|},$$

allora il metodo di Newton per radici semplici è del secondo ordine.

Dimostrazione. Sia $\varepsilon > 0$ sufficientemente piccolo, siano I^* e M tali che:

$$I^* := [\alpha - \varepsilon, \alpha + \varepsilon],$$
$$M = \frac{\max_{x \in I^*} |f''(x)|}{\max_{x \in I^*} |f'(x)|}.$$

Vediamo lo sviluppo di Taylor di f:

$$f(x) = f(x_n) + f'(x_n)(x - x_n) + f''(\xi)\frac{(x - x_n)^2}{2}.$$

Posto $x \equiv \alpha$ la radice di cui stiamo parlando,

$$f(\alpha) = f(x_n) + f'(x_n)(\alpha - x_n) + f''(\xi)\frac{(\alpha - x_n)^2}{2} = 0.$$

Esiste un intorno in cui f' sia diverso da 0.

$$\begin{aligned}
0 = f(\alpha) &= f(x_n) + f'(x_n)(\alpha - x_n) + f''(\xi)\tfrac{(\alpha - x_n)^2}{2} \\
&= \frac{1}{f'(x_n)}\left(f(x_n) + f'(x_n)(\alpha - x_n) + f''(\xi)\frac{(\alpha - x_n)^2}{2} \right) \\
&= \frac{f(x_n)}{f'(x_n)} + \frac{f'(x_n)}{f'(x_n)}(\alpha - x_n) + \frac{f''(\xi)}{2f'(x_n)}(\alpha - x_n)^2 \\
&= \underbrace{\frac{f(x_n)}{f'(x_n)} - x_n}_{(*)\ -x_{n+1}} + \alpha + \frac{f''(\xi)}{2f'(x_n)}(\alpha - x_n)^2 \\
&= \alpha - x_{n+1} + \frac{f''(\xi)}{2f'(x_n)}(\alpha - x_n)^2.
\end{aligned}$$

La $(*)$ deriva dalla relazione: $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$.

$$0 = \alpha - x_{n+1} + \frac{f''(\xi)}{2f'(x_n)}(\alpha - x_n)^2.$$

$$\begin{aligned}
|\alpha - x_{n+1}| &= \frac{f''(\xi)}{2f'(x_n)}(\alpha - x_n)^2 \leq M|\alpha - x_n|^2 \\
&\leq M|\alpha - x_n|^2 \\
&= \frac{1}{M}(M|\alpha - x_n|)^2.
\end{aligned}$$

Passando alla distanza tra α e il punto x_n si ha:

$$|\alpha - x_n| \leq M|\alpha - x_{n-1}|^2.$$

$$|\alpha - x_{n+1}| \leq \frac{1}{M} \left(MM|\alpha - x_{n-1}|\right)^2 = \frac{1}{M} \left(M|\alpha - x_{n-1}|\right)^{2^2}.$$

Valutando la relazione con tutti i punti intermedi, si raggiunge a concludere la seguente per x_0:

$$|\alpha - x_{n+1}| \leq \frac{1}{M} \left(M|\alpha - x_0|\right)^{2^{n+1}}.$$

Proposizione 10.13. *Se $x_0 \in I^*$ tale che $M|\alpha - x_0| < 1$ allora:*

$$\lim_{n \to +\infty} |\alpha - x_{n+1}| = 0.$$

Può succedere che durante le iterazioni esca dall'intervallo I^*?

$$|\alpha - x_1| \leq M|\alpha - x_0|^2 = \underbrace{M|\alpha - x_0|}_{<1} |\alpha - x_0| \leq |\alpha - x_0|.$$

Quindi se $x_0 \in I^*$ allora $x_1 \in I^*$.
x_1 soddisfa le condizioni? E' vera la condizione per ogni x_n nella successione?

Proposizione 10.14. *Per ogni $x_n \in I^*$ appartente alla successione si ha che:*

$$M|\alpha - x_n| < 1.$$

Dimostrazione. Si è già dimostrato per $n = 1$ (caso base), proseguiamo per induzione:

Ipotesi: $x_n \in I^*, \quad M|\alpha - x_n| < 1.$

Tesi: $x_{n+1} \in I^*, \quad M|\alpha - x_{n+1}| < 1.$

$$|\alpha - x_{n+1}| \leq M|\alpha - x_n|^2 = \underbrace{M|\alpha - x_n|}_{<1} |\alpha - x_n| < |\alpha - x_n|.$$

Allora x_{n+1} appartiene a I^*.

$$M|\alpha - x_{n+1}| \leq M^2|\alpha - x_n| < 1.$$

In conclusione se $x_0 \in I^*$ tale che $M|\alpha - x_0| < 1$, allora:

$$\{x_n\}_{\text{Newton}} \colon x_n \in I^* \quad M|\alpha - x_n| < 1.$$

■

E' importante che ciascuno dei punti mantenga le proprietà del punto x_0, perchè ciascuno degli x_n è un punto nuovo per il processo di Newton.

$$f(x) = f(x_n) + f'(x_n)(x - x_n) + f''(\xi)\frac{(x - x_n)^2}{2}.$$

$$f(\alpha) = f(x_n) + f'(x_n)(\alpha - x_n) + f''(\xi)\frac{(\alpha - x_n)^2}{2}.$$

$$\longrightarrow\ 0 = \frac{f(x_n)}{f'(x_n)} - x_n + \alpha + \frac{f''(\xi)}{2f'(x_n)}(\alpha - x_n)^2.$$

$$\longrightarrow\ 0 = \alpha - x_{n+1} + \frac{f''(\xi)}{2f'(x_n)}$$

$$-(\alpha + x_{n+1}) = (\alpha - x_n)^2\frac{f''(\xi)}{2f'(x_n)}.$$

$$\frac{(\alpha - x_{n+1})}{(\alpha - x_n)^2} = -\frac{f''(\xi)}{2f'(x_n)}.$$

$$\lim_{n\to+\infty} -\frac{f''(\xi)}{2f'(x_n)} = \lim_{n\to+\infty} \frac{(\alpha - x_{n+1})}{(\alpha - x_n)^2} = \frac{f''(\alpha)}{2f'(\alpha)} \qquad \substack{\alpha<\xi<x_n \\ |\alpha-\xi|<|\alpha-x_n|}.$$

■

Esercizio. $x^2 = 0$.

Applichiamo il teorema di Newton, attenzione però che non è una situazione contemplata dal teorema, perchè $f'(0) = 0$. Non è garantita la convergenza al second'ordine.

$$f(x) \in \mathcal{C}^2([a, b]),\ \alpha \text{ radice doppia.}$$
$$f(\alpha) = 0,\ f'(\alpha) = 0,\ f''(\alpha) \neq 0.$$

Verificare che il metodo di Newton perde in ordine di convergenza.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad e_n = \alpha - x_n.$$

$$\underbrace{\alpha - x_{n+1}}_{e_{n+1}} = \alpha - x_n + \frac{f(x_n)}{f'(x_n)}$$

Poiché $x_n = \alpha - e_n$ si ha:

$$\begin{aligned}
\alpha - x_{n+1} &= (\alpha - x_n) + \frac{f(\alpha - e_n)}{f'(\alpha - e_n)} \\
&= (\alpha - x_n) + \frac{\cancelto{0}{f(\alpha)} + \cancelto{0}{f'(\alpha)}(\alpha - e_n - \alpha) + f''(\xi_1)\frac{(\alpha - e_n - \alpha)^2}{2}}{\cancelto{0}{f'(x)} + f''(\xi_2)(\alpha - e_n \alpha)} \\
&= (\alpha - x_n) - \frac{f''(\xi_1)\frac{e_n^2}{2}}{f''(\xi_2)e_n}.
\end{aligned}$$

$$e_{n+1} = e_n - \frac{f''(\xi_1)\frac{e_n^{\not 2}}{2}}{f''(\xi_2)\cancel{e_n}}.$$

$$e_{n+1} = e_n \left(1 - \frac{1}{2}\frac{f''(\xi_1)}{f''(\xi_2)}\right).$$

$$\longrightarrow \frac{e_{n+1}}{e_n} = 1 - \frac{1}{2}\frac{f''(\xi_1)}{f''(\xi_2)}.$$

Poiché $f''(\xi_1)$ e $f''(\xi_2)$ vanno a coincidere per n che tende a $+\infty$, passando ai limiti:

$$\lim_{n \to +\infty} \frac{e_{n+1}}{e_n} = \frac{1}{2}.$$

Questo ci dice che il processo è *lineare*.

Quindi il problema è trovare x_0 adeguato a costruire una successione convergente. Ci sono situazioni in cui possiamo affermare la globale convergenza nel metodo di Newton (ovvero non dobbiamo preoccuparci di come prendere x_0)?

- Funzioni monotone crescenti convesse:

  $$f(x) \in \mathcal{C}^2([a,b]), \quad f(a) \cdot f(b) < 0.$$

  $\text{sign}(f'(x)) = \text{cost} > 0.$
  $\text{sign}(f''(x)) = \text{cost} > 0.$

  La scelta adeguata per x_0 è quella in cui l'immagine tramite f è di segno concorde con f'', in questo caso $f > 0$, ovvero per $x_0 \in [\alpha, b]$.

  In $[a, \alpha]$ non è assicurata la convergenza.

- Funzioni monotone crescenti concave:

  $$f(x) \in \mathcal{C}^2([a,b]), \quad f(a) \cdot f(b) < 0.$$

$\text{sign}(f'(x)) = \text{cost} > 0.$
$\text{sign}(f''(x)) = \text{cost} < 0.$

In quest'altro caso invece la convergenza è assicurata per $f < 0$, ovvero per $x_0 \in [a, \alpha]$ e non lo sarà quindi per $x_0 \in [\alpha, b]$.

- Funzioni monotone decrescenti concave:

$$f(x) \in \mathcal{C}^2([a,b]), \quad f(a) \cdot f(b) < 0.$$

$\text{sign}(f'(x)) = \text{cost} < 0.$
$\text{sign}(f''(x)) = \text{cost} < 0.$

Convergenza garantita per $f < 0$, ovvero per $x_0 \in [\alpha, b]$.

- Funzioni monotone decrescenti convesse:

$$f(x) \in \mathcal{C}^2([a,b]), \quad f(a) \cdot f(b) < 0.$$

$\text{sign}(f'(x)) = \text{cost} < 0.$
$\text{sign}(f''(x)) = \text{cost} > 0.$

Convergenza garantita per $f > 0$, ovvero per $x_0 \in [a, \alpha]$.

In conclusione la convergenza è garantita se vale la seguente proposizione:

Proposizione 10.15. *Sia $x_0 \in [a,b]$, $f \in \mathcal{C}^2([a,b])$ tale che:*

$$f(x_0) \cdot f''(x_0) > 0,$$

allora la successione $\{x_n\}$ Newtoniana converge ad α, ovvero

$$\lim_{n \to +\infty} x_n = \alpha.$$

Dimostrazione. (Nel caso di funzioni monotone crescenti convesse)

Sia $x_0 \in [a,b]$ tale che $f(x_0) \cdot f''(x_0) > 0$, sia $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$.

Tesi: $x_1 \in [\alpha, b]$.

$$f' \wedge f > 0 \ \longrightarrow \ \frac{f}{f'} > 0 \ \longrightarrow \ x_1 < x_0.$$

Dobbiamo dimostrare che $x_1 \geq \alpha$.

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + f''(\xi)\frac{(x - x_0)^2}{2}.$$

$$f(\alpha) = f(x_0) + f'(x_0)(\alpha - x_0) + f''(\xi)\frac{(\alpha - x_0)^2}{2} = 0.$$

$$f' \neq 0 \longrightarrow \underbrace{\frac{f(x_0)}{f'(x_0)} - x_0}_{-x_1} + \alpha + \frac{f''(\xi)}{f'(x_0)}\frac{(\alpha - x_0)^2}{2}$$

$$\longrightarrow 0 = \alpha - x_1 + \underbrace{\frac{f''(\xi)}{f'(\xi)}\frac{(\alpha - x_0)^2}{2}}_{>0} \longrightarrow \alpha - x_1 < 0.$$

$$\alpha < x_1.$$

■

Ora si deve dimostrare nel caso generico x_n, x_{n+1}.

Osservazione 10.16. La successione $\{x_n\}$ converge ad α in modo *decrescente*, ovvero ad ogni passo ci avviciniamo allo zero. Questo miglioramento non è garantito con il metodo dicotomico (non è monotono).

Osservazione 10.17. Con il metodo di Newton, al caso delle funzioni monotone crescenti convesse, abbiamo un'approssimazione sempre per eccesso, con il metodo delle corde invece l'approssimazione è per difetto.

Combinando i due metodi otteniamo un intervallo di appartenenza. Questo crea una condizione di stop al processo analoga a quella del processo dicotomico (quando l'intervallo è $< \delta$).

Come scegliere quando terminare il procedimento?
Facciamo un test sulle iterazioni: quando due iterazioni successive sono vicine al limite.

Problema: Se siamo in un caso in cui la convergenza sia lenta le iterazioni possono somigliarsi già dall'inizio.

10.2.4 Metodo delle secanti. (cenno)

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

Consideriamo i punti:

$$\begin{matrix} x^{(k-1)} & x^{(k)} \\ f(x^{(k-1)}) & f(x^{(k)}) \end{matrix}$$

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

$$\begin{cases} x^{(k+1)} = x^{(k)} - f(x^{(k)}) \cdot \frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})}, & k = 0, 1, \dots \\ x^{-1},\ x^0 \quad \text{dati.} \end{cases}$$

Nel caso in cui le immagini attraverso f di $x^{(0)}$ e $x^{(-1)}$ fossero "vicine" si potrebbe uscire dall'intervallo di partenza, inoltre una tale situazione potrebbe verificarsi anche durante le iterazioni. La convergenza non è garantita.

10.2.5 Regula Falsi. (cenno)

E' una variante del metodo delle secanti, però qui è assicurata la convergenza.

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

Deve accadere che:

$$f(x^{(k)}) \cdot f(x^{(k')}) < 0. \tag{10.1}$$

Tale vincolo si impone nella scelta dei punti.

Se vale la 10.1 allora il punto $x^{(1)}$, ottenuto come intersezione della secante con l'asse delle x, è interno all'intervallo $(x^{(0)}, x^{(-1)})$. In questo modo la convergenza è garantita.

Ora come punto, insieme ad $x^{(1)}$, prendiamo quello di indice maggiore tale che $f(x^{(1)}) \cdot f(x^{(k)}) < 0$.

Teorema 10.18. *Sia $f(x) \in \mathcal{C}^2(I)$, dove I è un intervallo contenente la radice α e $f''(\alpha) \neq 0$. Allora presi due punti $x^{(-1)}$ e $x^{(0)}$ sufficientemente vicini alla radice α, la successione generata dal metodo delle secanti converge ad α con ordine p:*

$$p = \frac{\left(1 + \sqrt{5}\right)}{2} \simeq 1.63.$$

10.2.6 Metodo delle tangenti per cercare le radici.

$$f(x) \equiv p(x) = a_0 x^n + a_1 x^{n-1} + \cdots + a_{n-2} x^2 + a_{n-1} x + a_n.$$
$$a_i \in \mathbb{R},\ a_0 \neq 0, \quad i = 0, \ldots, n.$$

Tangenti $p(x) = 0$.

$$x^{(0)} \text{ dato}, \quad x^{(i+1)} = x^{(i)} - \frac{p(x^{(i)})}{p'(x^{(i)})}, \quad i = 0, 1, \ldots$$

Il teorema ci dice che converge almeno linearmente. Se la radice è semplice converge in modo quadratico.

- $d_0 \equiv a_0$.
  - $k = 1, \ldots, n$.
    $d_k = d_{k-1} x + a_k$.
  $d_n = p(x)$.
- $d_0 \equiv a_0$.
  - $k = 1, \ldots, n$.
    $d_k = d_{k-1} x^{(i)} + a_k$.
  $d_n = p(x^{(i)})$.

$p(x) = (x - x^{(i)}) \cdot q(x) + r$.
Il resto è una costante perchè dividiamo per un polinomio lineare.

$p(x^{(i)}) = r$: lo utilizziamo per calcolare il denominatore.

$$p(x) = \left(x - x^{(i)}\right) \cdot q(x) + p\left(x^{(i)}\right).$$

$$\longrightarrow \quad q(x)|_{x=x^{(i)}} = \left.\frac{p(x) - p\left(x^{(i)}\right)}{x - x^{(i)}}\right|_{x=x^{(i)}} = p'(x^{(i)}).$$

$q(x)$ coincide con $p'(x^{(i)})$ (solo nel nodo $x^{(i)}$), quindi possiamo applicare il metodo anche senza conoscere la derivata.

Sviluppo di Taylor.

$$p(x) = p\left(x^{(i)}\right) + p'\left(x^{(i)}\right)\left(x - x^{(i)}\right) + p''\left(x^{(i)}\right)\frac{\left(x - x^{(i)}\right)^2}{2} + \cdots + p^{(n)}\left(x^{(i)}\right)\frac{\left(x - x^{(i)}\right)^n}{n!}.$$

$$\left.\frac{p(x) - p\left(x^{(i)}\right)}{\left(x - x^{(i)}\right)}\right|_{x=x^{(i)}} = \left.\left(p'\left(x^{(i)}\right) + \frac{p''\left(x^{(i)}\right)\left(x - x^{(i)}\right)}{2!} + \cdots + p^{(n)}\left(x^{(i)}\right)\frac{\left(x - x^{(i)}\right)^{n-1}}{n!}\right)\right|_{x=x^{(i)}}$$
$$= p'(x^{(i)}).$$

$$\begin{aligned} p'(x^{(i)}) &= a_0x^n + a_1x^{n-1} + \cdots + a_{n-2}x^2 + a_{n-1}x + a_n \\ &= \left(x - x^{(i)}\right)\left(b_0x^{n-1} + b_1x^{n-2} + \cdots + b_{n-2}x^1 + b_{n-1}\right) + r \\ &= b_0x^n + \cdots + b_{n-1}x - b_0x^{(i)}x^{n-1} - \cdots - b_{n-1}x^{(i)} + r \\ &= b_0x^n + x^{n-1}\left(b_1 - b_0x^{(i)}\right) + \cdots + x\left(b_{n-1} - b_{n-2}x^{(i)}\right) + r - b_{n-1}x^{(i)}. \end{aligned}$$

Sfruttando il principio di identità dei polinomni si ha:

$$\longrightarrow \qquad \begin{aligned} a_0 &= b_0 \\ a_1 &= b_1 - b_0x^{(i)} \\ a_2 &= b_2 - b_1x^{(i)} \\ &\vdots \\ a_k &= b_k - b_{k-1}x^{(i)} \\ &\vdots \\ a_n &= b_n - b_{n-1}x^{(i)} \end{aligned}$$

$b_0 \equiv a_0$.

- $k = 1, \ldots, n-1$.
  $b_k = b_{k-1}x^{(i)} + a_k$.

$r = b_{n-1}x^{(i)} + a_n$.

Ora abbiamo $q\left(x^{(i)}\right)$ e quindi anche $p'\left(x^{(i)}\right)$.

Teorema 10.19. *Sia $p(x)$ un polinomio di grado n a coefficienti reali con zeri tutti reali.*

$$\xi_n \leq \xi_{n-1} \leq \cdots \leq \xi_2 \leq \xi_1,$$

allora preso un punto $x^{(0)} > \xi_1$, la successione generata dal metodo di Newton converge in modo monotono *alla radice ξ_1.*

Osservazione 10.20. Non ci sono ipotesi sulla molteplicità della radice.

Problema successivo: trovare tutte le radici.

Esempio. Sia J una matrice tridiagonale irriducibile tale che:

$$\alpha_i \in \mathbb{R}, \quad i = 1, \ldots, n; \qquad \beta_j \in \mathbb{R}, \quad j = 1, \ldots, n; \quad \beta_j \neq 0.$$

$$J = \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 \\ \beta_2 & \alpha_2 & \beta_3 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ 0 & \cdots & 0 & \beta_n & \alpha_n \end{bmatrix}.$$

Poiché irriducibile la matrice J non può essere spezzata a blocchi nella ricerca degli autovalori.

$$\longrightarrow \ \det(J - \lambda I) = 0,$$

presenta solo n autovalori reali. Inoltre l'irriducibilità implica che gli n autovalori sono *distinti*.

Come costruiamo un intervallo adeguato?

$$\|Jx\| = \|\lambda x\| \ \Rightarrow \ \|J\| \cdot \|x\| \geq |\lambda| \cdot \|x\|.$$

Nota Bene. Stiamo utilizzando una norma compatibile: $\|\cdot\|_1$ o $\|\cdot\|_\infty$, la $\|\cdot\|_2$ no perchè richiede di conoscere gli autovalori, che invece vogliamo trovare.

L'intervallo $[I - \|J\|_\infty, \|J\|_\infty]$ va bene poiché ha convergenza verso l'autovalore più grande per il teorema 10.19.

Come si costruirebbe l'algoritmo newtoniano? cioé come calcoliamo $p(x^{(i)})$ e $p'(x^{(i)})$?

$$J - \lambda I = \begin{bmatrix} \alpha_1 - \lambda & \beta_2 & 0 & \cdots & 0 \\ \beta_2 & \alpha_2 - \lambda & \beta_3 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \beta_{n-1} & \alpha_{n-1} - \lambda & \beta_n \\ 0 & \cdots & 0 & \beta_n & \alpha_n - \lambda \end{bmatrix},$$

$d_0 \equiv 1.$

$d_1 = \alpha_1 - \lambda.$

- $k = 2, \ldots, n.$

  $d_k = (\alpha_k - \lambda) d_{k-1} - \beta_k^2 d_{k-2}.$

$d_n \equiv \det(J - \lambda I) = p(\lambda).$

Se poniamo $\overline{\lambda} = x^{(i)}$ otteniamo:

$$\overline{d}_n = \det\left(J - \overline{\lambda} I\right),$$

il polinomio caratteristico valutato in $\overline{\lambda}$.

$$d_k'(\lambda) = -d_{k-1}(\lambda) + (\alpha - \lambda) d_{k-1}' - \beta_k^2 d_{k-2}'(\lambda),$$

valutato in $\overline{\lambda} = x^{(i)}$ otteniamo il valore di $p'(x^{(i)})$.

Dimostrazione. (Teorema 10.19)

Ipotesi: $a_0 > 0$.

In $[\xi_1, \xi_2]$ la derivata prima ammette una radice α_1 (teorema di Rolle), se $\xi_2 \equiv \xi_1$ allora ξ_1 ha molteplicità 2 ed è comunque una radice del polinomio derivato.

Analogamente otteniamo $\alpha_2 \in [\xi_2, \xi_3], \ldots, \alpha_{n-1} \in [\xi_{n-1}, \xi_n]$, $n-1$ radici del polinomio derivato.

$$x > \alpha_1 \;\Rightarrow\; p'(x) > 0.$$

$p''(x)$?

In $[\alpha_1, \alpha_2]$ il polinomio ha uno zero (teorema di Rolle), quindi p'' ha coefficiente massimo maggiore di 0.

$$x > \alpha_1 \;\Rightarrow\; p''(x) > 0.$$

$p'''(x)$? Discorso analogo.

$$x \geq \alpha_1 \;\Rightarrow\; p'''(x) \geq 0.$$

Allora p e p' sono funzioni convesse per $x > \alpha_1$.

Consideriamo $x^{(0)}$ a destra di ξ_1. Costruiamo $x^{(1)}$ col metodo di Newton.

$$x^{(1)} = x^{(0)} - \frac{p(x^{(0)})}{p'(x^{(0)})} \tag{10.2}$$

$$p(x) = p\left(x^{(0)}\right) + p'\left(x^{(0)}\right)\left(x - x^{(0)}\right) + p''(c_0)\left(\frac{x - x^{(0)}}{2}\right)^2.$$

$$0 = p(\xi_1) = p\left(x^{(0)}\right) + p'\left(x^{(0)}\right)\left(\xi_1 - x^{(0)}\right) + \underbrace{p''(c_0)\left(\frac{\xi_1 - x^{(0)}}{2}\right)^2}_{>0}.$$

$$0 > p\left(x^{(0)}\right) + p'\left(x^{(0)}\right)\left(\xi_1 - x^{(0)}\right).$$

$$\left(x^{(0)} - x^{(1)}\right) p'\left(x^{(0)}\right) = p\left(x^{(0)}\right).$$

Da 10.2 si ha:

$$0 > p'\left(x^{(0)}\right)\left(\cancel{x^{(0)}} - x^{(1)} + \xi_1 - \cancel{x^{(0)}}\right) = \underbrace{p'\left(x^{(0)}\right)}_{>0}\left(\xi_1 - x^{(1)}\right).$$

$$\longrightarrow \xi_1 - x^{(1)} < 0 \;\Rightarrow\; x^{(1)} > \xi_1.$$

Si completa, analogamente, la dimostrazione per induzione.

Ipotesi: $\xi_1 < x^{(k)} < x^{(k-1)}$.

Tesi: $\xi_1 < x^{(k+1)} < x^{(k)}$.

■

Il teorema permette di calcolare anche la radice più piccola grazie alla convergenza *monotona* dall'altro lato. Ovvero in questo caso si parte a sinistra dell'intervallo da $[\xi_{n-1}, \xi_n]$.

Lo spettro di autovalori è molto ampio si hanno quindi dei problemi numerici nel calcolo. Nel caso generale come si calcola l'intervallo a cui appartengono tutte le radici del polinomio?

Esperimento Matlab.

Prendiamo un polinomio a coefficienti reali e ne calcoliamo le radici reali. Quindi cambiamo un coefficiente, ricalcoliamo le radici e scopriamo che non sono più reali.

Esempio. Minkinson.

$$p(x) = (x-1)\cdot(x-2)\cdots(x-20).$$

Modifica: a_{19} diventa 10^{-10}.

Otteniamo radici complesse coniugate, 5 o 6 coppie.

Teorema 10.21. *Per le radici ξ_i, con $i = 1, \ldots, n$, del polinomio p tale che:*

$$p(x) = a_0 x^n + \cdots + a_n,$$

si ha la seguente limitazione:

$$|\xi_i| \leq \max\left\{\frac{|a_n|}{|a_0|}, 1 + \frac{|a_{n-1}|}{|a_0|}, \ldots, 1 + \frac{|a_1|}{|a_0|}\right\},$$

$$|\xi_i| \leq \max\left\{1, \sum_{i=1}^{n} \frac{|a_i|}{|a_0|}\right\}.$$

Tecnica defrattiva.

Per calcolare ξ_2, una volta trovata ξ_1 poniamo:

$$p_2(x) = \frac{p(x)}{(x - \xi_1)},$$

a questo punto riapplichiamo il procedimento. Analogamente per il caso inverso (da ξ_n).

Che errore commettiamo facendo la divisione? ξ_1 è calcolata con la massima precisione macchina, però viene approssimata. Ne segue che i coefficienti di p_2 siano affetti da errore poiché ξ_1 è leggermente diversa da quella effettiva. Anche in questo caso ξ_2 sarà un'approssimazione. Etc.

Il problema è dunque la divisione, è possibile calcolare ξ_2 utilizzando i coefficienti del polinomio di partenza? (senza fare la divisione).

$$p(x) = a_0 x^n + \cdots + a_n.$$

$$\xi_1 \geq \xi_2 \geq \cdots \geq \xi_{n-1} \geq \xi_n.$$

$$x^{(0)} > \xi_1, \quad \lim_{k \to +\infty} x^{(k)} = \xi_1, \quad x^{(0)} < \xi_n.$$

$$\begin{cases} x^{(k+1)} = x^{(k)} - \frac{p\left(x^{(k)}\right)}{p'\left(x^{(k)}\right)}, & k = 0, 1, \ldots \\ \\ x^{(0)} \text{ dato.} \end{cases}$$

$$p_1(x) = \frac{p(x)}{x - \xi_1}.$$

$$p_1'(x) = \frac{p'(x)(x - \xi_1) - p(x)}{(x - \xi_1)^2}.$$

$$\begin{cases} x^{(k+1)} = x^{(k)} - \frac{p_1\left(x^{(k)}\right)}{p_1'\left(x^{(k)}\right)}, & k = 0, 1, \ldots \\ \\ x^{(0)} \text{ dato.} \end{cases}$$

$$x^{(k+1)} = x^{(k)} - \frac{\frac{p\left(x^{(k)}\right)}{\cancel{x^{(k)} - \xi_1}}}{\frac{p_1'\left(x^{(k)}\right)\left(x^{(k)} - \xi_1\right) - p\left(x^{(k)}\right)}{\left(x^{(k)} - \xi_1\right)^{\cancel{2}}}}.$$

$$x^{(k+1)} = x^{(k)} - \frac{p(x^{(k)})}{p'(x^{(k)}) - p(x^{(k)})\frac{1}{x^{(k)}-\xi_1}}.$$

Supponiamo di avere $\xi_1, \xi_2, \ldots, \xi_j$ note.

$$p_j(x) = \frac{p(x)}{(x-\xi_1)\cdot(x-\xi_2)\cdots(x-\xi_j)}.$$

$$\begin{cases} x^{(k+1)} = x^{(k)} - \frac{p_j\left(x^{(k)}\right)}{p_j'\left(x^{(k)}\right)}, \quad k = 0, 1, \ldots \\ \\ x^{(0)} \text{ dato.} \end{cases}$$

$$p_j'(x) = \frac{p'(x)(x-\xi_1)\cdot(x-\xi_2)\cdots(x-\xi_j) - p(x)\cdot\sum_{l=1}^{j}\prod_{\substack{i=1\\ i\neq l}}^{j}(x-\xi_i)}{\left((x-\xi_1)\cdot(x-\xi_2)\cdots(x-\xi_j)\right)^2}.$$

Moltiplicando e dividendo per $(x-\xi_l)$ si ha:

$$p_j'(x) = \frac{p'(x)(x-\xi_1)\cdot(x-\xi_2)\cdots(x-\xi_j) - p(x)\cdot\prod_{i=1}^{j}(x-\xi_i)\sum_{l=1}^{j}\frac{1}{(x-\xi_l)}}{\left((x-\xi_1)\cdot(x-\xi_2)\cdots(x-\xi_j)\right)^2}.$$

Segue che:

$$x^{(k+1)} = x^{(k)} - \frac{\frac{p(x^{(k)})}{(x^{(k)}-\xi_1)\cdot\cancel{(x^{(k)}-\xi_2)\cdots(x^{(k)}-\xi_j)}}}{\frac{p'(x^{(k)})\cancel{(x^{(k)}-\xi_1)\cdot(x^{(k)}-\xi_2)\cdots(x^{(k)}-\xi_j)} - p(x^{(k)})\cdot\prod_{i=1}^{j}(x^{(k)}-\xi_i)\sum_{l=1}^{j}\frac{1}{(x^{(k)}-\xi_l)}}{\left((x^{(k)}-\xi_1)\cdot(x^{(k)}-\xi_2)\cdots(x^{(k)}-\xi_j)\right)^{\cancel{2}}}}.$$

$$\longrightarrow x^{(k+1)} = x^{(k)} - \frac{p(x^{(k)})}{p'(x^{(k)}) - p(x^{(k)})\sum_{l=1}^{j}\frac{1}{(x^{(k)}-\xi_l)}}$$

Costruiamo così un nuovo polinomio utilizzando sempre $p(x)$.

