

Capitolo 1

Sistemi lineari

1.1 Introduzione al Calcolo Numerico

Il Calcolo Numerico è una disciplina che fa parte di un ampio settore della Matematica Applicata che prende il nome di Analisi Numerica. Si tratta di una materia che è al confine tra la Matematica e l'Informatica poiché cerca di risolvere i consueti problemi matematici utilizzando però una via algoritmica. In pratica i problemi vengono risolti indicando un processo che, in un numero finito di passi, fornisca una soluzione numerica e soprattutto che sia implementabile su un elaboratore. I problemi matematici che saranno affrontati nelle pagine seguenti sono problemi di base: risoluzione di sistemi lineari, approssimazione delle radici di funzioni non lineari, approssimazione di funzioni e dati sperimentali, calcolo di integrali definiti. Tali algoritmi di base molto spesso non sono altro se non un piccolo ingranaggio nella risoluzione di problemi ben più complessi.

1.1.1 Elementi di Algebra Lineare

Assegnati due numeri interi positivi $m, n \in \mathbb{N}$, $m, n \geq 1$, si definisce *matrice* una tabella avente m righe ed n colonne. Se gli elementi della matrice sono numeri reali allora l'insieme delle matrici aventi m righe ed n colonne si indica con $\mathbb{R}^{m \times n}$. Per convenzione le matrici si indicano con lettere maiuscole dell'alfabeto latino. Se $A \in \mathbb{R}^{m \times n}$ allora m ed n si dicono *dimensioni della matrice*. Se consideriamo due numeri interi $i, j \in \mathbb{N}$, tali che $1 \leq i \leq m$ e $1 \leq j \leq n$ questi identificano, rispettivamente, una riga ed una colonna della matrice A , e un elemento che si trova nella posizione (i, j) viene conven-

zionalmente indicato con a_{ij} . Una matrice di dimensione $m \times n$ può essere indicata nel seguente modo:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}.$$

Se $m = n$ allora la matrice A si dice *quadrata* di dimensione n o di ordine n altrimenti si dice *rettangolare*. Gli elementi a_{ij} di una matrice quadrata A di ordine n tali che $i = j$ sono detti *elementi principali* o *diagonali* e formano la cosiddetta diagonale principale di A .

Assegnata una matrice $A \in \mathbb{R}^{m \times n}$ si definisce *matrice trasposta di A* la matrice $B = A^T \in \mathbb{R}^{n \times m}$ tale che

$$b_{ij} = a_{ji}, \quad i = 1, \dots, n, \quad j = 1, \dots, m.$$

Se accade che $A = A^T$ allora la matrice è detta *simmetrica*. Una matrice simmetrica deve essere ovviamente quadrata. Gli elementi di una matrice che si trovano al di sopra della diagonale principale sono detti *sopradiagonali*, mentre quelli che si trovano al di sotto della stessa diagonale principale sono detti *sottodiagonali*. Se una matrice ha tutti gli elementi sopradiagonali e sottodiagonali uguali a zero viene detta *matrice diagonale*. Se invece ha solo gli elementi sopradiagonali nulli allora viene detta *triangolare inferiore*. Se ha gli elementi sottodiagonali nulli allora è detta *triangolare superiore*.

Assegnate due matrici $A, B \in \mathbb{R}^{m \times n}$ si definisce *somma* di A e B , e si denota con $C = A + B$, la matrice $C \in \mathbb{R}^{m \times n}$ i cui elementi sono:

$$c_{ij} = a_{ij} + b_{ij} \quad i = 1, \dots, m, \quad j = 1, \dots, n.$$

In modo analogo si definisce la *differenza* tra matrici, infatti $D = A - B$ è la matrice avente elementi:

$$d_{ij} = a_{ij} - b_{ij} \quad i = 1, \dots, m, \quad j = 1, \dots, n.$$

Se $\alpha \in \mathbb{R}$ ed $A \in \mathbb{R}^{m \times n}$ allora la matrice $C = \alpha A$ è definita da:

$$c_{ij} = \alpha a_{ij}.$$

Se $A \in \mathbb{R}^{m \times p}$ e $B \in \mathbb{R}^{p \times n}$ si definisce *prodotto* di A per B la matrice $C \in \mathbb{R}^{m \times n}$ i cui elementi sono

$$c_{ij} = \sum_{k=1}^p a_{ik} b_{kj} \quad i = 1, \dots, m, \quad j = 1, \dots, n.$$

Si noti che affinché tale prodotto abbia senso è necessario che il numero delle colonne di A coincida con il numero delle righe di B . Quando ciò accade le matrici si dicono *conformabili*, altrimenti si dicono *non conformabili*. Ad esempio nel nostro caso se $m \neq n$ allora il prodotto BA non ha senso. Ha sempre significato considerare i prodotti AB e BA se A e B sono matrici quadrate dello stesso ordine ($m = n$).

È facile verificare che il prodotto tra matrici gode della proprietà *associativa* ma in generale non di quella *commutativa*. Vale invece la seguente proprietà:

$$(AB)^T = B^T A^T.$$

Esempio 1.1.1 Siano A e B le seguenti matrici:

$$A = \begin{bmatrix} 3 & 1 & 0 \\ -1 & 2 & 1 \\ 3 & 1 & 1 \end{bmatrix}; \quad B = \begin{bmatrix} 2 & 1 & -1 \\ 0 & 1 & 1 \\ 2 & 1 & 1 \end{bmatrix}.$$

Calcoliamo la matrice $C = AB$. L'elemento c_{ij} è uguale alla somma dei prodotti degli elementi della i -esima riga di A per la j -esima colonna di B .

$$\begin{aligned} c_{11} &= 3 \cdot 2 + 1 \cdot 0 + 0 \cdot 2 = 6 \\ c_{12} &= 3 \cdot 1 + 1 \cdot 1 + 0 \cdot 1 = 4 \\ c_{13} &= 3 \cdot (-1) + 1 \cdot 1 + 0 \cdot 1 = -2 \\ c_{21} &= -1 \cdot 2 + 2 \cdot 0 + 1 \cdot 2 = 0 \\ c_{22} &= -1 \cdot 1 + 2 \cdot 1 + 1 \cdot 1 = 2 \\ c_{23} &= -1 \cdot (-1) + 2 \cdot 1 + 1 \cdot 1 = 4 \\ c_{31} &= 3 \cdot 2 + 1 \cdot 0 + 1 \cdot 2 = 8 \\ c_{32} &= 3 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 = 5 \\ c_{33} &= 3 \cdot (-1) + 1 \cdot 1 + 1 \cdot 1 = -1. \end{aligned}$$

In definitiva

$$C = \begin{bmatrix} 6 & 4 & -2 \\ 0 & 2 & 4 \\ 8 & 5 & -1 \end{bmatrix}.$$

Calcolando il prodotto $D = BA$ si trova invece:

$$D = \begin{bmatrix} 2 & 3 & 0 \\ 2 & 3 & 2 \\ 8 & 5 & 2 \end{bmatrix}$$

da cui risulta evidente che $AB \neq BA$.

Si definisce *matrice identità di ordine n* la matrice quadrata diagonale I_n avente tutti gli elementi principali uguali a 1:

$$I_n = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & 1 \end{bmatrix}.$$

La matrice identità è l'elemento neutro per il prodotto, cioè se $A \in \mathbb{R}^{n \times n}$ si ha

$$AI_n = I_nA = A.$$

Siano $A, B \in \mathbb{R}^{n \times n}$ le seguenti matrici

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

dove $A_{11}, B_{11} \in \mathbb{R}^{p \times p}$, $A_{12}, B_{12} \in \mathbb{R}^{p \times (n-p)}$, $A_{21}, B_{21} \in \mathbb{R}^{(n-p) \times p}$ e infine $A_{22}, B_{22} \in \mathbb{R}^{(n-p) \times (n-p)}$, con $p < n$, rappresentano a loro volta matrici e non semplici elementi. Si dice cioè che A e B sono state suddivise a blocchi. Il prodotto AB può essere calcolato utilizzando tale decomposizione delle matrici:

$$AB = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}.$$

Tale modo di effettuare il prodotto tra matrici viene detto *prodotto a blocchi* ed è molto utile quando le matrici hanno una particolare struttura (per esempio se uno dei blocchi è identicamente nullo oppure è uguale alla matrice identità). Per esempio se A e B sono matrici triangolari inferiori il blocco in posizione $(1, 2)$ è nullo perchè composto da elementi sopradiagonali, quindi le matrici possono essere divise nel seguente modo:

$$A = \begin{bmatrix} A_{11} & O \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & O \\ B_{21} & B_{22} \end{bmatrix}.$$

Il loro prodotto a blocchi ha come risultato:

$$AB = \begin{bmatrix} A_{11}B_{11} & O \\ A_{21}B_{11} + A_{22}B_{21} & A_{22}B_{22} \end{bmatrix},$$

evidenziando che il blocco $(1, 2)$ non va calcolato perchè identicamente nullo.

Definizione 1.1.1 *Una matrice che si ottiene da I_n scambiando alcune righe (o colonne) viene detta matrice di permutazione.*

Esempio 1.1.2 *Sia P la seguente matrice di permutazione:*

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

che è stata ottenuta da I_3 scambiando la prima riga con la terza. Consideriamo la seguente matrice A

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

e calcoliamo il prodotto PA :

$$PA = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{bmatrix}.$$

La moltiplicazione a sinistra di una matrice di permutazione per A ha l'effetto di scambiare le righe di A esattamente nello stesso modo con cui erano state scambiate le righe dell'identità per ottenere P . Calcoliamo ora il prodotto AP :

$$AP = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 2 & 1 \\ 6 & 5 & 4 \\ 9 & 8 & 7 \end{bmatrix}.$$

Invece la moltiplicazione a destra di una matrice di permutazione per A ha l'effetto di scambiare le colonne di A .

Data una matrice $A \in \mathbb{R}^{m \times n}$, una matrice $B \in \mathbb{R}^{h \times k}$, $0 < h \leq m$, $0 < k \leq n$, è detta *sottomatrice* di A se è ottenuta da A eliminando $m - h$ righe ed $n - k$ colonne. Data una matrice $A \in \mathbb{R}^{m \times n}$, una sottomatrice quadrata B di ordine $k \leq n$ di A è detta *principale* se gli elementi principali di B sono anche gli elementi principali di A . Una sottomatrice B principale di ordine k di A è detta *principale di testa* se è formata dagli elementi a_{ij} , $i, j = 1, \dots, k$.

Definizione 1.1.2 Se $A \in \mathbb{R}^{n \times n}$ è una matrice di ordine 1, si definisce determinante di A il numero

$$\det A = a_{11}.$$

Se la matrice A è quadrata di ordine n allora fissata una qualsiasi riga (colonna) di A , diciamo la i -esima (j -esima) allora applicando la cosiddetta regola di Laplace il determinante di A è:

$$\det A = \sum_{j=1}^n a_{ij} (-1)^{i+j} \det A_{ij}$$

dove A_{ij} è la matrice che si ottiene da A cancellando la i -esima riga e la j -esima colonna.

Il determinante è pure uguale a

$$\det A = \sum_{i=1}^n a_{ij} (-1)^{i+j} \det A_{ij},$$

cioè il determinante è indipendente dall'indice di riga (o di colonna) fissato. Se A è la matrice di ordine 2

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}.$$

allora

$$\det A = a_{11}a_{22} - a_{21}a_{12}.$$

Il determinante ha le seguenti proprietà:

1. Se A è una matrice triangolare o diagonale allora

$$\det A = \prod_{i=1}^n a_{ii};$$

2. $\det I = 1$;
3. $\det A^T = \det A$;
4. $\det AB = \det A \det B$ (Regola di Binet);
5. se $\alpha \in \mathbb{R}$ allora $\det \alpha A = \alpha^n \det A$.
6. $\det A = 0$ se una riga (o una colonna) è nulla, oppure una riga (o una colonna) è proporzionale ad un'altra riga (o colonna) oppure è combinazione lineare di due (o più) righe (o colonne) di A .
7. Se A è una matrice triangolare a blocchi

$$A = \begin{bmatrix} B & C \\ O & D \end{bmatrix}$$

con B e D matrici quadrate, allora

$$\det A = \det B \det D. \quad (1.1)$$

Una matrice A di ordine n si dice *non singolare* se il suo determinante è diverso da zero, in caso contrario viene detta *singolare*. Si definisce *inversa di* A la matrice A^{-1} tale che:

$$AA^{-1} = A^{-1}A = I_n$$

Per quello che riguarda il determinante della matrice inversa vale la seguente proprietà:

$$\det A^{-1} = \frac{1}{\det A}.$$

Ricordiamo che se A e B sono due matrici non singolari in base alla regola di Binet anche il loro prodotto è una matrice invertibile e inoltre:

$$(AB)^{-1} = B^{-1}A^{-1}.$$

Questa proprietà vale anche se abbiamo più di due matrici, cioè:

$$(ABC)^{-1} = C^{-1}B^{-1}A^{-1}.$$

Vettori

Se $A \in \mathbb{R}^{m \times 1}$ (o $A \in \mathbb{R}^{1 \times m}$), la matrice si riduce ad una sola colonna (o una sola riga) e viene detta *vettore colonna* (o *riga*) *ad m elementi o componenti*. Solitamente il termine vettore viene associato a vettori colonna e l'insieme dei vettori ad m componenti viene indicato con \mathbb{R}^m . Per le operazioni tra vettori valgono le stesse regole viste per le matrici, cioè la somma e la differenza sono possibili tra vettori dello stesso tipo e con lo stesso numero di componenti. Se \mathbf{x} è un vettore colonna di m elementi allora \mathbf{x}^T è un vettore riga sempre di m elementi. Se $A \in \mathbb{R}^{m \times n}$ e $\mathbf{x} \in \mathbb{R}^n$ è possibile definire il prodotto matrice per vettore nel seguente modo:

$$\mathbf{y} = A\mathbf{x}, \quad y_i = \sum_{j=1}^n a_{ij}x_j, \quad i = 1, \dots, m$$

quindi $\mathbf{y} \in \mathbb{R}^m$. Non è possibile effettuare il prodotto $A\mathbf{x}^T$ perchè le dimensioni non sono compatibili.

Esempio 1.1.3 *Sia*

$$A = \begin{bmatrix} 5 & 1 & 0 \\ -1 & 1 & 2 \\ 5 & -5 & 1 \end{bmatrix}$$

e sia \mathbf{x} il vettore

$$\mathbf{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

Calcoliamo il vettore prodotto $\mathbf{y} = A\mathbf{x}$:

$$\begin{aligned} y_1 &= 5 \cdot 1 + 1 \cdot 2 + 0 \cdot 3 = 7 \\ y_2 &= -1 \cdot 1 + 1 \cdot 2 + 2 \cdot 3 = 7 \\ y_3 &= 5 \cdot 1 - 5 \cdot 2 + 1 \cdot 3 = -2. \end{aligned}$$

Tra vettori sono consentite le seguenti operazioni:

1. *prodotto interno;*
2. *prodotto esterno.*

Il prodotto interno (o scalare), che viene spesso indicato come (\cdot, \cdot) , è definito nel seguente modo: siano $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, allora

$$(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y} = \sum_{i=1}^n x_i y_i = \alpha$$

e il risultato è un numero reale. Il prodotto scalare soddisfa le seguenti proprietà:

1. $\mathbf{x}^T \mathbf{x} \geq 0$ per ogni $\mathbf{x} \in \mathbb{R}^n$ e $(\mathbf{x}, \mathbf{x}) = 0$ se e solo se $\mathbf{x} = \mathbf{0}$;
2. $\mathbf{x}^T \mathbf{y} = \mathbf{y}^T \mathbf{x}$ per ogni $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$;
3. $(\alpha \mathbf{x})^T \mathbf{y} = \alpha (\mathbf{x}^T \mathbf{y})$ per ogni $\alpha \in \mathbb{R}$ e per ogni $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$;
4. $(\mathbf{x} + \mathbf{y})^T \mathbf{z} = \mathbf{x}^T \mathbf{z} + \mathbf{y}^T \mathbf{z}$ per ogni $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^n$.
5. se $\mathbf{x}^T \mathbf{y} = 0$ allora i due vettori si dicono *ortogonali*.

Se $\mathbf{x} \in \mathbb{R}^n$ e $\mathbf{y} \in \mathbb{R}^m$ allora il prodotto esterno viene definito nel seguente modo:

$$A = \mathbf{x} \mathbf{y}^T$$

e il risultato è una matrice di dimensione $n \times m$ i cui elementi sono:

$$a_{ij} = x_i y_j, \quad i = 1, \dots, n, \quad j = 1, \dots, m.$$

Esempio 1.1.4 Siano \mathbf{x} e \mathbf{y} i seguenti vettori:

$$\mathbf{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

e

$$\mathbf{y} = \begin{bmatrix} -1 \\ -2 \\ 4 \end{bmatrix}.$$

Calcoliamo prima il prodotto interno:

$$\mathbf{x}^T \mathbf{y} = 1 \cdot (-1) + 2 \cdot (-2) + 3 \cdot 4 = 7.$$

Osserviamo che tale operazione gode della proprietà commutativa, poiché $\mathbf{y}^T \mathbf{x} = 7$.

Per quello che riguarda il prodotto esterno, il risultato è la matrice

$$A = \mathbf{x}\mathbf{y}^T = \begin{bmatrix} -1 & -2 & 4 \\ -2 & -4 & 8 \\ -3 & -6 & 12 \end{bmatrix}.$$

Tale prodotto non gode della proprietà commutativa, infatti:

$$B = \mathbf{y}\mathbf{x}^T = \begin{bmatrix} -1 & -2 & -3 \\ -2 & -4 & -6 \\ 4 & 8 & 12 \end{bmatrix}.$$

Infatti $B \neq A$, anche se va osservato che $B = A^T$.