10.3 Criteri d'arresto.

Quale può essere un criterio di arresto per i processi iterativi? Introduciamo la ε tolleranza.

10.3.1 Controllo del resto.

Se siamo abbastanza vicini alla radice α, la seguente disequazione è verificata?

$$f\left(x^{(k)}\right) < \varepsilon$$

Se $x^{(k)}$ è in un intorno della radice il test sul residuo è soddisfacente, però la funzione potrebbe essere approssimata a zero anche in un punto $x^{(k)}$ lontano dalla radice e il test interromperebbe la computazione in modo errato.

Allo stesso modo se la pendenza della tangente alla curva nel punto in questione è molto elevata, otterremmo quindi una f molto grande ed eseguiremmo iterazioni in eccesso.

Sia α una radice di f, ovvero $f(\alpha) = 0$.

$$f\left(x^{(k)}\right) - f(\alpha) = f'(c_n)\left(x^{(k)} - \alpha\right).$$

c_n punto opportuno, dividendo per $f'(c_n)$ si ha:

$$\alpha - x^{(k)} = -\frac{f\left(x^{(k)}\right)}{f'(c_n)}.$$

$$|\alpha - x^{(k)}| = \frac{|f\left(x^{(k)}\right)|}{|f'(c_n)|} \simeq \frac{|f\left(x^{(k)}\right)|}{|f'(\alpha)|} \leq \frac{\varepsilon}{|f'(\alpha)|}.$$

Se $|f'(\alpha)| \simeq 1$ allora $|\alpha - x^{(k)}| \leq \varepsilon$.

Se $|f'(\alpha)| \simeq 0$ allora $|\alpha - x^{(k)}|$ può essere grande.

10.3.2 Differenza tra due iterate successive.

Esaminiamo la differenza tra due iterazioni successive, ovvero in un dato $x^{(k)}$ e $x^{(k+1)}$.

$$|x^{(k+1)} - x^{(k)}| < \varepsilon.$$

Normalmente non è esaustivo, quando la convergenza di successione è lenta probabilmente questo metodo "fa acqua". E' invece valido quando l'iterata (è di ordine maggiore di 1) ha una convergenza più che lineare ("muovendosi" velocemente $|x^{(k+1)} - x^{(k)}|$ è piccola).

Per il metodo di Newton ad esempio come arriverei a scrivere qualcosa?

$$x^{(k+1)} - x^{(k)} = -\frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)} \simeq -\frac{f\left(x^{(k)}\right)}{f'\left(c_n\right)} = \alpha - x^{(k)}.$$

Se facciamo il test $\varepsilon > |x^{(k+1)} - x^{(k)}| \simeq |\alpha - x^{(k)}|$.

Stiamo lavorando nelle vicinanze del punto α, occorre che il processo sia di ordine superiore ad 1.

10.3.3 Teorema di Sturm.

Consideriamo un polinomio $p(x) = a_0x^n + a_1x^{n-1} + \cdots + a_n$, con $a_0 \neq 0$. Definiamo una sequenza di Sturm (di polinomi)

$$p(x) \equiv p_0(x), p_1(x), p_2(x), \ldots, p_n(x);$$

che soddisfino le seguenti condizioni:

1. $p_0(x)$ ha tutte le radici semplici;
2. $\operatorname{sign} p_1(\xi) = -\operatorname{sign} p_0'(\xi)$ per ogni zero reale di $p_0(x)$;
3. $i = 1, 2, \ldots, n-1$

   $p_{i-1}(\xi) \cdot p_{i+1}(\xi) < 0$, se ξ è uno zero reale di $p_i(\xi)$;
4. il polinomio $p_n(x)$ non cambia di segno.

A cosa serve questa successione?

Teorema 10.22. *Sia $p_0(x), p_1(x), p_2(x), \ldots, p_n(x)$ una sequenza di Sturm, allora il numero degli zeri reali di $p_0(x)$ che cadono in $[a, b)$ è dato dalla differenza $w(b) - w(a)$, dove $w(x)$ indica il numero dei cambiamenti di segno della sequenza nel punto x.*

Esempio. Esempio numerico:
$p(x) = 13x^6 - 364x^5 + 2912x^4 - 9984x^3 + 166640x^2 - 13312x + 4096$.
$p_0(x) = p(x)$.
$p_1(x) = -26(3x^5 - 70x^4 + 448x^3 - 1152x^2 + 1280x - 50)$.
$p_2(x) = \frac{4}{9}(1001x^4 - 9152x^3 + 27456x^2 - 33280x + 14080)$.
$p_3(x) = -\frac{2304}{539}(134x^3 - 702x^2 + 1080x - 528)$.
$p_4(x) = \frac{784}{99}(51x^2 - 136x + 88)$.
$p_5(x) = -\frac{4094}{833}(19x - 22)$.
$p_6(x) = \frac{3136}{361}$.

Come si costruisce? Si parte dal polinomio assegnato e si costruisce $p_1(x)$:

$$p_1(x) = -p_0'(x).$$

$$p_0(x) = p_1(x)q_1(x) + r_1(x), \quad p_2(x) = r_1(x).$$

$$\longrightarrow p_0(x) = p_1(x)q_1(x) + q_2(x).$$

$$p_1(x) = p_2(x)q_2(x) + r_2(x), \quad p_3(x) = r_2(x).$$
$$p_2(x) = p_3(x)q_3(x) + r_3(x), \quad p_4(x) = r_3(x).$$
$$\vdots$$

Si trova così il massimo comun divisore tra $p(x)$ e $p'(x)$.

x	$p_0(x)$	$p_1(x)$	$p_2(x)$	$p_3(x)$	$p_4(x)$	$p_5(x)$	$p_6(x)$	$w(x)$
0	+	+	+	+	+	+	+	$0 = w(0)$
2	−	0	+	+	+	−	+	3

Alla seconda riga (per $x = 2$) possiamo dare il "valore" che vogliamo, a seconda se calcoliamo la derivata destra o sinistra.

Poiché $w(2) - w(0) = 3$ si ha che nell'intervallo $[0, 2]$ il polinomio ha 3 radici. Passando al limite ($x \to +\infty$) si possono trovare gli zeri totali.

Parte II

Calcolo Numerico.
Prof.ssa Aimi.

Capitolo 11

Introduzione

Si consideri il problema dello studio integrale. Sia $f \in C^0([a,b])$, allora:

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

Ma se non si conosce la primitva F?

$$\underbrace{\sum_{i=0}^{n} w_i f(x_i)}_{calcolo} + \underbrace{R_n(f)}_{errore}.$$

Si ha in genere un problema reale ed occorre quindi trasformarlo in un problema matematico tramite la fase di modellizzazione (semplifcazione) del problema. A questo punto dal problema matematico (ben posto?) si passa al problema numerico mediante la fase di discretizzazione, qui verrà introdotto un *errore analitico*. Infine si giunge alla soluzione *approssimata* con algoritmi dell'Analisi Numerica utilizzando il calcolatore, anche in questa fase vengono introdotti errori di calcolo dovuti alla finitezza della precisione di rappresentazione.

11.1 Errore analitico o di discretizzazione

Sia $f \in C^0([a,b])$, vorrei valutare f' in un punto x^* senza l'espressione della derivata $f'(x)$.

Dallo sviluppo di Taylor di f centrato in x^* relativo all'incremento h $(0 < h \leq 1)$ si ha:

$$f(x^* + h) = f(x^*) + hf'(x^*) + \frac{h^2}{2} f''(\psi_{x^*}), \qquad x^* < \psi_{x^*} < x^* + h.$$

Si ottiene quindi:

$$f'(x^*) = \underbrace{\frac{f(x^*+h)-f(x^*)}{h}}_{\text{calcolabile}} - \underbrace{\frac{h}{2}f''(\psi_{x^*})}_{\text{non calcolabile}} .$$

Tramite la discretizzazione si può approssimare quindi:

$$f'(x^*) \simeq \frac{f(x^*+h)-f(x^*)}{h}.$$

Costo dell'algoritmo.

Valutazioni di f: 2.

Somme algebriche: 2.

Divisioni: 1.

Il vantaggio è che non si deve calcolare f'. L'errore analitico E_1 si esprime come:

$$E_1(f, x^*, h) := -\frac{h}{2}f''(\psi_{x^*}) = \mathrm{O}(h).$$

Sempre dallo sviluppo di Taylor di f si ottiene:

$$f'(x^*) = \frac{f(x^*+h)-f(x^*-h)}{2h} - \frac{h^3}{6}f'''(\eta_{x^*}), \quad x^*-h < \eta_{x^*} < x^*+h.$$

Sviluppo di Taylor in x^*, incremento h, con resto di ordine tre:

$$f(x^*+h) = f(x^*) + hf'(x^*) + \frac{h^2}{2}f''(x^*) + \frac{h^3}{6}f'''(\eta_{x^*}), \qquad x^* < \eta_{x^*} < x^*+h.$$

Sviluppo di Taylor in x^*, decremento h, con resto di ordine tre:

$$f(x^*-h) = f(x^*) - hf'(x^*) + \frac{h^2}{2}f''(x^*) - \frac{h^3}{6}f'''(\eta_{x^*}), \qquad x^*-h < \eta_{x^*} < x^*.$$

$$\begin{aligned} f(x^*+h) - f(x^*-h) &= 2hf'(x^*) + \tfrac{h^3}{6}\left(f'''(\eta_{x^*}) + f'''(\eta_{x^*})\right) \\ &= 2hf'(x^*) + \tfrac{h^3}{6}\left(2f'''(\eta_{x^*})\right). \end{aligned}$$

$$f'(x^*) = \frac{f(x^*+h)-f(x^*-h) - \frac{h^3}{3}f'''(\eta_{x^*})}{2h}.$$

$$f'(x^*) = \underbrace{\frac{f(x^*+h)-f(x^*-h)}{2h}}_{\text{calcolabile}} - \underbrace{\frac{h^3}{6}f'''(\eta_{x^*})}_{\text{non calcolabile}} \qquad x^*-h < \eta_{x^*} < x^*+h.$$

Tramite la discretizzazione si può approssimare quindi:

$$f'(x^*) \simeq \frac{f(x^* + h) - f(x^*)}{2h}.$$

Costo dell'algoritmo.

Valutazioni di f: 2.

Somme algebriche: 3.

Divisioni: 1.

$$E_2(f, x^*, h) := -\frac{h^2}{6} f'''(\eta_{x^*}) = \mathrm{O}(h^2).$$

Il costo di questo algoritmo è all'incirca uguale al primo, la differenza tra i due è l'errore analitico. Il primo ha un errore dell'ordine di 10^{-2} mentre il secondo di 10^{-4}. Di contro il primo algoritmo ha meno ipotesi su f.

11.1.1 Problema Test o campione.

Un Test è un problema di cui si conosce la soluzione in forma chiusa, che viene utilizzato per testare le proprietà matematiche scritte sulla carta.

Esempio. $f(x) = x^2, \qquad x^* = 1, \qquad f'(1) = 2.$

$E_1 = (f, x^*, h) = -\frac{h}{2} f''(\psi_{x^*}) = -\frac{h}{2} 2 = -h.$

$E_2(f, x^*, h) = -\frac{h^2}{6} f'''(\eta_{x^*}) = \frac{h^2}{6} 0 = 0.$

Meglio la rappresentazione in scala logaritmica.

Esercizio. Calcolo numerico del limite:

$$\lim_{x \to +\infty} [x(\sqrt{x^2 + 1} - x)].$$

$$\lim_{x \to +\infty} [x(\sqrt{x^2 + 1} - x)] \cdot \frac{\sqrt{x^2 + 1} + x}{\sqrt{x^2 + 1} + x} = \lim_{x \to +\infty} \frac{x(\cancel{x^2} + 1 \cancel{- x^2})}{\sqrt{x^2 + x} + x} = \frac{1}{2}.$$

```
>> % x = array che contiene 10^1, 10^2, ..., 10^15
>> x = logspace(1,15,15);
>>
>> % creo un vettore di dimensione 15 (come x) i cui elementi hanno
>> % il valore della funzione calcolata in x
>> f1 = x.*(sqrt(x.^2 + 1)-x);
>>
>> % stessa funzione, calcolata usando la proprietà distributiva
>> f2 = x.*sqrt(x.^2+1)-x.^2;
>>
>> % f moltiplicata e divisa per sqrt(x^2+1) + x
>> f3 = x./(sqrt(x.^2 + 1) +x);
>>
>> % Creo un vettore di output y (matrice)
>> y = [x; f1; f2; f3];
>>
>> % funzione di stampa
>> fprintf('%.3e %18.16f %18.16f %18.16f \n',y)
1.000e+01 0.4987562112088995 0.4987562112088995 0.4987562112089027
1.000e+02 0.4999875006248544 0.4999875006251386 0.4999875006249609
1.000e+03 0.4999998750463419 0.4999998750863597 0.4999998750000625
1.000e+04 0.5000000055588316 0.5000000000000000 0.4999999987500000
1.000e+05 0.4999994416721165 0.5000000000000000 0.4999999999875001
1.000e+06 0.5000038072466850 0.5000000000000000 0.4999999999998750
1.000e+07 0.5029141902923584 0.5000000000000000 0.4999999999999987
1.000e+08 0.0000000000000000 0.0000000000000000 0.5000000000000000
1.000e+09 0.0000000000000000 0.0000000000000000 0.5000000000000000
1.000e+10 0.0000000000000000 0.0000000000000000 0.5000000000000000
1.000e+11 0.0000000000000000 0.0000000000000000 0.5000000000000000
1.000e+12 0.0000000000000000 0.0000000000000000 0.5000000000000000
1.000e+13 0.0000000000000000 0.0000000000000000 0.5000000000000000
1.000e+14 0.0000000000000000 0.0000000000000000 0.5000000000000000
1.000e+15 0.0000000000000000 0.0000000000000000 0.5000000000000000
>>
```

11.2 Errori di rappresentazione.

11.2.1 Rappresentazione in virgola mobile.

Si consideri la rappresentazione dei numeri reali nel sistema decimale:

$$\begin{aligned} x &= \underbrace{327.52}_{\text{mantissa}} \\ &= 3 \cdot 10^2 + 2 \cdot 10^1 + 7 \cdot 10^0 + 5 \cdot 10^{-1} + 2 \cdot 10^{-2}. \end{aligned}$$

Oppure una rappresentazione differente, sempre nel sistema decimale:

$$\begin{aligned} x &= \underbrace{0.32752}_{\text{nuova mantissa}} \\ &= 3 \cdot 10^{-1} + 2 \cdot 10^{-2} + 7 \cdot 10^{-3} + 5 \cdot 10^{-4} + 2 \cdot 10^{-5}. \end{aligned}$$

$$\begin{aligned} x &= 00.32752 \\ &= 3 \cdot 10^{-2} + 2 \cdot 10^{-3} + 7 \cdot 10^{-4} + 5 \cdot 10^{-5} + 2 \cdot 10^{-6}. \end{aligned}$$

Possiamo fissare una rappresentazione univoca di x con una convenzione:

$$x = \text{sgn}(x) \cdot m \cdot 10^p,$$

con m mantissa ($0.1 \leq m < 1$) e p peso.

Esempio. $x = 327.52 \quad m = 0.32752, \quad p = 3.$

Teorema 11.1. *Sia* $x \in \mathbb{R} \setminus \{0\}$ *e* $\beta \in \mathbb{N}, \beta \geq 2$. *Allora* $\exists!$ $p \in \mathbb{Z}$ *e* $\{d_i\}_{i=1}^{\infty}$, $d_i \in \mathbb{N}$ *tale che:*

$$\begin{array}{rcl} x & = & \text{sgn}(x) \cdot \sum_{i=1}^{\infty} d_i \beta^{-i} \cdot \beta^p \\ & = & \mp(.d_1 d_2 \cdots) \cdot \beta^p. \end{array}$$

Con $0 \leq d_i < \beta$, $d_1 \neq 0 e$ $d_i \neq \beta - 1$.

Definizione 11.2. Rappresentazione in virgola mobile normalizzata.

- β: base.
- p: caratteristica.
- d_i: cifra.
- $m := \sum_{i=1}^{\infty} d_i \beta^{-i}$: mantissa.

11.2.2 Algoritmi per il cambio di base.

Sia x un numero intero.

1. $i = 1 \quad c_i = x.$
2. $d_i = \text{mod}(c_i, \beta).$
3. $c_{i+1} = \text{int}(\frac{c_i}{\beta}).$
4. se $c_{i+1} = 0$ stop.
5. $i = i + 1$ e torna al punto 2.

Sia x tale che $0 < x < 1$.

1. $i = 1 \quad c_i = x.$
2. $d_i = \text{int}(c_i \beta).$
3. $c_{i+1} = c_i \beta - d_i.$

4. se $c_{i+1} = 0$ stop.

5. $i = i + 1$ e torna al punto 2.

Esercizio. $x = (0.1)_{10}$, effettuare il cambio di base con $\beta = 2$ e $\beta = 16$.

$\beta = 2$.
$c_1 = 0.1$
$d_1 = \text{int}(c_i\beta)$

i	1	2	3	4	5	6	...
c_i	0.1	0.2	0.4	0.8	0.6	0.2	...
$c_i\beta$	0.2	$\underline{0.4}$	0.8	1.6	1.2	$\underline{0.4}$	...
d_i	0	0	0	1	1	0	...

$(.0\overline{00110}\cdots)_2$.

Come si può notare $d_1 = 0$, quindi per ricondurci alla nostra convenzione occorre moltiplicare tale mantissa per 2^3 ovvero aumenttarne il peso.

Osservazione 11.3. Si può notare che in base 10 il numero in questo esempio ha una rappresentazione esatta, mentre passando alla base binaria occorrerà troncare la mantissa (poiché periodica), in una certo punto. Questo introduce un errore di rappresentazione.

$\beta = 16$.
$c_1 = 0.1$
$d_1 = \text{int}(c_i\beta)$

i	1	2	3	...
c_i	0.1	0.6	0.6	...
$c_i\beta$	1.6	$\underline{9.6}$	$\underline{9.6}$	...
d_i	1	9	9	...

$(.199\overline{99}\cdots)_{16}$.

Oppure dato che $16 = 2^4$ si ha che:
$(.\underbrace{0001}_{1\cdot 2^0}\ \overline{\underbrace{1001}_{1\cdot 2^0+1\cdot 2^3}\ \underbrace{1001}_{1\cdot 2^0+1\cdot 2^3}}\ \cdots)_2$.

Esercizio. $x = (026)_{10}$, effettuare il cambio di base con $\beta = 2$ e $\beta = 16$.

26 : 2	=	13	Resto:	0	0	
13 : 2	=	6	Resto:	1	1	
6 : 2	=	3	Resto:	0	0	
3 : 2	=	1	Resto:	1	1	
1 : 2	=	0	Resto:	1	1	$\uparrow$ dall'ultimo al primo

$(11010)_2$.

$$\begin{array}{rcrrr|c|l} 26:16 & = & 1 & \text{Resto: } 10 & A & \\ 1:16 & = & 0 & \text{Resto: } 1 & 1 & \uparrow \text{ dall'ultimo al primo} \end{array}$$

$(1A)_{16}$.

Oppure dato che $16 = 2^4$ si ha che:
$(1\underbrace{1010}_{(A)_{16}})_2$

Capitolo 12

Sistema Floating Point.

12.1 Definizione di un sistema Floating Point.

Definizione 12.1. (Sistema Floating Point) Siano β, t, m ed M interi positivi tale che: $\beta \geq 2, \quad t \geq 1, \quad m, n > 0$. L'insieme dei numeri macchina in base β con t cifre significative è l'insieme:

$$F(\beta, t, m, M) = \{0\} \cup \left\{ x \in \mathbb{R} \colon x = \operatorname{sgn}(x) \cdot \sum_{i=1}^{t} d_i \beta^{-i} \cdot \beta^p \right\},$$

tale che:

- $0 \leq d_i < \beta, \qquad i = 1, \ldots, t.$
- $d_1 \neq 0.$
- $-m \leq p \leq M.$

Osservazione 12.2. Se x appartiene ad un formato floating point può essere espresso mediante un finito numero di cifre.

$$x \in F(\beta, t, m, M),\ x \neq 0 \ \Rightarrow\ x = \pm(.\underbrace{d_1 \cdots d_t}_{\text{finite}})_\beta \cdot \beta^p.$$

Esempio. Prendiamo il seguente sistema floating point e vediamo alcuni esempi:

$F(2, 3, 1, 1)$

$$\beta = 2, \quad t = 3, \quad m = 1, \quad M = 1,$$

$d_1 = 1$, poiché non può essere uguale a zero.

$(.100)_2 = \frac{1}{2}$;

$(.101)_2 = \frac{1}{2} + \frac{1}{8} = \frac{5}{8}$;

$(.110)_2 = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}$;

$(.111)_2 = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} = \frac{7}{8}$.

Queste di cui sopra sono le mantisse del sistema, le possibili caratteristiche sono -1, 0 e 1 poiché $-1 \leq p \leq 1$.

$$F(2,3,1,1) = \left\{0, \frac{1}{4}, \pm\frac{9}{16}, \pm\frac{3}{8}, \pm\frac{7}{16}, \pm\frac{1}{2}, \pm\frac{9}{8}, \pm\frac{3}{4}, \pm\frac{7}{8}, \pm 9, \pm\frac{9}{4}, \pm\frac{3}{2}, \pm\frac{7}{4}\right\}.$$

I numeri sono equispaziati fra la successione potenze della base, ma non in tutto il range.

12.2 Proprietà.

Sia $F(\beta, t, m, M)$ un sistema floating point, allora valgono le seguenti proprietà:

1. La cardinalità di $F(\beta, t, m, M)$ è esprimibile come:

   $$2 \cdot (m + M + 1) \cdot (\beta - 1) \cdot \beta^{t-1} + 1.$$

   2 è dato dal segno, ogni numero può essere negativo o positivo (tutti i numeri nel formato hanno opposto). $(m + M + 1)$ è il numero dei valori che può assumere il peso p. $(\beta - 1) \cdot \beta^{t-1}$ è il numero dei valori che può assumere la mantissa. 1 è dato dalla rappresentazione di 0.

2. I numeri macchina sono equispaziati fra le successive potenze di β ma non in tutto il range.

3. Il più piccolo numero rappresentabile è:

   $$(.1\,0\,\cdots\,0)_\beta \cdot \beta^{-m} = \beta^{-m-1}.$$

   (`realmin`)

4. Il più grande numero rappresentabile è:

   $$\underbrace{(.\beta - 1\,\beta - 1\,\cdots\,\beta - 1)_\beta}_{1-\beta^{-t}} \cdot \beta^M \backsimeq {}^*\beta^M.$$

   (`realmax`)

[*] t molto grande.

12.3 Il sistema Floating Point usato da Matlab.

Definizione 12.3. Sia un sistema Floating Point tale che:

- $\beta = 2$;
- posti $p \in \mathbb{Z},\ d_0 = 1,\ d_i = \texttt{bit},\ (i \geq 1)$ e d_i definitivamente diverso da 1,

$$x \in \mathbb{R} \setminus \{0\} \ \Rightarrow\ x = \pm \sum_{i=0}^{+\infty} d_i 2^{-i} \cdot 2^p.$$

si dice che utilizza una rappresentazione *normalizzata*. Ovvero x si può così definire:

$$\begin{array}{rcll} x & = & \pm \left(1 + \sum_{i=1}^{+\infty} d_i 2^{-i}\right) \cdot 2^p & \\ & = & \pm \left(1.d_1\, d_2\, \dots\right) \cdot 2^p & \\ & = & \pm (1+f) \cdot 2^p & \text{con } 0 \leq f < 1. \end{array}$$

Un formato che utilizza una forma normalizzata si denota con F^*.

Esempio. $F^*(2,t,m,M) = \{0\} \cup \left\{x \in \mathbb{R}\colon\ x = \pm(1 + \sum_{i=1}^{t} d_i 2^{-i}) \cdot 2p\right\}$, con $d_i = \texttt{bit},\ 1 \leq i \leq t$ e $-m \leq p \leq M$.

Alcuni possibili formati sono:

Singola precisione: $F^*(2, 23, 128, 127)$;

Doppia precisione: $F^*(2, 52, 1024, 1023)$.

Matlab utilizza un sistema Floating Point conforme allo standard IEEE-754 a doppia precisione, ovvero del tipo $F^*(2, 52, 1024, 1023)$, ogni numero occupa 64 bit (8 byte).

12.4 Numeri macchina e rappresentazione dei reali.

L'insieme $F^*(2, 52, 1024, 1023)$, detto insieme dei numeri macchina, è finito, questo implica che dato un numero reale $x \in \mathbb{R}$, questo possa appartenere o meno all'insieme dei numeri macchina.

Se tale x appartiene all'insieme dei numeri macchina non ci sono problemi (saremmo comunque molto fortunati), la norma è che x non vi appertenga. Che errore commettiamo rappresentando $x \in \mathbb{R}$ tale che $x \notin F$?

Sia $x \in \mathbb{R}^+,\ x \notin F(\beta, t, m, M)$, allora:

$$x = \beta^p \sum_{i=1}^{\infty} d_i \beta^{-i}.$$

Problema: come si associa in modo adeguato ad x un numero di macchina $\tilde{x} \in F(\beta, t, m, M)$?

Distinguiamo due casi:

a) $p \notin [-m, M]$

   1. $p < -m \longrightarrow$ UNDERFLOW $\qquad \tilde{x} = 0$;
   2. $p > M \longrightarrow$ OVERFLOW $\qquad \nexists\, \tilde{x}$, in questo caso o si arresta il calcolo oppure si imposta $\tilde{x} =$ `Inf`.

b) $p \in [-m, M]$, ma $\exists i > t \colon d_i \neq 0$

   1. Troncamento di x alla t-esima cifra:

$$\tilde{x} = \mathrm{trn}(x) = \beta^p \sum_{i=1}^{t} d_i \beta^{-i}.$$

   2. Arrotondamento di x alla t-esima cifra:

$$\tilde{x} = \mathrm{arr}(x) = \beta^p \mathrm{trn}\left(\sum_{i=1}^{t+1} d_i \beta^{-i} + \frac{\beta}{2} \beta^{-t-1} \right).$$

Possiamo notare che se $d_{i+1} < \frac{\beta}{2}$ allora $\mathrm{arr}(x) = \mathrm{trn}(x)$, se invece $d_{i+1} > \frac{\beta}{2}$ si ha che $\mathrm{arr}(x) = \mathrm{trn}(x) + \beta^{p-t}$.

Nota Bene. Nell'effetuare l'arrotondamento può verificarsi overflow.

Esempio. $F(10, 3, 5, 5) \qquad x = 98273.$

$$\mathrm{trn}(x) = (.982) \cdot 10^5.$$

$$\mathrm{arr}(x) = (.983) \cdot 10^5.$$

Esempio. $F(10, 3, 5, 5) \qquad x = 99960.$

$$\mathrm{trn}(x) = (.999) \cdot 10^5.$$

$$\mathrm{arr}(x) = \cancel{(.1) \cdot 10^6} \qquad \text{overflow.}$$

Nota Bene. Abbiamo dimostrato un teorema, un numero macchina a dista dal successivo b per β^{p-t}.

12.5 Errori di rappresentazione.

Definizione 12.4. Siano $x \in \mathbb{R}$, $\tilde{x} \in F(\beta, t, m, M)$, si definiscono l'*errore assoluto* ed *errore relativo* come segue:

- $\tilde{x} - x$, errore assoluto.
- $\frac{\tilde{x}-x}{x}$, errore relativo.

Teorema 12.5. *Sia $x \in \mathbb{R}$, $\tilde{x} \in F(\beta, t, m, M)$, allora:*

$|\tilde{x} - x| < \beta^{p-t}$ *se* $\tilde{x} = trn(x)$.

$|\tilde{x} - x| \leq \frac{1}{2}\beta^{p-t}$ *se* $\tilde{x} = arr(x)$.

Definizione 12.6. Siano $x \in \mathbb{R}$, $\tilde{x} \in F(\beta, t, m, M)$, si dice *epsilon macchina* il numero `eps` dato da:

$$\texttt{eps} = \begin{cases} \beta^{1-t}, & \text{se } \tilde{x} = \text{trn}(x) \\ \frac{1}{2}\beta^{1-t} & \text{se } \tilde{x} = \text{arr}(x). \end{cases}$$

Osservazione 12.7. `eps` è independente da x.

Teorema 12.8. *Sia $x \in \mathbb{R}$, $\tilde{x} \in F(\beta, t, m, M)$, se $x \neq 0$ allora:*

$$\left|\frac{\tilde{x} - x}{x}\right| < \texttt{eps}.$$

Dimostrazione. $|\frac{\tilde{x}-x}{x}| \overset{?}{<} \texttt{eps}$.

$$\begin{array}{rcll} |x| & \geq & \beta^p d_1 \beta^{-1} & (\text{poiché } d_1 \neq 0) \\ & \geq & \beta^{p-1}. & (\text{poiché } d_1 \geq 1) \end{array}$$

Passando ai reciproci:

$$\frac{1}{|x|} \leq \frac{1}{\beta^{p-1}}.$$

Abbiamo quindi due casi:

Caso 1: $\tilde{x} = \text{trn}(x)$

$$\left|\frac{\tilde{x} - x}{x}\right| \leq \left|\frac{\tilde{x} - x}{\beta^{p-1}}\right| < \frac{\beta^{p-t}}{\beta^{p-1}} = \beta^{1-t}.$$

Caso 2: $\tilde{x} = \text{arr}(x)$

$$\left|\frac{\tilde{x} - x}{x}\right| \leq \left|\frac{\tilde{x} - x}{\beta^{p-1}}\right| \leq \frac{1}{2}\frac{\beta^{p-t}}{\beta^{p-1}} = \frac{1}{2}\beta^{1-t}.$$

■

Corollario 12.9. *Sia ε l'errore relativo di rappresentazione:*

$$\varepsilon = \left| \frac{\tilde{x} - x}{x} \right|,$$

allora il numero floating point $\tilde{x}$ è rappresentabile formalmente come:

$$\tilde{x} = x(1 + \varepsilon).$$

Dove $|\varepsilon| <$ `eps` *per il teorema precedente.*

Osservazione 12.10. Nel caso di underflow si ha un errore relativo del 100%:

$$\left| \frac{0 - x}{x} \right| = 1.$$

Osservazione 12.11. La caratterizzazione della precisione di macchina `eps` è il più piccolo numero positivo rappresentabile tale che:

$$(1 + \texttt{eps}) > 1.$$

Esempio. Per $\beta = 10$, $t = 8$ e con arrotondamento si ha:

$$\texttt{eps} = 5 \cdot 10^{-8} = \frac{1}{2}\beta^{-7}.$$

Allora $x = 1 + \texttt{eps} = 1.00000005 = 0.100000005 \cdot 10^1$. Questo non è un numero macchina poiché $d_{i+1} = 5 \neq 0$.

Pertanto $\tilde{x} = 0.10000001 \cdot 10^1 = 1.0000001 > 1$.

Vediamo un semplice programma Matlab per il calcolo di `eps`:

```
>> epsi = 1;
>> while epsi + 1 > 1
epsi = epsi*0.5;
end
>> epsi = epsi*2

epsi =

   2.2204e-16
```

12.6 Aritmetica di macchina (o aritmetica in virgola mobile)

Definizione 12.12. Siano $a, b \in F(\beta, t, m, M)$, data un'operazione $*$ sui numeri reali, un'operazione $\circledast$ su F è definita come segue:

$$a \circledast b := \text{arr}(a * b).$$

Il risultato può non essere un numero di macchina valido.

Osservazione 12.13. Per le operazioni sui numeri in virgola mobile non valgono le seguenti proprietà dei numeri reali:

1. Associtività della somma.
2. Associatività del prodotto.
3. Regola di cancellazione della somma.
4. Regola di cancellazione del prodotto.
5. Distributiva.
6. Semplificazione.

Esempio. Sia $F(10, 2, m, M)$ un dato formato Floating Point, allora:

1. $a = .11\beta^0,\ b = .13\beta^{-1},\ c = .14\beta^{-1}$:

$$(a \oplus b) \oplus c = .13\beta^0, \qquad a \oplus (b \oplus c) = .14\beta^0.$$

2. $a = .11\beta^1,\ b = .31\beta^1,\ c = .25\beta^1$:

$$(a \otimes b) \otimes c = .85\beta^1, \qquad a \otimes (b \otimes c) = .86\beta^1.$$

3. $a = .11\beta^0,\ b = .13\beta^{-1},\ c = .14\beta^{-1}$:

$$a \oplus b = .13\beta^0 = a \oplus c,\ a \neq 0 \ \wedge\ b \neq c.$$

4. $a = .51\beta^1,\ b = .22\beta^1,\ c = .52\beta^1$:

$$a \otimes b = .11\beta^2 = c \oplus b,\ b \neq 0 \ \wedge\ a \neq c.$$

5. $a = .11\beta^1,\ b = .23\beta^1,\ c = .24\beta^1$:

$$(a \otimes b) \oplus (a \otimes c) = .51\beta^1,$$

$$a \otimes (b \oplus c) = .52\beta^1.$$

6. $a = .70\beta^1,\ b = .10\beta^1$:

$$a \otimes (b \oslash a) = .98\beta^0 \neq b.$$

Osservazione 12.14. L'aritmetica di macchina soffre del *fenomeno di cancellazione numerica* (o perdita di cifre significative).

$a = 0.23371298 \cdot 10^{-4}$
$b = 0.33678429 \cdot 10^{2}$
$c = -0.33677811 \cdot 10^{2}$

In questo esempio b e c sono quasi uguali, ma di segno opposto.

I) $(a \oplus b) \oplus c = [0.33678452 \ominus 0.33677811] \cdot 10^2 = 0.64100000 \cdot 10^{-3}$.

II) $a \oplus (b \oplus c) = 0.23371298 \cdot 10^{-4} \oplus 0.61800000 \cdot 10^{-3} = 0.64137126 \cdot 10^{-3}$.

$a + b + c = 0.641371298 \cdot 10^{-3}$.

Capitolo 13

Condizionamento e stabilità algoritmica.

Definizione 13.1. Si dice *condizionamento* la risposta di un problema all'introduzione di errori nei dati, *a prescindere dall'algoritmo utilizzato.*

Definizione 13.2. Un problema si dice *ben condizionato* se a piccole perturbazioni sui dati corrispondono errori dello stesso ordine sui risultati; mal condizionato in caso contrario.

Definizione 13.3. Sia x un dato, $\tilde{x}$ tale dato affetto da errore, si dice *perturbazione* la quantità Δx tale che:

$$\Delta x = \tilde{x} - x.$$

Posti $y = f(x)$ e $\tilde{y} = f(\tilde{x})$ l'*errore sui risultati* si definisce come:

$$\Delta y = \tilde{y} - y.$$

$$\tilde{y} = f(\tilde{x}) = f(x + \Delta x) = f(x) + f'(x)\Delta x + \cdots$$
$$\underbrace{f(\tilde{x}) - f(x)}_{\Delta y} \cong f'(x)\Delta x.$$

Dove $\Delta y = Df(x)\Delta x$ è l'*errore assoluto.*

$$\frac{\Delta y}{y} = \frac{x}{f(x)} Df(x) \frac{\Delta x}{x}.$$

La quantità $\frac{\Delta y}{y}$ è l'errore relativo, mentre $\frac{x}{f(x)} Df(x)$ è l'*indice di condizionamento*, dipende dalla derivata di f dato x.

Esempio. $f(x) = \sqrt{1-x}$, con $x \lesssim 1$. L'indice di condizionamento in modulo è il seguente:

$$\left| \frac{x}{\sqrt{1-x}} \cdot \left(-\frac{1}{2\sqrt{1-x}} \right) \right| = \frac{|x|}{2(1-x)}.$$

x è prossimo al valore 1, il problema risulta quindi malcondizionato poiché più x si avvicina ad 1 più $\tilde{y} = f(\tilde{x})$ differisce da $f(x)$.

$$x = 0.9999,\ \tilde{x} = 0.9991,\ \frac{\Delta x}{x} = \frac{10^{-9}}{0.9999} = 1.0001 \cdot 10^{-5}.$$

$$\longrightarrow \frac{\Delta y}{y} = -0.513079 \cdot 10^{-1}.$$

13.1 Condizionamento delle funzioni in due variabili.

Siano x_1, x_2, y_1 e y_2 tali che:

$$y = f(x_1, x_2),\ \tilde{y} = f(\tilde{x_1}, \tilde{x_2}),$$

$$\tilde{x_1} = x_1 + \Delta x_1,\ \tilde{x_2} = x_2 + \Delta x_2.$$

Δx_1 e Δx_2 errori sui relativi dati.

$$\longrightarrow\ \Delta y := \tilde{y} - y = f(\tilde{x_1}, \tilde{x_2}) - f(x_1, x_2).$$

Dallo sviluppo di Taylor in due variabili si ha che:

$$\Delta y \simeq \frac{\partial f}{\partial x_1}(x_1, x_2)\Delta x_1 + \frac{\partial f}{\partial x_2}(x_1, x_2)\Delta x_2.$$

L'errore relativo è quindi esprimibile come:

$$\frac{\Delta y}{y} := \frac{\tilde{y} - y}{y} = \frac{\Delta y}{f(x_1, x_2)}.$$

$$\frac{\Delta y}{y} \simeq \underbrace{\frac{x_1}{f(x_1, x_2)} \frac{\partial f}{\partial x_1}(x_1, x_2)}_{\text{indice di condizionamento}} \frac{\Delta x_1}{x_1} + \frac{x_2}{f(x_1, x_2)} \frac{\partial f}{\partial x_2}(x_1, x_2) \frac{\Delta x_2}{x_2}.$$

Esempio. Osserviamo le seguenti funzioni ed i relativi indici di condizionamento.

$$f(x_1, x_2) = x_1 + x_2 \;\Rightarrow\; \frac{\Delta y}{y} \simeq \frac{x_1}{x_1 + x_2} \cdot \frac{\Delta x_1}{x_1} + \frac{x_2}{x_1 + x_2} \cdot \frac{\Delta x_2}{x_2}$$

L'indice di condizionamento in questo caso è $\frac{x_1}{x_1+x_2}$ e $\frac{x_2}{x_1+x_2}$ e il problema è mal condizionato.

$$f(x_1, x_2) = x_1 - x_2 \;\Rightarrow\; \frac{\Delta y}{y} \simeq \frac{x_1}{x_1 - x_2} \cdot \frac{\Delta x_1}{x_1} + \frac{x_2}{x_1 - x_2} \cdot \frac{\Delta x_2}{x_2}$$

Analogamente al primo esempio il problema risulta mal condizionato.

$$f(x_1, x_2) = x_1 \cdot x_2 \;\Rightarrow\; \frac{\Delta y}{y} \simeq 1 \cdot \frac{\Delta x_1}{x_1} + 1 \cdot \frac{\Delta x_2}{x_2}$$

In questo caso invece il problema è ben condizionato poichè l'indice di condizionamento è una costante (1).

$$f(x_1, x_2) = \frac{x_1}{x_2} \;\Rightarrow\; \frac{\Delta y}{y} \simeq 1 \cdot \frac{\Delta x_1}{x_1} + 1 \cdot \frac{\Delta x_2}{x_2}$$

$$f(x) = \sqrt{x} \;\Rightarrow\; \frac{\Delta y}{y} \simeq \frac{1}{2} \cdot \frac{\Delta x}{x}$$

13.2 Errore inerente ed errore algoritmico.

Dato un generico problema $y = f(x)$, se x è un dato esatto ma non è un valido numero di macchina (ovvero $x \notin F(\beta, t, m, M)$) occorre introdurre un dato perturbato $\tilde{x}$. Ad esempio:

$$\tilde{x} = \mathrm{fl}(x) = x(1 + \varepsilon) \quad |\varepsilon| < \texttt{eps}.$$

Con ε errore relativo di arrotondamento. In generale comunque, applicando la funzione f alla variabile $\tilde{x} \in F(\beta, t, m, M)$, non è detto che il risultato sia un numero di macchina, occorre quindi arrotondare anche questo:

$$\mathrm{fl}(f(\tilde{x})) = \mathrm{fl}(\tilde{y}) =: \overline{y}.$$

Quant'è l'errore complessivo?

$$\frac{\overline{y} - y}{y} = ?$$

$$\frac{\overline{y}-y}{y} := \frac{\text{fl}(f(\tilde{x}))-f(x)}{f(x)},$$

aggiungendo e togliendo $f(\tilde{x})$ al numeratore:

$$\frac{\overline{y}-y}{y} = \frac{\text{fl}(f(\tilde{x}))-f(\tilde{x})}{f(x)} + \frac{f(\tilde{x})-f(x)}{f(x)}$$

$$\frac{\overline{y}-y}{y} = \frac{\tilde{y}-y}{y} + \underbrace{\frac{\text{fl}(f(\tilde{x}))-f(\tilde{x})}{f(\tilde{x})}}_{\varepsilon_{\text{alg}}} \cdot \frac{f(\tilde{x})}{f(x)} = \varepsilon_{\text{in}} + \varepsilon_{\text{alg}}. \tag{13.1}$$

ε_{in} è l'*errore inerente* legato al condizionamento del problema, ε_{alg} è l'*errore algoritmico* dovuto agli errori di arrotondamento.

$$\varepsilon_{\text{in}} := \frac{f(\tilde{x})-f(x)}{f(x)} = \frac{\tilde{y}-y}{y}.$$

$$\varepsilon_{\text{in}} = \frac{f(\tilde{x})}{f(x)} - 1.$$

$$\longrightarrow \quad \frac{f(\tilde{x})}{f(x)} = 1 + \varepsilon_{\text{in}}.$$

Si ha inoltre che $\varepsilon_{\text{alg}}(1+\varepsilon_{\text{in}}) \simeq \varepsilon_{\text{alg}}$ perchè non teniamo conto degli errori del second'ordine ($\varepsilon_{\text{alg}} \cdot \varepsilon_{\text{in}} \simeq 10^{-16} \cdot 10^{-16} = 10^{-32}$ risulta una quantità troppo piccola, non ci interessa). Segue quindi l'uguaglianza dell'equazione 13.1.

Definizione 13.4. Un algoritmo si dice *numericamente stabile* se produce un'errore dello stesso ordine dell'errore inerente, altrimenti si dice *instabile*.

Esempio. $y = f(x_1, x_2) = x_1 + x_2; \quad x_1, x_2 \notin F(\beta, t, m, M)$.

x_1 è approssimato con $\tilde{x_1} \in F(\beta, t, m, M)$ tale che:

$$\tilde{x_1} = x_1 \cdot (1 + \varepsilon_1).$$

x_2 è approssimato con $\tilde{x_2} \in F(\beta, t, m, M)$ tale che:

$$\tilde{x_2} = x_2 \cdot (1 + \varepsilon_2).$$

In cui ε_1 e ε_2 sono errori relativi di rappresentazione commessi nell'approssimazione di x_1 e x_2 rispettivamente: $|\varepsilon_1| < \texttt{eps}$ e $|\varepsilon_2| < \texttt{eps}$.

In aritmetica di macchina:

$$\overline{y} = \tilde{x_1} \oplus \tilde{x_2} = \text{fl}(\tilde{x_1} + \tilde{x_2}) = (\tilde{x_1} + \tilde{x_2})(1 + \varepsilon).$$

Con ε errore relativo di arrotondamento sul risultato della somma.

Che errore complessivo viene commesso?

$$\begin{aligned}
\tfrac{\overline{y}-y}{y} &\overset{\text{def}}{=} \tfrac{1}{x_1+x_2}\left((\tilde{x_1}+\tilde{x_2})(1+\varepsilon)-(x_1+x_2)\right)\\
&= \tfrac{1}{x_1+x_2}\left((x_1\cdot(1+\varepsilon_1)+x_2\cdot(1+\varepsilon_2))(1+\varepsilon)-x_1-x_2\right)\\
&= \tfrac{1}{x_1+x_2}\left(\cancel{x_1}+x_1\varepsilon_1+\cancel{x_2}+x_2\varepsilon_2+x_1\varepsilon+x_1\varepsilon\varepsilon_1+x_2\varepsilon+x_2\varepsilon\varepsilon_2\right)\\
&\simeq \underbrace{\frac{x_1}{x_1+x_2}\cdot\varepsilon_1+\frac{x_2}{x_1+x_2}\cdot\varepsilon_2}_{\text{errore inerente: somma}}+\underbrace{\varepsilon}_{\text{errore algoritmico}}
\end{aligned}$$

$$\Rightarrow \frac{\overline{y}-y}{y}=\varepsilon_{\text{in}}+\varepsilon_{\text{alg}}.$$

Anche se ε_1 è piccolo, $\frac{x_1}{x_1+x_2}$ potrebbe "esplodere" se x_1 è circa uguale a x_2 ma di segno opposto. Questo ci dice che gli errori sui dati sono algoritmicamente trascinati.

In sintesi:

1. Si analizza il problema discreto: è ben condizionato?
   - NO (ε_{in} troppo grande): si cerca di riformulare il problema e si ritorna al punto 1.
   - SI (ε_{in} piccolo): si passa al punto 2.
2. Si produce un algoritmo numerico.
3. Si analizza l'algoritmo numerico: è stabile?
   - NO: si torna al punto 2.
   - SI: si passa al punto 4.
4. Si valutano gli algoritmi ottenuti fino ad ora, se è possibile crearne degli altri si torna al punto 2, altrimenti si passa al punto 5.
5. Tra tutti gli algoritmi numericamente stabili si sceglie il più efficiente (minor costo computazionale, minor impiego di memoria, etc).

13.3 Propagazione dell'errore.

Studio della propagazione dell'errore nel calcolo della funzione:

$$y=f(\underline{x})=\sum_{i=1}^{n}x_i,$$

dove $\underline{x}=(x_1,\ldots,x_n)$.

Condizionamento del problema:

sia $f(\underline{x}) \neq 0$,

$$\varepsilon_y \cong \frac{1}{f(\underline{x})} \cdot \sum_{i=1}^{n} x_i \varepsilon_{x_i}, \quad \text{coefficiente di approssimazione.}$$

In generale non possiamo stimare l'errore senza ulteriori ipotesi sui dati x_i: se tutti i dati sono dello stesso segno si ha

$$|\varepsilon_y| \simeq \left| \frac{1}{f(\underline{x})} \cdot \sum_{i=1}^{n} x_i \varepsilon_{x_i} \right| \leq \frac{1}{\cancel{|f(\underline{x})|}} \left(\sum_{i=1}^{n} \cancel{|x_i|} \right) \cdot \max_{i=1,\ldots,n} |\varepsilon_{x_i}|.$$

$$|\varepsilon_y| \simeq \max_{i=1,\ldots,n} |\varepsilon_{x_i}| \leq \texttt{eps} \;\Rightarrow\; \text{è ben condizionato.}$$

Se i dati fossero di segni non concordi il problema sarebbe malcondizionato.

13.3.1 Algoritmi per la funzione $\sum x$.

Analizziamo ora due algoritmi per calcolare la funzione:

$$y = f(\underline{x}) = \sum_{i=1}^{n} x_i, \quad \underline{x} = (x_1, \ldots, x_n).$$

Algoritmo 1.

Dati: $x_1, x_2, \ldots, x_n$. Possiamo supporli puliti perchè abbiamo già sistemato il condizionamento a priori.

Poniamo:
$z^{(1)} = x_1 + x_2$;
$z^{(i)} = z^{(i-1)} + x_{i+1}, \quad i = 2, \ldots, n-1$;
$f(\underline{x}) = z^{(n-1)}$.

Per n addendi l'errore complessivo finale sarà:

$$\varepsilon^{n-1} = \varepsilon^{(n-1)} + \underbrace{\frac{z^{(n-2)}}{z^{(n-1)}} \cdot \varepsilon^{n-2}}_{\text{errore trascinato dagli addendi}}.$$

Dove ε^{n-2} è l'errore complessivo che si è annidato sul penultimo nodo del grafo.

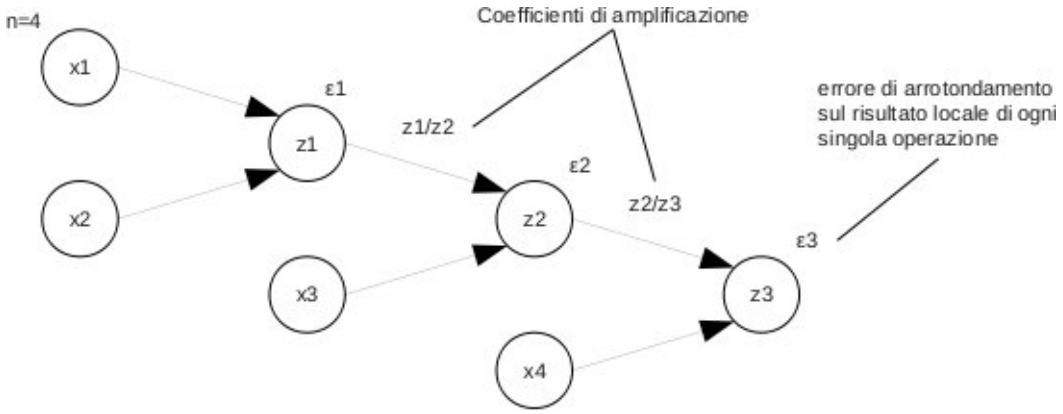


Figura 13.1: Algoritmo 1.

$$\begin{aligned}
\varepsilon^{n-1} &= \varepsilon^{(n-1)} + \frac{z^{(n-2)}}{z^{(n-1)}}\left(\varepsilon^{(n-2)} + \frac{z^{(n-3)}}{z^{(n-2)}}\varepsilon^{n-3}\right) \\
&= \varepsilon^{(n-1)} + \frac{z^{(n-2)}}{z^{(n-1)}}\varepsilon^{(n-2)} + \frac{z^{(n-3)}}{z^{(n-1)}}\left(\varepsilon^{(n-3)} + \frac{z^{(n-4)}}{z^{(n-3)}}\right) \\
&\vdots \\
&= \varepsilon^{(n-1)} + \frac{z^{(n-2)}}{z^{(n-1)}}\varepsilon^{(n-2)} + \frac{z^{(n-3)}}{z^{(n-1)}}\varepsilon^{(n-3)} + \cdots + \frac{z^{(1)}}{z^{(n-1)}}\varepsilon^{(1)} = \frac{1}{z^{(n-1)}}\sum_{i=1}^{n-1} z^{(1)}\varepsilon^{(i)}.
\end{aligned}$$

Allora:

$$\varepsilon_{\text{alg}} = \frac{1}{f(\underline{x})} \cdot \sum_{i=1}^{n-1}\left(\sum_{j=1}^{i+1} x_j\right) \cdot \varepsilon^{(i)}.$$

Osservazione 13.5. In generale *non è possibile* dare *limitazioni* per ε_{alg} che non dipendono dai dati.

Se $\{x_i\}$ sono tutti dello stesso segno, vale:

$$\left|\sum_{j=1}^{i+1} x_j\right| \leq |f(x)|, \quad i = 1, \ldots, n-1.$$

Allora:

$$|\varepsilon_{\text{alg}}| \leq \frac{1}{|f(\underline{x})|}\sum_{i=1}^{n-1} |f(x)|\left|\varepsilon^{(i)}\right| \leq (n-1)\cdot \texttt{eps} \qquad \text{crescita lineare.}$$

L'algoritmo è numericamente ben condizionato per n piccoli, altrimenti non lo è. In questo caso cosa possiamo fare?

Algoritmo 1.2.

Supponiamo di dare gli addendi in ordine di modulo non decrescente, la successione delle medie è ancora in modulo non decrescente:

$$\frac{1}{i}\left|\sum_{j=1}^{i} x_j\right| \leq \frac{1}{i+1}\left|\sum_{j=1}^{i+1} x_j\right| \leq \frac{1}{n}|f(\underline{x})|.$$

Allora:

$$|\varepsilon_{\text{alg}}| \leq \frac{1}{|f(\underline{x})|}\cdot\sum_{i=1}^{n-1}\left|\sum_{j=1}^{i+1} x_j\right|\cdot|\varepsilon^{(i)}| \leq \frac{1}{|f(\underline{x})|}\sum_{i=1}^{n-1}\frac{(i+1)f(\underline{x})}{n}\cdot \texttt{eps} =$$

$$= \frac{eps}{n}\sum_{i=1}^{n-1}(i+1) < \frac{n+1}{2}\cdot \texttt{eps}.$$

$$\longrightarrow\ |\varepsilon_{\text{alg}}| < \frac{n+1}{2}\cdot \texttt{eps} \qquad \text{crescita lineare.}$$

Osservazione 13.6. La stima è migliorata di un coefficiente $\frac{1}{2}$.

13.3.2 Algoritmo 2.

Addizioni in parallelo.

Sia $n = 2^p$, con p intero positivo.

Poniamo:
$v_j^{(0)} = x_j, \quad j = 1, \ldots, n;$
$v_j^{(i)} = v_{2j-1}^{(i-1)} + v_{2j}^{i-1}, \quad j = 1, \ldots, \frac{n}{2^i}; \quad i = i, \ldots, p;$
$f(\underline{x}) = v_1^{(p)}.$

Proposizione 13.7. *L'errore algoritmico è quantificabile come:*

$$\varepsilon_{alg} = \frac{1}{f(\underline{x})}\sum_{i=1}^{p}\sum_{j=1}^{\frac{n}{2^i}} v_j^{(i)}\eta_j^{(i)}.$$

Poiché, dati gli x_i dello stesso segno, si ha:

$$\sum_{j=1}^{\frac{n}{2^i}}\left|v_j^{(i)}\right| = |f(\underline{x})| \quad i = 0, \ldots, p$$

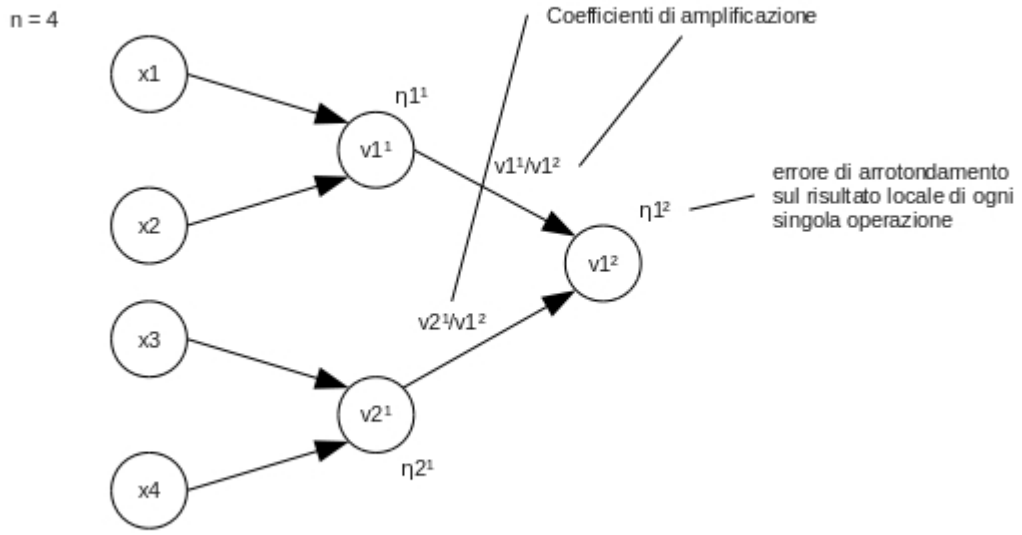


Figura 13.2: Algoritmo 2.

risulta quindi:

$$|\varepsilon_{alg}| < \max_{i,j} \left|\eta_j^{(i)}\right| \cdot \sum_{i=1}^{p} \sum_{j=1}^{\frac{n}{2^i}} \frac{\left|v_j^{(i)}\right|}{|f(\underline{x})|} = \max_{i,j} \left|v_j^{(i)}\right| \cdot p < \texttt{eps} \cdot \log_2 n.$$

$$|\varepsilon_{alg}| < \texttt{eps} \cdot \log_2 n.$$

Dimostrazione. Per $n = 2^k$ addendi l'errore algoritmico complessivo sarà:

$$\varepsilon_{\text{alg}} = \eta_1^k = \eta_1^{(k)} + \eta_1^{(k-1)} \cdot \frac{v_1^{(k-1)}}{v_1^{(k)}} + \eta_2^{(k-1)} \cdot \frac{v_2^{(k-1)}}{v_1^{(k)}}.$$

Ipotesi induttiva:

$$\eta_1^k = \frac{1}{v_1^{(k)}} \cdot \sum_{i=1}^{k} \sum_{j=1}^{\frac{n}{2^i}*} v_j^{(i)} \eta_j^{(i)}, \quad \eta_2^k = \frac{1}{v_2^{(k)}} \cdot \sum_{i=1}^{k} \sum_{j=1}^{2^{k+1-i}} v_j^{(i)} \eta_j^{(i)}.$$

Nota $*$: $\frac{n}{2^i} = \frac{2^k}{2^i} = 2^{k-i}$.

■

Capitolo 14

Integrazione Numerica.

Problema:
calcolo numerico dell'integrale definito di una funzione $f\colon [a,b] \to \mathbb{R}$, integrabile in $[a,b]$, cioé $\int_a^b f(x)dx$ mediante una formula del tipo:

$$\sum_{i=0}^{n} \omega_i f(x_i) = \omega_0 f(x_0) + \omega_1 f(x_1) + \cdots + \omega_n f(x_n),$$

dove $x_i \in [a,b],\ i = 0, \ldots, n$, sono detti *nodi* e sono distinti, $\omega_i,\ i = 0, \ldots, n$ sono detti *pesi*.

Analisi dell'errore:

$$R_n(f) := \int_a^b f(x)dx - \sum_{i=0}^{n} \omega_i f(x_i).$$

Possiamo calcolare $R_n(f)$ esattamente solo quando abbiamo $\int_a^b f(x)dx$, quando non è possibile ricavare esattamente il valore dell'integrale è possibile usare le formule di quadratura.

14.1 Formule (di quadratura) interpolatorie.

Fissato $n \in \mathbb{N}$ si scelgono $n+1$ nodi $\{x_i\}_{i=0}^n$ in $[a,b]$, con $x_i \neq x_j$ per $i \neq j$; quindi si approssima $f(x)$ con il suo polinomio interpolatore $p_n(x)$ nei nodi fissati. I punti base del polinomio sono: $x_0, \ldots, x_n$, in cui si calcolano i valori della funzione: $f(x_0), \ldots, f(x_n)$.

Osservazione 14.1. $f(x)$ deve essere definita nei nodi di integrazione.

Si ottiene allora:

$$\int_a^b f(x)dx = \int_a^b p_n(x)dx + \int_a^b E_n(x)dx.$$

$$\int_a^b f(x)dx = \sum_{i=0}^{n} f(x_i) \cdot \int_a^b L_i(x)dx + R_n(f).$$

$E_n(x)$ è l'errore di interpolazione, $L_i(x)$ sono i polinomi elementari di Lagrange. Il *resto* $R_n(f)$ è così definito:

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

se $f \in \mathcal{C}^{n+1}([a,b])$ si ha:

$$R_n(f) = \int_a^b w(x)\frac{f^{(n+1)}(\xi - x)}{(n+1)!}dx.$$

Posto $\omega_i = \int_a^b L_i(x)dx,\ i = 0, \ldots, n$ si ottiene, trascurando il resto, la formula di quadratura:

$$\int_a^b f(x)dx \simeq \sum_{i=0}^{n} \omega_i f(x_i). \tag{14.1}$$

Definizione 14.2. La formula 14.1, in cui i pesi sono definiti da:

$$\omega_i = \int_a^b L_i(x)dx, \quad i = 0, \ldots, n;$$

viene detta formula di *quadratura interpolatoria.*

Proposizione 14.3. *Una formula di quadratura interpolatoria integra esattamente almeno tutti i polinomi di grado minore o uguale ad n.*

Dimostrazione. Sia $p_n \in \mathcal{P}_n$, allora:

$$R_n(f) = \int_a^b E_n(x)dx = \int_a^b \left(f(x) - p_n(x)\right) dx = 0.$$

Poiché $f(x) = p_n(x)$ per esistenza ed unicità del polinomio interpolante.

■

Definizione 14.4. Una formula di quadratura si dice avere *grado di precisione* n se integra esattamente tutti i polinomi di grado minore o uguale ad n (resto nullo) ed esiste $q \in \mathcal{P}_{n+1}$ tale che $R_n(q) \neq 0$.

Osservazione 14.5. Una formula di quadratura interpolatoria ha grado di precisione almeno n.