Norme Vettoriali

La funzione $\|\cdot\| : \mathbb{R}^n \rightarrow \mathbb{R}$ si dice *norma* se per ogni vettore $\mathbf{x} \in \mathbb{R}^n$ $\|\mathbf{x}\|$ soddisfa:

1. $\|\mathbf{x}\| \geq 0$ e $\|\mathbf{x}\| = 0$ se e solo se $\mathbf{x} = \mathbf{0}$;
2. $\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\|$ per ogni $\alpha \in \mathbb{R}$;
3. $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ per ogni $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ (*disuguaglianza triangolare*).

Si può dimostrare che per ogni fissato p , $1 \leq p \leq \infty$, la funzione $\|\cdot\|_p$ che a $\mathbf{x} \in \mathbb{R}^n$ associa

$$\|\mathbf{x}\|_p = \left(\sum_{j=1}^n |x_j|^p \right)^{\frac{1}{p}}$$

è una norma su vettori. Queste norme prendono il nome di *norme Hölderiane*. Tra queste le più utilizzate sono:

$$\|\mathbf{x}\|_1 = \sum_{j=1}^n |x_j| \quad \text{norma 1}$$

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{j=1}^n |x_j|^2} \quad \text{norma 2 o norma euclidea}$$

$$\|\mathbf{x}\|_\infty = \max_{1 \leq j \leq n} |x_j| \quad \text{norma infinito.}$$

Norme su Matrici

Definizione 1.1.3 Una funzione $\|\cdot\| : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$ tale che per ogni matrice $A \in \mathbb{R}^{n \times n}$, $\|A\|$ soddisfa:

1. $\|A\| \geq 0$ e $\|A\| = 0$ se e solo se $A = 0$;
2. $\|\alpha A\| = |\alpha| \|A\|$ per ogni $\alpha \in \mathbb{R}$;
3. $\|A + B\| \leq \|A\| + \|B\|$ per ogni $A, B \in \mathbb{R}^{n \times n}$;
4. $\|A \cdot B\| \leq \|A\| \cdot \|B\|$ per ogni $A, B \in \mathbb{R}^{n \times n}$;

si dice norma di matrice.

Definizione 1.1.4 Si dice che una norma di matrice è compatibile con una norma di vettore se per ogni matrice A e per ogni vettore \mathbf{x} risulta

$$\|A\mathbf{x}\| \leq \|A\| \|\mathbf{x}\|.$$

Un modo per definire le norme di matrici compatibili con norme di vettori è il seguente. Sia $\mathbf{x} \neq 0$ con norma $\|\mathbf{x}\|$. Considerata la norma del vettore $A\mathbf{x}$, $\|A\mathbf{x}\|$, definiamo come norma di A il numero $\|A\|$ dato da

$$\|A\| = \sup_{\mathbf{x} \neq 0} \frac{\|A\mathbf{x}\|}{\|\mathbf{x}\|}. \quad (1.2)$$

$\|A\|$ è detta *norma naturale di A* oppure *norma di A indotta* dalla norma di vettore $\|\mathbf{x}\|$.

La (1.2) può anche scriversi:

$$\|A\| = \sup_{\|\mathbf{y}\|=1} \|A\mathbf{y}\|$$

anzi è possibile dimostrare che

$$\|A\| = \max_{\|\mathbf{y}\|=1} \|A\mathbf{y}\|.$$

Le norme matriciali indotte dalle norme su vettori $\|\mathbf{x}\|_1$ e $\|\mathbf{x}\|_\infty$ sono

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}|$$

$$\|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|.$$

1.2 Metodi diretti per sistemi lineari

Siano assegnati una matrice non singolare $A \in \mathbb{R}^{n \times n}$ ed un vettore $\mathbf{b} \in \mathbb{R}^n$. Risolvere un sistema lineare avente A come matrice dei coefficienti e \mathbf{b} come vettore dei termini noti significa trovare un vettore $\mathbf{x} \in \mathbb{R}^n$ tale che

$$A\mathbf{x} = \mathbf{b}. \quad (1.3)$$

Esplicitare la relazione (1.3) significa imporre le uguaglianze tra le componenti dei vettori a primo e secondo membro:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \vdots & \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n. \end{aligned} \quad (1.4)$$

Le (1.4) definiscono un *sistema di n equazioni algebriche lineari* nelle n incognite x_1, x_2, \dots, x_n . Il vettore \mathbf{x} viene detto *vettore soluzione*. Un metodo universalmente noto per risolvere il problema (1.3) è l'applicazione della cosiddetta *Regola di Cramer* la quale fornisce:

$$x_i = \frac{\det A_i}{\det A} \quad i = 1, \dots, n, \quad (1.5)$$

dove A_i è la matrice ottenuta da A sostituendo la sua i -esima colonna con il termine noto \mathbf{b} . Dalla (1.5) è evidente che per ottenere tutte le componenti del vettore soluzione è necessario il calcolo di $n + 1$ determinanti di ordine n . Si può facilmente dedurre che il numero di operazioni necessarie per il calcolo del determinante di una matrice di ordine n applicando la regola di Laplace è circa $n!$, quindi questa strada non permette di poter determinare velocemente la soluzione del nostro sistema. Basti pensare che se $n = 100$ il numero di operazioni per il calcolo di un solo determinante sarebbe all'incirca dell'ordine di 10^{157} .

1.2.1 Risoluzione di sistemi triangolari

Prima di affrontare la soluzione algoritmica di un sistema lineare vediamo qualche particolare sistema che può essere agevolmente risolto. Assumiamo che il sistema da risolvere abbia la seguente forma:

$$\begin{array}{cccccc}
 a_{11}x_1 & +a_{12}x_2 & \dots & +a_{1i}x_i & \dots & +a_{1n}x_n & = & b_1 \\
 & a_{22}x_2 & \dots & +a_{2i}x_i & \dots & +a_{2n}x_n & = & b_2 \\
 & & \ddots & \vdots & & \vdots & & \vdots \\
 & & & a_{ii}x_i & \dots & +a_{in}x_n & = & b_i \\
 & & & & \ddots & \vdots & & \vdots \\
 & & & & & a_{nn}x_n & = & b_n
 \end{array} \tag{1.6}$$

con $a_{ii} \neq 0$ per ogni i . In questo caso la matrice A è detta *triangolare superiore*. È evidente che in questo caso, la soluzione è immediatamente calcolabile. Infatti:

$$\left\{ \begin{array}{l} x_n = \frac{b_n}{a_{nn}} \\ \\ x_i = \frac{b_i - \sum_{j=i+1}^n a_{ij}x_j}{a_{ii}} \quad i = n-1, \dots, 1. \end{array} \right. \tag{1.7}$$

Il metodo (1.7) prende il nome di *metodo di sostituzione all'indietro*, poichè il vettore \mathbf{x} viene calcolato partendo dall'ultima componente. Anche per il seguente sistema il vettore soluzione è calcolabile in modo analogo.

$$\begin{array}{cccccc}
 a_{11}x_1 & & & & & & = & b_1 \\
 a_{21}x_1 & +a_{22}x_2 & & & & & = & b_2 \\
 \vdots & \vdots & \ddots & & & & \vdots & \\
 a_{i1}x_1 & +a_{i2}x_2 & \dots & +a_{ii}x_i & & & = & b_i \\
 \vdots & \vdots & & & \ddots & & \vdots & \\
 a_{n1}x_1 & +a_{n2}x_2 & \dots & +a_{ni}x_i & \dots & +a_{nn}x_n & = & b_n
 \end{array} \tag{1.8}$$

In questo caso la matrice dei coefficienti è *triangolare inferiore* e la soluzione viene calcolata con il *metodo di sostituzione in avanti*:

$$\begin{cases} x_1 = \frac{b_1}{a_{11}} \\ x_i = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j}{a_{ii}} \quad i = 2, \dots, n-1. \end{cases}$$

Concludiamo questo paragrafo facendo alcune considerazioni sul costo computazionale dei metodi di sostituzione. Per costo computazionale di un algoritmo si intende il numero di operazioni che esso richiede per fornire la soluzione di un determinato problema. Nel caso di algoritmi numerici le operazioni che si contano sono quelle aritmetiche che operano su dati reali. Considerano per esempio il metodo di sostituzione in avanti osserviamo che per calcolare x_1 è necessaria una sola operazione (una divisione), per calcolare x_2 le operazioni sono tre (un prodotto, una differenza e una divisione), mentre il generico x_i richiede $2i - 1$ operazioni ($i - 1$ prodotti, $i - 1$ differenze e una divisione), quindi indicato con C_s il numero totale di operazioni necessarie è:

$$C_s = \sum_{i=1}^n (2i - 1) = 2 \sum_{i=1}^n i - \sum_{i=1}^n 1 = 2 \frac{n(n+1)}{2} - n = n^2,$$

sfruttando la proprietà:

$$C_s = \sum_{i=1}^n i = \frac{n(n+1)}{2}.$$

1.2.2 Metodo di Eliminazione di Gauss

L'idea di base del metodo di Gauss è appunto quella di operare delle opportune trasformazioni sul sistema originale $A\mathbf{x} = \mathbf{b}$, che non costino eccessivamente, in modo da ottenere un sistema equivalente, cioè un sistema che ammetta la stessa soluzione di quello di partenza, ma che sia facile da risolvere, per esempio uno avente come matrice dei coefficienti una matrice triangolare superiore. Prima di descrivere il metodo vediamo un esempio.

Supponiamo che il sistema da risolvere sia:

$$\begin{array}{ccccrc} 2x_1 & +x_2 & +x_3 & & = & -1 \\ -6x_1 & -4x_2 & -5x_3 & +x_4 & = & 1 \\ -4x_1 & -6x_2 & -3x_3 & -x_4 & = & 2 \\ 2x_1 & -3x_2 & +7x_3 & -3x_4 & = & 0. \end{array}$$

Il vettore soluzione di un sistema lineare non cambia se ad un'equazione viene sommata la combinazione lineare di una, o più, equazioni dello stesso sistema. L'idea alla base del metodo di Gauss è quella di ottenere un sistema lineare con matrice dei coefficienti triangolare superiore effettuando opportune combinazioni lineari tra le equazioni. Poniamo

$$A^{(1)} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ -6 & -4 & -5 & 1 \\ -4 & -6 & -3 & -1 \\ 2 & -3 & 7 & -3 \end{bmatrix}, \quad \mathbf{b}^{(1)} = \begin{bmatrix} -1 \\ 1 \\ 2 \\ 0 \end{bmatrix}$$

rispettivamente la matrice dei coefficienti e il vettore dei termini noti del sistema di partenza. Cerchiamo ora di determinare un sistema lineare equivalente a quello iniziale ma che abbia gli elementi sottodiagonali della prima colonna uguali a zero. Lasciamo inalterata la prima equazione. Poniamo

$$l_{21} = -\frac{a_{21}}{a_{11}} = -\frac{-6}{2} = 3$$

e moltiplichiamo la prima equazione per l_{21} ottenendo:

$$6x_1 + 3x_2 + 3x_3 = -3.$$

La nuova seconda equazione sarà la somma tra la seconda equazione e la prima moltiplicata per l_{21} :

$$\begin{array}{ccccrc} -6x_1 & -4x_2 & -5x_3 & +x_4 & = & 1 \\ 6x_1 & +3x_2 & +3x_3 & & = & -3 \\ \hline & -x_2 & -2x_3 & +x_4 & = & -2 \quad \text{[Nuova seconda equazione].} \end{array}$$

Poniamo

$$l_{31} = -\frac{a_{31}^{(1)}}{a_{11}^{(1)}} = -\frac{-4}{2} = 2$$

e moltiplichiamo la prima equazione per l_{31} ottenendo:

$$4x_1 + 2x_2 + 2x_3 = -2.$$

La nuova terza equazione sarà la somma tra la terza equazione e la prima moltiplicata per l_{31} :

$$\begin{array}{rccccrc} -4x_1 & -6x_2 & -3x_3 & -x_4 & = & 2 \\ 4x_1 & +2x_2 & +2x_3 & & = & -2 \\ \hline & -4x_2 & -x_3 & -x_4 & = & 0 & \text{[Nuova terza equazione].} \end{array}$$

Poniamo ora

$$l_{41} = -\frac{a_{41}^{(1)}}{a_{11}^{(1)}} = -\frac{2}{2} = -1$$

e moltiplichiamo la prima equazione per l_{41} ottenendo:

$$-2x_1 - x_2 - x_3 = 1.$$

La nuova quarta equazione sarà la somma tra la quarta equazione e la prima moltiplicata per l_{41} :

$$\begin{array}{rccccrc} 2x_1 & -3x_2 & +7x_3 & -3x_4 & = & 0 \\ -2x_1 & -x_2 & -x_3 & & = & 1 \\ \hline & -4x_2 & +6x_3 & -3x_4 & = & 1 & \text{[Nuova quarta equazione].} \end{array}$$

Al secondo passo il sistema lineare è diventato:

$$\begin{array}{rccccrc} 2x_1 & +x_2 & +x_3 & & = & -1 \\ & -x_2 & -2x_3 & +x_4 & = & -2 \\ & -4x_2 & -x_3 & -x_4 & = & 0 \\ & -4x_2 & +6x_3 & -3x_4 & = & 1. \end{array}$$

La matrice dei coefficienti e il vettore dei termini noti sono diventati:

$$A^{(2)} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & -1 & -2 & 1 \\ 0 & -4 & -1 & -1 \\ 0 & -4 & 6 & -3 \end{bmatrix}, \quad \mathbf{b}^{(2)} = \begin{bmatrix} -1 \\ -2 \\ 0 \\ 1 \end{bmatrix}.$$

Cerchiamo ora di azzerare gli elementi sottodiagonali della seconda colonna, a partire da a_{32} , usando una tecnica simile. Innanzitutto osserviamo che non

conviene prendere in considerazione una combinazione lineare che coinvolga la prima equazione perchè avendo questa un elemento in prima posizione diverso da zero quando sommata alla terza equazione cancellerà l'elemento uguale a zero in prima posizione. Lasciamo quindi inalterate le prime due equazioni del sistema e prendiamo come equazione di riferimento la seconda. Poichè $a_{22}^{(2)} \neq 0$ poniamo

$$l_{32} = -\frac{a_{32}^{(2)}}{a_{22}^{(2)}} = -\frac{-4}{-1} = -4$$

e moltiplichiamo la seconda equazione per l_{32} ottenendo:

$$4x_2 + 8x_3 - 4x_4 = 8.$$

La nuova terza equazione sarà la somma tra la terza equazione e la seconda appena modificata:

$$\begin{array}{rclcrcl} -4x_2 & -x_3 & -x_4 & = & 0 & \\ 4x_2 & +8x_3 & -4x_4 & = & 8 & \\ \hline & 7x_3 & -5x_4 & = & 8 & \text{[Nuova terza equazione].} \end{array}$$

Poniamo

$$l_{42} = -\frac{a_{42}^{(2)}}{a_{22}^{(2)}} = -\frac{-4}{-1} = -4$$

e moltiplichiamo la seconda equazione per l_{42} ottenendo:

$$4x_2 + 8x_3 - 4x_4 = 8.$$

La nuova quarta equazione sarà la somma tra la quarta equazione e la seconda appena modificata:

$$\begin{array}{rclcrcl} -4x_2 & +6x_3 & -3x_4 & = & 1 & \\ 4x_2 & +8x_3 & -4x_4 & = & 8 & \\ \hline & 14x_3 & -7x_4 & = & 9 & \text{[Nuova quarta equazione].} \end{array}$$

Al terzo passo il sistema lineare è diventato:

$$\begin{array}{rclcrcl} 2x_1 & +x_2 & +x_3 & & = & -1 \\ & -x_2 & -2x_3 & +x_4 & = & -2 \\ & & 7x_3 & -5x_4 & = & 8 \\ & & 14x_3 & -7x_4 & = & 9. \end{array}$$

La matrice dei coefficienti e il vettore dei termini noti sono quindi

$$A^{(3)} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & -1 & -2 & 1 \\ 0 & 0 & 7 & -5 \\ 0 & 0 & 14 & -7 \end{bmatrix}, \quad \mathbf{b}^{(2)} = \begin{bmatrix} -1 \\ -2 \\ 8 \\ 9 \end{bmatrix}.$$

Resta da azzerare l'unico elemento sottodiagonali della terza colonna. Lasciamo inalterate le prime tre equazioni del sistema. Poniamo

$$l_{43} = -\frac{a_{43}^{(3)}}{a_{33}^{(3)}} = -\frac{14}{7} = -2$$

e moltiplichiamo la terza equazione per l_{43} ottenendo:

$$-14x_3 + 10x_4 = -16.$$

La nuova quarta equazione sarà la somma tra la quarta equazione e la terza appena modificata:

$$\begin{array}{rcl} 14x_3 & -7x_4 & = -16 \\ -14x_3 & +10x_4 & = 9 \\ \hline & 3x_4 & = -7 \quad \text{[Nuova quarta equazione].} \end{array}$$

Abbiamo ottenuto un sistema triangolare superiore:

$$\begin{array}{rcl} 2x_1 & +x_2 & +x_3 & & = -1 \\ & -x_2 & -2x_3 & +x_4 & = 4 \\ & & 7x_3 & -5x_4 & = 8 \\ & & & 3x_4 & = -7. \end{array}$$

La matrice dei coefficienti e il vettore dei termini noti sono diventati:

$$A^{(4)} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & -1 & -2 & 1 \\ 0 & 0 & 7 & -5 \\ 0 & 0 & 0 & 3 \end{bmatrix}, \quad \mathbf{b}^{(4)} = \begin{bmatrix} -1 \\ 4 \\ 8 \\ -7 \end{bmatrix}.$$

I numeri $l_{21}, l_{3,1}, \dots$ sono detti *moltiplicatori*.

Vediamo ora di calcolare le formule che consentano di calcolare gli elementi della matrice dei coefficienti e del vettore dei termini noti ad ogni passo del metodo di Gauss. Abbiamo detto che $A^{(1)}$ e $\mathbf{b}^{(1)}$ sono assegnati inizialmente, ipotizziamo per il momento che $a_{11}^{(1)} \neq 0$. Calcoliamo ora gli stessi dati al passo 2 tenendo presente che:

1. La prima equazione del sistema resta invariata;
2. Gli elementi sottodiagonali della prima colonna di $A^{(2)}$ sono nulli;
3. La i -esima equazione del sistema ($i \geq 2$) è ottenuta sommando alla medesima equazione la prima moltiplicata per $-a_{i1}^{(1)}/a_{11}^{(1)}$.