Teorema 14.6. *Fissato $n \in \mathbb{N}$, e fissati $n+1$ nodi distinti $\{x_i\}_{i=0}^n$ in $[a,b]$ esiste al più ($\exists!$) una formula di quadratura con grado di precisione* almeno *n, che quindi coincide con la formula di quadratura interpolatoria.*

Dimostrazione. Occorre imporre che $\sum_{i=0}^n \omega_i f(x_i)$ integri esattamente i polinomi di grado $\leq n$. Imponiamo quindi $n+1$ condizioni:

$$\begin{cases} \sum_{i=0}^n \omega_i \cdot 1 = \int_a^b 1dx = b-a & x^0 \\ \sum_{i=0}^n \omega_i \cdot x_i = \int_a^b x dx = \frac{b^2-a^2}{2} & x^1 \\ \sum_{i=0}^n \omega_i \cdot x_i^2 = \int_a^b x^2 dx = \frac{b^3-a^3}{3} & x^2 \\ \vdots & \\ \sum_{i=0}^n \omega_i \cdot x_i^n = \int_a^b x^n dx = \frac{b^{n+1}-a^{n+1}}{n+1} & x^n \end{cases}$$

$$\begin{cases} \omega_0 + \omega_1 + \cdots + \omega_n = b-a \\ \omega_0 x + \omega_1 x + \cdots + \omega_n x = \frac{b^2-a^2}{2} \\ \omega_0 x^2 + \omega_1 x^2 + \cdots + \omega_n x^2 = \frac{b^3-a^3}{3} \\ \vdots \\ \omega_0 x^n + \omega_1 x^n + \cdots + \omega_n x^n = \frac{b^{n+1}-a^{n+1}}{n+1} \end{cases}$$

La matrice del sistema risulta quindi:

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_0 & x_1 & \cdots & x_n \\ \vdots & \vdots & & \vdots \\ x_0^n & x_1^n & \cdots & x_n^n \end{bmatrix},$$

ovvero la matrice trasposta di Vandermonde. Se i nodi sono distinti, il determinante è diverso da 0. L'unicità è garantita dal fatto che il determinante non sia nullo, tuttavia a livello di calcolo (dei ω_i) è meglio percorrere un'altra strada.

■

Proposizione 14.7. *Il massimo grado di precisione ottenibile con una formula di quadratura a $n+1$ nodi distinti è $2n+1$.*

Dimostrazione. Sia $Q(x) = \prod_{j=0}^{n}(x - x_j)^2 \in \mathcal{P}_{2n+1}$. Allora:

$$R_n(Q) = \underbrace{\int_a^b Q(x)dx}_{>0} - \underbrace{\sum \omega_i Q(x_i)}_{=0} > 0.$$

■

$R_n(Q)$ sono le formule di quadratura gaussiana. Cosa possiamo cambiare?
La scelta dei nodi (non lo vedremo nel corso di Analisi Numerica 1).

14.2 Formule di quadratura di Newton-Cotes chiuse.

Sono formule di quadratura *interpolatorie* dove i nodi $x_i^{(n)},\ i = 0, \dots, n$ sono fissati ed equidistanti in $[a, b]$. Fissato $n \in \mathbb{N}^+$ e posto $h = \frac{b-a}{n}$, i nodi sono dati da:

$$x_i^{(n)} = a + ih, \quad i = 0, \dots, n.$$

I pesi sono:

$$\omega_i = \int_a^b L_i(x)dx = \int_a^b \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}dx,$$

ponendo $x = a + sh$ si ha:

$$\int_a^b \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}dx = \int_0^n \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{\cancel{a} + sh\cancel{-a} - jh}{\cancel{a} + ih\cancel{-a} - jh}ds.$$

$$\omega_i = h\int_0^n \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{s - j}{i - j}ds = h \cdot \alpha_i^{(n)}.$$

Nota Bene. Si dicono formule chiuse quando a, b sono nodi, mentre si dicono aperte quando $a, b \notin \{x_i\}$.

Definizione 14.8. $\alpha_i^{(n)}$ è detta *costante di Newton-Cotes*, indipendente da a e b.

$$\int_a^b f(x)dx \approx h\sum_{i=0}^{n} \alpha_i^{(n)} f(x_i^{(n)}).$$

Osservazione 14.9. Valgono le seguenti proprietà:

- $\alpha_i^{(n)} = \alpha_{n-i}^{(n)}$;
- $\sum_{i=0}^{n} \alpha_i^{(n)} = n$.

Dimostrazione.

- $\alpha_i^{(n)} = \alpha_{n-i}^{(n)}$;

$$\alpha_{n-i}^{(n)} = \int_0^n \prod_{\substack{j=0 \\ j\neq n-i}}^{n} \frac{s-j}{n-i-j} ds$$

ponendo $k = n - j$ si ha:

$$\alpha_{n-i}^{(n)} = \int_0^n \prod_{\substack{k=0 \\ k\neq i}}^{n} \frac{s-n+k}{k-i} ds = \int_0^n \prod_{\substack{k=0 \\ k\neq i}}^{n} \frac{n-s-k}{i-k} ds$$

ponendo $t = n - s$:

$$\alpha_{n-i}^{(n)} = \int_0^n \prod_{\substack{k=0 \\ k\neq i}}^{n} \frac{t-k}{i-k} dt = \alpha_i^{(n)}.$$

- $\sum_{i=0}^{n} \alpha_i^{(n)} = n$.

$$\int_a^b 1 \cdot dx = h \sum_{i=0}^{n} \alpha_i^{(n)} \;\rightsquigarrow\; \sum_{i=0}^{n} \alpha_i^{(n)} = \frac{b-a}{h} = n.$$

■

14.2.1 Formula dei trapezi ($n = 1$).

$$x_0^{(1)} = a, \quad x_1^{(1)} = a + h = a + \frac{b-a}{1} = b.$$

$$\alpha_0^{(1)} = \int_0^1 \frac{s-1}{0-1} ds = \left[-\frac{s^2}{2} + s \right]_0^1 = \frac{1}{2}.$$

Inoltre per simmetria si ha $\alpha_0^{(1)} = \alpha_1^{(1)} = \frac{1}{2}$.

$$\int_a^b f(x)dx = \frac{b-a}{2}[f(a) + f(b)] + R_1(f).$$

14.2.2 Formula di Cavalieri-Simpson ($n = 2$).

$$x_0^{(2)} = a, \quad x_1^{(2)} = a + h = a + \frac{b-a}{2} = \frac{a+b}{2}, \quad x_2^{(2)} = b.$$

$$\alpha_0^{(n)} = \int_0^2 \frac{s-1}{0-1} \cdot \frac{s-2}{0-2} ds = \frac{1}{2} \int_0^2 (1-s)(2-s) ds =$$

$$= \frac{1}{2} \int_0^2 (s^2 - 2s - s + 2) ds = \frac{1}{2} \left[\frac{1}{3} s^3 - s^2 - \frac{1}{2} s^2 + 2s \right]_0^2 =$$

$$= \frac{1}{2} \left(\frac{2^3}{3} - 3\frac{2^2}{2} + 2 \cdot 2 \right) = \frac{4}{3} - 3 + 2 = \frac{1}{3}.$$

$$\alpha_1^{(2)} = \int_0^2 \frac{s-0}{1-0} \cdot \frac{s-2}{1-2} ds = \int_0^2 s(2-s) ds = \int_0^2 -(s^2 + 2s) ds =$$

$$= \left[\frac{2s^2}{2} - \frac{s^3}{3} \right]_0^2 = \frac{2 \cdot 2^2}{2} - \frac{2^3}{3} = \frac{4}{3}.$$

$$\alpha_2^{(n)} = \alpha_0^{(n)} = \frac{1}{3}.$$

$$\alpha_0^{(n)} + \alpha_1^{(n)} + \alpha_2^{(n)} = \frac{1}{3} + \frac{4}{3} + \frac{1}{3} = 2 = n \qquad \to \text{ok.}$$

$$\omega_0 = h \cdot \alpha_0^{(n)} = \frac{b-a}{2} \cdot \frac{1}{3} = \frac{b-a}{6}.$$

$$\omega_1 = h \cdot \alpha_1^{(n)} = \frac{b-a}{2} \cdot \frac{4}{3} = \frac{2(b-a)}{3}.$$

$$\omega_2 = h \cdot \alpha_2^{(n)} = \frac{b-a}{2} \cdot \frac{1}{3} = \frac{b-a}{6}.$$

$$\int_a^b f(x) dx = \frac{b-a}{2} \sum_{i=0}^{2} \alpha_i^{(2)} f(x_i) = \frac{b-a}{2} \left[\alpha_0^{(n)} f(x_0) + \alpha_1^{(n)} f(x_1) + \alpha_2^{(n)} f(x_2) \right] =$$

$$= \frac{b-a}{2} \left[\frac{1}{3} f(a) + \frac{4}{3} f\left(\frac{a+b}{2} \right) + \frac{1}{3} f(b) \right] =$$

$$= \frac{b-a}{6} f(a) + \frac{2(b-a)}{3} f\left(\frac{a+b}{2} \right) + \frac{b-a}{6} f(b).$$

$$\int_a^b f(x) dx = \frac{h}{3} \left[f(a) + 4f\left(\frac{a+b}{2} \right) + f(b) \right].$$

14.2.3 Formula dei tre ottavi ($n = 3$).

$$x_0^{(3)} = a, \quad x_1^{(3)} = a + h = a + \frac{b-a}{3}, \quad x_2^{(3)} = a + \frac{2(b-a)}{3}, \quad x_3^{(3)} = b.$$

$$\alpha_0^{(n)} = \int_0^3 \frac{s-1}{0-1} \cdot \frac{s-2}{0-2} \cdot \frac{s-3}{0-3} ds = \int_0^3 (1-s)\frac{(2-s)}{2}\frac{(3-s)}{3} ds.$$

$$\alpha_0^{(n)} = \frac{1}{6}\int_0^3 (1-s)(2-s)(3-s) ds = \frac{1}{6}\int_0^3 \left(-s^3 + 6s^2 - 11s + 6\right) ds.$$

$$\alpha_0^{(n)} = \frac{1}{6}\left[-\frac{s^4}{4} + 2s^3 - \frac{11s^2}{2} + 6s\right]_0^3 = \frac{1}{6}\left[-\frac{81}{4} + 54 - \frac{99}{2} + 18\right].$$

$$\alpha_0^{(n)} = \frac{1}{6}\left[\frac{-81-198}{4} + \frac{216}{4} + \frac{72}{4}\right] = \frac{1}{6}\cdot\frac{9}{4} = \frac{1}{2}\cdot\frac{3}{4} = \frac{3}{8}.$$

Possiamo, questa volta, calcolare $\alpha_1^{(3)}$ senza utilizzare le formule integrali, poiché:

$$\alpha_0^{(3)} + \alpha_1^{(3)} + \alpha_2^{(3)} + \alpha_3^{(3)} = 3.$$

Dai calcoli precedenti e dal fatto che $\alpha_1^{(3)} = \alpha_2^{(3)}$ otteniamo:

$$\frac{3}{8} + \alpha_1^{(3)} + \alpha_1^{(3)} + \frac{3}{8} = 3 \implies 2 \cdot \alpha_1^{(3)} = 3 - \frac{\cancel{6}^{3}}{\cancel{8}_{4}}$$

$$\alpha_1^{(3)} = \frac{3}{2} - \frac{3}{8} = \frac{12-3}{8} = \frac{9}{8} = \alpha_2^{(3)}.$$

$$\int_a^b f(x)dx = h\left[\frac{3}{8}f(a) + \frac{9}{8}f\left(a + \frac{b-a}{3}\right) + \frac{9}{8}f\left(a + \frac{2(b-a)}{3}\right) + \frac{3}{8}f(b)\right].$$

$$\int_a^b f(x)dx = \frac{3}{8}h\left[f(a) + 3f\left(a + \frac{b-a}{3}\right) + 3f\left(a + \frac{2(b-a)}{3}\right) + f(b)\right] + R_3(f).$$

Esercizio. Dato un integrale:

$$\int_0^1 x^3 dx,$$

approssimarlo con la formula dei trapezi e di Cavalieri-Simpson, determinare per questo problema test (solo soluzione chiusa) l'errore assoluto per le due formule.

Svolgimento. *Calcoliamo per primo il valore dell'integrale definito:*

$$\int_0^1 x^3 dx = \left[\frac{x^4}{4}\right]_0^1 = \frac{1}{4}.$$

- *Formula dei trapezi:*

$$\int_0^1 x^3 dx = h(f(a)+f(b)) = \frac{1}{2}(1+0) = \frac{1}{2} + R_1(f).$$

- *Formula di Cavalieri-Simpson:*

$$\int_0^1 x^3 dx = \frac{h}{3}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right) = \frac{1}{6}\left(0 + \frac{1}{2} + 1\right) =$$

$$= \frac{1}{12} + \frac{1}{6} = \frac{3}{12} = \frac{1}{4} + R_2(f).$$

Si ha quindi che:

$$R_1(f) = -\frac{1}{4}, \quad R_2(f) = 0.$$

La formula dei trapezi ha grado di precisione almeno 1*, è quindi giusto che commetta un errore, la formula di Cavalieri-Simpson invece ha grado di precisione almeno* 2*. Questa funzione però è di grado* 3 *e viene integrata perfettamente. E' questo un caso?*

No. E' una proprietà delle formule di Newton-Cotes con n pari (numero dispari di nodi) che hanno precisione almeno $n+1$.

Osservazione 14.10. Le formule di Newton-Cotes hanno grado di precisione almeno n essendo formule interpolatorie.

Proposizione 14.11. *La formula dei trapezi ha grado* 1 *e precisione* esattamente 1.

Dimostrazione. Sia $q(x) = (x-a)(x-b) \in \mathcal{P}_2$, allora:

$$\begin{aligned} R_1(q) &= \int_a^b q(x)dx - h\sum_{i=0}^{1} \alpha_i^{(1)} q(x_i^{(1)}) \\ &= \underbrace{\int_a^b q(x)dx}_{<0} - \tfrac{b-a}{2}\underbrace{[q(a)+q(b)]}_{\equiv 0} \neq 0. \end{aligned}$$

■

Proposizione 14.12. *Se n è pari (numero di nodi dispari), le formule di quadratura di Newton-Cotes a $n+1$ nodi hanno almeno grado di precisione $n+1$.*

Dimostrazione. Le formule di quadratura considerate, essendo interpolatorie, integrano esattamente tutti i polinomi di grado minore o uguale ad n.

Sia ora $q \in \mathcal{P}_{n+1}$, possiamo allora scrivere q nella seguente forma:

$$q(x) = \overline{a}(x - x_{\frac{n}{2}})^{n+1} + q_n(x), \quad x_{\frac{n}{2}} = \frac{a+b}{2},\ q_n \in \mathcal{P}_n.$$

$$\longrightarrow \int_a^b q(x)dx = \overline{a} \int_a^b (x - x_{\frac{n}{2}})^{n+1} dx + \int_a^b q_n(x)dx.$$

Poiché il polinomio $(x - x_{\frac{n}{2}})^{n+1}$ è di grado dispari e la funzione è antisimmetrica rispetto al punto medio dell'intervallo di integrazione si ha:

$$\int_a^b (x - x_{\frac{n}{2}})^{n+1} dx = 0.$$

$$\longrightarrow \int_a^b q(x)dx = \int_a^b q_n(x)dx = h \sum_{i=0}^{n} \alpha_i^{(n)} q_n(x_i).$$

Poiché $x_i - x_{\frac{n}{2}} = -(x_{n-1} - x_{\frac{n}{2}})$ per $i = 0, \ldots, n$ e $\alpha_{n-i}^{(n)} = \alpha_i^{(n)}$ si ha:

$$= \sum_{i=0}^{n} \alpha_i^{(n)} (x_i - x_{\frac{n}{2}})^{n+1} \equiv 0.$$

$$\longrightarrow \int_a^b q(x)dx = h \left(\overline{a} \sum_{i=0}^{n} \alpha_i^{(n)} (x_i - x_{\frac{n}{2}})^{n+1} + \sum_{i=0}^{n} \alpha_i^{(n)} q_n(x_i) \right)$$

$$= h \cdot \sum_{i=0}^{n} \alpha_i^{(n)} q_n(x_i)$$

■

Proposizione 14.13. *La formula di Cavalieri-Simpson ha grado di precisione* esattamente 3.

Dimostrazione. Sia $q(x) = (x-a)(x-\frac{a+b}{2})^2(x-b) \in \overline{\mathcal{P}}_n$, allora:

$$R_2(q) = \int_a^b q(x)dx - h\sum_{i=0}^{2} \alpha_i^{(2)} q(x_i^{(2)}).$$

$$R_2(q) = \underbrace{\int_a^b q(x)dx}_{<0} - \frac{h}{3}\underbrace{\left[q(a) + 4q(\frac{a+b}{2}) + q(b)\right]}_{\equiv 0} \neq 0.$$

■

14.2.4 Errore di integrazione numerica.

Per le varie formule di quadratura viste fino ad ora abbiamo visto che approssimano un risultato che si discosta dal valore reale per un resto che abbiamo denotato con R_i. Cerchiamo ora di stimare questo errore.

Teorema 14.14. *Sia $n \in \mathbb{N}$ dispari e $f \in \mathcal{C}^{n+1}([a,b])$, allora esiste $\xi \in (a,b)$ tale che:*

$$R_n(f) = K_n \frac{f^{(n+1)}(\xi)}{(n+1)!} \cdot h^{n+2}, \quad h = \frac{b-a}{n}.$$

$$K_n = \int_0^n \mathcal{T}_n(t)dt, \quad \textit{con } \mathcal{T}_n = \prod_{i=0}^{n}(t-i).$$

Nota Bene. ξ non è dato saperlo, ovvero non possiamo calcolare esattamente l'errore, però lo possiamo stimare.

In particolare per la formula dei trapezi si ha:

$$R_1(f) = -\frac{1}{12} f''(\xi) h^3.$$

Dimostrazione. Caso $n = 1$:

$$R_1(f) = \int_a^b [f(x) - p(x)]\, dx = \int_a^b \underbrace{(x-a)(x-b)\frac{f''(\xi)}{2!}}_{\text{(*) errore di interpolazione}} dx.$$

Si applica dunque il Teorema della media generalizzato al prodotto di due funzioni: una continua g $(= (*))$ e un'altra che non cambia segno $r = 1$ nell'intervallo di interpolazione:

$$R_1(f) = \frac{f''(\xi)}{2!} \int_a^b (x-a)(x-b)dx.$$

Ponendo $x = a + th$, $h = b - a$ si ha:

$$R_1(f) = \frac{f''(\xi)}{2!}\int_0^1 th(t-1)hhdt = h^3\frac{f''(\xi)}{2!}\int_0^1 \underbrace{(t-0)(t-1)}_{\mathcal{T}(t)}dt.$$

$$R_1(f) = h^3\frac{f''(\xi)}{2!}\left[\frac{t^3}{3} - \frac{t^2}{2}\right]_0^1 = -\frac{1}{12}f''(\xi)h^3.$$

■

Se $b-a = 1000$ avremmo $R_1 \sim 1000^3$, ovvero un errore enorme! Dobbiamo quindi integrare su intervalli tali che $h < 1$.

Teorema 14.15. *Sia $n \in \mathbb{N}$ pari e $f \in \mathcal{C}^{n+2}([a,b])$, allora esiste $\xi \in (a,b)$ tale che:*

$$R_n(f) = \overline{K}_n\frac{f^{(n+2)}(\xi)}{(n+2)!}h^{n+3}, h = \frac{b-a}{n}.$$

$$\overline{K}_n = \int_0^n t\mathcal{T}(t)dt, \quad con\ \mathcal{T}(t) = \prod_{i=0}^{n}(t-i).$$

In particolare per la formula di Cavalieri-Simpson si ha:

$$R_2(f) = -\frac{1}{90}f^{(iv)}(\xi)h^5.$$

Dimostrazione. Caso $n = 2$ (formule di quadratura di Cavalieri-Simpson):

$$R_2(f) = \int_a^b f(x)dx - h\sum_{i=0}^{2}\alpha_i^{(2)}f(x_i^{(2)}).$$

Sia $\overline{p} \in \mathcal{P}_3$ il polinomio interpolante $f(x)$ nei nodi:

$$x_0^{(2)} = a,\ x_1^{(2)} = \frac{a+b}{2},\ x_2^{(2)} = b,$$

e sia $f'(x)$, calcolata nel nodo $x_1^{(2)}$, uguale a $c = a + h = \frac{a+b}{2}$. Ricordiamo inoltre che la formula di quadratura di Cavalieri-Simpson ha grado di precisione 3.

$$R_2(f) = \int_a^b [f(x) - \overline{p}(x)]\,dx = \int_a^b (x-a)(x-c)^2(x-b)\frac{f^{(iv)}(\xi_x)}{4!}dx.$$

Sfruttando il teorema della media generalizzata si ha:

$$R_2(f) = \frac{f^{(iv)}(\xi_x)}{4!} \int_a^b (x-a)(x-c)^2(x-b)dx.$$

Ponendo $x = a + th$ e ricordando che $h = \frac{b-a}{2}$ si ha:

$$\begin{aligned} R_2(f) &= \frac{f^{(iv)}(\xi_x)}{4!} \int_0^2 t(t-1)^2(t-2)h^5 dt \\ &= h^5 \frac{f^{(iv)}(\xi_x)}{4!} \underbrace{\int_0^2 t\mathcal{T}(t)dt}_{\neq 0} - h^5 \frac{f^{(iv)}(\xi_x)}{4!} \underbrace{\int_0^2 \mathcal{T}(t)dt}_{=0} \quad (\to 0) \\ &= h^5 \frac{f^{(iv)}(\xi_x)}{4!} \left[\frac{1}{5}t^5 - \frac{3}{4}t^4 + \frac{2}{3}t^3\right]_0^2 \\ &= -\frac{1}{90} \cdot f^{(iv)}(\xi_x) \cdot h^5. \end{aligned}$$

■

Esercizio. Approssimare l'integrale:

$$\int_0^1 e^{-x^2} dx$$

commettendo un errore non superiore a $0.5 \cdot 10^-2$.

Svolgimento.

- *Utilizziamo le formule chiuse di Newton Cotes.*
- *Poniamo $n = 1$ con $E_1 = -\frac{1}{12}h^3 f''(\xi)$.*
  In questo caso $f''(x) = 2(2x^2 - 1)e^{-x^2}$ e $h = 1$, si ha dunque:

$$\max_{x \in (0,1)} |f''(x)| = 2,$$

  facendo i calcoli otteniamo:

$$E_1 \leq \frac{1}{6}.$$

Il risultato non è soddisfacente.

- *Poniamo $n = 2$ con $E_2 = -\frac{1}{90}h^5 f^{(iv)}(\xi)$.*

  In questo caso $f^{(iv)}(x) = 4(4x^4 - 12x^2 + 3)e^{-x^2}$ e $h = \frac{1}{2}$, si ha dunque:

$$\max_{x\in(0,1)} |f^{(iv)}(x)| = 12,$$

  facendo i calcoli otteniamo:

$$E_2 \leq \frac{1}{240}.$$

Il risultato è soddisfacente.

- *Calcoliamo ora l'integrale con la formula dei trapezi:*

$$\int_0^1 e^{-x^2} dx = \frac{1}{6}\left[1 + 4e^{-\frac{1}{4}} + e^{-1}\right] = 0.7472$$

  Commettendo un errore che al più sarà $0,0041 \leq 0.5 \cdot 10^{-2}$.

14.3 Formule di quadratura di Newton-Cotes aperte.

Le formule aperte di Newton-Cotes sono sempre formule interpolatorie, in cui i nodi sono equispaziati, ma non interpolano gli estremi dell'intervallo.

Fissato $n \in \mathbb{N}$ e posto $h = \frac{b-a}{n+2}$ i nodi sono dati da:

$$x_i = a + (i+1)h, \quad i = 0, \ldots, n.$$

Osservazione 14.16. Vediamo i nodi più esterni:

$i = 0 \longrightarrow x_0 = a + h \neq a;$

$i = n \longrightarrow x_n = a + (n+1)h < b.$

Data una generica formula interpolatoria analizziamo i pesi ω_i analogamente a quanto fatto per le formule chiuse.

$$\omega_i = \int_a^b L_i(x)dx = \int_a^b \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j} dx,$$

ponendo $x = a + sh$ si ha:

$$\int_{-1}^{n+1} \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{\cancel{a} + sh \cancel{-a} - jh}{\cancel{a} + ih \cancel{-a} - jh} ds.$$

$$\omega_i = h \int_{-1}^{n+1} \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{s-j}{i-j} ds = h \cdot \alpha_i^{(n)}.$$

La struttura dei pesi è simile a quella delle formule chiuse, cambiano gli estremi di integrazione ma godono delle stesse proprietà. In particolare i pesi non dipendono dagli estremi.

Queste formule sono utili nel caso in cui si devono studiare funzioni non integrabili agli estremi.

Esempio. Il seguente integrale non è calcolabile mediante formule chiuse:

$$\int_0^1 \log(x) dx.$$

14.3.1 Formula del punto medio ($n = 0$).

$$h = \frac{b-a}{2}, \quad x_0 = a + h = \frac{a+b}{2}, \quad \omega_0 = b - a = h \cdot 2 \;\Rightarrow\; \alpha_0^{(0)} = 2.$$

$$\int_a^b f(x) dx = (b-a) \cdot f\left(\frac{a+b}{2}\right) + R_0(f).$$

La formula del punto medio integra *perfettamente* i polinomi di primo grado (e inferiori), questo si può osservare chiaramente nell figura 14.1.

14.3.2 Errore di integrazione numerica.

Analogamente a quanto fatto per le formule di Newton-Cotes chiuse, analizziamo il resto dell'integrazione per stimarne l'errore.

Teorema 14.17. *Sia $n \in \mathbb{N}$ dispari e $f \in \mathcal{C}^{n+1}([a,b])$, allora esiste $\xi \in (a,b)$ tale che:*

$$R_n(f) = K_n \cdot \frac{f^{n+1}(\xi)}{(n+1)!} \cdot h^{n+2}, \quad h = \frac{b-a}{n+2};$$

$$K_n = \int_{-1}^{n+1} \mathcal{T}_n dt, \quad \mathcal{T}_n = \prod_{i=0}^{n} (t-i).$$

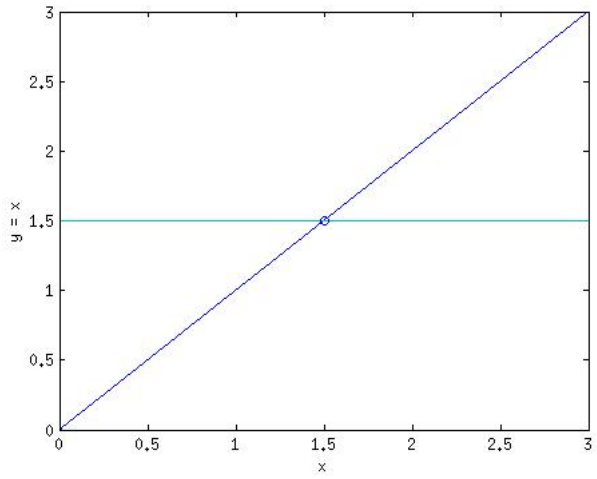


Figura 14.1: Integrale della funzione $f(x) = x$ nell'intervallo $(0, 3)$ mediante la formula del punto medio.

Teorema 14.18. *Sia $n \in \mathbb{N}$ pari e $f \in \mathcal{C}^{n+2}([a,b])$, allora esiste $\xi \in (a,b)$ tale che:*

$$R_n(f) = K_n \cdot \frac{f^{n+2}(\xi)}{(n+2)!} \cdot h^{n+3}, \quad h = \frac{b-a}{n+2};$$

$$K_n = \int_{-1}^{n+1} \mathcal{T}_n dt, \quad \mathcal{T}_n = \prod_{i=0}^{n} (t-i).$$

Corollario 14.19. *Le formule di Newton-Cotes aperte con n dispari hanno grado di precisione n, quelle con n pari hanno grado di precisione $n+1$.*

Osservazione 14.20. In particolare la formula del punto medio integra *esattamente* tutti i polinomi di grado ≤ 1.

Proposizione 14.21. *Sia $f \in \mathcal{C}^2([a,b])$, ovvero $n = 0$, allora l'errore risulta:*

$$R_0(f) = \frac{1}{3} f''(\xi) h^3, \quad h = \frac{b-a}{2}.$$

Dimostrazione. Poniamo f nella generica "forma di secondo grado" $f(x) = (x-c)^2$, il punto medio $x_0 = \frac{a+b}{2}$ e p il polinomio al più di primo grado che interpola f nel punto x_0. Sia ora la derivata prima calcolata in x_0 uguale a c, ovvero $f'(x_0) = c$.

$$
\begin{aligned}
R_0(f) &= \int_a^b f(x)dx - h \cdot \alpha_0 \cdot f(x_0) && \text{per def.} \\
&= \int_a^b f(x)dx - \underbrace{h \cdot \alpha_0 \cdot p(x_0)}_{*=\int_a^b p(x)dx} && \text{def. di interpolazione} \\
&= \int_a^b [f(x) - p(x)]\, dx && * \\
&= \int_a^b \frac{(x-c)^2 f''(\xi)}{2!} dx && \text{def. di errore di interp.} \\
&= \frac{f''(\xi)}{2} \int_a^b (x-c)^2 dx. && \text{teo. media gen.}
\end{aligned}
$$

Poniamo ora $x = a + (t+1) \cdot h$, abbiamo che:

$$
\begin{aligned}
R_0(f) &= \frac{f''(\xi)}{2} \int_{-1}^{1} t^2 h^2 h dt \\
&= \frac{f''(\xi)}{2} \int_{-1}^{1} t^2 h^3 dt \\
&= \frac{f''(\xi)}{2} \cdot h^3 \int_{-1}^{1} t^2 dt \\
&= \frac{f''(\xi)}{2} \cdot h^3 \left[\frac{t^3}{3}\right]_{-1}^{1} \\
&= \frac{1}{3} \cdot h^3 \cdot f''(\xi).
\end{aligned}
$$

■

Parte III

Matlab.

Capitolo 15

Introduzione a Matlab.

MATLAB (abbreviazione di Matrix Laboratory) è un ambiente per il calcolo numerico e l'analisi statistica che comprende anche l'omonimo linguaggio di programmazione creato dalla The MathWorks. MATLAB consente di manipolare matrici, visualizzare funzioni e dati, implementare algoritmi, creare interfacce utente, e interfacciarsi con altri programmi.

Nota Bene. Il corso e *sopratutto l'esame* di Laboratorio di Calcolo Numerico vertirà su esercizi da svolgersi in ambiente Matlab, per esercitarsi è possibile usufruire dei computer in laboratorio. Per coloro che volessero (o debbano) esercitarsi a casa, poiché Matlab è un software proprietario, l'unico modo (legale) per ottenere il programma è l'acquisto. Tuttavia esiste un'alternativa "free", Octave, che permette di fare gran parte delle operazioni di Matlab con gli stessi comandi. D'ora in poi, nel seguente testo, per specificare differenze o particolarità riguardanti Octave utilizzeremo un "ambiente" apposito come il seguente.

Octave. GNU Octave è un'applicazione software per l'analisi numerica parzialmente compatibile con MATLAB. Octave è rilasciato sotto licenza GPL, e quindi può essere liberamente copiato e usato. Il programma gira sotto sistemi Unix e Linux, oltre che su Windows e MAC OS X.
Ha un insieme di funzionalità fornite per il calcolo matriciale come rango e determinante o specialistiche come decomposizione ai valori singolari (SVD), fattorizzazione LU; sebbene le consenta la soluzione numerica di sistemi lineari non svolge calcolo simbolico o altre attività tipiche di un sistema di algebra computazionale. ♠

15.1 Introduzione.

Questo testo è di fatto una raccolta di appunti del corso di Laboratorio di Calcolo Numerico del dipartimento di Matematica dell'Università di Parma, non ha lo scopo di sostituirne le lezioni o i libri di testo consigliati. Pertanto sconsiglio di utilizzarlo come unica fonte di studio o di esercitazione pratica.