Fissiamo quindi un'equazione i , $i \geq 2$, e calcoliamone i coefficienti $a_{ij}^{(2)}$ e $b_i^{(2)}$:

$$\begin{array}{cccccccccc}
 a_{i1}^{(1)} & a_{i2}^{(1)} & a_{i3}^{(1)} & \dots & a_{ij}^{(1)} & \dots & a_{in}^{(1)} & b_i^{(1)} & + \\
 -\frac{a_{i1}^{(1)}}{a_{11}^{(1)}} \times & a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \dots & a_{1j}^{(1)} & \dots & a_{1n}^{(1)} & b_1^{(1)} & = \\
 \hline
 0 & a_{i2}^{(2)} & a_{i3}^{(2)} & \dots & a_{ij}^{(2)} & \dots & a_{in}^{(2)} & b_i^{(2)} & &
 \end{array}$$

dove

$$a_{ij}^{(2)} = a_{ij}^{(1)} - \frac{a_{i1}^{(1)}}{a_{11}^{(1)}} a_{1j}^{(1)}, \quad i, j = 2, \dots, n$$

e

$$b_i^{(2)} = b_i^{(1)} - \frac{a_{i1}^{(1)}}{a_{11}^{(1)}} b_1^{(1)}, \quad i = 2, \dots, n.$$

Se ipotizziamo che $a_{22}^{(2)} \neq 0$ possiamo calcolare gli elementi del sistema al passo 3 tenendo presente che:

1. La prime 2 equazioni del sistema restano invariate;
2. Gli elementi sottodiagonali della prime 2 colonna di $A^{(3)}$ sono nulli;
3. La i -esima equazione del sistema ($i \geq 3$) è ottenuta sommando alla medesima equazione la seconda moltiplicata per $-a_{i2}^{(2)}/a_{22}^{(2)}$.

Fissiamo quindi un'equazione i , $i \geq 3$, e calcoliamone i coefficienti $a_{ij}^{(3)}$ e $b_i^{(3)}$:

$$\begin{array}{cccccccc}
 0 & a_{i2}^{(2)} & a_{i3}^{(2)} & \dots & a_{ij}^{(2)} & \dots & a_{in}^{(2)} & b_i^{(2)} & + \\
 -\frac{a_{i2}^{(2)}}{a_{22}^{(2)}} \times & 0 & a_{22}^{(2)} & a_{23}^{(2)} & \dots & a_{2j}^{(2)} & \dots & a_{2n}^{(2)} & b_2^{(2)} & = \\
 \hline
 0 & 0 & a_{i3}^{(3)} & \dots & a_{ij}^{(3)} & \dots & a_{in}^{(3)} & b_i^{(3)} & &
 \end{array}$$

dove

$$a_{ij}^{(3)} = a_{ij}^{(2)} - \frac{a_{i2}^{(2)}}{a_{22}^{(2)}} a_{2j}^{(2)}, \quad i, j = 3, \dots, n$$

e

$$b_i^{(3)} = b_i^{(2)} - \frac{a_{i2}^{(2)}}{a_{22}^{(2)}} b_2^{(2)}, \quad i = 3, \dots, n.$$

Avendo ricavato esplicitamente le formule per i primi due passi del metodo di Gauss è semplice ricavare quelle per un generico passo k . La matrice $A^{(k)}$ ha gli elementi sottodiagonali delle prime $k - 1$ colonne uguali a zero, e, supposto $a_{kk}^{(k)} \neq 0$, gli elementi di $A^{(k+1)}$ e di $\mathbf{b}^{(k+1)}$ sono quindi:

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} a_{kj}^{(k)}, \quad i, j = k + 1, \dots, n \quad (1.9)$$

e

$$b_i^{(k+1)} = b_i^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} b_k^{(k)}, \quad i = k + 1, \dots, n. \quad (1.10)$$

Il valore di k varia da 1 (matrice dei coefficienti e vettori dei termini noti iniziali) fino a $n - 1$, infatti la matrice $A^{(n)}$ avrà gli elementi sottodiagonali delle prime $n - 1$ colonne uguali a zero.

Tutto il discorso fatto finora va bene se gli elementi $a_{kk}^{(k)}$ sono diversi da zero per ogni k , prima di affrontare come modificare il metodo di Gauss qualora tale situazione non si verifichi consideriamo una formulazione alternativa dello stesso metodo.

Torniamo ora all'esempio visto in precedenza. Definiamo ora una matrice, che chiamiamo $L^{(1)}$ triangolare inferiore, con elementi diagonali uguali a 1,

e con tutti gli altri elementi uguali a zero tranne quelli della prima colonna che poniamo uguali ai moltiplicatori:

$$L^{(1)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}.$$

Se calcoliamo il prodotto $L^{(1)}A^{(1)}$ risulta:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 0 \\ -6 & -4 & -5 & 1 \\ -4 & -6 & -3 & -1 \\ 2 & -3 & 7 & -3 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & -1 & -2 & 1 \\ 0 & -4 & -1 & -1 \\ 0 & -4 & 6 & -3 \end{bmatrix} = A^{(2)}.$$

In modo analogo si può verificare che $L^{(1)}\mathbf{b}^{(1)} = \mathbf{b}^{(2)}$. Tale proprietà vale anche per i passi successivi e può essere dimostrata teoricamente.

Sia $L^{(1)}$ una matrice triangolare inferiore $n \times n$ così definita:

$$L^{(1)} = \begin{bmatrix} 1 & & & & \\ l_{21} & 1 & & & 0 \\ l_{31} & 0 & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ l_{n1} & 0 & \dots & 0 & 1 \end{bmatrix},$$

con

$$l_{i1} = -\frac{a_{i1}^{(1)}}{a_{11}^{(1)}} \quad i = 2, \dots, n. \quad (1.11)$$

Per calcolare la matrice prodotto $L^{(1)}A^{(1)}$ decomponiamo a blocchi le due matrici nel seguente modo:

$$L^{(1)} = \begin{bmatrix} 1 & 0^T \\ \mathbf{l}_1 & I_{n-1} \end{bmatrix} \quad A^{(1)} = \begin{bmatrix} a_{11}^{(1)} & \mathbf{r}_1^T \\ \mathbf{c}_1 & A_{22}^{(1)} \end{bmatrix}$$

avendo indicato con \mathbf{l}_1 il vettore i cui elementi sono definiti da (1.11), con \mathbf{r}_1^T gli elementi della prima riga di $A^{(1)}$, escluso il primo, con \mathbf{c}_1 gli elementi della prima colonna di $A^{(1)}$, escluso il primo, e con I_{n-1} la matrice identità

di ordine $n - 1$. Calcoliamo ora il prodotto tra queste due matrici:

$$\begin{aligned} L^{(1)}A^{(1)} &= \begin{bmatrix} 1 & 0^T \\ \mathbf{l}_1 & I_{n-1} \end{bmatrix} \begin{bmatrix} a_{11}^{(1)} & \mathbf{r}_1^T \\ \mathbf{c}_1 & A_{22}^{(1)} \end{bmatrix} = \\ &= \begin{bmatrix} a_{11}^{(1)} & \mathbf{r}_1^T \\ \mathbf{l}_1 a_{11}^{(1)} + \mathbf{c}_1 & A_{22}^{(1)} + \mathbf{l}_1 \mathbf{r}_1^T \end{bmatrix}. \end{aligned}$$

Osserviamo preliminarmente che la prima riga della matrice prodotto coincide con la prima riga di $A^{(1)}$. Per gli elementi della prima colonna:

$$\mathbf{l}_1 a_{11}^{(1)} + \mathbf{c}_1 = \begin{bmatrix} -a_{21}^{(1)} \\ -a_{31}^{(1)} \\ \vdots \\ -a_{n1}^{(1)} \end{bmatrix} + \begin{bmatrix} a_{21}^{(1)} \\ a_{31}^{(1)} \\ \vdots \\ a_{n1}^{(1)} \end{bmatrix} = 0.$$

Per un generico elemento (i, j) , $2 \leq i, j \leq n$, si ha:

$$(A_{22}^{(1)} + \mathbf{l}_1 \mathbf{r}_1^T)_{ij} = a_{ij}^{(1)} - \frac{a_{i1}^{(1)}}{a_{11}^{(1)}} a_{1j}^{(1)} = a_{ij}^{(2)}$$

quindi $L^{(1)}A^{(1)} = A^{(2)}$. In modo simile si può verificare che $\mathbf{b}^{(2)} = L^{(1)}\mathbf{b}^{(1)}$. Siamo passati dal sistema $A^{(1)}\mathbf{x} = \mathbf{b}^{(1)}$ al sistema equivalente ad esso $A^{(2)}\mathbf{x} = \mathbf{b}^{(2)}$. Al passo successivo definiamo la matrice

$$L^{(2)} = \begin{bmatrix} 1 & & & & \\ 0 & 1 & & & 0 \\ 0 & l_{32} & 0 & 1 & \\ \vdots & \vdots & \vdots & \ddots & \ddots \\ 0 & l_{n2} & 0 & \dots & 0 & 1 \end{bmatrix}$$

con

$$l_{i2} = -\frac{a_{i2}^{(2)}}{a_{22}^{(2)}} \quad i = 3, \dots, n.$$

In modo analogo a quanto già visto si può verificare che $A^{(3)} = L^{(2)}A^{(2)}$ e che $\mathbf{b}^{(3)} = L^{(2)}\mathbf{b}^{(2)}$. Assumiamo ora di aver effettuato $k - 1$ passi del tipo sopra descritto e dunque di essere giunti al sistema equivalente:

$$A^{(k)}\mathbf{x} = \mathbf{b}^{(k)}$$

dove $A^{(k)} = L^{(k-1)} A^{(k-1)}$, e con

$$A^{(k-1)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1,k-2}^{(1)} & a_{1,k-1}^{(1)} & \cdots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & & a_{2,k-2}^{(2)} & a_{2,k-1}^{(2)} & \cdots & a_{2n}^{(2)} \\ & & \ddots & \vdots & \vdots & & \vdots \\ & & & a_{k-2,k-2}^{(k-2)} & a_{k-2,k-1}^{(k-2)} & \cdots & a_{k-2,n}^{(k-2)} \\ & 0 & & 0 & a_{k-1,k-1}^{(k-1)} & \cdots & a_{k-1,n}^{(k-1)} \\ & & & \vdots & \vdots & & \vdots \\ & & & 0 & a_{n,k-1}^{(k-1)} & \cdots & a_{nn}^{(k-1)} \end{bmatrix},$$

e

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

Le matrici $L^{(k)}$ sono dette *matrici elementari di Gauss*, quindi

$$A^{(k)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1k}^{(1)} & \cdots & a_{1n}^{(1)} \\ & a_{22}^{(2)} & & a_{2k}^{(2)} & \cdots & a_{2n}^{(2)} \\ & & \ddots & \vdots & & \vdots \\ & & & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ & 0 & & \vdots & & \vdots \\ & & & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} \end{bmatrix}$$

L'algoritmo di eliminazione di Gauss è quindi composto dai seguenti passi:

1° passo:

$$A^{(1)} \mathbf{x} = \mathbf{b}^{(1)} \Leftrightarrow L^{(1)} A^{(1)} \mathbf{x} = L^{(1)} \mathbf{b}^{(1)} \Leftrightarrow A^{(2)} \mathbf{x} = \mathbf{b}^{(2)}$$

2° passo:

$$A^{(2)} \mathbf{x} = \mathbf{b}^{(2)} \Leftrightarrow L^{(2)} A^{(2)} \mathbf{x} = L^{(2)} \mathbf{b}^{(2)} \Leftrightarrow A^{(3)} \mathbf{x} = \mathbf{b}^{(3)}$$

\vdots
 $(n-1)^{mo}$ passo:

$$A^{(n-1)}\mathbf{x} = \mathbf{b}^{(n-1)} \Leftrightarrow L^{(n-1)}A^{(n-1)}\mathbf{x} = L^{(n-1)}\mathbf{b}^{(n-1)} \Leftrightarrow A^{(n)}\mathbf{x} = \mathbf{b}^{(n)}$$

dove $A^{(n)}\mathbf{x} = \mathbf{b}^{(n)}$ ha la seguente forma:

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & \cdots & a_{2n}^{(2)} \\ 0 & 0 & \ddots & & \vdots \\ \vdots & \vdots & \ddots & a_{n-1, n-1}^{(n-1)} & a_{n-1, n}^{(n-1)} \\ 0 & 0 & \cdots & 0 & a_{nn}^{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_{n-1}^{(n-1)} \\ b_n^{(n)} \end{bmatrix}$$

e gli elementi $a_{ij}^{(k)}$ e $b_i^{(k)}$ sono dati dalle relazioni (1.9) e (1.10).

Dal procedimento appena descritto seguono due importanti proprietà che legano la matrice dei coefficienti $A^{(1)}$ e tutte le $A^{(k)}$:

1. $A^{(n)} = L^{(n-1)}L^{(n-2)} \dots L^{(1)}A^{(1)}$;
2. $A^{(k)} = L^{(k-1)}L^{(k-2)} \dots L^{(1)}A^{(1)}$.

Applicando la regola di Binet sul determinante del prodotto di matrici e, poichè

$$\det L^{(k)} = 1, \quad k = 1, \dots, n-1$$

segue che:

$$\det A^{(k)} = \det A^{(1)}, \tag{1.12}$$

per ogni k , ed in particolare quando $k = n$:

$$\det A^{(n)} = \det A^{(1)}. \tag{1.13}$$

La relazione (1.13) fornisce un modo alternativo per calcolare il determinante della matrice di partenza A poichè la matrice $A^{(n)}$ è triangolare superiore ed il suo determinante è noto esplicitamente:

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

Nell'eseguire il metodo di Gauss si è fatta l'implicita ipotesi (vedi formule (1.9) e (1.10)) che i cosiddetti *elementi pivotali* $a_{kk}^{(k)}$ siano non nulli per

$k = 1, 2, \dots, n - 1$. In vero questa non è un'ipotesi limitante in quanto la non singolarità di A permette, con un opportuno scambio di righe in $A^{(k)}$, di ricondursi a questo caso. Infatti scambiare due righe in $A^{(k)}$ significa sostanzialmente scambiare due equazioni nel sistema $A^{(k)}\mathbf{x} = \mathbf{b}^{(k)}$ e ciò non altera la natura del sistema stesso.

Consideriamo la matrice $A^{(k)}$ e supponiamo $a_{kk}^{(k)} = 0$. In questo caso possiamo scegliere un elemento sottodiagonale appartenente alla k -esima colonna, supponiamo $a_{ik}^{(k)}$ diverso da zero, scambiare le equazioni di indice i e k e procedere perchè in questo modo l'elemento pivotale è diverso da zero. In ipotesi di non singolarità della matrice A possiamo dimostrare tale elemento diverso da zero esiste sicuramente. Infatti supponendo che, oltre all'elemento pivotale siano nulli anche gli $a_{ik}^{(k)}$ per $i = k + 1, \dots, n$, allora $A^{(k)}$ ha la seguente struttura:

$$A^{(k)} = \begin{bmatrix} a_{11}^{(1)} & \cdots & a_{1,k-1}^{(1)} & a_{1k}^{(1)} & a_{1,k+1}^{(1)} & \cdots & a_{1n}^{(1)} \\ & \ddots & \vdots & \vdots & \vdots & & \vdots \\ & & a_{k-1,k-1}^{(k-1)} & a_{k-1,k}^{(k-1)} & a_{k-1,k+1}^{(k-1)} & \cdots & a_{k-1,n}^{(k-1)} \\ & & & 0 & a_{k,k+1}^{(k)} & & a_{kn}^{(k)} \\ & 0 & & \vdots & \vdots & & \vdots \\ & & & 0 & a_{n,k+1}^{(k)} & \cdots & a_{nn}^{(k)} \end{bmatrix}$$

Partizioniamo $A^{(k)}$ nel seguente modo

$$A^{(k)} = \begin{bmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ 0 & A_{22}^{(k)} \end{bmatrix}$$

con $A_{11}^{(k)} \in \mathbb{R}^{(k-1) \times (k-1)}$. Il determinante di $A^{(k)}$ è

$$\det A^{(k)} = \det A_{11}^{(k)} \det A_{22}^{(k)} = 0$$

perchè la matrice $A_{22}^{(k)}$ ha una colonna nulla. Poichè tutte la matrici $A^{(k)}$ hanno, per ogni k lo stesso determinante di A , dovrebbe essere $\det A = 0$ e questo contrasta con l'ipotesi fatta. Quindi possiamo concludere che se $a_{kk}^{(k)} = 0$ e $\det A \neq 0$ deve necessariamente esistere un elemento $a_{ik}^{(k)} \neq 0$, con $i \in \{k + 1, k + 2, \dots, n\}$.

1.2.5 Esistenza e Unicità della Fattorizzazione LU

Per le condizioni di esistenza della fattorizzazione LU si deve considerare un importante risultato preliminare che però non dimostriamo.

Detta A_k la sottomatrice principale di testa di ordine k della matrice A esiste una relazione che lega gli elementi pivotali e i minori principali di una matrice (cioè i determinanti delle sottomatrici di testa di A). In particolare se

$$A_k = \begin{bmatrix} a_{11} & \dots & a_{1k} \\ \vdots & & \vdots \\ a_{k1} & \dots & a_{kk} \end{bmatrix}$$

allora

$$a_{kk}^{(k)} = \frac{\det A_k}{\det A_{k-1}} \quad k = 2, \dots, n$$

e $a_{11}^{(1)} = \det A_1 = a_{11}$.

Una naturale conseguenza di questo risultato è il seguente teorema di esistenza e unicità della fattorizzazione LU .

Teorema 1.2.1 *Sia A una matrice quadrata di ordine n non singolare. Sia A_k la sottomatrice principale di testa di ordine k di A e sia $\det A_k \neq 0$ per $k = 1, 2, \dots, n$, cioè siano i minori principali di testa non nulli. Allora esistono un'unica matrice L triangolare inferiore con elementi diagonali uguali a 1 ed un'unica U triangolare superiore tali che: $A = LU$.*

1.2.6 Strategia di Pivoting Parziale

Le strategie di pivoting nel metodo di Gauss hanno principalmente lo scopo di evitare gli elementi pivotali nulli. Infatti al k -esimo passo la matrice $A^{(k)}$ si presenta nella forma:

$$\begin{bmatrix} a_{11}^{(1)} & \dots & a_{1,k-1}^{(1)} & a_{1k}^{(1)} & \dots & a_{1n}^{(1)} \\ & \ddots & \vdots & \vdots & & \vdots \\ & & a_{k-1,k-1}^{(k-1)} & a_{k-1,k}^{(k-1)} & \dots & a_{k-1,n}^{(k-1)} \\ & & 0 & a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ & & \vdots & \vdots & & \vdots \\ & & 0 & a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{bmatrix}.$$

Il passo successivo consiste nell'azzerare gli elementi al di sotto dell'elemento $a_{kk}^{(k)}$ situati nella k -esima colonna. La strategia di *Pivoting parziale* prevede che prima di fare ciò si ricerchi l'elemento di massimo modulo tra gli elementi $a_{kk}^{(k)}, a_{k+1,k}^{(k)}, \dots, a_{nk}^{(k)}$ e si scambi la riga in cui si trova questo elemento con la k -esima qualora esso sia diverso da $a_{kk}^{(k)}$. In altri termini il pivoting parziale richiede le seguenti operazioni:

1. determinare l'elemento $a_{rk}^{(k)}$ tale che

$$|a_{rk}^{(k)}| = \max_{k \leq i \leq n} |a_{ik}^{(k)}|;$$

2. effettuare lo scambio tra la r -esima e la k -esima riga.

1.2.7 Fattorizzazione di Matrici Non Singolari

Abbiamo visto che se A , matrice quadrata di ordine n , è non singolare e tutti i suoi minori principali di testa fino all'ordine $n - 1$ sono non nulli allora A ammette un'unica fattorizzazione LU . Se A è solo non singolare non è detto che ammetta fattorizzazione LU . In particolare si può dimostrare che è sempre possibile trovare una *matrice di permutazione* P (vedere definizione 1.1.1) tale che $PA = LU$.

Infatti vale il seguente risultato.

Teorema 1.2.2 *Se A è una matrice quadrata non singolare allora esistono una matrice di permutazione P , una matrice L triangolare inferiore con elementi uguali a 1 sulla diagonale principale ed una matrice U triangolare superiore tali che:*

$$PA = LU.$$

1.2.8 Classi di Matrici che non hanno bisogno di Pivoting

Tra le classi di matrici che non hanno bisogno di alcuna strategia di pivoting è opportuno citare le matrici a predominanza per righe e le matrici simmetriche e definite positive.

Definizione 1.2.1 *A si dice a predominanza diagonale per colonne se:*

$$|a_{jj}| \geq \sum_{i=1, i \neq j}^n |a_{ij}|, \quad j = 1, 2, \dots, n.$$

Infatti è possibile dimostrare che la predominanza diagonale è invariante sotto trasformazioni elementari di Gauss, cioè le sottomatrici trasformate ad ogni passo sono pure a predominanza diagonale per colonne di conseguenza non è necessario utilizzare alcuna strategia di pivoting.

Definizione 1.2.2 Una matrice quadrata A , simmetrica, di ordine n si dice definita positiva se e solo se:

$$\mathbf{x}^T A \mathbf{x} > 0 \quad \text{per ogni } \mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq 0.$$

Uno dei criteri possibili per determinare se una matrice è definita positiva è fornito dal *criterio di Sylvester* che stabilisce che una matrice A simmetrica di ordine n è definita positiva se e solo se $\det A_k > 0$ per $k = 1, 2, \dots, n$, dove A_k è la sottomatrice principale di testa di ordine k . Poichè è noto che

$$a_{kk}^{(k)} = \frac{\det A_k}{\det A_{k-1}} \quad k = 1, \dots, n, \quad \det A_0 = 1$$

dove gli $a_{kk}^{(k)}$ sono gli elementi pivotali nel metodo di Gauss. Ad ogni passo l'elemento pivotale è positivo quindi non è necessaria alcuna strategia di pivoting.

1.2.9 Fattorizzazione Diretta

Quando risolviamo un sistema lineare con il metodo di Gauss dobbiamo calcolare circa $n^3/3$ risultati intermedi. È comunque possibile riarrangiare i calcoli in modo tale da valutare direttamente gli elementi di L ed U . Il metodo che ne risulta prende il nome di *Metodo di Doolittle*.

Fissata la matrice A , quadrata di ordine n , imponiamo che risulti

$$A = LU$$

con L matrice triangolare inferiore con elementi diagonali uguali a 1 e U matrice triangolare superiore. Una volta note tali matrici il sistema di partenza $A\mathbf{x} = \mathbf{b}$ viene scritto come

$$LU\mathbf{x} = \mathbf{b}$$

e, posto $U\mathbf{x} = \mathbf{y}$, il vettore \mathbf{x} viene trovato prima risolvendo il sistema triangolare inferiore

$$L\mathbf{y} = \mathbf{b}$$

e poi quello triangolare superiore

$$U\mathbf{x} = \mathbf{y}.$$

Imponiamo quindi che la matrice A ammetta fattorizzazione LU :

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \dots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & \vdots \\ \vdots & \vdots & \dots & 0 \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & u_{nn} \end{bmatrix}.$$

Deve essere

$$a_{ij} = \sum_{k=1}^{\min(i,j)} l_{ik}u_{kj} \quad i, j = 1, \dots, n. \quad (1.14)$$

Uguagliando la parte triangolare superiore di A abbiamo

$$a_{ij} = \sum_{k=1}^i l_{ik}u_{kj} \quad j \geq i \quad (1.15)$$

ovvero

$$a_{ij} = \sum_{k=1}^{i-1} l_{ik}u_{kj} + l_{ii}u_{ij} \quad j \geq i.$$

Imponendo $l_{ii} = 1$ risulta

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj} \quad j \geq i \quad (1.16)$$

e ovviamente $u_{1j} = a_{1j}$, per $j = 1, \dots, n$. Uguagliamo ora la parte triangolare strettamente inferiore di A . Sempre da (1.14) risulta:

$$a_{ij} = \sum_{k=1}^j l_{ik}u_{kj} \quad i > j \quad (1.17)$$

ovvero

$$a_{ij} = \sum_{k=1}^{j-1} l_{ik}u_{kj} + l_{ij}u_{jj} \quad i > j$$

da cui

$$l_{ij} = \frac{1}{u_{jj}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj} \right) \quad i \geq j. \quad (1.18)$$

Si osservi che le formule (1.16) e (1.18) vanno implementate secondo uno degli schemi riportati in Figura 1.1.

Ogni schema rappresenta in modo stilizzato una matrice la cui parte triangolare superiore indica la matrice U mentre quella triangolare inferiore la matrice L mentre i numeri indicano l'ordine con cui gli elementi saranno calcolati. Per esempio applicando la tecnica di Crout si segue il seguente ordine:

- 1° Passo: Calcolo della prima riga di U ;
- 2° Passo: Calcolo della seconda riga di L ;
- 3° Passo: Calcolo della seconda riga di U ;
- 4° Passo: Calcolo della terza riga di L ;
- 5° Passo: Calcolo della terza riga di U ;
- 6° Passo: Calcolo della quarta riga di L ;
- 7° Passo: Calcolo della quarta riga di U ;

e così via procedendo per righe in modo alternato. Nel caso della tecnica di Doolittle si seguono i seguenti passi:

- 1° Passo: Calcolo della prima riga di U ;
- 2° Passo: Calcolo della prima colonna di L ;
- 3° Passo: Calcolo della seconda riga di U ;
- 4° Passo: Calcolo della seconda colonna di L ;
- 5° Passo: Calcolo della terza riga di U ;

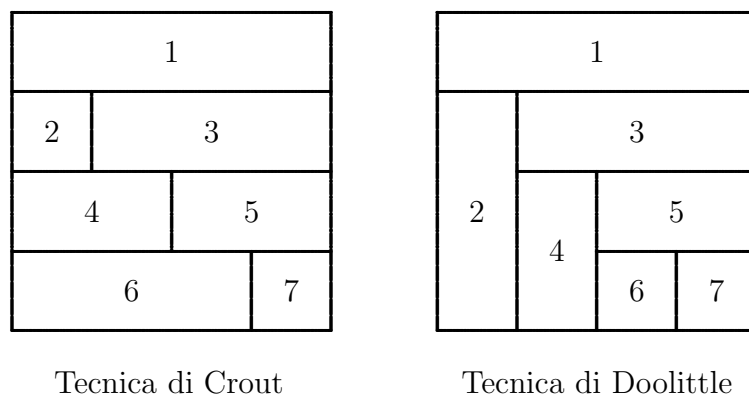


Figura 1.1: Tecniche di calcolo diretto della fattorizzazione di una matrice

- 6° Passo: Calcolo della terza colonna di L ;
- 7° Passo: Calcolo della quarta riga di U .

Lo svantaggio del metodo di fattorizzazione diretto risiede essenzialmente nella maggiore difficoltà, rispetto al metodo di Gauss, di poter programmare una strategia di pivot. Per matrici simmetriche e definite positive, poichè non è necessaria alcuna strategia di pivoting, il metodo diventa particolarmente vantaggioso.

1.2.10 Condizionamento di sistemi lineari

Uno dei più importanti aspetti legati all'uso di algoritmi numerici per la soluzione di problemi matematici è l'affidabilità dei risultati ottenuti. I metodi descritti in questo capitolo funzionerebbero sempre bene (cioè fornirebbero risultati numericamente affidabili) se non si dovesse aver a che fare con una serie di problemi legati alla struttura dell'elaboratore che stiamo utilizzando. Un primo problema che ci troviamo ad affrontare è il modo con cui i numeri reali sono rappresentati nella memoria di un elaboratore. Giusto per rendersi conto delle difficoltà che si incontrano va osservato che i numeri reali sono infiniti mentre la memoria di un calcolatore ha una capacità finita. Una seconda osservazione consiste nel fatto che un numero reale ammette molteplici rappresentazioni. Per esempio il numero 12.47 può essere scritto in diversi modi

$$12.47 = 1.247 \times 10^1 = 0.1247 \times 10^2 = 1247 \times 10^{-2}.$$

Questa prima ambiguità viene risolta convenzionalmente utilizzando come rappresentazione la seguente:

$$x = \text{segno}(x) q \times 10^p$$

dove $\text{segno}(x) = \pm 1$, $q = 0.d_1d_2d_3 \dots d_k \dots$ è un numero reale positivo minore di 1 con le cifre d_i comprese tra 0 e 9 se si utilizza la rappresentazione in base 10, e si impone $d_1 \neq 0$, cioè

$$\frac{1}{10} \leq q < 1.$$

Lo stesso discorso si può ripetere scegliendo una qualsiasi base $\beta \in \mathbb{N}$, cioè

$$x = \text{segno}(x) q \times \beta^p, \quad \text{dove } q = 0.d_1d_2d_3 \dots d_k \dots$$

con le cifre d_i comprese tra 0 e $\beta - 1$, e $d_1 \neq 0$, cioè

$$\frac{1}{\beta} \leq q < 1.$$

Assegnato $x \in \mathbb{R}$, $x \neq 0$, l'espressione

$$x = \text{segno}(x) \beta^p \times 0.d_1d_2 \dots d_k \dots$$

prende il nome di *rappresentazione in base β di x* . Il numero p viene detto *esponente* (o *caratteristica*), i valori d_i sono le cifre della rappresentazione, mentre $0.d_1d_2 \dots d_k \dots$ si dice *mantissa*. Il numero x viene normalmente rappresentato con la cosiddetta *notazione posizionale* $x = \text{segno}(x)(.d_1d_2d_3 \dots) \times \beta^p$, che viene detta *normalizzata*. In alcuni casi è ammessa una rappresentazione in notazione posizionale tale che $d_1 = 0$, che viene detta *denormalizzata*. Quindi un qualunque numero reale $x \neq 0$ può essere rappresentato con *infinite cifre* nella mantissa. Inoltre come è noto l'insieme dei numeri reali ha cardinalità infinita. Poichè un elaboratore è dotato di *memoria finita* non è possibile memorizzare:

- a) gli infiniti numeri reali
- b) le infinite (in generale) cifre di un numero reale.

Risulta perciò importante stabilire dei criteri che permettano di rappresentare in macchina i numeri reali che è possibile rappresentare in modo da commettere il minimo errore possibile.

Assegnati i numeri $\beta, t, m, M \in \mathbb{N}$ con $\beta \geq 2$, $t \geq 1$, $m, M > 0$, si dice *insieme dei numeri di macchina con rappresentazione normalizzata in base β con t cifre significative* l'insieme:

$$\mathcal{F}(\beta, t, m, M) = \{ x \in \mathbb{R} : x = \pm \beta^p \times 0.d_1d_2 \dots d_t \text{ con } 0 \leq d_i \leq \beta - 1,$$

$$d_1 \neq 0, -m \leq p \leq M \} \cup \{0\}.$$

Infatti poichè zero sfugge alla rappresentazione in base normalizzata viene assegnato per definizione all'insieme \mathcal{F} . Normalmente lo zero viene rappresentato con mantissa nulla ed esponente $-m$.

Osserviamo che un elaboratore che abbia le seguenti caratteristiche:

- t campi di memoria per la mantissa, ciascuno dei quali può assumere β differenti configurazioni (e perciò può memorizzare una cifra d_i),
- un campo di memoria che può assumere $m + M + 1$ differenti configurazioni (e perciò può memorizzare i differenti valori p dell'esponente),
- un campo che può assumere due differenti configurazioni (e perciò può memorizzare il segno $+ o -$),

è in grado di rappresentare tutti gli elementi dell'insieme $\mathcal{F}(\beta, t, m, M)$.

Assumiamo ora che $x \in \mathbb{R}$ e che abbia la seguente rappresentazione in base β :

$$x = \text{segno}(x) \beta^p \times 0.d_1d_2 \dots d_t$$

con $d_1 \neq 0$ e $p \in [-m, M]$. Allora è evidente che $x \in \mathcal{F}(\beta, t, m, M)$ e pertanto verrà rappresentato esattamente su un qualunque elaboratore che utilizzi $\mathcal{F}(\beta, t, m, M)$ come insieme dei numeri di macchina.

Assumiamo ora che $x \in \mathbb{R}$ ma $x \notin \mathcal{F}(\beta, t, m, M)$. In questo caso si pone il problema di associare ad x un numero di macchina che lo rappresenti in modo da commettere il più piccolo errore possibile. Per risolvere questo problema facciamo innanzitutto, e solo per semplicità di esposizione, le seguenti ipotesi $x \in \mathbb{R}$, $x > 0$ e β numero pari.

Distinguiamo quindi i seguenti casi:

- a) $p \notin [-m, M]$. Se $p < -m$ allora x è più piccolo del più piccolo numero di macchina: in questo caso si dice che si è verificato un *underflow* (l'elaboratore interrompe la sequenza di calcoli e segnala con un messaggio l'underflow). Se $p > M$ allora vuol dire che x è più grande del più grande numero di macchina e in questo caso si dice che si è verificato un *overflow* (anche in questo caso l'elaboratore si ferma e segnala l'overflow).
- b) $p \in [-m, M]$, $x = \beta^p \times 0.d_1d_2 \dots d_t \dots$, ed esiste un $k > t$ tale che $d_k \neq 0$. Anche in questo caso poichè x ha più di t cifre significative $x \notin \mathcal{F}$. È però possibile rappresentare x mediante un numero in \mathcal{F} con

un'opportuna operazione di taglio delle cifre decimali che seguono la t -esima. In pratica i criteri di taglio sono i seguenti:

1. *troncamento di x alla t -esima cifra significativa*

$$\tilde{x} = \text{tr}(x) = \beta^p \times 0.d_1d_2 \dots d_t$$

2. *arrotondamento di x alla t -esima cifra significativa*

$$\tilde{x} = \text{arr}(x) = \beta^p \times 0.d_1d_2 \dots \tilde{d}_t$$

dove

$$\tilde{d}_t = \begin{cases} d_t + 1 & \text{se } d_{t+1} \geq \beta/2 \\ d_t & \text{se } d_{t+1} < \beta/2. \end{cases}$$

Se $x \in \mathbb{R}$ e \tilde{x} è la sua rappresentazione di macchina, chiameremo *errore assoluto* la quantità

$$E_a = |x - \tilde{x}|$$

mentre per $x \neq 0$ chiameremo *errore relativo* la quantità

$$E_r = \frac{|x - \tilde{x}|}{|x|}.$$

Nel seguito assumeremo $x > 0$ e supporremo anche che la rappresentazione di x in $\mathcal{F}(\beta, t, m, M)$ non dia luogo ad underflow o overflow.

Teorema 1.2.3 *Sia $x = \pm\beta^p 0.d_1d_2 \dots d_t \dots$ tale che la sua rappresentazione macchina non dia luogo a fenomeni di underflow o overflow, allora risulta:*

$$\begin{aligned} |\text{tr}(x) - x| &< \beta^{p-t} \\ |\text{arr}(x) - x| &\leq \frac{1}{2}\beta^{p-t} \end{aligned}$$

dove il segno di uguaglianza vale se e solo se $d_{t+1} = \frac{\beta}{2}$ e $d_{t+i} = 0$ per $i \geq 2$.