Inoltre non verranno definite tutte le funzioni o risolti tutti gli esercizi proposti durante il corso ed alcune altre cose saranno date per scontate come ad esempio la descrizione o definizione di:

- Command Window,
- Command History,
- Workspace,
- alcune funzionio comandi ben descritti sul libro di testo.

15.2 Help.

Il primo comando utile di Matlab è `help`, serve, dato un comando o una funzione, per ottenerne le istruzioni d'uso:

```
>> help nomecomando
>> help nomefunzione
```

15.3 Variabili ed ambiente di lavoro.

In Matlab tutte le variabili sono array, nello specifico una variabile singola è un array di dimensione 1. Nell'ambiente di lavoro (workspace) le variabili vengono memorizzate e vi risiedono fino alla fine della sessione di lavoro o fino a quando non viene utilizzato il comando `clear` che consente di cancellarle, non vengono cancellati lo schermo e la history dei comandi. Per cancellare il contenuto dello schermo si utilizza il comando `clc`.

```
>> clear
```

Per definire (dichiarazione e inizializzazione) una variabile è sufficiente scrivere la seguente istruzione:

```
>> x = 1

x =

     1
```

In questo modo verrà salvata nel Workspace la variabile di nome x, valore 1, size 1×1 e valore minimo e massimo dell'array. Di default gli attributi visibili delle variabili del Workspace dovrebbero essere questi, è comunque possibile cambiarli dal menù a tendina "view/Choose Columns".

E' possibile evitare di visualizzare a schermo l'assegnamento della variabile semplicemente utilizzando il carattere ";" a fine istruzione. Questo è molto utile per array o matrici di grandi dimensioni.

```
>> v = [ 1 2 3 4 5 6 7 8 9 ]

v =

     1     2     3     4     5     6     7     8     9


>> v = [ 1 2 3 4 5 6 7 8 9 ];
>>
```

Esiste una variabile speciale in Matlab chiamata `ans`, questa è generata automaticamente se non viene esplicitamente dato un nome ad una variabile, tipicamente quando non viene assegnato il risultato di un operazione. Proviamo a moltiplicare il nostro vettore v per la variabile x senza assegnamento:

```
>> x*v

ans =

     1     2     3     4     5     6     7     8     9
```

Utilizzare la variabile `ans` non è un errore, occorre però fare attenzione poiché questa può essere sovrascritta durante il lavoro.

Nell'ambiente di lavoro è inoltre possibile cambiare il formato di rappresentazione dei numeri, di default l'ambiente dovrebbe essere inpostato sul formato `short` ma è possibile cambiarlo con il comando `format`.

```
>> x = 1/2

x =

    0.5000

>> format long
>> x = 1/2

x =

   0.500000000000000

>>
```

Inizialmente x ha quattro cifre dopo la virgola, impostando il formato `long` diventano quindici.

Octave. In Octave le cifre del formato short sono cinque dopo la virgola, si noti inoltre la differenza di stampa nella Command Window (che in Octave si chiama Octave Terminal).

```
>>> x = 1/2
x =  0.50000
>>> format long
>>> x = 1/2
x =  0.500000000000000
```

♠

Matlab lavora con un sistema floating point in doppia precisione (si possono aumentare i numeri dopo la virgola).

15.3.1 Variabili speciali.

In Matlab esistono alcune variabili speciali:

`i`, `j`	unità immaginarie ($i^2 = -1,\ j^2 = -1$),
`pi`	pi greco,
`realmax`	il più grande numero di macchina,
`realmin`	il più piccolo numero di macchina,
`Inf`	∞, ovvero un numero maggiore di `realmax`,
`nan`	Not a number (0/0, etc.),
`eps`	epsilon macchina.

```
>> pi

ans =

    3.1416
```

15.3.2 Funzioni Built-In.

In Matlab è possibile utilizzare una lista di funzioni "pre-costruite" come `sin`, `cos`, etc... con la sintassi seguente:

`sin(x)`	calcola il seno della variabile `x`,
`cos(x)`	calcola il coseno delella variabile `x`,
`sqrt(x)`	calcola la radice quadrata di `x`,
`abs(x)`	calcola il valore assoluto di `x`,
`log(x)`	calcola il logaritmo naturale di `x`,
`log10(x)`	calcola il logaritmo dixin base 10.

```
>> log(2)

ans =

    0.6931

>> log10(2)

ans =

    0.3010
```

Molte altre funzioni sono disponibili, per poterle visualizzare tutte è possibile utilizzare il browser apposito cliccando sulla barra di navigazione su Help/Function Browser, oppure premendo MAIUSC+F1.

Octave. La maggior parte delle funzioni sono presenti su Octave con lo stesso nome.

```
>>> log(2)
ans =  0.69315
```

Però, come già sottolineato, il formato short differisce per una cifra decimale, è quindi consigliabile usare il formato long. Inoltre in Octave per visualizzare la lista è necessario andare sull'Help, sempre sulla barra di navigazione: Help/Octave Help (anche premendo F1) e quindi in fondo all'indice delle pagine di aiuto, dopo le appendici, cliccare su *function index*.
♠

Per avere informazioni su funzioni specifiche è possibile usare due comandi: `help` e `lookfor`, del primo abbiamo già parlato, il secondo invece è molto utile poiché permette di verificare tutti i comandi che si riferiscano ad una parola chiave. La sintassi è: `lookfor nome`, dove `nome` può specificare il nome di un comando o di una parola chiave, in inglese.

```
>> lookfor absolute
abs                          - Absolute value.
genelowvalfilter             - filters genes with low absolute expression levels.
mamadnorm                    - normalizes microarray data by median absolute deviation (MAD).
abs2active                   - Convert constraints from absolute format to active format.
active2abs                   - Convert constraints from active format to absolute format.
imabsdiff                    - Absolute difference of two images.
mae                          - Mean absolute error performance function.
dmae                         - Mean absolute error performance derivative function.
circlepick                   - Pick bad triangles using an absolute tolerance
mad                          - Mean/median absolute deviation.
>>
```

15.3.3 Modificare, Salvare e Caricare il Workspace.

Nel Workspace sono presenti tutte le variabili che noi dichiariamo con la semplice sintassi descritta. Ogni volta che eseguiamo un assegnamento modifichiamo il Workspace, nello specifico modifichiamo la variabile a sinistra dell'assegnamento, se una funzione non ha tale variabile di default viene modificata `ans`.

Esempio. Modifica del Workspace.

```
>> x = 1

x =

     1
>> x = [1 2 3]

x =

     1     2     3
```

In questo esempio la variabile `x` viene inizializzata come un array di un solo elemento con valore 1, il comando successivo la modifica in un array di 3 elementi. D'ora in avanti, durante il lavoro, ogni volta che utilizzeremo `x` questa sarà come l'ultima modifica effetuata.

Per evitare di cancellare dati importanti è consigliabile dare nomi diversi alle variabili e sopratutto significativi, quando le variabili in gioco cominciano a diventare numerose è possibile usare il comando `exist` per verificare l'esistenza o meno.

Esempio. Ricerca di una variabile.

```
>> exist x

ans =

     1

>> exist y

ans =

     0

>>
```

In questo caso `x` esiste (1 = vero) mentre `y` no.

E' possibile che si debba interrompere una sessione di lavoro, o che questa debba essere suddivisa in intervalli temporali per vari motivi. Ovviamente

non occorre ogni volta che si riavvia Matlab reinserire tutte le variabili manualmente per ricominciare il lavoro, con il comando `save nomefile` è infatti possibile salvare l'intero contenuto del Workspace all'interno del file specificato. Per ricarice i dati salvati si usa il comando `load nomefile`.

Esercizio. Si provi la seguente lista di comandi:

```
>> x = 1;
>> save walker
>> clear
>> exist x
>> load walker
>> exist x
```

15.4 Manipolare i dati.

Come abbiamo già detto, i dati in Matlab sono tutti espressi in array. Con la sintassi `>> x = 2;` si genera un array monodimenzionale di un solo elemento (1×1). Ma come si creano vettori di più elementi?

```
>> x = [1 2 3 4 5];
>> y = [1,2,3,4,5]

y =

     1     2     3     4     5
```

Questi due comandi generano due vettori riga (`x` e `y`) completamente uguali a parte per il nome, ovvero all'interno di parentesi elencare numeri –elementi– separati da uno spazio o una virgola genera un vettore riga.

Per generare un vettore colonna invece gli elmenti devono essere separati da un "a capo" o da un punto e virgola.

```
>> x = [1
2
3
4
5];
>> y = [1;2;3;4;5]

y =

     1
     2
     3
     4
     5

>>
```

Per generare una matrice si devono mischiare le due modalità, ovvero inserire una colonna di righe o una riga di colonne (è possibile usare anche le variabili come righe o colonne).

```
>> Y = [1 2 ; 3 4]

Y =

     1     2
     3     4
```

15.4.1 Trasposta.

Dato un array (vettore o matrice) è possibile calcolarne semplicemente la *trasposta* applicando un apice (') alla fine della definizione del vettore o al nome del vettore.

```
>> y = [1;2;3;4;5]'

y =

     1     2     3     4     5

>> x'

ans =

     1     2     3     4     5
```

15.4.2 Accesso agli elementi e dimensioni

Per accedere ad un elemento specifico di un vettore o di una matrice è sufficiente utilizzare le parentesi rotonde applicate al nome dell'array indicando l'indice o gli indici dell'elemento desiderato.

```
>> x(2)

ans =

     2

>> X(2,1)

ans =

     0
```

Dato un array è possibile conoscerne la lunghezza con il comando `length nomevettore`, oppure le dimensioni con il comando `size nomevettore`.

```
>> length x

ans =

     1

>> size X

ans =

     1     1
```

15.4.3 linspace(a,b,n).

```
>> x = linspace(a,b,n);
```

Crea un vettore equispaziato da `a` a `b` di n elementi. Il parametro `n` è facoltativo, se non presente di default Matlab imposta 100 elementi.

15.4.4 logspace(a,b,n).

```
>> x = linspace(a,b,n);
```

Crea un vettore di n elementi intervallati logaritmicamente da $10^{\texttt{a}}$ a $10^{\texttt{b}}$. Il parametro `n` è facoltativo, se non presente di default Matlab imposta 50 elementi. Utile per creare grafici in scala logaritmica per dati di grandi dimensioni.

15.4.5 Assegnazione a blocchi.

Matlab da la possibilità di eseguire operazioni fondamentali senza il bisogno di accedere agli elementi di vettori e matrici direttamente magari usando comandi come cicli `for` o `while`. Ad esempio è possibile creare matrici con assegnazioni a blocchi.

```
>> x = [1 2];
>> y = [3 4];
>> z = [x y x]

z =

     1     2     3     4     1     2

>> B = [x;y]

B =

     1     2
     3     4
```

Nota Bene. I blocchi devono essere di dimensioni compatibili.

```
>> C = [y' B; 5 6 7]

C =

     3     1     2
     4     3     4
     5     6     7

>> C = [y B; 5 6 7]
??? Error using ==> horzcat
CAT arguments dimensions are not consistent.
```

15.4.6 Notazione "due punti" o "colon".

E' possibile usare, alternativamente a `linspace` la notazione "colon" per creare vettori equispaziati, utile sopratutto per array di piccole dimensioni, la sintassi è la seguente:

```
vettore = inizio : passo : fine
```

Il `passo` se non specificato di default è 1, ovvero il vettore che ne risulta avrà come valore iniziale `inizio` e di seguito gli altri elementi saranno incrementati di 1 fino all'ultimo elemento (`fine`), un valore negativo invece crea il vettore dal valore più alto a quello più basso.

```
>> x = (2:5)

x =

     2     3     4     5

>> y = (5:2)

y =

   Empty matrix: 1-by-0

>> z = (5:-1:2)

z =

     5     4     3     2
```

Nota Bene. Si noti che `y` è un vettore vuoto di dimensioni 1×0

Un altro utilizzo per la notazione "colon" è per selezionare un'intera colonna o una riga in una matrice.

```
>> C(:,1)

ans =

     3
     4
     5

>> C(1,:)

ans =

     3     1     2
```

Oppure per la selezione di un "range" all'interno di un vettore o una matrice.

```
>> C(1:2,2:3)

ans =

     1     2
     3     4
```

Qui ad esempio abbiamo estratto la sottomatrice di `C` data dalla prima alla seconda riga e dalla seconda alla terza colonna.

```
>> A = [1 2 3; 4 5 6; 7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9
```

L'ultima utilità che vediamo per la notazione "colon" è quella per la modifica di una matrce o un vettore, partiamo dalla matrice `A` di cui sopra ed applichiamo la notazione per alcune modifiche.

```
>> A(1,:) = (2:2:6)

A =

     2     4     6
     4     5     6
     7     8     9
```

Qui alla prima riga abbiamo sostituito il vettore dato da `2:2:6`.

```
>> A(:,3) = (1:3)'

A =

     1     2     1
     4     5     2
     7     8     3
```

Alla terza colonna abbiamo sostituito il vettore colonna `(1:3)'`.

```
>> A(:,1) = []

A =

     2     3
     5     6
     8     9
```

Come si nota, è possibile anche eliminare un intero vettore colonna o riga, modificando la struttura della matrice.

15.4.7 zeros(), ones(), rand() e eye()

Alcune funzioni utili per costruire vettori e matrici particolari sono:

`zeros(righe, colonne)`	crea un array contenete solo elementi `0`,
`ones(righe, colonne)`	crea un array contenete solo elementi `1`,
`rand(righe, colonne)`	crea un array contenete elementi casuali,
`eye(righe, colonne)`	crea una matrice identità.

I parametri `righe` e `colonne` indicano il numero di righe e di colonne della matrice risultante.

```
>> zeros(2,2)

ans =

     0     0
     0     0

>> ones(2,2)

ans =

     1     1
     1     1

>> rand(2,2)

ans =

    0.8147    0.1270
    0.9058    0.9134

>> eye(2,4)

ans =

     1     0     0     0
     0     1     0     0
```

Come si può notare è possibile dare dimensioni differenti dalla canonica forma quadrata, anche nella matrice identità.

15.4.8 Operatori su vettori e matrici.

Sia `x` un vettore di n elementi tale che $x_i \in$ `x` per ogni $i \in [0, n-1]$, sia `A` una matrice di $n \times m$ elementi. In Matlab sono definite le seguenti funzioni:

`a = sum(x)`	genera lo scalare $\mathtt{a} = \sum_i x_i$,
`a = prod(x)`	genera lo scalare $\mathtt{a} = \prod_i x_i$,
`a = max(x)`	genera lo scalare $\mathtt{a} = \max x_i, \quad i = 1, \ldots n$,
`a = min(x)`	genera lo scalare $\mathtt{a} = \min x_i, \quad i = 1, \ldots n$,
`a = norm(x)`	genera lo scalare $\mathtt{a} = \|x\|_2$,
`a = norm(x,1)`	genera lo scalare $\mathtt{a} = \|x\|_1$,
`a = norm(x,inf)`	genera lo scalare $\mathtt{a} = \|x\|_\infty$,
`A = diag(x)`	crea una matrice con diagonale il vettore `x`,
`a = norm(A)`	genera lo scalare $\mathtt{a} = \|A\|_2$,
`a = norm(A,)1`	genera lo scalare $\mathtt{a} = \|A\|_1$,
`a = norm(A,inf)`	genera lo scalare $\mathtt{a} = \|A\|_\infty$,
`x = sum(A)`	genera il vettore riga $\mathtt{x} = x_{i,j} = \sum_i a_{i,j}, \quad j = 1, \ldots n$,
`x = max(A)`	genera il vettore riga $\mathtt{x} = x_{i,j} = \max a_{i,j}, \quad j = 1, \ldots n$,
`x = min(A)`	genera il vettore riga $\mathtt{x} = x_{i,j} = \min a_{i,j}, \quad j = 1, \ldots n$,
`x = diag(A)`	genera il vettore colonna dato dalla diagonale di `A`,
`B = abs(A)`	genera la matrice dei valori assoluti,
`U = triu(A)`	genera la matrice triangolare superiore con gli elementi non nulli uguali a quelli di `A`.
`U = tril(A)`	genera la matrice triangolare inferiore con gli elementi non nulli uguali a quelli di `A`.

15.4.9 Operatori algebrici sugli array.

Siano `A` e `B` array (matrici o vettori), Matlab definisce due tipi generici di operazioni: le operazioni puntuali e le operazioni tra array. Le operazioni tra array sono quelle con la consueta sintassi, le operazioni puntuali invece hanno come sintazzi un punto prima dell'operazione desiderata. Ad esempio un prodotto matriciale riga per colonna si effettua con il seguente comando: `A*B`, se invece avessimo voluto moltiplicare elemento per elemento (nella stessa posizione) il comando sarebbe stato: `A.*B`.

```
>> A = 2*ones(2,2)

A =

     2     2
     2     2

>> A.^2

ans =

     4     4
     4     4

>> A^2
```

```
ans =

     8     8
     8     8
```

Si noti che `A.^2` equivale al comando `A.*A`, ovvero moltiplica ciascun elemento per se stesso, invece `A^2` equivale a `A*A`, ovvero prodotto riga per colonna.

Come il prodotto è possibile utilizzare tutte le operazioni su vettori, matrici, scalari, etc.

Capitolo 16

Lavorare con Matlab.

16.1 Funzioni dell'algebra lineare.

Come si è visto nell'introduzione a Matlab, i dati su cui si lavora sono matrici e vettori. In questa sezione vedremo quindi l'utilizzo delle principali funzioni Matlab basate sull'algebra lineare.

- `d = det(A)` calcola il determinante di A.
- `B = inv(A)` calcola la matrice inversa di A.
- `H = hilb(n)` costruisce la matrice di Hilbert: $H(i,j) = \frac{1}{i+j-1}$.
- `V = vander(v)` costruisce la matrice di Vandermonde di ordine $n =$ `length(v)`: $V(i,j) = v(i)^{(n-j)}$.
- `[M,D] = eig(A)` calcola gli autovalori e autovettori di A.
- `A\b` calcola la soluzione del sistema lineare $Ax = b$ con l'eliminazione di Gauss.

Esempio. Matrice di Hilbert e suo determinante.

```
>> format rat
>> A = hilb(4)

A =

       1            1/2          1/3          1/4
       1/2          1/3          1/4          1/5
       1/3          1/4          1/5          1/6
       1/4          1/5          1/6          1/7

>> m1 = A(1,1)
```

```
m1 =

       1

>> m2 = A(1:2,1:2), dm2 = det(m2)

m2 =

       1              1/2
       1/2            1/3


dm2 =

       1/12

>> m3 = A(1:3,1:3), dm3 = det(m3)

m3 =

       1              1/2            1/3
       1/2            1/3            1/4
       1/3            1/4            1/5


dm3 =

       1/2160

>> m4 = det(A)

m4 =

       1/6048000

>>
```

In questo esempio vediamo un nuovo formato: `format rat` che visualizza la notazione razionale dei nostri dati.

Le funzioni `hilb()` e `det()`, si comportano esattamente come ci si aspetta: con la prima si costruisce una matrice di ordine 4, mentre con la seconda un *vettore* 1×1 il cui unico elemento è il valore del determinante.

Esempio. Matrice di Vandermonde e calcolo dell'inversa.

```
>> v = [1:4]; V = vander(v)

V =

       1              1              1              1
       8              4              2              1
      27              9              3              1
      64             16              4              1

>> inv_V = inv(V)

inv_V =
```

```
      -1/6            1/2          -1/2           1/6
       3/2           -4             7/2          -1
     -13/3           19/2          -7            11/6
       4             -6             4            -1

>>
```

`v` è il vettore $[1, 2, 3, 4]$, viene quindi creata la matrice V di Vandermonde come ci si aspetta dal comando `vander()`, quindi con la funzione `inv()` si calcola l'inversa.

Esempio. Autovalori e autovettori.

```
>> v = [0.1:0.1:0.5]; [M,D] = eig(vander(v))

M =

    -641/1844      -903/1345      1206/1345       521/892      -1236/1405
    -521/1355      -397/702        739/2270      -174/235        411/1310
   -3585/8339      -963/2413     -1189/10328      129/394        615/1933
    -605/1244      -249/1691     -1018/3795      -243/4162      -1259/7785
    -976/1753        24/109        194/2795        87/25672       95/5587


D =

     939/535          0              0              0              0
       0           -452/1511         0              0              0
       0              0            169/3621         0              0
       0              0              0             8/32799         0
       0              0              0              0           -100/20753

>>
```

Esempio. Risoluzione di un sistema lineare.

```
>> A = vander(1:3); b=[3;7;13];
>> x = A\b

x =

       1
       1
       1

>> A*x

ans =

       3
       7
      13
```

16.2 File Diario.

Il comando `diary` permette di registrare l'attività svolta in una sessione di lavoro con Matlab. L'utilizzo è il seguente:

- `diary` registra una sessione di lavoro e la salva in un file di nome diary.
- `diary nomefile` registra una sessione di lavoro e la salva in un file di nome `nomefile`.
- `diary off` disattiva la registrazione.
- `diary on` riattiva la registrazione.

I file vengono salvati in formato ASCII.

Esempio. Utilizzo del comando `diary`.

```
>> diary esempio
>> x = linspace(0,pi,10);
>> y = (15120 -6900*x.^2 + 313*x.^4)./(15120+660*x.^2 +13*x.^4);
>> [x',y']

ans =

       0             1
    355/1017       857/912
    710/1017      1313/1714
    355/339     441131/882261
   1420/1017       207/1192
   1775/1017      -109/628
    710/339       -546/1093
   1102/451       -554/725
   2840/1017    -15007/16079
    355/113      -4069/4143

>> diary off
>> % fine registrazione
>>
>> type esempio

x = linspace(0,pi,10);
y = (15120 -6900*x.^2 + 313*x.^4)./(15120+660*x.^2 +13*x.^4);
[x',y']

ans =

       0             1
    355/1017       857/912
    710/1017      1313/1714
    355/339     441131/882261
   1420/1017       207/1192
   1775/1017      -109/628
    710/339       -546/1093
   1102/451       -554/725
   2840/1017    -15007/16079
```

```
    355/113      -4069/4143

diary off

>>
```

Il comando `type nomefile` infine permette di visualizzare il contenuto di `nomefile`.

16.3 Tipi di file Matlab.

In Matlab ci sono tre tipi di file:

- Mat File, con estensione `.mat`.

  ```
  >> save esempio
  ```

  salva il file `esempio.mat`, contente l'ambiente di lavoro Matlab.

- M-file, con estensione `.m`, per memorizzare programmi e funzioni Matlab.

- File dati, con estensione `.txt`, in formato ASCII.

In Matlab possiamo operare in due modi, in modalità interattiva (a riga di comando dalla shell) oppure tramite l'esecuzione di programmi o script.

16.4 Creare e utilizzare M-file.

Gli M-file possono contenere script o funzioni. La principale differenza "semantica" tra lo script e la function è che nel primo le variabili sono globali, e quindi vengono salvate nel workspace, nelle function invece le variabili sono memorizzate localmente e non vanno a sporcare l'ambiente di lavoro.

Inoltre una function ha delle variabili in input ed output, mentre lo script modifica e crea variabili date nel workspace, in una function l'unico modo di modificare variabili è quello di usare il meccanismo di output.

Per creare un M-file è necessario:

1. selezionare `New` dal menù `File` e poi `M-file`;
2. digitare i comandi che compongono lo script o la function;
3. selezionare `save` dal menù `File` della nuova finestra;
4. dare un nome coerente al file se si tratta di uno script, o il nome esatto della function.

16.4.1 Function.

Le function Matlab sono utili quando occorre ripetere più volte una serie di comandi, in modo particolare quando questi comandi hanno un senso particolare, ad esempio un algoritmo.

Per quanto riguarda la sintassi, la prima riga dell'M-file deve contenere la parola chiave `function` seguita dai parametri di output racchiusi tra parentesi quadre, il nome della funzione ed i nomi delle variabili in input tra parentesi rotonde.

Esempio. Il codice seguente, salvato in un file `risolvi.m`, implementa una funzione che genera il vettore soluzione `x` del problema lineare $Ax = b$.

```
function [x] = risolvi(A,b)
% risolvi risolve il generico problema Ax = b
% con metodo di Gauss.

x = A\b;

end
```

Il suo utilizzo è il seguente.

```
>> A = rand(4);
>> b = rand(4,1);
>> x1 = A\b

x1 =

   17.2819
    0.8395
  -15.9067
    1.0883

>> x = risolvi(A,b)

x =

   17.2819
    0.8395
  -15.9067
    1.0883

>>
```

Questo è solo un esempio di come possono essere utilizzate le funzioni, in genere convengono quando l'algoritmo richiede più linee di codice.

Esempio. La seguente funzione, invece è più complessa, ha come parametri di output due variabili: A_n e P. Di conseguenza il metodo di invocazione è differente e, come si può notare, ha molte più righe di codice ed ha più senso di essere scritta come funzione di quella vista in precedenza.

```
function [An,P] = permuta(A)
% Data una matrice A analizza l'elemento
% di indice (1,1) e se è uguale a zero effettua uno scambio di righe.
% Utilizza la tecnica del pivoting parziale.

n = size(A);
An = A;
P = eye(n(1));

if(A(1,1)==0)
   [ma, j] = max(abs(A(:,1)));
   if(ma ~= 0)
    An(1,:) = A(j,:);
    An(j,:) = A(1,:);
    P(1,:) = P(j,:);
    P(j,:)= eye(1,n);
   end
end
```

Ecco come si utilizza, da notare che l'output è un array di due elementi.

```
>> A = [ 0 1 1 2 4
1 0 3 3 3
5 5 5 5 5]

A =

     0     1     1     2     4
     1     0     3     3     3
     5     5     5     5     5

>> [An, P] = permuta(A)

An =

     5     5     5     5     5
     1     0     3     3     3
     0     1     1     2     4


P =

     0     0     1
     0     1     0
     1     0     0
```

I dati di input, se si verifica il workspace, sono inalterati. Gli unici dati aggiunti sono quelli di output, delle variabili interne alla funzione non vi è traccia.

16.4.2 Script.

Gli script Matlab sono di fatto file di comandi, contengono una sequenza di comandi Matlab, comprese anche funzioni definite da utente.

Tutte le variabili contenute negli script vengono salvate nel workspace. Vediamo dunque un esempio di prima in forma di script.

Esempio. Il seguente codice è inserito nel file `esempio.m`.

```
A = rand(4)
b = rand(4,1)
x = A\b
```

Per eseguirlo è sufficiente digitare `esempio`, ovvero il nome dello script salvato nella directory corrente.

```
>> esempio

A =

    0.3804    0.5308    0.5688    0.1622
    0.5678    0.7792    0.4694    0.7943
    0.0759    0.9340    0.0119    0.3112
    0.0540    0.1299    0.3371    0.5285


b =

    0.1656
    0.6020
    0.2630
    0.6541


x =

   -0.7515
    0.0062
    0.5057
    0.9902
```

Parte IV

Esercizi.

Capitolo 17

Esercitazioni di Laboratorio Computazionale Numerico.

17.1 Esercitazione I.

1. Supponendo che le variabili a, b, c, d, e, f, g siano scalari, scrivere istruzioni di assegnazione in Matlab per calcolare il valore delle seguenti espressioni:

$$x = 1 + \frac{a}{b} + \frac{c}{f^2} \qquad s = \frac{b-a}{d-c} \qquad z = \left(1 - \frac{1}{e^5}\right)^{-1}$$

$$r = \frac{1}{\frac{1}{a} + \frac{1}{b} + \frac{1}{c} + \frac{1}{d}} \qquad y = ab \cdot \frac{1}{c} \cdot \frac{f^2}{2} \qquad t = 7\left(g^{\frac{1}{3}}\right) + 4g^{0.58}$$

con $a = 1.12$, $b = 2.34$, $c = 0.72$, $d = 0.81$, $e = 3$, $f = 19.83$, $g = 20$. Visualizzare i risultati in format short e format long.

Svolgimento.

```
>> a = 1.12;
>> b = 2.34;
>> c = 0.72;
>> d = 0.81;
>> e = 3;
>> f = 19.83;
>> g = 20;
>>
>> x = 1 + (a/b) + (c/f^2)

x =

    1.4805

>> s = (b-a)/(d-c);
>> z = (1-1/e^5)^-1;
```

```
>> r = 1/(1/a + 1/b + 1/c + 1/d);
>> y = a*b*(1/c)*((f^2)/2);
>> t = 7*(g^(1/3))+4*g^(0.58);
>>
>> x, s, z, r, y, t

x =

    1.4805


s =

   13.5556


z =

    1.0041


r =

    0.2536


y =

  715.6766


t =

   41.7340

>> format long
>>
>> x, s, z, r, y, t

x =

   1.480463473251643


s =

  13.555555555555541


z =

   1.004132231404959


r =

   0.253571274994625


y =

     7.156765979999999e+02
```

```
t =

  41.733956653314806

>>
```

2. Dopo aver cancellato le variabili del workspace, individuare, descrivere e correggere gli errori nelle seguenti istruzioni di assegnazione:

   - `a=2y+(((3+1)9)`,
   - `b==2*sin[3]`,
   - digitare `c=e^0.5` per calcolare $e^{0.5}$ con e numero di Nepero,
   - per il calcolo di $log(4 - \frac{8}{4-2})$ digitare `d=log(4-8/4*2)`.

   Svolgimento.

```
>> clear x s z r y t
>>
>> a = 2y+(((3+1)9)
??? a = 2y+(((3+1)9)
         |
Error: Unexpected MATLAB expression.
```

In questo comando abbiamo due tipi di errori: innanzitutto le moltiplicazioni (ove `2y` *si intende ovviamente* $2 \cdot y$*) si effettuano con l'operatore* `*`*, inoltre le parentesi non sono bilanciate.*

```
>> b==2*sin[3]
??? b==2*sin[3]
            |
Error: Unbalanced or unexpected
parenthesis or bracket.
```

La funzione sin*, come tutte le funzioni in Matlab, utilizza le parentesi rotonde (le quadre sono utilizzate per altri scopi).*

```
>> c = e^0.5

c =

   1.732050807568877
```

`e` *in questo caso è uno scalare definito da utente, il numero di Nepero in Matlab è calcolato dalla dunzione* `exp()` *con argomento* 1*:* `exp(1)`*.*

```
>> d = log(4-8/4*2)

d =
```

```
    -Inf

>>
```

Gli operatori `/` *e* `*` *hanno la stessa precedenza, quindi la sopracitata istruzione calcola* $\log(4-\frac{8}{4}\cdot 2)$ *eseguendo in ordine (da sinistra verso destra) le operazioni. L'istruzione corretta è quindi* `d=log(4-8/(4*2))`.