Teorema 1.2.4 *Sia $x = \pm\beta^p 0.d_1d_2 \dots d_t \dots$, $x \neq 0$, se \tilde{x} è la sua rappresentazione di macchina cioè $\tilde{x} \in \mathcal{F}(\beta, t, m, M)$, allora*

$$\left| \frac{\tilde{x} - x}{x} \right| < u$$

$$\left| \frac{\tilde{x} - x}{\tilde{x}} \right| < u$$

dove

$$u = \begin{cases} \beta^{-t+1} & \text{se } \tilde{x} = \text{tr}(x) \\ \frac{1}{2}\beta^{-t+1} & \text{se } \tilde{x} = \text{arr}(x). \end{cases}$$

La quantità u che interviene nel precedente teorema si chiama *precisione di macchina* o *zero macchina*. La rappresentazione di $x \in \mathbb{R}$ attraverso $\tilde{x} \in \mathcal{F}(\beta, t, m, M)$ si dice *rappresentazione in virgola mobile di x* o *rappresentazione floating point*, con troncamento se $\tilde{x} = \text{tr}(x)$, con arrotondamento se $\tilde{x} = \text{arr}(x)$.

Se \tilde{x} è un'approssimazione di $x \in \mathbb{R}$ con un errore relativo minore di β^{1-t} , si dice che t cifre della rappresentazione in base β sono *significative*.

Un problema simile si presenta anche quando si effettuano delle operazioni aritmetiche su numeri reali. In fatti non è garantito, in generale, che un'operazione aritmetica su due numeri macchina fornisca come risultato un numero di macchina. Per esempio considerati $x = (.11)10^0$ e $y = (.11)10^{-2}$, $x, y \in \mathcal{F}(10, 2, m, M)$, si ha:

$$x + y = (.1111)10^0 \notin \mathcal{F}(10, 2, m, M)$$

Per poter realizzare la naturale ed importante *Proprietà di chiusura* di una operazione in un certo insieme risulta importante definire delle *operazioni di macchina* che permettano appunto di realizzare tale proprietà. Un requisito essenziale che si richiede nel costruire un'aritmetica di macchina è il seguente. Indicata con \cdot una delle quattro operazioni aritmetiche $+$, $-$, \times , \div e con \odot la corrispondente operazione di macchina dev'essere:

$$x \odot y = (x \cdot y)(1 + \varepsilon), \quad |\varepsilon| < u \quad (1.19)$$

per ogni $x, y \in \mathcal{F}(\beta, t, m, M)$ tali che $x \odot y$ non dia luogo ad overflow o underflow. Si può dimostrare che

$$x \odot y = \text{tr}(x \cdot y)$$

e

$$x \odot y = \text{arr}(x \cdot y)$$

soddisfano la (1.19) e dunque danno luogo ad operazioni di macchina. Le quattro operazioni così definite danno luogo alla *aritmetica di macchina* o

aritmetica finita.

Si può dimostrare che per le operazioni di macchina non valgono alcune proprietà, come per esempio l'associatività dell'addizione e della moltiplicazione o la distributività della moltiplicazione rispetto all'addizione, che invece sono valide per le operazioni tra numeri reali.

Supponiamo ora di voler valutare la differenza tra due numeri reali x e y . Siano $fl(x)$ e $fl(y)$ rispettivamente le loro rappresentazioni di macchina. Vogliamo vedere quale è l'errore relativo che viene commesso dall'elaboratore quando calcola $x - y$.

$$\begin{aligned} fl(x) \ominus fl(y) &= [fl(x) - fl(y)](1 + \varepsilon) = \\ &= [x(1 + \varepsilon_x) - y(1 + \varepsilon_y)](1 + \varepsilon) = \\ &= (x + x\varepsilon_x - y - y\varepsilon_y)(1 + \varepsilon) = \\ &= (x - y) + (x - y)\varepsilon + x\varepsilon_x - y\varepsilon_y + x\varepsilon\varepsilon_x - y\varepsilon\varepsilon_y. \end{aligned}$$

Una maggiorazione per l'errore relativo è la seguente

$$\begin{aligned} \frac{|(fl(x) \ominus fl(y)) - (x - y)|}{|x - y|} &\leq |\varepsilon| + \frac{|x|}{|x - y|}(|\varepsilon_x| + |\varepsilon||\varepsilon_x|) + \\ &+ \frac{|y|}{|x - y|}(|\varepsilon_y| + |\varepsilon||\varepsilon_y|). \end{aligned} \tag{1.20}$$

Se x e y hanno segno opposto risulta

$$\max(|x|, |y|) \leq |x - y|$$

e dalla (1.20) segue la maggiorazione

$$\frac{|(fl(x) \ominus fl(y)) - (x - y)|}{|x - y|} \leq 3u + O(u^2)$$

dove u è la precisione di macchina ed $O(u^2)$ indica la maggiorazione per i cosiddetti termini quadratici dell'errore (cioè i prodotti del tipo $\varepsilon\varepsilon_x$, infatti tali quantità sono maggiorabili singolarmente in modulo dalla precisione di macchina ed il loro prodotto è una quantità trascurabile).

Se x e y hanno lo stesso segno allora l'errore relativo può essere molto grande

quanto più x e y sono vicini. Questo fenomeno prende il nome di *cancellazione di cifre significative*.

Tornando al problema della risoluzione di un sistema lineare è ovvio che gli errori nella rappresentazione dei dati del problema A e \mathbf{b} hanno come conseguenza il fatto che, invece di risolvere il sistema

$$A\mathbf{x} = \mathbf{b}$$

si risolve il sistema perturbato

$$(A + \delta A)(\mathbf{x} + \delta \mathbf{x}) = \mathbf{b} + \delta \mathbf{b}.$$

La quantità $\delta \mathbf{x}$ rappresenta quanto la soluzione del sistema che effettivamente stiamo risolvendo è distante dalla soluzione del sistema che invece vorremmo risolvere. Questa proprietà, che è tipica del problema che vogliamo risolvere e non del metodo che stiamo utilizzando, prende il nome di *condizionamento del problema* e si tratta di qualcosa che non è neanche strettamente legato ai sistemi lineari in quanto può essere definito per un qualsiasi problema matematico. In particolare si parla di *problema bencondizionato* se piccole perturbazioni sui dati iniziali provocano una piccola perturbazione dei dati di uscita (quindi se le quantità δA , $\delta \mathbf{b}$ e $\delta \mathbf{x}$ sono tutte piccole). Si parla di *problema malcondizionato* se piccole perturbazioni sui dati iniziali provocano un grande cambiamento dei dati di uscita (quindi se le quantità δA e $\delta \mathbf{b}$ sono piccole e invece $\delta \mathbf{x}$ è molto grande). Nel caso dei sistemi lineari si può provare che vale la seguente relazione

$$\frac{\|\delta \mathbf{x}\|}{\|\mathbf{x}\|} \leq \|A\| \|A^{-1}\| \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta \mathbf{b}\|}{\|\mathbf{b}\|} \right).$$

La quantità $\|A\| \|A^{-1}\|$ prende il nome di *indice di condizionamento di A* e viene considerato a tutti gli effetti come una misura del condizionamento del sistema. In particolare esso misura di quanto una matrice è vicina ad essere singolare.

Capitolo 2

Interpolazione e Quadratura di Funzioni

2.1 Introduzione

Nel campo del Calcolo Numerico si possono incontrare diversi casi nei quali è richiesta l'approssimazione di una funzione (o di una grandezza incognita): 1) non è nota l'espressione analitica della funzione $f(x)$ ma si conosce il valore che assume in un insieme finito di punti x_1, x_2, \dots, x_n . Si potrebbe pensare anche che tali valori siano delle misure di una grandezza fisica incognita valutate in differenti istanti di tempo.

2) Si conosce l'espressione analitica della funzione $f(x)$ ma è così complicata dal punto di vista computazionale che è più conveniente cercare un'espressione semplice partendo dal valore che essa assume in un insieme finito di punti.

In questo capitolo analizzeremo un particolare tipo di approssimazione di funzioni cioè la cosiddetta interpolazione che richiede che la funzione approssimante assume in determinate ascisse esattamente lo stesso valore di $f(x)$.

In entrambi i casi appena citati è noto, date certe informazioni supplementari, che la funzione approssimante va ricercata della forma:

$$f(x) \simeq g(x; a_0, a_1, \dots, a_n). \quad (2.1)$$

Se i parametri a_0, a_1, \dots, a_n sono definiti dalla condizione di coincidenza di f e g nei punti x_0, x_1, \dots, x_n , allora tale procedimento di approssimazione

CAPITOLO 2. INTERPOLAZIONE E QUADRATURA DI FUNZIONI 42

si chiama appunto *Interpolazione*. Invece se $x \notin [\min_i x_i, \max_i x_i]$ allora si parla di *Estrapolazione*.

Tra i procedimenti di interpolazione il più usato è quello in cui si cerca la funzione g in (2.1) nella forma

$$g(x; a_0, a_1, \dots, a_n) = \sum_{i=0}^n a_i \Phi_i(x)$$

dove $\Phi_i(x)$, per $i = 0, \dots, n$, sono funzioni fissate e i valori di a_i , $i = 0, \dots, n$, sono determinati in base alle condizioni di coincidenza di f con la funzione approssimante nei punti di interpolazione (detti anche *odi*), x_j , cioè si pone

$$f(x_j) = \sum_{i=0}^n a_i \Phi_i(x_j) \quad j = 0, \dots, n. \quad (2.2)$$

Il processo di determinazione degli a_i attraverso la risoluzione del sistema (2.2) si chiama *metodo dei coefficienti indeterminati*. Il caso più studiato è quello dell'interpolazione polinomiale, in cui si pone:

$$\Phi_i(x) = x^i \quad i = 0, \dots, n$$

e perciò la funzione approssimante g assume la forma

$$\sum_{i=0}^n a_i x^i;$$

mentre le condizioni di coincidenza diventano

$$f(x_j) = \sum_{i=0}^n a_i x_j^i \quad j = 0, \dots, n. \quad (2.3)$$

Nell'ipotesi che i nodi x_j siano tutti distinti si può provare che la matrice dei coefficienti del sistema (2.3) è non singolare (*matrice di Vandermonde*) e pertanto il problema dell'interpolazione ammette sempre un'unica soluzione. Il polinomio interpolante fornito da (2.3) può essere ottenuto anche in modo esplicito. Va comunque notato che in generale è molto difficile ottenere una funzione di interpolazione in forma esplicita, il che rende necessario risolvere il sistema (2.3).

2.1.1 Il Polinomio Interpolante di Lagrange

Al fine di dare una forma esplicita al polinomio interpolante, scriviamo il candidato polinomio nella seguente forma:

$$L_n(x) = \sum_{k=0}^n l_{nk}(x) f(x_k) \quad (2.4)$$

dove gli $l_{nk}(x)$ sono per il momento generici polinomi di grado n . Imponendo le condizioni di interpolazione

$$L_n(x_i) = f(x_i) \quad i = 0, \dots, n$$

deve essere, per ogni i :

$$L_n(x_i) = \sum_{k=0}^n l_{nk}(x_i) f(x_k) = f(x_i)$$

ed è evidente che se

$$l_{nk}(x_i) = \begin{cases} 0 & \text{se } k \neq i \\ 1 & \text{se } k = i \end{cases} \quad (2.5)$$

allora esse sono soddisfatte. In particolare la prima condizione di (2.5) indica che $l_{nk}(x)$ si annulla negli n nodi $x_0, x_1, \dots, x_{k-1}, x_{k+1}, \dots, x_n$ e quindi deve avere la seguente struttura:

$$l_{nk}(x) = c_k \prod_{i=0, i \neq k}^n (x - x_i)$$

mentre imponendo la seconda condizione di (2.5)

$$l_{nk}(x_k) = c_k \prod_{i=0, i \neq k}^n (x_k - x_i) = 1$$

si trova immediatamente:

$$c_k = \frac{1}{\prod_{i=0, i \neq k}^n (x_k - x_i)}.$$

In definitiva il polinomio interpolante ha la seguente forma:

$$L_n(x) = \sum_{k=0}^n \left(\prod_{i=0, i \neq k}^n \frac{x - x_i}{x_k - x_i} \right) f(x_k). \quad (2.6)$$

Il polinomio (2.6) prende il nome di *Polinomio di Lagrange* mentre i polinomi:

$$l_{nk}(x) = \prod_{i=0, i \neq k}^n \frac{x - x_i}{x_k - x_i}; \quad k = 0, 1, \dots, n$$

si chiamano *Polinomi Fondamentali di Lagrange*.

2.1.2 Il Resto del Polinomio di Lagrange

Assumiamo che la funzione interpolata $f(x)$ sia di classe $\mathcal{C}^{n+1}([a, b])$ e valutiamo l'errore che si commette nel sostituire $f(x)$ con $L_n(x)$ in un punto $x \neq x_i$. Supponiamo che l'intervallo $[a, b]$ sia tale da contenere sia i nodi x_i che l'ulteriore punto x . Sia dunque

$$e(x) = f(x) - L_n(x)$$

l'errore (o resto) commesso nell'interpolazione della funzione $f(x)$. Poichè

$$e(x_i) = f(x_i) - L_n(x_i) = 0 \quad i = 0, \dots, n$$

è facile congetturare per $e(x)$ la seguente espressione:

$$e(x) = c(x)\omega_{n+1}(x)$$

dove

$$\omega_{n+1}(x) = \prod_{i=0}^n (x - x_i)$$

è il cosiddetto *polinomio nodale* mentre $c(x)$ è una funzione da determinare. Definiamo ora la funzione

$$\Phi(t; x) = f(t) - L_n(t) - c(x)\omega_{n+1}(t)$$

dove t è una variabile ed x è un valore fissato. Calcoliamo la funzione $\Phi(t; x)$ nei nodi x_i :

$$\Phi(x_i; x) = f(x_i) - L_n(x_i) - c(x)\omega_{n+1}(x_i) = 0$$

e anche nel punto x :

$$\Phi(x; x) = f(x) - L_n(x) - c(x)\omega_{n+1}(x) = e(x) - c(x)\omega_{n+1}(x) = 0$$

pertanto la funzione $\Phi(t; x)$ (che è derivabile con continuità $n+1$ volte poichè $f(x)$ è di classe \mathcal{C}^{n+1}) ammette almeno $n+2$ zeri distinti. Applicando il teorema di Rolle segue che $\Phi'(t; x)$ ammette almeno $n+1$ zeri distinti. Riapplicando lo stesso teorema segue che $\Phi''(t; x)$ ammette almeno n zeri distinti. Così proseguendo segue che

$$\exists \xi_x \in [a, b] \ni \Phi^{(n+1)}(\xi_x; x) = 0.$$

Calcoliamo ora la derivata di ordine $n+1$ della funzione $\Phi(t; x)$, osservando innanzitutto che la derivata di tale ordine del polinomio $L_n(x)$ è identicamente nulla. Pertanto

$$\Phi^{(n+1)}(t; x) = f^{(n+1)}(t) - c(x) \frac{d^{n+1}}{dt^{n+1}} \omega_{n+1}(t).$$

Calcoliamo la derivata di ordine $n+1$ del polinomio nodale. Osserviamo innanzitutto che

$$\omega_{n+1}(t) = \prod_{i=0}^n (t - x_i) = t^{n+1} + p_n(t)$$

dove $p_n(t)$ è un polinomio di grado al più n . Quindi

$$\frac{d^{n+1}}{dt^{n+1}} \omega_{n+1}(t) = \frac{d^{n+1}}{dt^{n+1}} t^{n+1}.$$

Poichè

$$\frac{d}{dt} t^{n+1} = (n+1)t^n$$

e

$$\frac{d^2}{dt^2} t^{n+1} = (n+1)nt^{n-1}$$

è facile dedurre che

$$\frac{d^{n+1}}{dt^{n+1}} t^{n+1} = \frac{d^{n+1}}{dt^{n+1}} \omega_{n+1}(t) = (n+1)!.$$

Pertanto

$$\Phi^{(n+1)}(t; x) = f^{(n+1)}(t) - c(x)(n+1)!$$

e quindi

$$\Phi^{(n+1)}(\xi_x; x) = f^{(n+1)}(\xi_x) - c(x)(n+1)! = 0$$

cioè

$$c(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}$$

e in definitiva

$$e(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \omega_{n+1}(x). \quad (2.7)$$

Esempio 2.1.1 *Supponiamo di voler calcolare il polinomio interpolante di Lagrange passante per i punti $(-1, -1)$, $(0, 1)$, $(1, -1)$, $(3, 2)$ e $(5, 6)$. Il grado di tale polinomio è 4, quindi definiamo i nodi*

$$x_0 = -1, \quad x_1 = 0, \quad x_2 = 1, \quad x_3 = 3, \quad x_4 = 5,$$

cui corrispondono le ordinate che indichiamo con y_i , $i = 0, \dots, 4$:

$$y_0 = -1, \quad y_1 = 1, \quad y_2 = -1, \quad y_3 = 2, \quad y_4 = 6.$$

Scriviamo ora l'espressione del polinomio $L_4(x)$:

$$L_4(x) = l_{4,0}(x)y_0 + l_{4,1}(x)y_1 + l_{4,2}(x)y_2 + l_{4,3}(x)y_3 + l_{4,4}(x)y_4 \quad (2.8)$$

e calcoliamo i 5 polinomi fondamentali di Lagrange:

$$\begin{aligned} l_{4,0}(x) &= \frac{(x-0)(x-1)(x-3)(x-5)}{(-1-0)(-1-1)(-1-3)(-1-5)} = \\ &= \frac{1}{48} x(x-1)(x-3)(x-5) \\ l_{4,1}(x) &= \frac{(x+1)(x-1)(x-3)(x-5)}{(0+1)(0-1)(0-3)(0-5)} = \\ &= -\frac{1}{15}(x+1)(x-1)(x-3)(x-5) \\ l_{4,2}(x) &= \frac{(x+1)(x-0)(x-3)(x-5)}{(1+1)(1-0)(1-3)(1-5)} = \\ &= \frac{1}{16} x(x+1)(x-3)(x-5) \end{aligned}$$

$$\begin{aligned}
l_{4,3}(x) &= \frac{(x+1)(x-0)(x-1)(x-5)}{(3+1)(3-0)(3-1)(3-5)} = \\
&= -\frac{1}{48}x(x+1)(x-1)(x-5) \\
l_{4,4}(x) &= \frac{(x+1)(x-0)(x-1)(x-3)}{(5+1)(5-0)(5-1)(5-3)} = \\
&= \frac{1}{240}x(x+1)(x-1)(x-3)
\end{aligned}$$

Sostituendo in (2.8) il valore della funzione nei nodi si ottiene l'espressione finale del polinomio interpolante:

$$L_4(x) = -l_{4,0}(x) + l_{4,1}(x) - l_{4,2}(x) + 2l_{4,3}(x) + 6l_{4,4}(x).$$

Se vogliamo calcolare il valore approssimato della funzione $f(x)$ in un'ascissa diversa dai nodi, per esempio $x = 2$ allora dobbiamo calcolare il valore del polinomio interpolante $L_4(2)$.