3. Mediante una sequenza di istruzioni di assegnazione in Matlab:
   - Calcolare il raggio di una sfera che ha un volume del 30% più grande di una sfera di raggio 5 cm.
   - Considerando le seguenti approssimazioni polinomiali della funzione e^x :

     $$e^x \simeq p_1(x) := 1 + x, \quad e^x \simeq p_2(x) := 1 + x + \frac{x^2}{2}$$

     si calcolino l'errore assoluto

     $$e_a(x) = |e^x - p_i(x)| \quad i = 1, 2$$

     e l'errore relativo

     $$e_r(x) = \frac{|e^x - p_i(x)|}{e^x} \quad i = 1, 2$$

     in $x = 0, 1$.
   - Calcolare le radici delle equazioni:

     $$2t^2 - 4t - 1 = 0 \quad x^4 + 2x^2 - 3 = 0 \quad x^3 = 2197.$$

Svolgimento.

- ```
  >> v1 = (4/3)*pi*(5^3);
  >> v2 = v1*(130/100);
  >> r2 = ((3/4)*(1/pi)*v2)^(1/3)

  r2 =

     5.456964415305529

  >>
  ```
- ```
  >> p10 = 1 + 0;
  >> p11 = 1 + 1;
  >> p20 = 1 + 0 + (0^2)/2;
  >> p21 = 1 + 1 + (1^2)/2;
  >>
  >> e_a10 = abs(exp(0) - p10)
  ```

```
e_a10 =

     0

>> e_a11 = abs(exp(1) - p11)

e_a11 =

   0.718281828459046

>> e_a20 = abs(exp(0) - p20)

e_a20 =

     0

>> e_a21 = abs(exp(1) - p21)

e_a21 =

   0.218281828459046

>> e_r10 = abs(exp(0) - p10)/(exp(0))

e_r10 =

     0

>> e_r11 = abs(exp(1) - p11)/(exp(1))

e_r11 =

   0.264241117657115

>> e_r20 = abs(exp(0) - p20)/(exp(0))

e_r20 =

     0

>> e_r21 = abs(exp(1) - p21)/(exp(1))

e_r21 =

   0.080301397071394

>>
```

– *Utilizzando* $x_{1,2} = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$ *e alcune operazioni algebriche:*

```
>> t1 = (4 + sqrt((-4)^2-4*2*(-1)))/(2*2);
>> t2 = (4 - sqrt((-4)^2-4*2*(-1)))/(2*2);
>> t = [t1 t2]

t =

   2.224744871391589  -0.224744871391589

>>
>> y1 = (-2 + sqrt((2)^2-4*2*(-3)))/(2*2);
>> y2 = (-2 - sqrt((2)^2-4*2*(-3)))/(2*2);
>> x1 =  sqrt(y1);
```

```
>> x2 = -x1;
>> x = [x2 x1]

x =

  -0.907124939317785   0.907124939317785

>>
>> x = (2197)^(1/3)

x =

  12.999999999999998

>>
```

4. Generare il vettore riga e il vettore colonna y di elementi equidistanti $1, 2, ..., 10$ e $10, 9, ..., 1$ rispettivamente e farne il prodotto scalare. Generare inoltre il vettore colonna z costituito dai valori della funzione seno in 11 elementi equidistanti nell'intervallo $[0, 1]$.

   Svolgimento.

```
>> y1 = linspace(1,10,10)';
>> y2 = linspace(10,1,10);
>> y1'.*y2

ans =

    10    18    24    28    30    30    28    24    18    10

>>
>> z = sin(linspace(0,1,11))'

z =

                   0
   0.099833416646828
   0.198669330795061
   0.295520206661340
   0.389418342308651
   0.479425538604203
   0.564642473395035
   0.644217687237691
   0.717356090899523
   0.783326909627483
   0.841470984807897

>>
```

5. Generare il vettore riga x e il vettore colonna y di elementi equidistanti $25, 28, 31, ..., 91$ e $100, 98, 96, ..., 10$ rispettivamente. Generare inoltre il vettore colonna z costituito da 33 elementi equidistanti nell'intervallo $[-15, -10]$.

Svolgimento.

```
>> x = (25:1:91);
>> y = (100:-1:10);
>> z = linspace(-15,-10, 33);
>>
```

6. Applicare la formula di de Moivre $z^n = \rho^n(\cos(n\theta) + i\sin(n\theta))$ al numero complesso $z = 1 + i = \sqrt{2}(\cos(\frac{\pi}{4}) + i\sin(\frac{\pi}{4}))$ con $n = 60$ e controllare il risultato ottenuto con Matlab.

17.2 Esercitazione II.

1. Dato il vettore x di elementi equidistanti $-5, -4, \ldots, 8, 9$ determinare l'elemento massimo, minimo, di valore assoluto massimo, di valore assoluto minimo, la somma di tutti gli elementi e la somma dei valori assoluti di tutti gli elementi. **Svolgimento.**

```
>> x = linspace(-5,9,15)

x =

    -5    -4    -3    -2    -1     0     1     2     3     4     5     6     7     8     9

>> % elmento massimo
>> max(x)

ans =

     9

>> % elmento minimo
>> min(x)

ans =

    -5

>> % valore assoluto massimo
>> max(abs(x))

ans =

     9

>> % valore assoluto minimo
>> min(abs(x))

ans =

     0

>> % somma degli elementi di x
>> sum(x)
```

```
ans =

    30

>> % somma dei valori assoluti di x
>> sum(abs(x))

ans =

    60
```

2. Dopo aver definito il vettore $x = [1 : -0.1 : 0]$, spiegare il significato dei seguenti comandi Matlab:

```
>> x([1 4 3]);
>> x([1:2:7 10]) = zeros(1,5);
>> x([1 2 5]) = [0.5*ones(1,2) -0.3];
>> y = x(end:-1:1);
```

Svolgimento.

```
>> x([1 4 3])

ans =

   1.000000000000000   0.700000000000000   0.800000000000000
```

Il comando seleziona gli elementi di indici 1, 4 *e* 3 *e li inserisce in un vettore temporaneo.*

```
>> x([1:2:7 10]) = zeros(1,5)

x =

  Columns 1 through 4

                  0   0.900000000000000                   0   0.700000000000000

  Columns 5 through 8

                  0   0.500000000000000                   0   0.300000000000000

  Columns 9 through 11

   0.200000000000000                   0                   0
```

Seleziona dal primo al settimo elemento di x *con passo* 2 *ed il decimo elemento e li sostituisce con gli elementi di un vettore di cinque elementi costituito di soli zeri.*

```
>> x([1 2 5]) = [0.5*ones(1,2) -0.3]

x =

  Columns 1 through 4
```

```
 0.500000000000000   0.500000000000000                   0   0.700000000000000

Columns 5 through 8

-0.300000000000000   0.500000000000000                   0   0.300000000000000

Columns 9 through 11

 0.200000000000000                   0                   0
```

Seleziona gli elementi di x di indice 1, 2 e 5 e li sostituisce con gli elementi del vettore $[0.5, 0.5, 0.3]$.

```
>> y = x(end:-1:1)

y =

  Columns 1 through 4

                  0                   0   0.200000000000000   0.300000000000000

  Columns 5 through 8

                  0   0.500000000000000  -0.300000000000000   0.700000000000000

  Columns 9 through 11

                  0   0.500000000000000   0.500000000000000
```

Crea un nuovo vettore y a cui assegna gli elementi del vettore x in ordine invertito.

3. Usare le variabili e le operazioni vettoriali per osservare la convergenza in $\mathbb{N}$ delle successioni:

$$\left(1+\frac{1}{n}\right)^n \to e, \quad \frac{4n}{n+2} \to 4, \quad \log\left(1+\sqrt{\frac{n}{n+1}}\right) \to \log 2.$$

Svolgimento.

```
>> n = linspace(1,10000,10000);
>> one = ones(1,10000);
>> e = (one + one./n).^n;
>> exp(1) - e(10000)

ans =

   1.3590e-04

>> q = (4*n)./(n+2*one);
>> 4 - q(10000)

ans =

   7.9984e-04
```

```
>> l=log(one + sqrt(n./(n+one)));
>> log(2) -l(10000)

ans =

   2.4998e-05
```

4. Usare le variabili e le operazioni vettoriali per osservare, per qualche $n \in \mathbb{N}$, che la somma dei primi n numeri naturali è:

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2},$$

e che la serie (detta di Mengoli):

$$\sum_{i=0}^{n} \frac{1}{(i+1)(i+2)} = 1 - \frac{1}{n+2},$$

converge a 1.

Svolgimento.

```
>> n = 10;
>> sum = n*(n+1)/2

sum =

    55

>> sum_1 = 0;
>> for i=1:10
sum_1 = sum_1 +i;
end
>> sum_1

sum_1 =

    55
```

`sum` *e* `sum_1` *risultano uguali.*

```
>> n = linspace(1,1000,1000);
>> one = ones(1,1000);
>> one - one./(one*2 + n);
>> meng = one - one./(one*2 + n);
>>
>> meng_1 = 0;
>> for i=0:1000
meng_1 = meng_1 + 1/((i+1)*(i+2));
end
>> meng(1000), meng_1

ans =
```

```
    0.9990

meng_1 =

    0.9990

>>
```

5. Definire la matrice:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

e comprendere il significato dei seguenti comandi Matlab:

```
>> size(A);
>> B = A.*A;
>> B = A*A;
>> B = A'*A;
>> A(1:2,4), A(:,3), A(1:2,:), A(:,[2 4]), A([2 3 3]);
>> A(3,2) = A(1,1);
>> A(2,:) = A(2,:)-A(2,1)/A(1,1)*A(1,:);
>> [r,c] = max(A);
```

Svolgimento.

```
>> A = [1 2 3 4; 5 6 7 8; 9 10 11 12]

A =

     1     2     3     4
     5     6     7     8
     9    10    11    12

>> size(A)

ans =

     3     4
```

Costruita la matrice `A` *il comando* `size(A)` *restituisce il numero di righe e di colonne in un vettore* 2×1.

```
>> B = A.*A

B =

     1     4     9    16
    25    36    49    64
    81   100   121   144
```

L'operatore `.*` *effettua il prodotto puntuale tra gli elementi delle due matrici, in questo caso due matrici* `A` *e salva il risultato nella matrice* `B`. *Un'operazione analoga sarebbe potuta essere* `B=A.^2` .

```
>> B = A*A
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

L'operatore `*` *effettua il prodotto righe per colonne di due matrici, in questo caso, essendo la matrice* A *della forma* 3×4, *le dimnesioni non sono compatibili con tale operazione.*

```
>> B = A'*A

B =

   107   122   137   152
   122   140   158   176
   137   158   179   200
   152   176   200   224
```

In questo caso invece, applicando l'operatore `'` *si esegue prima la trasposizione della matrice* A, *e solo in seguito (tale operatore ha infatti la precedenza su ogni altro) quello di prodotto riga per colonna. L'operazione risulta quindi valida e viene eseguito ciò che ci si aspetta.*

```
>> A(1:2,4), A(:,3), A(1:2,:), A(:,[2 4]), A([2 3 3])

ans =

     4
     8


ans =

     3
     7
    11


ans =

     1     2     3     4
     5     6     7     8


ans =

     2     4
     6     8
    10    12


ans =

     5     9     9
```

Le operazioni di cui sopra sono delle selezioni di elementi della matrice, che vengono salvati in un array temporaneo. L'operatore `( , )` *applicato ad una matrice come sopra seleziona gli elementi indicati dagli indici di riga (prima della virgola) e di colonna (dopo la virgola).*

```
>> A(3,2) = A(1,1)

A =

     1     2     3     4
     5     6     7     8
     9     1    11    12
```

Questa operazione assegna all'elemnto di indice $3, 2$ *il valore di indice* $1, 1$ *della matrice* A*.*

```
>> A(2,:) = A(2,:)-A(2,1)/A(1,1)*A(1,:)

A =

     1     2     3     4
     0    -4    -8   -12
     9     1    11    12
```

Alla seconda riga della matrice, assegna una combinazione lineare della seconda e della prima.

```
>> [r,c] = max(A)

r =

     9     2    11    12


c =

     3     1     3     3
```

La funzione `max()` *crea un array con i valori massimi di ogni colonna. La scrittura* `[r,c]=` *prima della funzione, invoca la funzione che salva gli elementi massimi nell'array* `r` *e cerca gli indici di riga di tali elementi e li salva nell'array* `c` *.*

6. Utilizzare il comando `diag` per generare una matrice tridiagonale A di dimensione 9×9 i cui elementi della diagonale principale coincidono con -2 e quelli delle codiagonali con 1. Calcolare A^{-1} (e commentare il risultato). Successivamente scambiare in A dapprima le righe 3 e 6, e di seguito, le colonne 1 e 4.

   Svolgimento.

```
>> d = -2*ones(1,9);
>> d1 = ones(1,8);
>> A = diag(d) + diag(d1,1) + diag(d1,-1);
>> Ainv = inv(A);
>> Ainv

Ainv =

   -0.9000   -0.8000   -0.7000   -0.6000   -0.5000   -0.4000   -0.3000   -0.2000   -0.1000
   -0.8000   -1.6000   -1.4000   -1.2000   -1.0000   -0.8000   -0.6000   -0.4000   -0.2000
   -0.7000   -1.4000   -2.1000   -1.8000   -1.5000   -1.2000   -0.9000   -0.6000   -0.3000
   -0.6000   -1.2000   -1.8000   -2.4000   -2.0000   -1.6000   -1.2000   -0.8000   -0.4000
   -0.5000   -1.0000   -1.5000   -2.0000   -2.5000   -2.0000   -1.5000   -1.0000   -0.5000
   -0.4000   -0.8000   -1.2000   -1.6000   -2.0000   -2.4000   -1.8000   -1.2000   -0.6000
   -0.3000   -0.6000   -0.9000   -1.2000   -1.5000   -1.8000   -2.1000   -1.4000   -0.7000
   -0.2000   -0.4000   -0.6000   -0.8000   -1.0000   -1.2000   -1.4000   -1.6000   -0.8000
   -0.1000   -0.2000   -0.3000   -0.4000   -0.5000   -0.6000   -0.7000   -0.8000   -0.9000

>> A

A =

    -2     1     0     0     0     0     0     0     0
     1    -2     1     0     0     0     0     0     0
     0     1    -2     1     0     0     0     0     0
     0     0     1    -2     1     0     0     0     0
     0     0     0     1    -2     1     0     0     0
     0     0     0     0     1    -2     1     0     0
     0     0     0     0     0     1    -2     1     0
     0     0     0     0     0     0     1    -2     1
     0     0     0     0     0     0     0     1    -2

>> % La matrice inversa di una tridiagonale, non mantiene tale proprietâ
>>
>> tmp = A(3,:)

tmp =

     0     1    -2     1     0     0     0     0     0

>> A(3,:) = A(6,:);
>> A(6,:) = tmp;
>> tmp = A(:,1);
>> A(:,1) = A(:,4);
>> A(:,4) = tmp;
>>
>> A

A =

     0     1     0    -2     0     0     0     0     0
     0    -2     1     1     0     0     0     0     0
     0     0     0     0     1    -2     1     0     0
    -2     0     1     0     1     0     0     0     0
     1     0     0     0    -2     1     0     0     0
     1     1    -2     0     0     0     0     0     0
     0     0     0     0     0     1    -2     1     0
     0     0     0     0     0     0     1    -2     1
     0     0     0     0     0     0     0     1    -2

>>
```

7. Definire la matrice:

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 6 & 6 & 6 & 6 & 6 & 6 & 6 & 6 \\ 7 & 7 & 7 & 7 & 7 & 7 & 7 & 7 \\ 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 \end{bmatrix}$$

e successivamente:

a) generare le matrici S triangolare superiore e I triangolare inferiore i cui elementi non nulli coincidano con gli elementi omonimi di A; successivamente, porre gli elementi della diagonale principale della matrice S uguali a 0 e quelli della matrice I uguali a 1;

b) generare le matrici B_1, B_2 e B_3 rispettivamente triangolare, bidiagonale superiore e bidiagonale inferiore, i cui elementi coincidano con gli elementi omonimi di A.

Svolgimento.

```
>> A = [1:8]'*ones(1,8)

A =

     1     1     1     1     1     1     1     1
     2     2     2     2     2     2     2     2
     3     3     3     3     3     3     3     3
     4     4     4     4     4     4     4     4
     5     5     5     5     5     5     5     5
     6     6     6     6     6     6     6     6
     7     7     7     7     7     7     7     7
     8     8     8     8     8     8     8     8

>> S = triu(A)

S =

     1     1     1     1     1     1     1     1
     0     2     2     2     2     2     2     2
     0     0     3     3     3     3     3     3
     0     0     0     4     4     4     4     4
     0     0     0     0     5     5     5     5
     0     0     0     0     0     6     6     6
     0     0     0     0     0     0     7     7
     0     0     0     0     0     0     0     8

>> I = tril(A)
```

```
I =

     1     0     0     0     0     0     0     0
     2     2     0     0     0     0     0     0
     3     3     3     0     0     0     0     0
     4     4     4     4     0     0     0     0
     5     5     5     5     5     0     0     0
     6     6     6     6     6     6     0     0
     7     7     7     7     7     7     7     0
     8     8     8     8     8     8     8     8

>> S = triu(A,1)

S =

     0     1     1     1     1     1     1     1
     0     0     2     2     2     2     2     2
     0     0     0     3     3     3     3     3
     0     0     0     0     4     4     4     4
     0     0     0     0     0     5     5     5
     0     0     0     0     0     0     6     6
     0     0     0     0     0     0     0     7
     0     0     0     0     0     0     0     0

>> I = tril(A,-1);
>> I = eye(8) + I

I =

     1     0     0     0     0     0     0     0
     2     1     0     0     0     0     0     0
     3     3     1     0     0     0     0     0
     4     4     4     1     0     0     0     0
     5     5     5     5     1     0     0     0
     6     6     6     6     6     1     0     0
     7     7     7     7     7     7     1     0
     8     8     8     8     8     8     8     1

>> B1 = diag(diag(A)) + diag(diag(A,-1),-1) + diag(diag(A,1),1)

B1 =

     1     1     0     0     0     0     0     0
     2     2     2     0     0     0     0     0
     0     3     3     3     0     0     0     0
     0     0     4     4     4     0     0     0
     0     0     0     5     5     5     0     0
     0     0     0     0     6     6     6     0
     0     0     0     0     0     7     7     7
     0     0     0     0     0     0     8     8

>>

>> B2 = diag(diag(A)) + diag(diag(A,1),1)

B2 =

     1     1     0     0     0     0     0     0
     0     2     2     0     0     0     0     0
     0     0     3     3     0     0     0     0
     0     0     0     4     4     0     0     0
```

```
     0     0     0     0     5     5     0     0
     0     0     0     0     0     6     6     0
     0     0     0     0     0     0     7     7
     0     0     0     0     0     0     0     8

>> B3 = diag(diag(A)) + diag(diag(A,-1),-1)

B3 =

     1     0     0     0     0     0     0     0
     2     2     0     0     0     0     0     0
     0     3     3     0     0     0     0     0
     0     0     4     4     0     0     0     0
     0     0     0     5     5     0     0     0
     0     0     0     0     6     6     0     0
     0     0     0     0     0     7     7     0
     0     0     0     0     0     0     8     8
```

8. E matrice elementare è definita da:

$$E(\alpha, \mathbf{u}, \mathbf{v}) = I - \alpha \mathbf{u}\mathbf{v}^t \qquad \alpha \in \mathbb{R},\ \mathbf{u}, \mathbf{v} \in \mathbb{R}^n.$$

Se $\alpha \mathbf{v}^t \mathbf{u} \neq 1$, $E(\alpha, \mathbf{u}, \mathbf{v})$ è *invertibile* e la sua inversa è $E(\beta, \mathbf{u}, \mathbf{v})$ con

$$\beta = \frac{\alpha}{\alpha \mathbf{v}^t \mathbf{u} - 1}.$$

Generare le matrici elementari (e le loro inverse) associate a:

$$\mathbf{u}_1 = \begin{pmatrix} 9 \\ 1 \\ 5 \\ 1 \end{pmatrix},\ \mathbf{v}_1 = \begin{pmatrix} 3 \\ 2 \\ 5 \\ 5 \end{pmatrix},\ \alpha_1 = 7, \quad \mathbf{u}_2 = \begin{pmatrix} 2 \\ 1 \\ -3 \\ -1 \end{pmatrix},\ \mathbf{v}_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix},\ \alpha_2 = 1.$$

Svolgimento.

```
>> u = [9;1;5;1];
>> u1 = [9;1;5;1];
>> v1 = [3;2;5;5];
>> a1 = 7;
>>
>> E1 = eye(4) - a1*(u1*v1')

E1 =

  -188  -126  -315  -315
   -21   -13   -35   -35
  -105   -70  -174  -175
   -21   -14   -35   -34

>> u2 = [2;1;-3;-1];
>> v2 = [1;1;1;1];
>> a2 = 1;
>>
```

```
>> E2 = eye(4) - a2*(u2*v2');
>>
>> b1 = a1/(a1*v1'*u1-1);
>> b2 = a2/(a2*v2'*u2-1);
>>
>> invE1 = eye(4) - b1*(u1*v1');
>> invE2 = eye(4) - b2*(u2*v2');
>>
>> abs(invE1 - inv(E1))

ans =

   1.0e-14 *

    0.2331    0.9048    0.2109    0.2220
    0.0507    0.0222    0.0791    0.0805
    0.2109    0.5024    0.2331    0.2387
    0.0493    0.0201    0.0791    0.0777

>>
>> abs(invE2 - inv(E2))

ans =

   1.0e-15 *

    0.4441    0.4441    0.2220    0.2220
    0.2220    0.2220    0.1110    0.1110
    0.4441    0.4441    0.3331         0
    0.1110    0.1110         0    0.2220
```

Si può osservare che la differenza, in valore assoluto, delle matrici inverse così ottenute è prossima allo zero.

9. Sia $A \in \mathbb{R}^{n\times n}$, costruire le matrici $A^{(i)} = E_{i-1}A_{i-1}$ derivanti dal prodotto di matrici elementari $E_i(1, \mathbf{m}_i, \mathbf{e}_i) = I - \mathbf{m}_i\mathbf{e}_i^t$, con $\mathbf{m}_i, \mathbf{e}_i \in \mathbb{R}^4$,

$$A = \begin{pmatrix} -27 & -6 & 15 & 8 \\ -9 & 53 & 6 & 2 \\ -45 & -5 & 29 & 6 \\ -9 & -1 & -5 & 53 \end{pmatrix},$$

definite dai vettori $\mathbf{e}_i$ della base canonica di $\mathbb{R}^n$ e $\mathbf{m}_i$ tali che:

$$\mathbf{m}_i = (0, \ldots, 0, m_{i+1,1}, \ldots, m_{n,i})^t, \quad m_{r,i} = \frac{a_{r,i}}{a_{i,i}},\ r = i+1, \ldots, n.$$

Osservare la struttura delle matrici inverse di E_i e delle matrici $A^{(i+1)}$.

Svolgimento. *Posta $A^{(1)} \in \mathbb{R}^{n\times n}$ tale che:*

$$A^{(1)} = \begin{pmatrix} -27 & -6 & 15 & 8 \\ -9 & 53 & 6 & 2 \\ -45 & -5 & 29 & 6 \\ -9 & -1 & -5 & 53 \end{pmatrix}$$

Si calcolano i coefficienti $m_{i,j}$ come richiesto.

```
>> A1 = [ -27 -6 15 8
-9 53 6 2
-45 -5 29 6
-9 -1 -5 53]

A1 =

   -27    -6    15     8
    -9    53     6     2
   -45    -5    29     6
    -9    -1    -5    53

>> m21 = A1(2,1)/A1(1,1);
>> m31 = A1(3,1)/A1(1,1);
>> m41 = A1(4,1)/A1(1,1);


>> m1 = [0; m21; m31; m41]

m1 =

         0
    0.3333
    1.6667
    0.3333

>> E1 = eye(4) -m1*[1 0  0 0]

E1 =

    1.0000         0         0         0
   -0.3333    1.0000         0         0
   -1.6667         0    1.0000         0
   -0.3333         0         0    1.0000

>> A2 = E1*A1

A2 =

  -27.0000   -6.0000   15.0000    8.0000
         0   55.0000    1.0000   -0.6667
         0    5.0000    4.0000   -7.3333
         0    1.0000  -10.0000   50.3333

>> m32 = A2(3,2)/A2(2,2);
>> m42 = A2(4,2)/A2(2,2);
>>
>> m2 = [0; 0; m32; m42]

m2 =

         0
         0
    0.0909
    0.0182

>> E2 = eye(4) -m2*[ 0 1 0 0]

E2 =
```

```
    1.0000         0         0         0
         0    1.0000         0         0
         0   -0.0909    1.0000         0
         0   -0.0182         0    1.0000

>> A3 = E2*A2

A3 =

  -27.0000   -6.0000   15.0000    8.0000
         0   55.0000    1.0000   -0.6667
         0         0    3.9091   -7.2727
         0         0  -10.0182   50.3455
>> m43 = A3(4,3)/A3(3,3);
>> m3 = [0; 0; 0; m43]

m3 =

         0
         0
         0
   -2.5628

>> E3 = eye(4) -m3*[ 0 0 1 0]

E3 =

    1.0000         0         0         0
         0    1.0000         0         0
         0         0    1.0000         0
         0         0    2.5628    1.0000

>> A4 = E3*A3

A4 =

  -27.0000   -6.0000   15.0000    8.0000
         0   55.0000    1.0000   -0.6667
         0         0    3.9091   -7.2727
         0         0         0   31.7070

>> inv(E1), inv(E2), inv(E3)

ans =

    1.0000         0         0         0
    0.3333    1.0000         0         0
    1.6667         0    1.0000         0
    0.3333         0         0    1.0000


ans =

    1.0000         0         0         0
         0    1.0000         0         0
         0    0.0909    1.0000         0
         0    0.0182         0    1.0000


ans =

    1.0000         0         0         0
```

```
0    1.0000         0         0
0         0    1.0000         0
0         0   -2.5628    1.0000
```

Le inverse delle matrici elementari E_i sono matrici, sempre elementari, mantengono la forma triangolare inferiore, ma i coefficienti al di sotto della diagonale principale sono di segno opposto.

17.3 Esercitazione III.

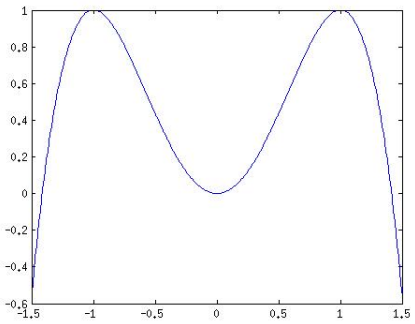
1. a) Dire cosa produce la seguente sequenza di comandi Matlab:

```
>> x = linspace(-3/2,3/2,300);
>> y = 1-(1-x.^2).^2;
>> plot(x,y);
```

   b) Scrivere e stampare un file script di Matlab in grado di generare una matrice triangolare superiore A di dimensione 8×8, con gli elementi della diagonale principale uguale a 0, quelli della prima sopradiagonale uguali a -1. quelli della seconda sopradiagonale -2 e ... quelli dell'ultima sopradiagonale uguali a -7. Quindi ridefinire gli elementi $A(i, j)$ della parte triangolare inferiore in modo tale che $A(i, j) = -A(j, i)$. Denotata con B la matrice modificata, verificare se è simmetrica.

Svolgimento.

a) La sequenza di comandi generano un vettore `x` *di elementi equispaziati da* $-\frac{3}{2}$ *a* $\frac{3}{2}$ *ed un vettore* `y` *di elementi che rappresentano l'immagine del vettore* `x` *attraverso la funzione* $f(x) = 1-(1-x^2)^2$. *Il comando* `plot(x,y)` *generano il grafico della funzione.*



b) Il seguente codice è scritto nel file `es3_2.m` *ed è lo script che opera come richiesto:*

```
A = zeros(8,8);
for i=7:-1:1
  A = A + diag(-i*(ones(i,1)),8-i);
end
```

Infatti, stampando la matrice A si ha:

```
>> A

A =
```

```
     0    -7    -6    -5    -4    -3    -2    -1
     0     0    -7    -6    -5    -4    -3    -2
     0     0     0    -7    -6    -5    -4    -3
     0     0     0     0    -7    -6    -5    -4
     0     0     0     0     0    -7    -6    -5
     0     0     0     0     0     0    -7    -6
     0     0     0     0     0     0     0    -7
     0     0     0     0     0     0     0     0
```

Si effettua ora la costruzione della matrice B come rischiesto, mediante una semplice operazione algebrica sulle matrici:

```
>> B = A - A'

B =

     0    -7    -6    -5    -4    -3    -2    -1
     7     0    -7    -6    -5    -4    -3    -2
     6     7     0    -7    -6    -5    -4    -3
     5     6     7     0    -7    -6    -5    -4
     4     5     6     7     0    -7    -6    -5
     3     4     5     6     7     0    -7    -6
     2     3     4     5     6     7     0    -7
     1     2     3     4     5     6     7     0

>>
```

Ora verifichiamo la simmetria, ricordiamo che una matrice A è simmetrica se e solo se $A \cdot A^{-1} = I$.

```
>> B * inv(B)

ans =

    1.0000    0.0000   -0.0000   -0.0000    0.0000   -0.0000    0.0000    0.0000
    0.0000    1.0000   -0.0000   -0.0000    0.0000    0.0000   -0.0000    0.0000
    0.0000         0    1.0000    0.0000   -0.0000    0.0000    0.0000    0.0000
   -0.0000    0.0000   -0.0000    1.0000   -0.0000    0.0000    0.0000   -0.0000
   -0.0000    0.0000   -0.0000   -0.0000    1.0000    0.0000   -0.0000   -0.0000
    0.0000    0.0000    0.0000   -0.0000   -0.0000    1.0000   -0.0000    0.0000
   -0.0000    0.0000   -0.0000   -0.0000         0    0.0000    1.0000   -0.0000
    0.0000   -0.0000    0.0000    0.0000    0.0000         0   -0.0000    1.0000
```

La matrice risulta simmetrica.

2. Dopo aver visualizzato in Matlab il valore `realmin` $\simeq 10^{-m}$, predisporre e visualizzare in `format long e` un vettore di 21 elementi logaritmicamente equispaziati fra $10^{-}m$ e 10^{-m-20}. Commentare i ruslitati ottenuti.

   Svolgimento.