Nelle figure 2.1-2.5 sono riportati i grafici dei cinque polinomi fondamentali di Lagrange: gli asterischi evidenziano il valore assunto da tali polinomi nei nodi di interpolazione. Nella figura 2.6 è tracciato il grafico del polinomio interpolante di Lagrange, i cerchi evidenziano ancora una volta i punti di interpolazione.

Il polinomio interpolante di Lagrange ha il pregio di fornire un'espressione esplicita del polinomio stesso ma ha un grosso svantaggio. Infatti se, una volta calcolato $L_n(x)$, supponiamo di venire a conoscenza di un ulteriore valore $(x_{n+1}, f(x_{n+1}))$ si devono ripetere tutti i calcoli. Questa motivazione ci spinge a cercare una diversa rappresentazione del polinomio interpolante.

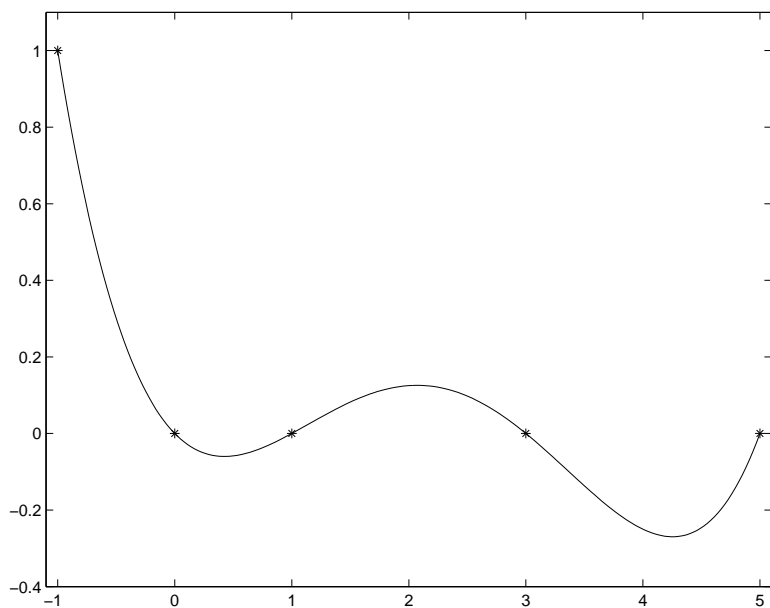


Figura 2.1: Grafico del polinomio $l_{40}(x)$.

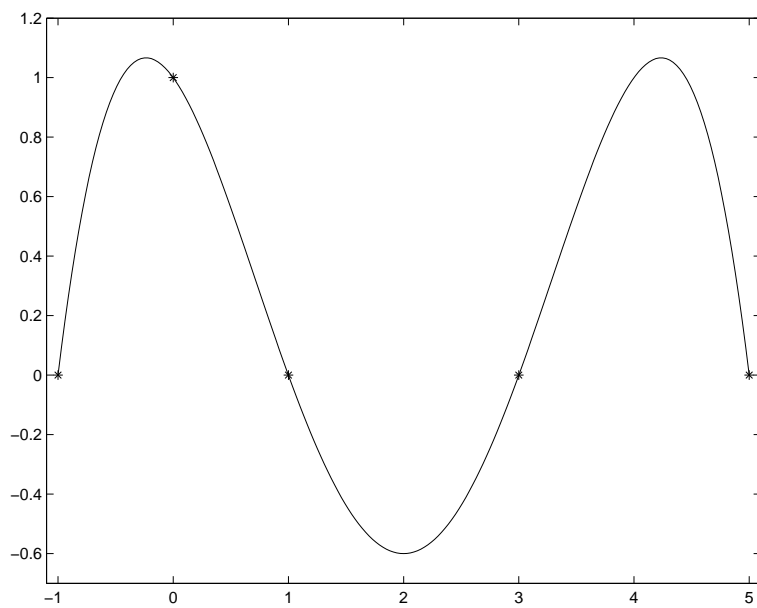


Figura 2.2: Grafico del polinomio $l_{41}(x)$.

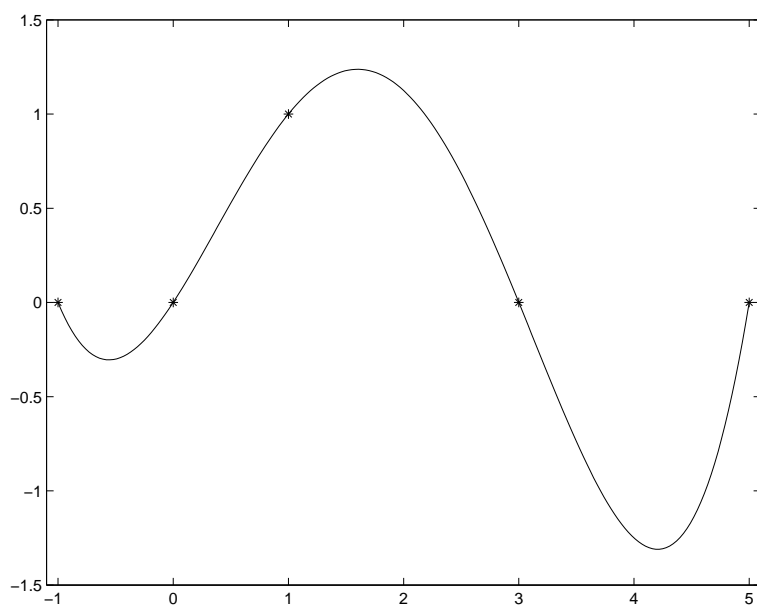


Figura 2.3: Grafico del polinomio $l_{42}(x)$.

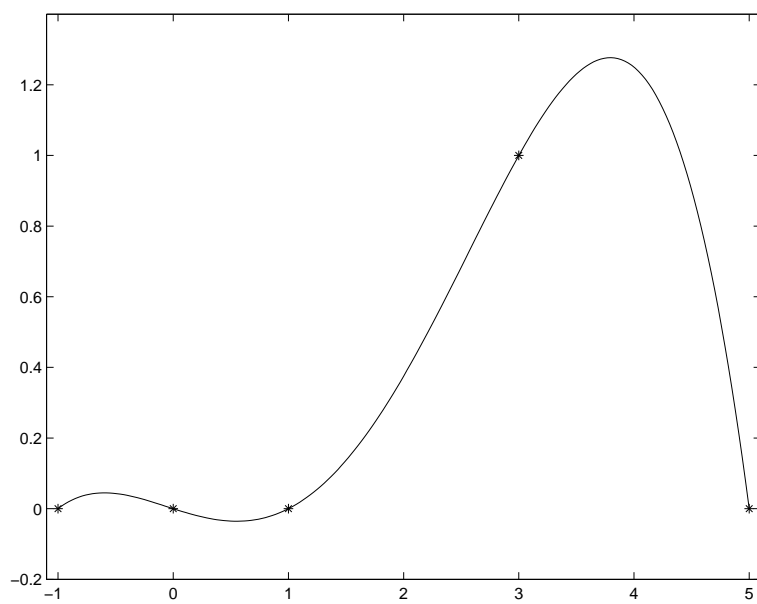


Figura 2.4: Grafico del polinomio $l_{43}(x)$.

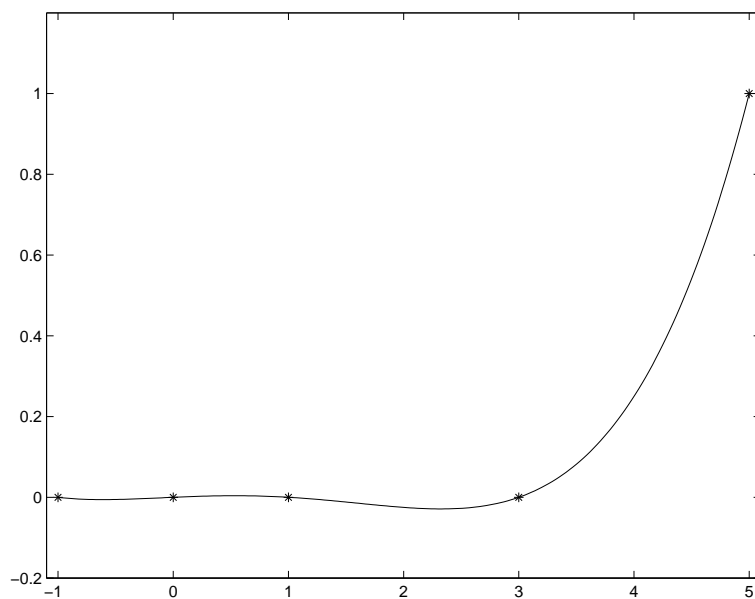


Figura 2.5: Grafico del polinomio $l_{44}(x)$.

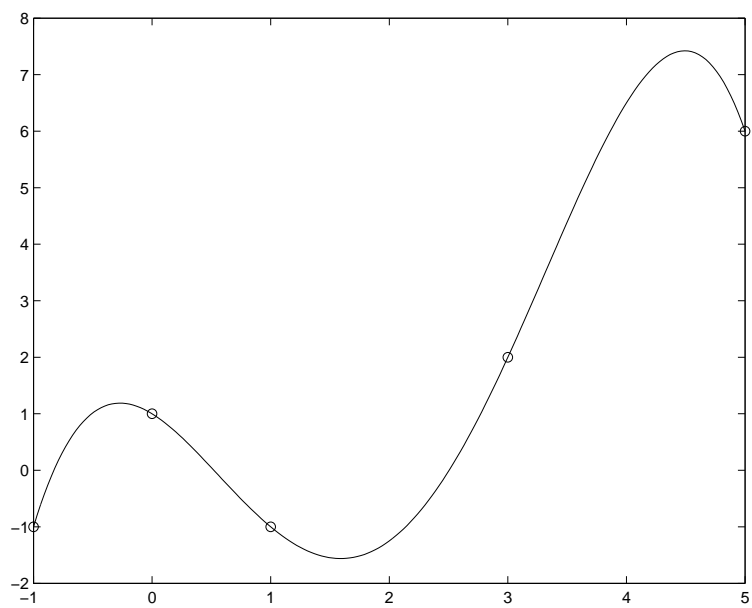


Figura 2.6: Grafico del polinomio interpolante di Lagrange $L_4(x)$.

2.2 Formule di Quadratura di Tipo Interpolatorio

Siano assegnati due valori a, b , con $a < b$, ed una funzione f integrabile sull'intervallo (a, b) . Il problema che ci poniamo è quello di costruire degli algoritmi numerici che ci permettano di valutare, con errore misurabile, il numero

$$I(f) = \int_a^b f(x) dx.$$

Diversi sono i motivi che possono portare alla richiesta di un algoritmo numerico per questi problemi.

Per esempio pur essendo in grado di calcolare una primitiva della funzione f , questa risulta così complicata da preferire un approccio di tipo numerico. Non è da trascurare poi il fatto che il coinvolgimento di funzioni, elementari e non, nella primitiva e la loro valutazione negli estremi a e b comporta comunque un'approssimazione dei risultati. Un'altra eventualità è che f sia nota solo in un numero finito di punti o comunque può essere valutata in ogni valore dell'argomento solo attraverso una routine. In questi casi l'approccio analitico non è neanche da prendere in considerazione.

Supponiamo dunque di conoscere la funzione $f(x)$ nei punti distinti x_0, x_1, \dots, x_n prefissati o scelti da noi, ed esaminiamo la costruzione di formule del tipo

$$\sum_{k=0}^n w_k f(x_k) \tag{2.9}$$

che approssimi realizzare $I(f)$.

Formule di tipo (2.9) si dicono *di quadratura*, i numeri reali x_0, x_1, \dots, x_n e w_0, \dots, w_n si chiamano rispettivamente *odi* e *pesi* della formula di quadratura.

Il modo più semplice ed immediato per costruire formule di tipo (2.9) è quello di sostituire la funzione integranda $f(x)$ con il polinomio di Lagrange $L_n(x)$ interpolante $f(x)$ nei nodi $x_i, i = 0, \dots, n$. Posto infatti

$$f(x) = L_n(x) + e(x)$$

dove $e(x)$ è la funzione errore, abbiamo:

$$\begin{aligned} \int_a^b f(x)dx &= \int_a^b [L_n(x) + e(x)]dx = \\ &= \int_a^b L_n(x)dx + \int_a^b e(x)dx = \\ &= \int_a^b \sum_{k=0}^n l_{nk}(x)f(x_k)dx + \int_a^b e(x)dx = \\ &= \sum_{k=0}^n \left(\int_a^b l_{nk}(x)dx \right) f(x_k) + \int_a^b e(x)dx. \end{aligned}$$

Ponendo

$$w_k = \int_a^b l_{nk}(x)dx \quad k = 0, 1, \dots, n \quad (2.10)$$

e

$$R_{n+1}(f) = \int_a^b e(x)dx \quad (2.11)$$

otteniamo

$$I(f) \simeq \sum_{k=0}^n w_k f(x_k)$$

con un errore stabilito dalla relazione (2.11). Le formule di quadratura con pesi definiti dalle formule (2.10) si dicono *interpolatorie*. La quantità $R_{n+1}(f)$ prende il nome di *Resto della formula di quadratura*. Un utile concetto per misurare il grado di accuratezza con cui una formula di quadratura, interpolatoria o meno, approssima un integrale è il seguente.

Definizione 2.2.1 *Una formula di quadratura ha grado di precisione q se fornisce il valore esatto dell'integrale quando la funzione integranda è un qualunque polinomio di grado al più q ed inoltre esiste un polinomio di grado $q + 1$ tale che l'errore è diverso da zero.*

È evidente da questa definizione che ogni formula di tipo interpolatorio con nodi x_0, x_1, \dots, x_n ha grado di precisione almeno n .

2.3 Formule di Newton-Cotes

Suddividiamo l'intervallo $[a, b]$ in n intervallini di ampiezza h , con

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

e definiamo i nodi

$$x_i = a + ih \quad i = 0, 1, \dots, n.$$

La formula di quadratura interpolatoria costruita su tali nodi, cioè

$$\int_a^b f(x)dx = \sum_{i=0}^n w_i f(x_i) + R_{n+1}(f)$$

è detta *Formula di Newton-Cotes*.

Una proprietà di cui godono i pesi delle formule di Newton-Cotes è la cosiddetta *proprietà di simmetria*. Infatti poichè i nodi sono a due a due simmetrici rispetto al punto medio c dell'intervallo $[a, b]$, cioè $c = (x_i + x_{n-i})/2$, per ogni i , tale proprietà si ripercuote sui pesi che infatti sono a due a due uguali, cioè $w_i = w_{n-i}$, per ogni i .

Vediamo ora due esempi di formule di Newton-Cotes.

2.3.1 Formula dei Trapezi

Siano $x_0 = a$, $x_1 = b$ e $h = b - a$.

$$\begin{aligned} T_2 &= w_0 f(x_0) + w_1 f(x_1) \\ w_0 &= \int_a^b l_{1,0}(x) dx = \int_a^b \frac{x - x_1}{x_0 - x_1} dx = \int_a^b \frac{x - b}{a - b} dx = \\ &= \frac{1}{a - b} [(x - b)^2]_{x=a}^{x=b} = \frac{h}{2}. \end{aligned}$$

Poichè i nodi scelti sono simmetrici rispetto al punto medio $c = (a + b)/2$ è

$$w_1 = w_0 = \frac{h}{2}.$$

Otteniamo dunque la formula

$$T_2 = \frac{h}{2} [f(a) + f(b)].$$

che viene detta *Formula dei Trapezi*. Per quanto concerne il resto abbiamo

$$R_2(f) = \frac{1}{2} \int_a^b (x-a)(x-b) f''(\xi_x) dx.$$

Prima di vedere come tale espressione può essere manipolata dimostriamo il seguente teorema che è noto come *teorema della media generalizzato*.