```
>> realmin

ans =

  2.2251e-308
```

```
>> format long e
>> m = 308;
>> rm = logspace(-m,-m-20,21)

rm =

  Columns 1 through 3

    9.999999999999999e-309    1.000000000000002e-309    9.999999999999969e-311

  Columns 4 through 6

    9.999999999999475e-312    9.999999999984653e-313    1.000000000013287e-313

  Columns 7 through 9

    9.999999999638807e-315    9.999999984816838e-316    9.999999836597144e-317

  Columns 10 through 12

    1.000000230692537e-317    9.999987484955998e-319    9.999888671826830e-320

  Columns 13 through 15

    9.999888671826830e-321    9.980126045993180e-322    9.881312916824931e-323

  Columns 16 through 18

    9.881312916824931e-324                         0                         0

  Columns 19 through 21

                         0                         0                         0

>>
```

Gli elementi del vettore `er` *risultano equispaziati fino al sedicesimo elemento, dopodichè si ha il fenomeno di underflow.*

3. Al variare del parametro $p = 10^{\alpha}$, con $\alpha = 1 : 10$, calcolare mediante le note formule risolutive, le radici dell'equazione di quarto grado:

$$x^4 - bx^2 + 1 = 0,$$

con $b = \frac{1+p^2}{p}$. In seguito, dopo aver tradotto tali formule in istruzioni di assegnazione Matlab, predisporre una tabella con gli errori relativi commessi da Matlab nel calcolo numerico delle radici dell'equazione assegnata. Motivare i risultati ottenuti.

Svolgimento. *Iniziamo a calcolare le radici con le formule note:*

```
>> p = logspace(1,10,10);
>> b = (1+p.^2)./p;
>>
>> % Per risolvere x^4 -bx^2 + 1 = 0 poniamo t = x^2 e
>> % risolvo l'equazione t^2 - bt + 1 = 0
```

```
>>
>> % Calcolo del delta
>> delta = sqrt(b.^2-4)./2;
>>
>> % Calcolo dei due possibili valori di t
>> t1 = (b + delta)./2;
>> t2 = (b - delta)./2;
>>
>> % Calcolo dei valori delle x
>> x1 = sqrt(t1);
>> x2 = -sqrt(t1);
>> x3 = sqrt(t2);
>> x4 = -sqrt(t2);
>>
>> % Inseriamo ora i risultati ottenuti in una tabella X
>> X = [x2' x1' x4' x3']

X =

   1.0e+04 *

  -0.000274317334487   0.000274317334487  -0.000160468065359   0.000160468065359
  -0.000866039837421   0.000866039837421  -0.000500074994376   0.000500074994376
  -0.002738613243961   0.002738613243961  -0.001581141201791   0.001581141201791
  -0.008660254052278   0.008660254052278  -0.005000000075000   0.005000000075000
  -0.027386127875715   0.027386127875715  -0.015811388303214   0.015811388303214
  -0.086602540378458   0.086602540378458  -0.050000000000075   0.050000000000075
  -0.273861278752584   0.273861278752584  -0.158113883008421   0.158113883008421
  -0.866025403784439   0.866025403784439  -0.500000000000000   0.500000000000000
  -2.738612787525831   2.738612787525831  -1.581138830084190   1.581138830084190
  -8.660254037844387   8.660254037844387  -5.000000000000000   5.000000000000000
```

Abbiamo così ottenuto una matrice contenente, per ogni riga, le quattro radici del polinomio al variare del parametro b.

Calcoliamo ora le radici con il metodo di Matlab `roots` *e calcoliamo l'errore assoluto tra le due soluzioni.*

```
>> for i = 1 : 10
Roots(i,:) = roots([1 0 -b(i) 0 1]);
end
>> Roots

Roots =

   1.0e+05 *

  -0.000031622776602   0.000031622776602  -0.000003162277660   0.000003162277660
  -0.000100000000000   0.000100000000000  -0.000001000000000   0.000001000000000
  -0.000316227766017   0.000316227766017  -0.000000316227766   0.000000316227766
  -0.001000000000000   0.001000000000000  -0.000000100000000   0.000000100000000
  -0.003162277660168   0.003162277660168  -0.000000031622777   0.000000031622777
  -0.010000000000000   0.010000000000000  -0.000000010000000   0.000000010000000
  -0.031622776601684   0.031622776601684  -0.000000003162278   0.000000003162278
  -0.100000000000000   0.100000000000000  -0.000000001000000   0.000000001000000
  -0.316227766016838   0.316227766016838  -0.000000000316228   0.000000000316228
  -1.000000000000000   1.000000000000000   0.000000000100000  -0.000000000100000

>> Ea = abs(X-Roots)
```

```
Ea =

   1.0e+04 *

   0.000041910431530   0.000041910431530   0.000128845288757   0.000128845288757
   0.000133960162579   0.000133960162579   0.000490074994376   0.000490074994376
   0.000423664416207   0.000423664416207   0.001577978924130   0.001577978924130
   0.001339745947722   0.001339745947722   0.004999000075000   0.004999000075000
   0.004236648725969   0.004236648725969   0.015811072075448   0.015811072075448
   0.013397459621542   0.013397459621542   0.049999900000075   0.049999900000075
   0.042366487264254   0.042366487264254   0.158113851385645   0.158113851385645
   0.133974596215562   0.133974596215562   0.499999990000000   0.499999990000000
   0.423664872642549   0.423664872642550   1.581138826921912   1.581138826921912
   1.339745962155612   1.339745962155615   5.000000001000000   5.000000001000000

>>
```

Errore relativo:

```
>> Er = abs((Roots-X)./X)

Er =

   0.152780835408468   0.152780835408469   0.802934144367142   0.802934144367141
   0.154681293851390   0.154681293851391   0.980002999325169   0.980002999325169
   0.154700345929209   0.154700345929209   0.998000002999993   0.998000002999993
   0.154700536454750   0.154700536454750   0.999800000003000   0.999800000003000
   0.154700538360007   0.154700538360007   0.999980000000003   0.999980000000003
   0.154700538379059   0.154700538379060   0.999998000000000   0.999998000000000
   0.154700538379250   0.154700538379250   0.999999800000000   0.999999800000000
   0.154700538379252   0.154700538379252   0.999999980000000   0.999999980000000
   0.154700538379252   0.154700538379252   0.999999998000000   0.999999998000000
   0.154700538379251   0.154700538379252   1.000000000200000   1.000000000200000

>>
```

4. In relazione al calcolo numerico della derivata di $f(x) = e^x$, in $x = 1$, si considerino le seguenti approssimazioni:

$$\frac{f(x+h) - f(x)}{h}, \quad \frac{f(x+h) - f(x-h)}{2h}.$$

Si indichi l'errore analitico per entrambe le discretizzazioni cosiderate e si predispongano con Matlab, al variare del parametro $h = 10^{-\alpha}$, $\alpha = 1, \ldots, 20$:

a) una tabella in `format short e` contenente i valori di h e i corrispondenti errori analitici;

b) un grafico di tali errori in scala logaritmica.

Analizzare i risultati ottenuti.

Svolgimento.

a) Poniamo `e` *come il valore* e^x*, con* $x = 1$ *e calcoliamo i valori delle approssimazioni* `p` *utilizzando i parametri indicati.*

```
>> e = exp(1);
>> for i=1:20
h(i) = 10^(-i);
end
>>
>> p1 = (exp(1+h)-e)./h;
>> p2 = (exp(1+h)-exp(1-h))./(2*h);
>>
>> e1 = e -p1;
>> e2 = e -p2;
>>
>> O = [h; e1; e2];
>> fprintf('%e \t %e \t %e \n',O)
1.000000e-01   -1.405601e-01   -4.532735e-03
1.000000e-02   -1.363683e-02   -4.530492e-05
1.000000e-03   -1.359594e-03   -4.530467e-07
1.000000e-04   -1.359186e-04   -4.530565e-09
1.000000e-05   -1.359145e-05   -5.858736e-11
1.000000e-06   -1.358527e-06   1.634577e-10
1.000000e-07   -1.355058e-07   -5.858691e-11
1.000000e-08   5.101167e-08   6.602751e-09
1.000000e-09   2.286474e-07   6.602751e-09
1.000000e-10   2.893183e-06   6.727366e-07
1.000000e-11   1.177497e-05   -1.042949e-05
1.000000e-12   1.177497e-05   -2.102696e-04
1.000000e-13   4.896756e-03   4.558642e-04
1.000000e-14   5.374657e-02   9.337648e-03
1.000000e-15   5.374657e-02   -1.682980e-01
1.000000e-16   2.718282e+00   4.978358e-01
1.000000e-17   2.718282e+00   2.718282e+00
1.000000e-18   2.718282e+00   2.718282e+00
1.000000e-19   2.718282e+00   2.718282e+00
1.000000e-20   2.718282e+00   2.718282e+00
>>
```

5. Usando il comando Matlab `Hilb(n)` costruire la matrice di Hilbert di ordine 10. Costruire poi il vettore b in modo che il sistema lineare $Hx = b$ abbia solunzione il vettore x con tutte le componenti uguali a 1.

   Risolvere il sistema lineare $Hx = b$ usando i comandi Matlab. Costruire poi il vettore $c = b+z$ dove $z^t = [0.0010 \cdots 0]$ e risolvere il nuovo sistema $Hy = c$.

   Calcolare $e_r = \frac{\|x-y\|_2}{\|x\|_2}$.

   Svolgimento. *Calcloiamo* H *utilizzando il comando specificato, e mediante operazioni elementari ricaviamo il vettore* b *e quindi* x*.*

```
>> H = hilb(10);
>> b = H*ones(10,1);
>>
>> x = H\b

x =
```

```
   0.999999998715226
   1.000000110249369
   0.999997664058654
   1.000021147216568
   0.999899477767458
   1.000275541218743
   0.999549027556600
   1.000434891240842
   0.999772107416504
   1.000050035409456

>>
```

Come si può notare il vettore x è perturbato da errori algoritmici.

```
>> z = [0.001; zeros(9,1)];
>> c = b+z

c =

   2.929968253968254
   2.019877344877345
   1.603210678210678
   1.346800421800422
   1.168228993228993
   1.034895659895660
   0.930728993228993
   0.846695379783615
   0.777250935339171
   0.718771403175428

>> y = H\c

y =

   1.0e+03 *

   0.001099997010956
  -0.003949743297673
   0.080194557919823
  -0.599550712574902
   2.523285644515019
  -6.304657469284754
   9.609548217661349
  -8.749585606735506
   4.376268391416600
  -0.922663274650903

>> er = norm(x-y)/norm(x)

er =

     4.851631507736105e+0
```

17.4 Esercitazione IV.

1. a) Rappresentare graficamente su un intervallo $[a, b]$ la funzione

$$f(x) = 2\sin(8x) - \ln(x^2 + 1)$$

usando una griglia di punti equispaziati dell'intervallo $[a, b]$.

b) Rappresentare la funzione $f(x)$ definita da:

$$f(x) = \begin{cases} x^2 & x \leq 3 \\ 9 & x > 3 \end{cases}$$

sull'intervallo $[2, 4]$.

Svolgimento.

a) Svolgiamo l'esercizio utilizzando due griglie di punti ed usiamo il comando `plot` *per visualizzare i due grafici in una sola figura. Cogliamo l'occasione per vedere la differenza di rappresentazione di una funzione in base alla scelta del numero di punti.*

```
>> % Definiamo il nostro intervallo [a,b] come per
>> % per il punto b), ovvero a = 2 e b = 4.
>> a = 2;
>> b = 4;
>> x = linspace(a,b,1000);
>>
>> % Calcoliamo ora la f(x) del punto a)
>> Fxa = 2*sin(8*x)-log(x.^2 +1);
>>
>> % Proviamo ora a utilizzare una griglia con meno punti, per poi
>> % confrontarne i grafici.
>> x1 = linspace(a,b,20);
>> ya = 2*sin(8*x1)-log(x1.^2 +1);
>>
>> % Stampa dei due grafici.
>> plot(x,Fxa,x1,ya,'r')
```

b) Analogamente al punto a) vediamo due griglie.

```
>> % Calcoliamo ora l'intervallo per la funzione f(x) per il punto
>> % b) suddividendola in due intervalli x = x1b U x2b
>> x1b = linspace(2,3,500);
>> x2b = linspace(3,4,500);
>>
>> % Calcoliamo ora i valori della funzione definita a tratti, escludendo il
>> % primo punto del primo vettore.
>> fxb1 = x1b.^2;
>> fxb2 = 9*ones(1,500);
>>
>> % Ripetiamo i calcoli utilizzando meno punti.
>> x1bb = linspace(2,3,10);
>> x2bb = linspace(3,4,10);
>>
>> fxbb1 = x1bb.^2;
```

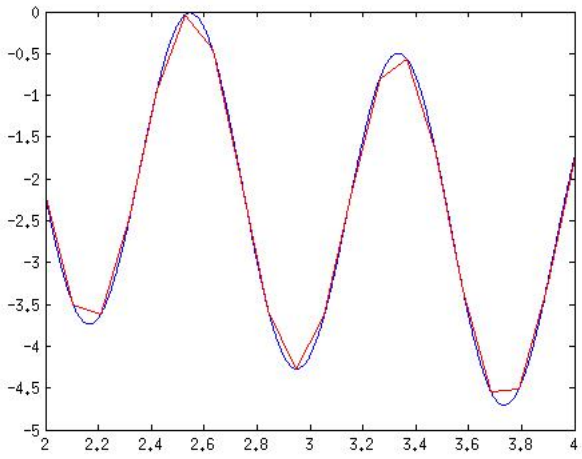


Figura 17.1: esercizio 1.a

```
>> fxbb2 = 9*ones(1,10);
>>
>> % Stampiamo i grafici facendo attenzione ad escludere il primo punto
>> % della seconda parte del vettore.
>> plot(x1b,fxb1,x2b(2:500),fx2b(2:500), x1bb,fx1bb,'r',x2bb(2:10),fxbb2(2:10),'r')
??? Undefined function or method 'fx2b' for input arguments of type 'double'.

>> plot(x1b,fxb1,x2b(2:500),fxb2(2:500), x1bb,fx1bb,'r',x2bb(2:10),fxbb2(2:10),'r')
??? Undefined function or variable 'fx1bb'.

>> plot(x1b,fxb1,x2b(2:500),fxb2(2:500), x1bb,fxbb1,'r',x2bb(2:10),fxbb2(2:10),'r')
>>
```

2. a) Dopo aver visualizzato in Matlab il valore `realmin` $\simeq 10^{-m}$, predisporre e visualizzare in formato `format long e` un vettore di 21 elementi logaritmicamente equispaziati fra 10^{-m} e 10^{-m-20}.

   b) Visualizzare i valori `realmax`, `realmax`$\cdot 10$, `realmax`$\cdot(1 + eps)$.

   c) Visualizzare i valori $1 + 10^{-17}$ e $1 + 10^{17}$.

Svolgimento.

a)

```
>> format long e
>> realmin

ans =

    2.225073858507201e-308

>> logspace(-m,-m-20,21)'

ans =
```

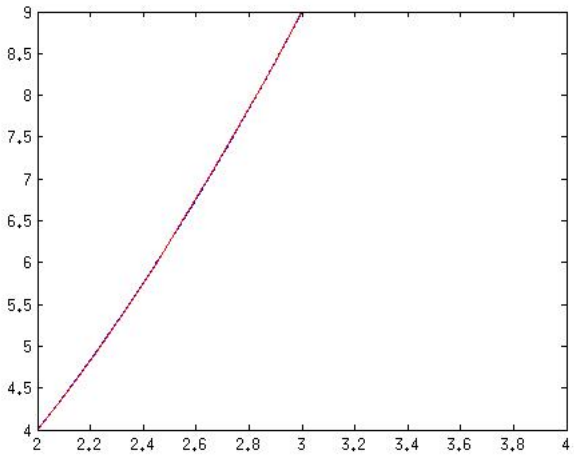


Figura 17.2: esercizio 1.b

```
9.999999999999999e-309
1.000000000000002e-309
9.999999999999969e-311
9.999999999999475e-312
9.999999999984653e-313
1.000000000013287e-313
9.999999999638807e-315
9.999999984816838e-316
9.999999836597144e-317
1.000000230692537e-317
9.999987484955998e-319
9.999888671826830e-320
9.999888671826830e-321
9.980126045993180e-322
9.881312916824931e-323
9.881312916824931e-324
                     0
                     0
                     0
                     0
                     0
```

Con il comando `logspace` *troviamo alcuni valori più piccoli di* `realim`*, che ricordiamo, è il più piccolo numero* normalizzato *in doppia precisione rappresentabile. Come si può vedere infatti, i numeri trovati non sono normalizzati (non iniziano con un* 1 *prima della virgola oppure hanno più zeri subito dopo) e di conseguenza è possibile utilizzare più* `bits` *per l'esponente (o caratteristica) e meno per la mantissa. Ad un certo punto la caratteristica esce comunque dal range ($p < -m$) e si ha quindi il fenomeno*

dell'underflow.

b) Valutiamo i seguenti risultati:

```
>> realmax

ans =

    1.797693134862316e+308

>> realmax*10

ans =

   Inf

>> realmax*(1+eps)

ans =

   Inf
```

Abbiamo qui il fenomeno di overflow.

c)

```
>> 1+10^17

ans =

     1.000000000000000e+17

>> 1+10^-17

ans =

     1

>>
```

In questi due casi abbiamo il fenomeno di arrotondamento, e poiché gli addendi di 1 *sono minori di* $\frac{\beta}{2}$ *i risultati vengono troncati.*

3. Scrivere un file script in Matlab in grado di calcolare la "precisione di macchina" in base 2, ricordando che tale valore è caratterizzato come il più piccolo numero macchina positivo tale che $fl(1 + eps) > 1$.

   Svolgimento. *Lo script Matlab, salvato in un file di nome* `epsilon.m` *è il seguente:*

```
epsi =1;
while (1+epsi)>1
    epsi = (1/2)*epsi;
end
epsi=epsi*2
```

Richiamando lo script nell'ambiente di lavoro si ha il seguente risultato:

```
>> eps

ans =
```

```
   2.2204e-16

>> epsilon

epsi =

   2.2204e-16

>>
```

4. Vedere esercizio 3 della terza esercitazione.

5. Osservare l'andamento del resto nello sviluppo di Taylor seguente:

$$\sin(x+1) = \sin(1) + \cos(1) \cdot x + R(x).$$

Fare un grafico in scala logaritmica dell'andamento del resto in funzione di x.

Svolgimento. *Per vedere l'andamento del resto, otteniamo mediante operazioni algebriche i valori di* $R(x) = s \in (x+1) - (\sin(1) + \cos(1) \cdot x$ *su un'intervallo di punti logaritmicamente equispaziati.*

```
>> x = logspace(-13,0,1000);
>>
>> % Calcolo di R(x)
>> R = sin(x+1) - (sin(1) + cos(1).*x);
>>
>> plot(x,R)
>>
```

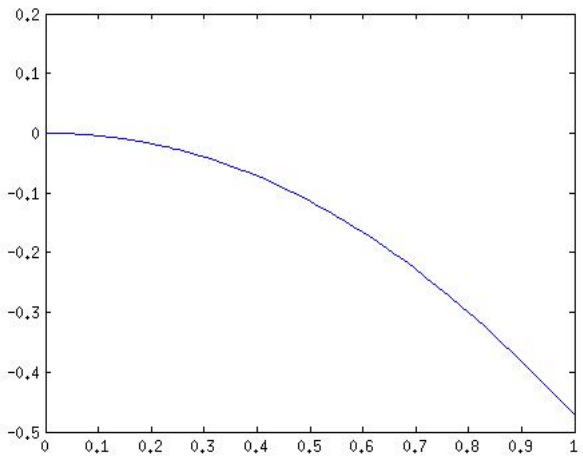


Figura 17.3: esercizio 5

6. Osservare la convergenza nel calcolo dei limiti delle seguenti funzioni:

   - $x \cdot (\sqrt{(x^2+1)} - x)$;
   - $x \cdot \sqrt{(x^2+1)} - x^2$;
   - $x/(\sqrt{(x^2+1)} + x)$.

   Svolgimento. *Come nell'esercizio della sezione 11.1.1 dedicata ai test campione, si può notare che il limite a* $+\infty$ *tende ad* $\frac{1}{2}$.

```
>> x = logspace(-13,0,15);
>>
>> f1 = x.*(sqrt(x.^2 + 1)-x);
>> f2 = x.*sqrt(x.^2+1)-x.^2;
>> f3 = x./(sqrt(x.^2 + 1) +x);
>>
>> % Creo un vettore di output y (matrice)
>> y = [x; f1; f2; f3];
>>
>> % funzione di stampa
>> format long e
>> fprintf('%.3e %18.16f %18.16f %18.16f \n',y)
1.000e-13 0.0000000000001000 0.0000000000001000 0.0000000000001000
8.483e-13 0.0000000000008483 0.0000000000008483 0.0000000000008483
7.197e-12 0.0000000000071969 0.0000000000071969 0.0000000000071969
6.105e-11 0.0000000000610540 0.0000000000610540 0.0000000000610540
5.179e-10 0.0000000005179475 0.0000000005179475 0.0000000005179475
4.394e-09 0.0000000043939705 0.0000000043939705 0.0000000043939705
3.728e-08 0.0000000372759358 0.0000000372759358 0.0000000372759358
3.162e-07 0.0000003162276660 0.0000003162276660 0.0000003162276660
2.683e-06 0.0000026826885984 0.0000026826885984 0.0000026826885984
2.276e-05 0.0000227579413192 0.0000227579413192 0.0000227579413192
1.931e-04 0.0001930325005496 0.0001930325005496 0.0001930325005496
1.638e-03 0.0016352132081426 0.0016352132081426 0.0016352132081426
1.389e-02 0.0137032264540035 0.0137032264540035 0.0137032264540035
1.179e-01 0.1047980301759449 0.1047980301759449 0.1047980301759449
1.000e+00 0.4142135623730951 0.4142135623730951 0.4142135623730951
```

   Intuitivamente dai valori della tabella stampata sulla command window il limite tende a 0*, vediamo un grafico tabulando con* 1000 *punti ancora più vicini a a* 0.

```
>> x = logspace(-15,0,1000);
>> f1 = x.*(sqrt(x.^2 + 1)-x);
>> f2 = x.*sqrt(x.^2+1)-x.^2;
>> f3 = x./(sqrt(x.^2 + 1) +x);
>> plot(x,f1)
>> xlabel('x')
>>
>> plot(x,f1)
>> xlabel('asse delle ascisse')
>>
```

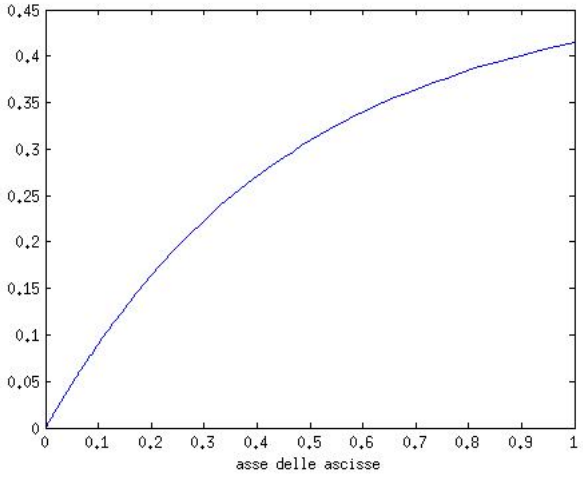


Figura 17.4: esercizio 6, si può notare che $\lim_{x \to o} f(x) = 0$.

17.5 Esercitazione V.

1. Studiare il condizionamento della funzione $f(x) = \cos(x)$. Fornire esempi numerici che motivino i risultati ottenuti.

   Svolgimento. *Studiare il condizionamento significa valutare se un problema è ben condizionato, ovvero se a piccole perturbazioni sui dati corrispondono errori dello stesso ordine.*

   Sia x un dato, e $\tilde{x}$ lo stesso dato affetto da errore, $\Delta x = \tilde{x} - x$.

   $$\textit{Imponiamo } \Delta x = 10^{-13}, \textit{ ovvero } 10^{-8} = \tilde{x} - x \Rightarrow \tilde{x} = 10^{-8} + x.$$

   Posto $y = \cos(x)$ e $\tilde{y} = \cos(\tilde{x})$, il problema è trovare $\Delta y = \tilde{y} - y$. Dalla definizione e passando allo sviluppo di Taylor si ha:

   $$\tilde{y} = f(x + \Delta x) = f(x) + f'(x)\Delta x + \cdots$$

   $$\tilde{y} - f(x) = f'(x)\Delta x + \cdots$$

   da cui segue che $\Delta y = \tilde{y} - y \simeq f'(x)\Delta x$.

   Tornando ai dati del problema si ha:

   $$\Delta y = -\sin(x) \cdot 10^{-13}.$$

   Poiché l'immagine della funzione sin *è l'intervallo $[-1, 1]$ possiamo concludere che:*

   $$\Delta y \subseteq [-1, 1] \cdot 10^{-13} = [-10^{-8}, 10^{-8}].$$

   Da questo deduciamo che l'errore assoluto non cambia ordine di grandezza, tuttavia noi vogliamo valutare l'errore relativo sui dati, ovvero trovare $\frac{\Delta y}{y}$.

   Valutiamo ora l'indice di condizionamento:

   $$\frac{x}{\cos(x)} \cdot (-1)\sin(x) = -x \cdot \tan(x).$$

   Da questo possiamo notare che, ad esempio in un intorno di $\frac{\pi}{2}$ l'indice di condizionamento "esplode" diventando $\simeq \infty$.

   Vediamo quindi con Matlab il nostro caso:

   $$\frac{\Delta y}{y} = \frac{x}{f(x)} Df(x) \frac{\Delta x}{x} = \frac{\pi}{\cos(\pi)}(-1)\sin(\pi)\frac{10^{-8}}{\pi}.$$

```
>> x = pi/2;
>> y = cos(x)

y =

     6.123233995736766e-17

>>
>> deltax = 10^(-8);
>>
>> xt = x - deltax;
>> yt = cos(xt)

yt =

     1.000000000045763e-08
```

Già il confronto tra questi due risultati ci dice che il divario tra le due soluzioni è eccessivamente elevato.

```
>> deltay = y-yt

deltay =

    -9.999999939225290e-09

>> deltay_ = -sin(x)*deltax

deltay_ =

    -1.000000000000000e-08
```

Qui sopra possiamo notare che, sia utilizzando la soluzione algebrica $\Delta y = y - \tilde{y}$ *che quella analitica, l'ordine di grandezza dell'errore relativo non cambia da quello di partenza sui dati.*

```
>> % Calcolo dell'errore relativo sui dati.
>> erx = deltax/x;
>>
>> % Calcolo dell'indice di condizionamento cond.
>> cond = (x/cos(x))*(-sin(x));
>>
>> % Errore relativo sui risultati.
>> ery = cond*erx

ery =

    -1.633123935319537e+08

>>
```

L'ordine dell'errore relativo sui risultati è, in valore assoluto, di 10^8*, molto più elevato di quello sui dati di partenza. Il problema è malcondizionato.*

2. Si approssimi il valore di e^{-9}, utilizzando lo sviluppo in serie di e^x:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Quanti termini sono necessari per stabilizzare il risultato a 6 cifre?

Ripetere il calcolo ricordando che è possibile scrivere $e^{-x} = \frac{1}{e^x}$. Costruire, per un confronto, una tabella e un grafico degli errori relativi dei valori ottenuti sommando n termini usando i due diversi algoritmi.

Svolgimento. *Con due cicli* `while` *calcoliamo l'approssimazione di* e^{-9} *con una tolleranza* `t` *dell'ordine di* 10^{-6}*, stampiamo quindi il valore di* `i` *che rappresenta il numero di termini utilizzati.*

```
>> format long
>> d = exp(-9);
>> x = -9;
>> a1 = 1 + x;
>>
>> err = d-a1;
>> t = 10^-6;
>> i = 2;
>>
>> while(abs(err)>t)
a1 = a1 + (x^i)/factorial(i);
err = d - a1;
i = i + 1;
end
>> i

i =

    34

>> a1

a1 =

     1.226613221354248e-04

>> d

d =

     1.234098040866796e-04

>> x = 9;
>> s = 1 + x;
>> a = 1/s;
>> err = d - a;
>> i = 2;
>> t = 10^-6;
>> while(abs(err)>t)
s = s + (x^i)/factorial(i);
a = 1/s;
err = d - a;
i = i + 1;
```

```
end;
>> i

i =

    18

>> a

a =

     1.240698022398142e-04
```

Il primo metodo impiega ben 34 *termini, mentre il secondo quasi la metà, questo ci dice che la convergenza del primo è più lenta.*

Calcoliamo ora due vettori contenenti le prime 34 *approssimazioni secondo i due metodi e confrontiamole.*

```
>> x = 9;
>>
>> a1 = 1;
>> i = 2;
>> a1(2) = 1 - x;
>>
>> for i=3:35
a1(i) = a1(i-1) + ((-x)^(i-1))/factorial(i-1);
end
>> a1(35)

ans =

     1.236033868055714e-04

>> s = 1 + x;
>> a2 = 1;
>> a2(2) = 1/s;
>>
>> s(2) = 1 + x;
>>
>> for i=3:35
s(i) = s(i-1) + (x^(i-1))/factorial(i-1);
a2(i) = 1/s(i);
end
>> a2(34)

ans =

     1.234098041059322e-04

>> d = exp(-9)

d =

     1.234098040866796e-04

>> ea1 = d - a1;
>> ea2 = d - a2;
>> er1 = ea1/d;
```

```
>> er2 = ea2/d;
>>
>> [er1', er2']

ans =

   1.0e+06 *

  -0.008102083927575  -0.008102083927575
   0.064825671420603  -0.000809308392758
  -0.263349227646200  -0.000159457107477
   0.721175469554209  -0.000046110953067
  -1.494005099146711  -0.000017193845473
   2.493319924514945  -0.000007643750523
  -3.487667610977539  -0.000003836038004
   4.202173506084228  -0.000002087401582
  -4.448897750610261  -0.000001194654415
   4.202173506084228  -0.000000702393539
  -3.583790624940812  -0.000000416454028
   2.786543664079675  -0.000000245317013
  -1.991207052685690  -0.000000141847842
   1.316466520459563  -0.000000079739600
  -0.809895062276671  -0.000000043260166
   0.465921887365069  -0.000000022532170
  -0.251725146808410  -0.000000011230636
   0.128205635989315  -0.000000005348020
  -0.061759755409548  -0.000000002432304
   0.028223851042545  -0.000000001057070
  -0.012268771860897  -0.000000000439445
   0.005085209383436  -0.000000000174982
  -0.002014146580155  -0.000000000066832
   0.000763862275163  -0.000000000024520
  -0.000277891045581  -0.000000000008653
   0.000097140149887  -0.000000000002941
  -0.000032678340852  -0.000000000000964
   0.000010594489394  -0.000000000000305
  -0.000003314634614  -0.000000000000093
   0.000001001990078  -0.000000000000028
  -0.000000292997329  -0.000000000000008
   0.000000082966757  -0.000000000000002
  -0.000000022773142  -0.000000000000001
   0.000000006065012  -0.000000000000000
  -0.000000001568617  -0.000000000000000

>> n = linspace(1,35,35);
>> plot(n,er1,n,er2,'r');
>>
```

Il primo metodo ha una convergenza molto più lenta, inoltre, come si può vedere dal grafico, l'errore oscilla costantemente mentre nel secondo questo fenomeno non si verifica.

3. Siano $A = \text{hilb}(1000)$ e $B = \text{rand}(1000)$.

   a) Costruire il vettore b in modo che il sistema $Ax = b$ sia risolto da $x = \text{ones}(1000, 1)$, e il vettore c in modo che il sistema $By = c$ sia risolto da $y = \text{ones}(1000, 1)$;

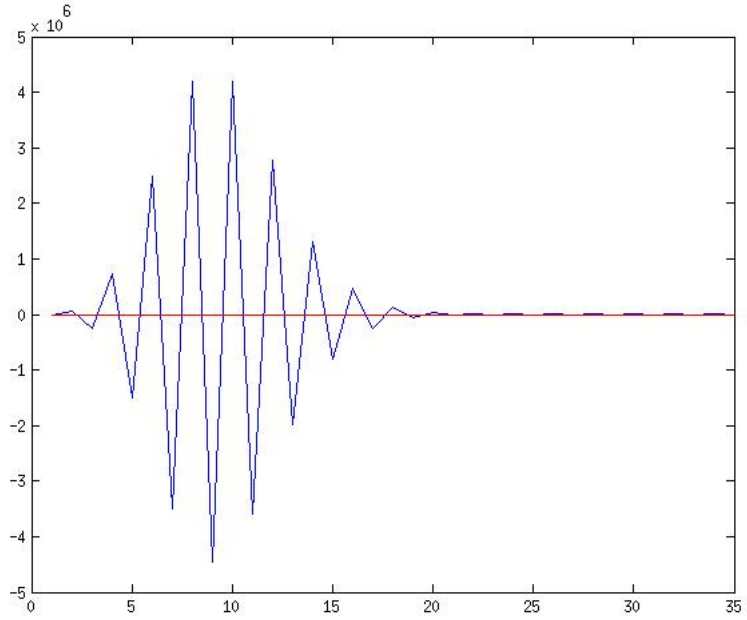


Figura 17.5: esercizio 2.

b) Calcolare i vettori x e y come soluzione dei sistemi lineari $Ax = b$ e $By = c$ utilizzando il comando \ (`mldivide`) di Matlab;

c) Calcolare gli errori relativi rapportandoli al numero di condizionamento delle corrispondenti matrici. Cosa si osserva?

d) Rappresentare in un grafico in scala semilogaritmica il condizionamento della matrice di Hilbert di dimensioni variabili da 2×2 a 50×50.

Svolgimento. *Costruiamo le matrici come indicato.*

```
>> H = hilb(1000);
>> B = rand(1000);
```

a) Calcoliamo i vettori c e b mediante prodotto riga per colonna delle matrici e dei vettori costruiti, sfruttando la simmetria della relazione "=".