Teorema 2.3.1 *Siano $f, g : [a, b] \rightarrow \mathbb{R}$, funzioni continue con $g(x)$ a segno costante e $g(x) \neq 0$ per ogni $x \in]a, b[$. Allora*

$$\int_a^b f(x)g(x)dx = f(\xi) \int_a^b g(x)dx \quad \xi \in [a, b]. \quad \square$$

Poichè la funzione $(x-a)(x-b)$ è a segno costante segue:

$$R_2(f) = \frac{1}{2} f''(\eta) \int_a^b (x-a)(x-b) dx$$

posto $x = a + ht$ otteniamo

$$R_2(f) = \frac{1}{2} f''(\eta) h^3 \int_0^1 t(t-1) dt = -\frac{1}{12} h^3 f''(\eta).$$

2.3.2 Formula di Simpson

Siano $x_0 = a$, $x_2 = b$ mentre poniamo $x_1 = c$, punto medio dell'intervallo $[a, b]$. Allora

$$S_3 = w_0 f(a) + w_1 f(c) + w_2 f(b).$$

Posto

$$h = \frac{b-a}{2}$$

abbiamo

$$w_0 = \int_a^b l_{2,0}(x) dx = \int_a^b \frac{(x-c)(x-b)}{(a-c)(a-b)} dx.$$

Effettuando il cambio di variabile $x = c + ht$ è facile calcolare quest'ultimo integrale, infatti

$$x = a \Rightarrow a = c + ht \Rightarrow a - c = ht \Rightarrow -h = ht \Rightarrow t = -1$$

e

$$x = b \Rightarrow b = c + ht \Rightarrow b - c = ht \Rightarrow h = ht \Rightarrow t = 1.$$

Inoltre $a - c = -h$ e $a - b = -2h$ mentre

$$x - c = c + ht - c = ht, \quad x - b = c + ht - b = c - b + ht = -h + ht = h(t - 1),$$

ed il differenziale $dx = hdt$ cosicchè

$$\begin{aligned} w_0 &= \int_a^b \frac{(x - c)(x - b)}{(a - c)(a - b)} = \int_{-1}^1 \frac{hth(t - 1)}{(-h)(-2h)} hdt = \\ &= \frac{h}{2} \int_{-1}^1 (t^2 - t)dt = \frac{h}{2} \int_{-1}^1 t^2 dt = \frac{h}{2} \left[\frac{t^3}{3} \right]_{-1}^1 = \frac{h}{3}. \end{aligned}$$

Per la simmetria è anche

$$w_2 = w_0 = \frac{h}{3}$$

mentre possiamo calcolare w_1 senza ricorrere alla definizione. Infatti possiamo notare che la formula deve fornire il valore esatto dell'integrale quando la funzione è costante nell'intervallo $[a, b]$, quindi possiamo imporre che, prendendo $f(x) = 1$ in $[a, b]$, sia

$$\int_a^b dx = b - a = \frac{h}{3}(f(a) + f(b)) + w_1 f(c)$$

da cui segue

$$w_1 = b - a - \frac{2}{3}h = 2h - \frac{2}{3}h = \frac{4}{3}h.$$

Dunque

$$S_3 = \frac{h}{3} [f(a) + 4f(c) + f(b)].$$

Questa formula prende il nome di *Formula di Simpson*. Per quanto riguarda l'errore si può dimostrare, e qui ne omettiamo la prova, che vale la seguente relazione

$$R_3(f) = -h^5 \frac{f^{(4)}(\sigma)}{90} \quad \eta, \sigma \in (a, b),$$

che assicura che la formula ha grado di precisione 3.

2.4 Formule di Quadratura Composte

Come abbiamo già avuto modo di vedere le formule di quadratura interpolatorie vengono costruite approssimando su tutto l'intervallo di integrazione la funzione integranda con un unico polinomio, quello interpolante la funzione sui nodi scelti. Per formule convergenti la precisione desiderata si ottiene prendendo n sufficientemente grande. In tal modo comunque, per ogni fissato n , bisogna costruire la corrispondente formula di quadratura. Una strategia alternativa che ha il pregio di evitare la costruzione di una nuova formula di quadratura, e che spesso produce risultati più apprezzabili, è quella delle *formule composte*. L'idea delle formule composte è la seguente.

Scelta una formula di quadratura l'intervallo di integrazione (a, b) viene suddiviso in N intervallini di ampiezza h ,

$$h = \frac{b - a}{N} \quad (2.12)$$

sicchè

$$\int_a^b f(x)dx = \sum_{i=0}^{N-1} \int_{x_i}^{x_{i+1}} f(x)dx$$

dove i punti x_i sono:

$$x_i = a + ih \quad i = 0, \dots, N \quad (2.13)$$

quindi la formula di quadratura viene applicata ad ognuno degli intervallini $[x_i, x_{i+1}]$. Il grado di precisione della formula di quadratura composta coincide con il grado di precisione della formula da cui deriva. Descriviamo ora la *Formula dei Trapezi Composta*.

2.4.1 Formula dei Trapezi Composta

Per quanto visto in precedenza suddividiamo l'intervallo $[a, b]$ in N sottointervalli, ognuno di ampiezza data da h , come in (2.12), e con i nodi x_i definiti in (2.13). Appliciamo quindi in ciascuno degli N intervalli $[x_i, x_{i+1}]$ la formula

dei trapezi. Otteniamo

$$\begin{aligned} \int_a^b f(x)dx &= \sum_{i=0}^{N-1} \int_{x_i}^{x_{i+1}} f(x)dx = \\ &= \sum_{i=0}^{N-1} \left[\frac{h}{2} (f(x_i) + f(x_{i+1})) - \frac{1}{12} h^3 f''(\eta_i) \right] \quad \eta_i \in (x_i, x_{i+1}). \end{aligned}$$

Scrivendo diversamente la stessa espressione

$$\begin{aligned} \int_a^b f(x)dx &= \frac{h}{2} (f(x_0) + f(x_N)) + h \sum_{i=1}^{N-1} f(x_i) - \frac{1}{12} h^3 \sum_{i=0}^{N-1} f''(\eta_i) = \\ &= \frac{h}{2} (f(x_0) + f(x_N)) + h \sum_{i=1}^{N-1} f(x_i) - \frac{1}{12} h^3 N f''(\eta) \end{aligned}$$

dove $\eta \in (a, b)$. L'esistenza di tale punto η è garantito dal cosiddetto *Teorema della media nel discreto* applicato a $f''(x)$, che stabilisce che se $g(x)$ è una funzione continua in un intervallo $[a, b]$ e $\eta_i \in [a, b]$ $i = 1, N$, sono N punti distinti, allora esiste un punto $\eta \in (a, b)$ tale che

$$\sum_{i=1}^N g(\eta_i) = N g(\eta).$$

Dunque la formula dei trapezi composta è data da:

$$T_C(h) = \frac{h}{2} (f(x_0) + f(x_N)) + h \sum_{i=1}^{N-1} f(x_i)$$

con resto

$$R_T = -\frac{1}{12} h^3 N f''(\eta) = -\frac{1}{12} \frac{(b-a)^3}{N^3} N f''(\eta) = -\frac{1}{12} \frac{(b-a)^3}{N^2} f''(\eta).$$

Quest'ultima formula talvolta può essere utile per ottenere a priori una suddivisione dell'intervallo $[a, b]$ in un numero di intervalli che permetta un errore non superiore ad una prefissata tolleranza. Infatti

$$|R_T| \leq \frac{1}{12} \frac{(b-a)^3}{N^2} M, \quad M = \max_{x \in [a, b]} |f''(x)|.$$

Imponendo che $|R_T| \leq \varepsilon$, precisione prefissata, segue

$$N_\varepsilon \geq \sqrt{\frac{(b-a)^3 M}{12\varepsilon}}. \quad (2.14)$$

Tuttavia questo numero spesso risulta una stima eccessiva a causa della maggiorazione della derivata seconda tramite M .

Capitolo 3

Equazioni non Lineari

3.1 Introduzione

Le radici di un'equazione non lineare $f(x) = 0$ non possono, in generale, essere espresse esplicitamente e anche se ciò è possibile spesso l'espressione si presenta in forma talmente complicata da essere praticamente inutilizzabile. Di conseguenza per poter risolvere equazioni di questo tipo siamo obbligati ad utilizzare metodi numerici che sono, in generale, di tipo iterativo, cioè partendo da una (o più) approssimazioni della radice, producono una successione x_0, x_1, x_2, \dots , convergente alla radice. Per alcuni di questi metodi per ottenere la convergenza è sufficiente conoscere un intervallo $[a, b]$ che contiene la soluzione, altri metodi richiedono invece la conoscenza di una buona approssimazione iniziale. Talvolta è opportuno utilizzare in maniera combinata due metodi, uno del primo tipo e uno del secondo.

Prima di analizzare alcuni metodi per l'approssimazione delle radici dell'equazione $f(x) = 0$ diamo la definizione di molteplicità di una radice.

Definizione 3.1.1 *Sia $f \in \mathcal{C}^r([a, b])$ per un intero $r > 0$. Una radice α di $f(x)$ si dice di molteplicità r se*

$$\lim_{x \rightarrow \alpha} \frac{f(x)}{(x - \alpha)^r} = c \quad c \neq 0, c \neq \pm\infty. \quad (3.1)$$

3.1.1 Metodo di Bisezione

Sia $f : [a, b] \rightarrow \mathbb{R}$, $f \in \mathcal{C}([a, b])$, e sia $f(a)f(b) < 0$. Sotto tali ipotesi esiste sicuramente almeno un punto nell'intervallo $[a, b]$ in cui f si annulla. L'idea

alla base del *Metodo di Bisezione* consiste nel costruire una successione di intervalli $\{I_k\}_{k=0}^{\infty}$, con $I_0 = [a_0, b_0] \equiv [a, b]$, tali che:

1. $I_{k+1} \subset I_k$;
2. $\alpha \in I_k, \forall k \geq 0$;
3. l'ampiezza di I_k tende a zero per $k \rightarrow +\infty$.

La successione degli I_k viene costruita nel seguente modo. Innanzitutto si pone

$$I_0 = [a_0, b_0] = [a, b]$$

e si calcola il punto medio

$$c_1 = \frac{a_0 + b_0}{2}.$$

Se $f(c_1) = 0$ allora $\alpha = c_1$, altrimenti si pone:

$$I_1 = [a_1, b_1] \equiv \begin{cases} a_1 = a_0 & b_1 = c_1 & \text{se } f(a_0)f(c_1) < 0 \\ a_1 = c_1 & b_1 = b_0 & \text{se } f(a_0)f(c_1) > 0. \end{cases}$$

Ora, a partire da $I_1 = [a_1, b_1]$, si ripete la stessa procedura. In generale al passo k si calcola

$$c_{k+1} = \frac{a_k + b_k}{2}.$$

Se $f(c_{k+1}) = 0$ allora $\alpha = c_{k+1}$, altrimenti si pone:

$$I_{k+1} = [a_{k+1}, b_{k+1}] \equiv \begin{cases} a_{k+1} = a_k & b_{k+1} = c_k & \text{se } f(a_k)f(c_{k+1}) < 0 \\ a_{k+1} = c_{k+1} & b_{k+1} = b_k & \text{se } f(a_k)f(c_{k+1}) > 0. \end{cases}$$

La successione di intervalli I_k così costruita soddisfa automaticamente le condizioni 1) e 2). Per quanto riguarda la 3) abbiamo:

$$b_k - a_k = \frac{b_{k-1} - a_{k-1}}{2} = \frac{b_0 - a_0}{2^k}$$

e dunque l'ampiezza di I_k tende a zero quando $k \rightarrow +\infty$. La continuità di f garantisce poi la convergenza delle successioni $\{a_k\}_{k=0}^{\infty}$ e $\{b_k\}_{k=0}^{\infty}$ ad una radice di $f(x)$.

Infatti, per costruzione, $\{b_k\}_{k=0}^{\infty}$ è una successione monotona decrescente, limitata inferiormente e dunque convergente, mentre $\{a_k\}_{k=0}^{\infty}$ è una successione monotona crescente, limitata superiormente e dunque convergente. Allora da:

$$\begin{aligned} \lim_{k \rightarrow +\infty} (b_k - a_k) &= 0 \\ \lim_{k \rightarrow +\infty} a_k &= \xi_1 \quad \lim_{k \rightarrow +\infty} b_k = \xi_2 \end{aligned}$$

segue

$$\xi_1 = \xi_2 = \xi.$$

Ancora poichè $f(a_k)f(b_k) \leq 0$ per ogni k , si ha

$$\lim_{k \rightarrow +\infty} f(a_k)f(b_k) = f\left(\lim_{k \rightarrow +\infty} a_k\right) f\left(\lim_{k \rightarrow +\infty} b_k\right) \leq 0 \Leftrightarrow f^2(\xi) \leq 0$$

e quest'ultima ha senso solo se $f(\xi) = 0$. Quindi in definitiva $\alpha = \xi$.

Generalmente costruendo le successioni $\{a_k\}$ e $\{b_k\}$ accade che la condizione $f(c_k) = 0$, per un certo valore k , non si verifica mai a causa degli errori di arrotondamento. Quindi è necessario stabilire un opportuno criterio di stop che ci permetta di fermare la procedura quando riteniamo di aver raggiunto una precisione soddisfacente. Per esempio si può imporre:

$$b_k - a_k \leq \varepsilon \tag{3.2}$$

dove ε è una prefissata tolleranza. La (3.2) determina anche un limite per il numero di iterate infatti:

$$\frac{b_0 - a_0}{2^k} \leq \varepsilon \quad \Rightarrow \quad k > \log_2 \left(\frac{b_0 - a_0}{\varepsilon} \right).$$

Poichè $b_k - \alpha \leq b_k - a_k$, il criterio (3.2) garantisce che α è approssimata da c_{k+1} con un errore assoluto minore di ε . Se $0 \notin [a, b]$ si può usare come criterio di stop

$$\frac{b_k - a_k}{\min(|a_k|, |b_k|)} \leq \varepsilon \tag{3.3}$$

che garantisce che α è approssimata da c_{k+1} con un errore relativo minore di ε . Un ulteriore criterio di stop è fornito dal test:

$$|f(c_k)| \leq \varepsilon. \tag{3.4}$$

È comunque buona norma utilizzare due criteri di stop insieme, per esempio (3.2) e (3.4) oppure (3.3) e (3.4).

3.1.2 Il metodo della falsa posizione

Una variante del metodo delle bisezioni è appunto il metodo della falsa posizione. Partendo sempre da una funzione $f(x)$ continua in un intervallo $[a, b]$ tale che $f(a)f(b) < 0$, in questo caso si approssima la radice considerando l'intersezione della retta passante per i punti $(a, f(a))$ e $(b, f(b))$ con l'asse x . L'equazione della retta è

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

pertanto il punto c_1 , sua intersezione con l'asse x , è:

$$c_1 = b - f(b) \frac{b - a}{f(b) - f(a)}.$$

Si testa a questo punto l'appartenenza della radice α ad uno dei due intervalli $[a, c_1]$ e $[c_1, b]$ e si procede esattamente come nel caso del metodo delle bisezioni, ponendo

$$[a_1, b_1] \equiv \begin{cases} a_1 = a, & b_1 = c_1 & \text{se } f(a)f(c_1) < 0 \\ a_1 = c_1, & b_1 = b & \text{se } f(a)f(c_1) > 0. \end{cases}$$

Ad un generico passo k si calcola

$$c_k = b_{k-1} - f(b_{k-1}) \frac{b_{k-1} - a_{k-1}}{f(b_{k-1}) - f(a_{k-1})}$$

e si pone

$$[a_k, b_k] \equiv \begin{cases} a_k = a_{k-1} & b_k = c_k & \text{se } f(a_{k-1})f(c_k) < 0 \\ a_k = c_k & b_k = b_{k-1} & \text{se } f(a_{k-1})f(c_k) > 0. \end{cases}$$

Anche per questo metodo è possibile dimostrare la convergenza nella sola ipotesi di continuità della funzione $f(x)$.

3.2 Metodi di Iterazione Funzionale

Il metodo di bisezione può essere applicato ad una vastissima classe di funzioni, in quanto per la sua costruzione è richiesta solo la continuità della

funzione. Tuttavia ha lo svantaggio di risultare piuttosto lento, infatti ad ogni passo si guadagna in precisione una cifra binaria. Poichè $10^{-1} \simeq 2^{-3.3}$ per passare da t a $t + 1$ cifre decimali significative sono mediamente necessarie 3.3 iterate. Inoltre la velocità di convergenza è indipendente dalla funzione $f(x)$ poichè il metodo utilizza esclusivamente il segno assunto dalla funzione in determinati punti e non il suo valore. Il metodo delle bisezioni può essere comunque utilizzato con profitto per determinare delle buone approssimazioni della radice α che possono essere utilizzate dai metodi iterativi che ci apprestiamo a trattare.

Infatti richiedendo alla f supplementari condizioni di regolarità è possibile individuare una vasta classe di metodi che forniscono le stesse approssimazioni del metodo di bisezione utilizzando però un numero di iterate molto minore. In generale questi metodi sono del tipo:

$$x_{k+1} = g(x_k) \quad k = 0, 1, 2, \dots \quad (3.5)$$

dove x_0 è un'assegnato valore iniziale e forniscono un'approssimazione delle soluzioni dell'equazione

$$x = g(x). \quad (3.6)$$

Ogni punto α tale che $\alpha = g(\alpha)$ si dice *punto fisso* o *punto unito* di g .

Per poter applicare uno schema del tipo (3.5) all'equazione $f(x) = 0$, bisogna prima trasformare questa nella forma (3.6). Ad esempio se $[a, b]$ è l'intervallo di definizione di f ed $h(x)$ è una qualunque funzione tale che $h(x) \neq 0$, per ogni $x \in [a, b]$, si può porre:

$$g(x) = x - \frac{f(x)}{h(x)}. \quad (3.7)$$

Ovviamente ogni punto fisso di g è uno zero di f e viceversa.

Teorema 3.2.1 *Sia $g \in \mathcal{C}([a, b])$ e assumiamo che la successione $\{x_k\}$ generata da (3.5) sia contenuta in $[a, b]$. Allora se tale successione converge, il limite è il punto fisso di g .*

Dimostrazione.

$$\alpha = \lim_{k \rightarrow +\infty} x_{k+1} = \lim_{k \rightarrow +\infty} g(x_k) = g\left(\lim_{k \rightarrow +\infty} x_k\right) = g(\alpha). \quad \square$$

Teorema 3.2.2 *Sia α punto fisso di g e $g \in \mathcal{C}^1([\alpha - \rho, \alpha + \rho])$, per qualche $\rho > 0$. Scelto x_0 tale che*

$$|x_0 - \alpha| \leq \rho$$

per la successione $\{x_k\}_{k=0}^{\infty}$ generata da (3.5) si ha che se $|g'(x)| < 1$, per $|x - \alpha| \leq \rho$, allora $|x_k - \alpha| \leq \rho$, per ogni k , e la successione $\{x_k\}$ converge a α .