```
>> x = ones(1000,1);
>> y = ones(1000,1);
>>
>> b = A*x;
>> c = B*y;
```

b) Calcoliamo i vettori soluzione dei sistemi lineari di cui sopra x_1 e y_1, per distinguerli da quelli iniziali.

```
>> x1 = A\b;
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 1.878903e-22.
>> y1 = B\c;
>>
```

Matlab già ora ci avverte che i risultati su x_1 potrebbero essere affetti da errore.

c) Utilizziamo il comando `cond()` *per calcolare l'indice di condizionamento delle due matrici, calcoliamo quindi gli errori relativi sulle soluzioni.*

```
>> ca = cond(A)

ca =

   3.6665e+21

>> cb = cond(B)

cb =

   2.8027e+05

>> eax = x - x1;
>> eay = y - y1;
>> erx = erx./x;
>> ery = ery./y;
>>
>> max(abs(erx))

ans =

   1.2879e+04

>> max(abs(ery))

ans =

   6.2221e-12
```

L'indice di condizionamento della matrice A è molto elevato, infatti il massimo dell'errore relativo in modulo è molto più elevato di quello calcolato per il vettore y relativo alla matrice B, che ha infatti un indice di condizionamento molto minore.

d) Calcoliamo l'indice di condizionamento delle matrici di Hilbert come richiesto e salviamole nel vettore `CondH`*. Utilizziamo quindi il comando* `semilogy()` *che disegna il grafico in scala semilogaritmica lungo l'asse delle y, utilizzando una scala lineare per l'asse delle x.*

```
>> for i=2:50
CondH(i) = cond(hilb(i));
end
>> semilogy(CondH)
```

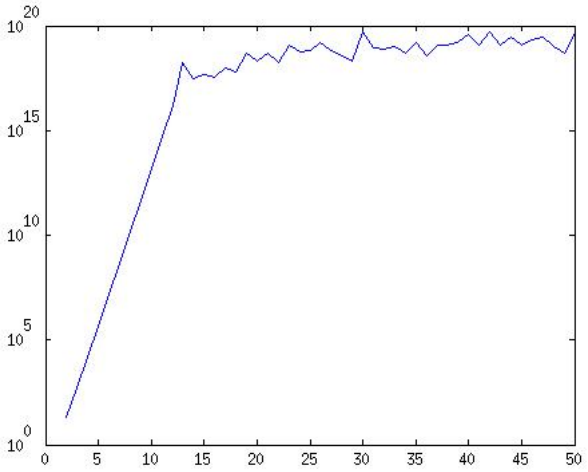


Figura 17.6: esercizio 3.

4. Data la matrice A, e i vettori b, x^* tali che:

$$A = \begin{bmatrix} -4 & -1 & 1 & 1 \\ 0 & -4 & -1 & 1 \\ -1 & -1 & 4 & 1 \\ 1 & -1 & 0 & 4 \end{bmatrix}, \quad b = \begin{bmatrix} -3 \\ -4 \\ 3 \\ 4 \end{bmatrix}, \quad x^* = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix};$$

preso un vettore $x^{(0)} \in \mathbb{R}^4$ arbitrario, calcolare e tabulare al variare di $k = 0, \ldots, 10$, le seguenti quantità:

$$\|x^* - x^{(k)}\|_1, \quad \|x^* - x^{(k)}\|_2, \quad \|x^* - x^{(k)}\|_\infty,$$

dove

$$x^{(k+1)} = (I - D^{-1}A)x^{(k)} + D^{-1}b, \quad k = 0, \ldots, 10.$$

I è la matrice identità e D^{-1} è l'inversa della matrice diagonale D estratta dalla matrice A. Mettere in un grafico la tabella costruita nella prima parte dell'esercizio.

Svolgimento. *Iniziamo ad impostare i dati.*

```
>> A = [-4 -1 1 1; 0 -4 -1 1; -1 -1 4 1; 1 -1 0 4]

A =

    -4    -1     1     1
     0    -4    -1     1
    -1    -1     4     1
     1    -1     0     4

>> b = [-3; -4; 3; 4];
>> x_ = ones(4,1);
>>
>> D = diag(diag(A))

D =

    -4     0     0     0
     0    -4     0     0
     0     0     4     0
     0     0     0     4

>> I = eye(4);
>> Dinv = inv(D);
>>
>> x = rand(4,1);
```

Ora costruiamo la matrice contente i vettori $x^{(k+1)}$ e quella con $x^* - x^{(k)}$.

```
>> for i=2:10
x(:,i) = (I-Dinv*A)*x(:,i-1) + Dinv*b;
end
>>
>> % le colonne di ex sono i vettori x-x^k
>> for i=1:10
ex(:,i) = x_ - x(:,i);
end
```

Infine calcoliamo le tre norme per ogni k richiesto e stampiamo la tabella relativa.

```
>> for i=1:10
n1(i) = norm(ex(:,i));
n2(i) = norm(ex(:,i),2);
n1(i) = norm(ex(:,i),1);
ninf(i) = norm(ex(:,i),inf);
end
>> O = [linspace(1,10,10);n1;n2;ninf];
>> fprintf('\n k\t Norma 1 \t Norma 2 \t Norma Inf\n\n'),...
fprintf('%i \t %1.6f \t %1.6f \t %1.6f\n', O)

 k  Norma 1   Norma 2   Norma Inf

1   2.444721   1.294401   0.902460
2   0.473086   0.262154   0.204245
3   0.185871   0.113172   0.101262
4   0.071560   0.043558   0.029724
5   0.022182   0.015684   0.014769
6   0.013994   0.008012   0.005531
7   0.005840   0.003309   0.002223
8   0.001676   0.001079   0.000921
```

```
9   0.001095   0.000568   0.000346
10   0.000443   0.000272   0.000191
>>
```

17.6 Esercitazione VI.

1. Determinare i coefficienti a_i dei polinomi di grado 10, 15 e 20 interpolanti la funzione $f(x) = e^x + 1$ nei nodi equispaziati dell'intervallo $[-1, 1]$ (suggerimento: usare la matrice di Vandermonde e poi risolvere il sistema lineare). Successivamente considerare i nuovi dati perturbati $\tilde{f}(x_i) = f(x_i) + \varepsilon_i$ con $\varepsilon_i = (-1)^i 10^{-5}$ e calcolare i coefficienti $\tilde{a}_i$ del polinomio perturbato $\tilde{p}(x)$ interpolante i dati $(x_i, \tilde{f}(x_i))$. Confrontare i grafici dei due polinomi.

   Calcolare:
   $$\max |a_i - \tilde{a}_i| \quad \text{e} \quad \max |p(t) - \tilde{p}(t)|$$
   dove t è un vettore formato da 101 punti equispaziati in $[-1, 1]$.

   Ripetere l'esercizio usando i nodi di Chebishev:
   $$x_i = \cos\left(\frac{(2i+1)\pi}{2n+2}\right), \quad i = 1, \ldots, n.$$

   Commentare i risultati ottenuti.

   Svolgimento. *Iniziamo con il calcolo dei tre grafici con punti equispaziati.*

```
>> % Per avere polinomi di grado N occorrono N+1 punti.
>> x10 = linspace(-1,1,11);
>> x15 = linspace(-1,1,16);
>> x20 = linspace(-1,1,21);
>>
>> % Costruzione delle matrici di Vandermonde.
>> V10 = vander(x10);
>> V15 = vander(x15);
>> V20 = vander(x20);
>>
>> % Costruizione delle immagini della funzione.
>> b10 = exp(x10)+1;
>> b15 = exp(x15)+1;
>> b20 = exp(x20)+1;
>>
>> % Calcolo dei coefficienti a dei polinomi.
>> a10 = V10\b10';
>> a15 = V15\b15';
>> a20 = V20\b20';
>>
>> % Introduciamo i vettori delle perturbazioni
>> for i = 1 : 11
eps10(i) = ((-1)^i)*10^(-5);
```

```
end
>> for i = 1 : 16
eps15(i) = ((-1)^i)*10^(-5);
end
>> for i = 1 : 21
eps20(i) = ((-1)^i)*10^(-5);
end
>>
>> % Calcolo delle immagini della funzione perturbate
>> b10_p = b10 + eps10;
>> b15_p = b15 + eps15;
>> b20_p = b20 + eps20;
>>
>> % Calcolo dei coefficienti perturbati
>> a10_p = V10\b10_p';
>> a15_p = V15\b15_p';
>> a20_p = V20\b20_p';
>>
>> % t è il vettore  di punti in cui valutare i polinomi.
>> t = linspace(-1,1,101);
>>
>> % Valutazione dei polinomi al variare di t.
>> y10 = polyval(a10, t);
>> y15 = polyval(a15, t);
>> y20 = polyval(a20, t);
>>
>> % Valutazione dei polinomi perturbati al variare di t.
>> y10_p = polyval(a10_p, t);
>> y15_p = polyval(a15_p, t);
>> y20_p = polyval(a20_p, t);
>>
>> % Stampa.
>> plot(t, y10, t, y10_p,'r');
>> plot(t, y15, t, y15_p,'r');
>> plot(t, y20, t, y20_p,'r');
>>
```

Gli errori massimi sui coefficienti e sulle valutazioni dei polinomi sono i seguenti:

```
>> format long
>> % Calcolo degli errori.
>> E_coeff = [max(abs(a10-a10_p)) max(abs(a15-a15_p)) max(abs(a20-a20_p))]

E_coeff =

   1.0e+03 *

   0.000057870370372   0.011238023832376   2.568468176961571

>> E_y = [max(abs(y10-y10_p)) max(abs(y15-y15_p)) max(abs(y20-y20_p))]

E_y =

   0.000291264924889   0.005083726754011   0.106893964401543

>>
```

Vediamo ora con i nodi di Chebishev come si comportano i nostri polinomi.

```
>> for i = 1 : 11
cheb10(i) = cos(((2*(i-1)+1)*pi)/(2*10+2));
end
>> for i = 1 : 16
cheb15(i) = cos(((2*(i-1)+1)*pi)/(2*15+2));
end
>> for i = 1 : 21
cheb20(i) = cos(((2*(i-1)+1)*pi)/(2*20+2));
end
>>
>> for i = 1 : 21
cCheb20(i) = cos(((2*(i)+1)*pi)/(2*21+2));
end
>>
>>
>> V10_c = vander(cheb10);
>> V15_c = vander(cheb15);
>> V20_c = vander(cheb20);
>>
>> b10_c = exp(cheb10)+1;
>> b15_c = exp(cheb15)+1;
>> b20_c = exp(cheb20)+1;
>>
>> a10_c = V10_c\b10_c';
>> a15_c = V15_c\b15_c';
>> a20_c = V20_c\b20_c';
>>
>> b10_cp = b10_c + eps10;
>> b15_cp = b15_c + eps15;
>> b20_cp = b20_c + eps20;
>>
>> a10_cp = V10_c\b10_cp';
>> a15_cp = V15_c\b15_cp';
>> a20_cp = V20_c\b20_cp';
>>
>> t = linspace(-1,1,101);
>> y10_c = polyval(a10_c,t);
>> y10_cp = polyval(a10_cp,t);
>> plot(t, y10_c, t, y10_cp,'r');
>>
>> y15_c = polyval(a15_c,t);
>> y15_cp = polyval(a15_cp,t);
>> plot(t, y15_c, t, y15_cp,'r');
>>
>> y20_c = polyval(a20_c,t);
>> y20_cp = polyval(a20_cp,t);
>> plot(t, y20_c, t, y20_cp,'r');
>>
```

Analizzando gli errori n modulo si ha:

```
> E_y = [max(abs(y10_c-y10_cp)) max(abs(y15_c-y15_cp)) max(abs(y20_c-y20_cp))]

E_y =

   1.0e-04 *

   0.248943037695071   0.272777793646206   0.290082490588262

>> E_coeff = [max(abs(a10_c-a10_cp)) max(abs(a15_c-a15_cp)) max(abs(a20_c-a20_cp))]
```

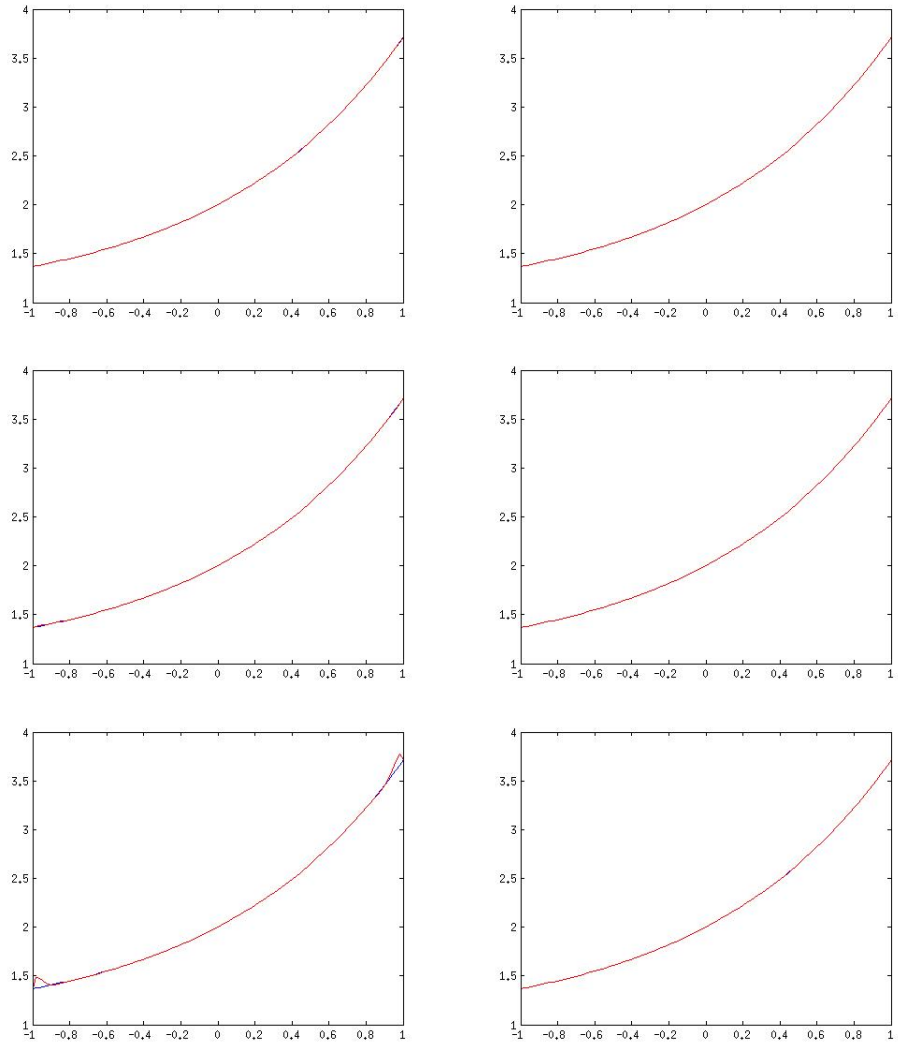


Figura 17.7: Sulla sinistra vediamo la curva calcolata con i nodi equispaziati, sulla destra con i nodi di Chebishev.

```
E_coeff =

   0.015792758827331   1.120696937190668  79.113672059826612

>>
```

Dai grafici si può notare che all'aumentare del numero dei punti nel polinomio interpolatore calcolato con nodi equispaziati l'errore aumenta (sopratutto agli estremi), mentre con i nodi di Chebyshev questo

fenomeno non si verifica.

Inoltre i grafici di sinistra hanno linee più distanti rispetto a quelli di destra, purtroppo lo zoom non è sufficiente, però ad occhio si può notare, oltre alla differenza agli estremi, che le curve di sinistra sembrano più "spesse". Questo perchè sono più distanti e quindi meno precise, come l'analisi sugli errori ha evidenziato.

2. Approssimare la radice quadrata di $x = 0.6$ considerando l'interpolazione nella forma di Lagrange con nodi i tre quadrati perfetti $x_0 = 0.49$, $x_1 = 0.64$ e $x_2 = 0.81$. Stimare il resto di interpolazione e calcolare lo scarto rispetto al valore ottenuto con il comando `sqrt` di Matlab.

   Ripetere le operazioni aggiungendo il nodo $x = 0.36$.

   Svolgimento. *Per stimare l'errore occorre calcolare la derivata terza nel caso dei* 3 *punti assegnati e quindi applicare il teorema dell'errore, o resto.*

$$|e(x)| = \frac{|f^{(n+1)}(\xi_x)|}{(n+1)!}|\omega_n(x)| \leq \frac{M}{(n+1)!}|\omega_n(x)| \leq \frac{M}{(n+1)!}(b-a)^{n+1}.$$

$$|e(x)| \leq \frac{1}{6} \cdot \frac{3}{8 \cdot \xi_x^{\frac{5}{2}}} \cdot (0.81 - 0.49)^3$$

Per calcolare ξ_x usiamo un vettore equispaziato tra 0.49 *e* 0.81.

```
>> x = [0.49 0.64 0.81];
>> y = sqrt(x);
>>
>> p = polyfit(x,y,2);
>>
>> polyval(p,0.6);
>> root = polyval(p,0.6)

root =

    0.7744

>> r = sqrt(0.6)

r =

    0.7746

>> err = abs(r-root)

err =

   1.8490e-04

>> vett = linspace(0.49,0.81);
```

```
>> M = max(abs(3./(8*sqrt(vett.^5))))

M =

    2.2312

>> ex = (M/6)*(0.81-0.49)^3

ex =

    0.0122
```

Come si può notare `ex` *è un'approssimazione corretta (anche se poco precisa) dell'errore* `err`.

Ora vediamo cosa accade aggiungendo il dato come richiesto, analogamente per il calcolo dell'errore prendiamo un ξ_x *compreso tra* 0.36 *e* 0.81.

```
>>
>>
>> x1 = [0.36 0.49 0.64 0.81];
>> y1 = sqrt(x1);
>>
>> p1 = polyfit(x1,y1,3);
>>
>> root1 = polyval(p1,0.6)

root1 =

    0.7747

>> err1 = abs(r-root1)

err1 =

   6.3964e-05

>>
>> vett1 = linspace(0.36,0.81);
>> M1 = max(abs(-15./(16*sqrt(vett1.^7))))

M1 =

   33.4898

>> ex1 = (M1/6)*(0.81-0.36)^3

ex1 =

    0.5086

>>
```

Aumentando la distanza tra due punti la precisione dell'approssimazione dell'errore peggiora, come ci aspettavamo.

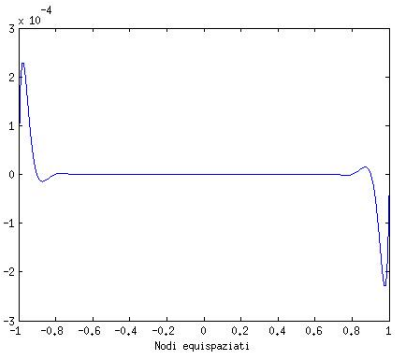
3. Disegnare sull'intervallo $[-1, 1]$ il grafico della funzione $|\omega_n(x)| = \prod_{i=0}^{n}(x - x_i)$ per alcuni valori di n considerando nodi equispaziati. Ripetere l'esercizio considerando i nodi di Chebyshev. Confrontare sullo stesso sistema di riferimento i due grafici.

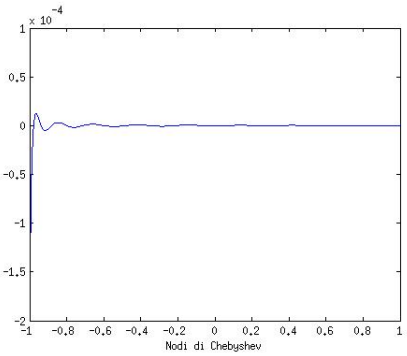
Svolgimento. *Iniziamo a definire queste due funzioni per costruire i nodi di Chebishev e per calcolare la funzione ω_n dato un vettore di nodi.*

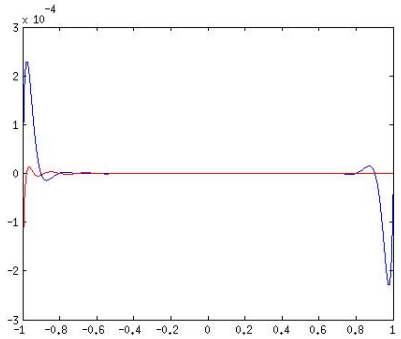
```
function [x] = chebyshev(a,b,n)
%chebyshev calcola n nodi di chebyshev

for i=1:n
    x(i) = ((b-a)/2)*cos((2*i-1)*pi/(2*n+2)) + (a+b)/2;
end

end


function [y] = omega_x(x, vec)
% omega_x := prod(x-x_i)

 n = length(x);
 y = zeros(n,1);
 for i = 1 : n
 y(i) = prod(x(i)-vec);
 end

end
```

A questo punto creiamo un vettore di dati e due vettori di nodi e vediamo i risultati.

```
>> dati = linspace(-1,1,200);
>>
>> nodi_e = linspace(-1,1,21);
>> nodi_c = chebyshev(-1,1,21);
>>
>> y_e = omega_x(dati, nodi_e);
>> y_c = omega_x(dati, nodi_c);
>>
>> plot(dati, y_e);
>> plot(dati, y_c);
>>
>> plot(dati, y_e, dati, y_c, 'r');
>>
```


x 10^-4
3
2
1
0
-1
-2
-3
-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1
Nodi equispaziati


x 10^-4
1
0.5
0
-0.5
-1
-1.5
-2
-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1
Nodi di Chebyshev


x 10^-4
3
2
1
0
-1
-2
-3
-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1

17.7 Esercitazione VII.

1. Date le coppie di valori (x_i, y_i), $i = 0, \ldots, n$, ottenuti tabulando a passo costante le funzioni $y(x) = \sin(2\pi x)$ e $y = \frac{1}{1+25(2x-1)^2}$ sull'intervallo $[0, 1]$, costruire le seguenti funzioni lineari:

$$s(x) = y_i + m_i(x - x_i), \quad m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, \quad x \in [x_i, x_{i+1}],\ i = 0, \ldots, n-1$$

e il polinomio interpolatore di Lagrange $p(x)$ di grado n.

Indicato con $x_i^* = \frac{x_i + x_{i+1}}{2}$ il punto medio di ciascun sottointervallo, tabulare l'errore assoluto:

$$e_i^s = y(x_i^*) - s(x_i^*), \quad e_i^p = y(x_i^*) - p(x_i^*), \qquad i = 0, \ldots, n-1.$$

Successivamente fare il grafico delle funzioni $s(x)$, $p(x)$, $y(x)$ sull'intervallo $[0, 1]$.

Svolgimento.

Capitolo 18

Esercizi.

Esercizio. É data la seguente tabella di valori:

x	-1	0	α	2	3
y	-1	0	1	-1	-1

dove α è un parametro reale diverso da -1, 0, 2 e 3.

Calcolare con un massimo errore assoluto: $e_a = 10^{-3}$, i valori di α che rendono minimo il *grado* del polinomio che interpola i punti assegnati.

Suggerimento: Costruire la tabella delle differenze divise contenente anche α.

Parte V

Appendice.

Appendice A

Norme.

A.1 Norma di un vettore.

La norma di un vettore è un'applicazione $\|\cdot\|:\mathbb{R}^n\to\mathbb{R}^+$ tale che:

(1) $\|x\|\geq 0, \quad \forall x\in\mathbb{R}^n$.

(2) $\|x\|=0\Leftrightarrow x=0$.

(3) $\|ax\|=|a|\cdot\|x\|, \quad \forall x\in\mathbb{R}^n$.

(4) $\forall x,y\in\mathbb{R}^n \quad \|x+y\|\leq\|x\|+\|y\|$.

Definizione A.1. Si definisce la norma p con $1\leq p<+\infty$ tale che:

$$\|x\|_p=\left(\sum_{i=1}^{n}|x_i|^p\right)^{\frac{1}{p}}.$$

$$\|x\|_\infty=\max_{1\leq i\leq m}|x_i|.$$

Esempio. Sia $x=(1,-2)$.

$$\|x\|_1=3, \quad \|x\|_2=\sqrt{5}, \quad \|x\|_\infty=2.$$

Definizione A.2. Due norme si dicono *topologicamente equivalenti* se esistono due costanti $\alpha,\beta\in\mathbb{R}, 0<\alpha\leq\beta$ tali che:

$$\alpha\|x\|''\leq\|x\|'\leq\beta\|x\|'.$$

Teorema A.3. $\forall x\in\mathbb{R}^n$ *si ha:*

1. $\|x\|_\infty\leq\|x\|_2\leq\sqrt{n}\|x\|_\infty$.
2. $\|x\|_2\leq\|x\|_1\sqrt{n}\|x\|_2$.
3. $\|x\|_\infty\leq\|x\|_1\leq n\|x\|_\infty$.

A.2 Norma di una matrice.

La norma di una matrice è un'applicazione analoga a quella sui vettori, valgono tutti i punti di cui sopra e in aggiunta:

(5) $\|A \cdot B\| \leq \|A\| \cdot \|B\|$.

Esempio. Esempio di norma che non soddisfa la proprietà (5):

$$A = B = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad C = AB = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}.$$

$$\max_{i,j} |c_{i,j}| = 2, \quad \max_{i,j} |a_{i,j}| = \max_{i,j} |b_{i,j}| = 1.$$

$$2 \leq 1 \longrightarrow \text{Assurdo.}$$

Definizione A.4. Si dice *norma naturale* o *norma indotta da un vettore* della matrice A il numero reale $\|A\|$ tale che:

$$\|A\| = \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|}.$$

Proprietà:

(1) $\|I\| = 1$.

(2) $\|Ax\| \leq \|A\|\|x\|$.

(3) $\rho(A) \leq \|A\|$.

$\rho(A)$ è il massimo autovalore di A in modulo.

Nel condizionamento:

$$1 = \|I\| = \|A \cdot A^{-1}\| \leq \|A\| \cdot \|A\| = \text{cond}(A).$$

Proposizione A.5. *La norma naturale indotta da* $\|\cdot\|_1$ *è tale che:*

$$\|A\|_1 = \max_j \sum_{i=1}^{n} |a_{i,j}|, \qquad A \in \mathbb{R}^{n \times n}.$$

Dimostrazione.

$$\|Ax\|_1 = \left\| \sum_{j=1}^{n} a_{i,j} x_j \right\|_1$$

$$= \sum_{i=1}^{n} \left| \sum_{j=1}^{n} a_{i,j} x_j \right| \leq \sum_{i=1}^{n} \sum_{j=1}^{n} |a_{i,j}||x_j|$$

$$= \sum_{j=1}^{n} |x_j| \sum_{i=1}^{n} |a_{i,j}| \leq \sum_{i=1}^{n} |x_j| \max_j \sum_{i=1}^{n} |a_{i,j}|$$

$$= \|x\|_1 (\max_j \sum_{i=1}^{n} |a_{i,j}|).$$

$$\longrightarrow \frac{\|Ax\|_1}{\|x\|_1} \leq \sum_{i=1}^{n} |a_{i,j}|.$$

■

Norme indotte:

1. $\|A\|_1 = \max_j \sum_{i=1}^{n} |a_{i,j}|$.
2. $\|A\|_2 = \sqrt{\rho(A^T A)}$.
3. $\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{i,j}|$.

A.2.1 Norma di Frobenius.

Definizione A.6. Sia $A \in \mathbb{R}^{n \times n}$, la *norma di Frobenius* è definita come segue:

$$\|A\|_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} a_{i,j}^2 \right)^{\frac{1}{2}}.$$

E' una norma compatibile con la norma due di vettore.

$$\|A\|_2 \leq \|A\|_F \|x\|_2.$$

La sintassi del comando Matlab, data una matrcie quadrata X, è la seguente:

```
>>> norm(X,'fro')
```

Proprietà:

1. $\frac{1}{\sqrt{n}}\|A\|_\infty \leq \|A\|_2 \leq \sqrt{n}\|A\|_\infty$.
2. $\frac{1}{\sqrt{n}}\|A\|_1 \leq \|A\|_2 \leq \sqrt{n}\|A\|_1$.
3. $\max_{i,j} |a_{i,j}| \leq \|A\|_2 \leq n \max_{i,j} |a_{i,j}|$.
4. $\|A\|_2 \leq \sqrt{\|A\|_1 \|A\|_\infty}$.