Dimostrazione. Sia

$$\lambda = \max_{|x-\alpha| \leq \rho} |g'(x)| < 1.$$

Proviamo per induzione che tutti gli elementi della successione $\{x_k\}$ sono contenuti nell'intervallo di centro α e ampiezza 2ρ . Per $k = 0$ si ha banalmente $x_0 \in [\alpha - \rho, \alpha + \rho]$. Assumiamo che $|x_k - \alpha| \leq \rho$ e dimostriamolo per $k + 1$.

$$|x_{k+1} - \alpha| = |g(x_k) - g(\alpha)| = |g'(\xi_k)| |x_k - \alpha|$$

dove $|\xi_k - \alpha| < |x_k - \alpha| \leq \rho$. Pertanto

$$|x_{k+1} - \alpha| \leq \lambda |x_k - \alpha| < |x_k - \alpha| \leq \rho.$$

Proviamo ora che:

$$\lim_{k \rightarrow +\infty} x_k = \alpha.$$

Da $|x_{k+1} - \alpha| \leq \lambda |x_k - \alpha|$ segue

$$|x_{k+1} - \alpha| \leq \lambda^{k+1} |x_0 - \alpha|.$$

Conseguentemente qualunque sia x_0 si ha:

$$\lim_{k \rightarrow +\infty} |x_k - \alpha| = 0 \Leftrightarrow \lim_{k \rightarrow +\infty} x_k = \alpha. \quad \square$$

Definizione 3.2.1 *Un metodo iterativo del tipo (3.5) si dice localmente convergente ad una soluzione α del problema $f(x) = 0$ se esiste un intervallo $[a, b]$ contenente α tale che, per ogni $x_0 \in [a, b]$, la successione generata da (3.5) converge a α .*

Una volta determinata una condizione sufficiente per la convergenza della successione $\{x_k\}$ ad un punto fisso di $g(x)$ si deve essere sicuri che tale punto

fisso è unico. Infatti se, oltre ad α esistesse anche $\beta \in [a, b]$ tale che $\beta = g(\beta)$, con $\alpha \neq \beta$, allora

$$|\alpha - \beta| = |g(\alpha) - g(\beta)| = |g'(\xi)| |\alpha - \beta|$$

con $\xi \in [a, b]$. Poichè $|g'(\xi)| < 1$ si ha:

$$|\alpha - \beta| < |\alpha - \beta|$$

e ciò è assurdo.

Come abbiamo già visto nel caso del metodo delle bisezioni anche per metodi di iterazione funzionale è necessario definire dei criteri di arresto per il calcolo delle iterazioni. I criteri di arresto per un processo iterativo basato sullo schema (3.5) sono i seguenti:

$$\begin{aligned} |x_{k+1} - x_k| &\leq \varepsilon \\ \frac{|x_{k+1} - x_k|}{\min(|x_{k+1}|, |x_k|)} &\leq \varepsilon \quad |x_{k+1}|, |x_k| \neq 0 \\ |f(x_k)| &\leq \varepsilon \end{aligned}$$

dove ε è una prefissata tolleranza dipendente dalla precisione con cui si vuole approssimare una soluzione.

Per confrontare differenti metodi iterativi che approssimano la stessa radice α di $f(x) = 0$, si può considerare la velocità con cui tali successioni convergono verso α . Lo studio della velocità di convergenza passa attraverso il concetto di ordine del metodo.

Definizione 3.2.2 Sia $\{x_k\}_{k=0}^{\infty}$ una successione convergente ad α e tale che $x_k \neq \alpha$, per ogni k . Se esiste un numero reale $p \geq 1$ tale che

$$\lim_{k \rightarrow +\infty} \frac{|x_{k+1} - \alpha|}{|x_k - \alpha|^p} = \gamma \quad \text{con} \quad \begin{cases} 0 < \gamma \leq 1 & \text{se } p = 1 \\ \gamma > 0 & \text{se } p > 1 \end{cases} \quad (3.8)$$

allora si dice che la successione ha ordine di convergenza p . La costante γ prende il nome di costante asintotica di convergenza.

In particolare se $p = 1$ e $0 < \gamma < 1$ allora la convergenza si dice *lineare*, se $p = 1$ e $\gamma = 1$ allora la convergenza si dice *sublineare*, mentre se $p > 1$ allora la convergenza si dice *superlineare*.

Osservazione. La relazione (3.8) implica che esiste una costante positiva β ($\beta \simeq \gamma$) tale che, per k sufficientemente grande:

$$|x_{k+1} - \alpha| \leq \beta |x_k - \alpha|^p \quad (3.9)$$

La (3.9) indica che la riduzione di errore assoluto ad ogni passo è tanto maggiore quanto più alto è l'ordine di convergenza e, a parità di ordine, quanto più piccola è la costante asintotica di convergenza. I metodi aventi convergenza lineare sono caratterizzati dal fatto che $g'(\alpha) \neq 0$ mentre il seguente teorema stabilisce una relazione tra la funzione $g(x)$ e l'ordine di convergenza di un metodo nel caso sia superlineare.

Teorema 3.2.3 (Enunciato). *Sia α punto fisso di $g \in \mathcal{C}^p([a, b])$, $p \geq 2$, intero. Scelto $x_0 \in [a, b]$ la successione $\{x_k\}$ generata dal metodo (3.5) converge a α con ordine p se e soltanto se*

$$g'(\alpha) = g''(\alpha) = \dots = g^{(p-1)}(\alpha) = 0, \quad g^{(p)}(\alpha) \neq 0. \quad \square$$

Definizione 3.2.3 *Un metodo iterativo convergente ad α si dice di ordine p (di ordine almeno p) se tutte le successioni ottenute al variare del punto iniziale in un opportuno intorno di α convergono con ordine di convergenza p (almeno p).*

3.2.1 Metodo di Newton-Raphson

Nell'ipotesi che f sia derivabile ed ammetta derivata prima continua allora un altro procedimento per l'approssimazione dello zero della funzione $f(x)$ è il *metodo di Newton-Raphson*, noto anche come *metodo delle tangenti*. Nella Figura 3.1 è riportata l'interpretazione geometrica di tale metodo. A partire dall'approssimazione x_0 si considera la retta tangente la funzione f passante per il punto P_0 di coordinate $(x_0, f(x_0))$. Si calcola l'ascissa x_1 del punto di intersezione tra tale retta tangente e l'asse delle x e si ripete il procedimento a partire dal punto P_1 di coordinate $(x_1, f(x_1))$.

È facile vedere che il metodo definisce il seguente processo iterativo:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad k = 0, 1, 2, \dots \quad (3.10)$$

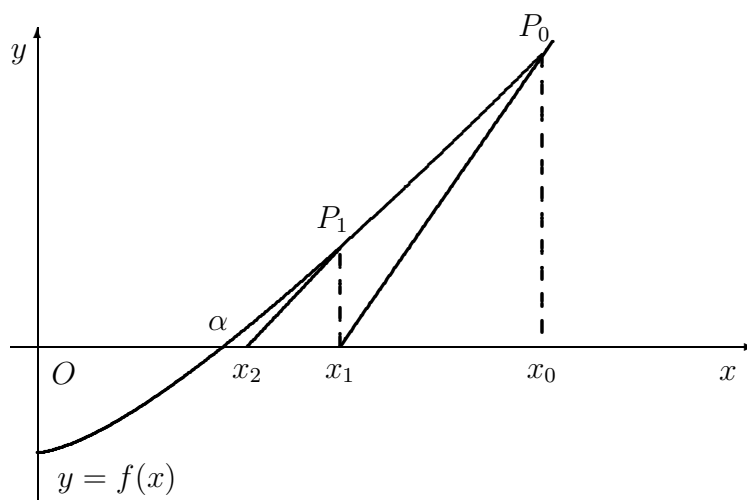


Figura 3.1: Metodo di Newton-Raphson

che equivale, scegliendo in (3.7) $h(x) = f'(x)$, al metodo di iterazione funzionale in cui la funzione $g(x)$ è

$$g(x) = x - \frac{f(x)}{f'(x)} \quad (3.11)$$

Per esaminare l'ordine di convergenza del metodo in questione consideriamo prima il α è radice semplice dell'equazione $f(x) = 0$. Supponiamo che per ipotesi $f'(x) \neq 0$, per $x \in [a, b]$, dove $[a, b]$ è un opportuno intervallo contenente α . Calcoliamo quindi la derivata prima di $g(x)$:

$$g'(x) = \frac{f(x)f''(x)}{[f'(x)]^2}. \quad (3.12)$$

Poichè α è semplice risulta $f'(\alpha) \neq 0$ e quindi:

$$g'(\alpha) = \frac{f(\alpha)f''(\alpha)}{[f'(\alpha)]^2} = 0$$

esiste quindi un intorno di α nel quale $|g'(x)| < 1$, per ogni x , e per il teorema (3.2.2) comunque si sceglie un punto iniziale appartenente a tale intorno il metodo di Newton-Raphson risulta convergente.

Se la radice α ha molteplicità $r > 1$ l'ordine di convergenza del metodo non

è più 2. Se x_0 è sufficientemente vicino ad α è $|g'(x)| < 1$ e quindi per il teorema 3.2.2 il metodo è ancora convergente ma l'ordine di convergenza è 1. *Osservazione.* Se applicando ripetutamente la formula di Newton-Raphson accade che la derivata prima della funzione $f(x)$ si mantiene sensibilmente costante allora si può porre

$$M = f'(x)$$

e applicare la formula

$$x_{k+1} = x_k - \frac{f(x_k)}{M} \quad (3.13)$$

anzichè la (3.10). La (3.13) definisce un metodo che viene detto *metodo di Newton semplificato* oppure *metodo della direzione costante* in quanto geometricamente equivale all'applicazione del metodo di Newton in cui anzichè prendere la retta tangente la curva f si considera la retta avente coefficiente angolare uguale a M .

3.2.2 I Metodi delle Secanti e delle Secanti a due punti

Il metodo delle secanti è definito dalla relazione

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - c)}{f(x_k) - f(c)}$$

dove $c \in [a, b]$. Il significato geometrico di tale metodo è il seguente: ad un generico passo k si considera la retta congiungente i punti di coordinate $(x_k, f(x_k))$ e $(c, f(c))$ e si pone x_{k+1} pari al punto di intersezione di tale retta con l'asse x . Dalla formula si evince che la funzione iteratrice del metodo è

$$g(x) = x - \frac{f(x)(x - c)}{f(x) - f(c)}.$$

In base alla teoria vista nei paragrafi precedenti il metodo ha ordine di convergenze 1 se $g'(\alpha) \neq 0$. Può avere ordine di convergenza almeno 1 se $g'(\alpha) = 0$. Tale eventualità si verifica se la tangente alla curva in α ha lo stesso coefficiente angolare della retta congiungente i punti $(\alpha, 0)$ e $(c, f(c))$.

Poichè il metodo delle secanti ha lo svantaggio di avere, solitamente, convergenza lineare mentre il metodo di Newton-Raphson, pur avendo convergenza quadratica, ha lo svantaggio di richiedere, ad ogni passo, due valutazioni di funzioni: $f(x_k)$ ed $f'(x_k)$, quindi se il costo computazionale di $f'(x_k)$ è molto

più elevato rispetto a quello di $f(x_k)$ può essere più conveniente l'uso di metodi che necessitano solo del calcolo del valore della funzione $f(x)$. Un metodo di questo tipo è **il metodo delle secanti a due punti**, che consiste nel considerare, alla k -esima iterata, la retta passante per i punti di coordinate $(x_{k-1}, f(x_{k-1}))$ e $(x_k, f(x_k))$ e definire x_{k+1} come il punto d'intersezione di tale retta con l'asse x . Ne risulta il seguente metodo:

$$x_{k+1} = x_k - f(x_k) \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}. \quad (3.14)$$

Poichè il metodo delle secanti a due punti non rientra tra quelli di iterazione funzionale, per ricavare l'ordine del metodo non si possono applicare i teoremi enunciati in precedenza, pertanto si deve procedere in un modo più che sperimentale. Infatti in questo caso si può vedere che l'ordine non è neanche un numero intero ma irrazionale:

$$p = \frac{1 + \sqrt{5}}{2}.$$

Il metodo delle secanti a due punti ha comunque una convergenza di tipo superlineare, come il metodo di Newton-Raphson, ma ha il vantaggio di richiedere solo una valutazione funzionale ad ogni passo (tranne al primo passo) e di non richiedere il calcolo della derivata prima.

Capitolo 4

II MATLAB

4.1 Introduzione al MATLAB

Il Matlab (acronimo delle parole inglesi *MATrix LABoratory*) è un software basato sulla manipolazione di matrici molto utilizzato nel campo della ricerca scientifica, non solo matematica, a causa della sua grande portabilità (infatti è disponibile sia per grandi workstation che per comuni personal computers), unita ad una notevole facilità d'uso e alle potenzialità di calcolo. Inoltre l'uso del Matlab è reso facile dalla presenza di un manuale dei comandi in linea, che può essere invocato tramite il comando `help`, e dalla presenza del comando `demo` che presenta numerosi e significativi esempi di applicazioni di tutte le funzioni Matlab. Nelle seguenti pagine faremo riferimento alla versione Matlab denotata con 5.1.

Per lanciare il Matlab in ambiente UNIX o Linux è sufficiente digitare il comando `matlab`, mentre in ambiente Windows o Mac si deve effettuare un doppio click sull'icona del programma. A questo punto compare il prompt del software

```
>>
```

mentre per uscire si deve digitare `exit` oppure `quit`.

Il comando `help` come già detto fornisce tutte le informazioni relative ad un particolare comando oppure una lista di tutti gli argomenti per i quali è presente un aiuto. La sintassi del comando è semplice:

```
>> help
```

oppure

```
>> help comando
```

Per esempio per sapere l'uso del comando `load`, che descriveremo in dettaglio nel seguito, è sufficiente scrivere

```
>> help load
```

Anche il comando `demo` ha una sintassi molto semplice:

```
>> demo
```

a questo punto compariranno sullo schermo alcuni menu e basterà scegliere, tramite il mouse, l'argomento del quale si vuole vedere una dimostrazione. Il Matlab può essere considerato un interprete le cui istruzioni sono del tipo:

```
variabile = espressione
```

oppure

```
variabile
```

In quest'ultimo caso, quando cioè un'istruzione è costituita solo dal nome di una variabile viene interpretata come la visualizzazione del valore di tale variabile. Vediamo i seguenti esempi.

```
>> b=5;
>> b
ans =
     5
>>
```

```
>> b=5
b =
     5
>>
```

Nel primo caso il valore di output di `b` è stato attribuito alla variabile di comodo `ans` (abbreviazione per la parola inglese *answer*). Questo modo di procedere viene utilizzato anche quando si chiede di valutare un'espressione di tipo numerico senza l'ausilio di variabili.

```
>> 3+4
ans =
     7
>>
```

Ogni espressione introdotta viene interpretata e calcolata. Ogni istruzione può essere scritta anche su due righe purchè prima di andare a capo vengano scritti 3 punti "...". Più espressioni possono essere scritte sulla stessa riga purchè siano separate da una virgola o dal punto e virgola. Se una riga di un file Matlab inizia con % allora tale riga viene considerata come un commento. Il Matlab fa distinzione tra lettere minuscole e maiuscole, quindi se abbiamo definito una variabile A e facciamo riferimento a questa scrivendo a essa non viene riconosciuta.

4.2 Assegnazione di matrici

La prima cosa da imparare del Matlab è come manipolare le matrici che costituiscono la struttura fondamentale dei dati. Una matrice è una tabella di elementi caratterizzata da due dimensioni: il numero delle righe e quello delle colonne. I vettori sono matrici aventi una delle dimensioni uguali a 1. Infatti esistono due tipi di vettori: i *vettori riga* aventi dimensione $1 \times n$, e i *vettori colonna* aventi dimensione $n \times 1$. I dati scalari sono matrici di dimensione 1×1 . Le matrici possono essere introdotte in diversi modi, per esempio possono essere assegnate esplicitamente, o caricate da file di dati esterni, o assegnate utilizzando generate da funzioni predefinite.

Per esempio l'istruzione

```
>> A = [1 2 3; 4 5 6; 7 8 9];
```

assegna alla variabile A una matrice di tre righe e tre colonne. Gli elementi di una riga della matrice possono essere separate da virgole o dallo spazio, mentre le diverse righe sono separate da un punto e virgola. Se alla fine dell'assegnazione viene messo il punto e virgola allora la matrice non viene visualizzata sullo schermo. In generale se vogliamo assegnare ad A una matrice ad m righe ed n colonne la sintassi è la seguente:

```
>> A = [riga 1; riga 2; ...; riga m];
```

Per assegnare ad una variabile x un vettore riga si ha

```
>> x = [3 -4 5];
```

gli elementi possono anche essere separati da una virgola

```
>> x = [3,-4,5];
```

Per assegnare invece ad una variabile un vettore colonna basta separare gli elementi con un punto e virgola:

```
>> y = [1;-3;6];
```

La stessa matrice A dell'esempio visto in precedenza può essere assegnata anche a blocchi:

```
>> A = [ 1 2 3; 4 5 6];
```

```
>> b = [ 7 8 9];
```

```
>> A = [ A; b];
```

mentre in modo analogo si può anche aggiungere una colonna:

```
>> A = [-1 2 3; 0 5 6; -5 4 3];
```

```
>> x = [-7; 0; 9];
```

```
>> A = [ A, x];
```

Descriviamo ora alcune funzioni predefinite che forniscono in output determinate matrici.

```
>> A=rand(m,n)
```

costruisce una matrice $m \times n$ di elementi casuali uniformemente distribuiti tra 0 e 1;

```
>> A=zeros(m,n)
```

costruisce una matrice $m \times n$ di elementi nulli;

```
>> A=ones(m,n)
```

costruisce una matrice $m \times n$ di elementi tutti uguali a 1;

```
>> A=eye(m,n)
```

costruisce una matrice $m \times n$ i cui elementi sono uguali a 1 sulla diagonale principale e 0 altrove; Per le funzioni appena viste se uno dei due parametri è omesso allora la matrice costruita viene considerata quadrata.

Il dimensionamento delle matrici è automatico. Per esempio se si pone

```
>> B = [1 2 3; 4 5 6];
```

e successivamente

```
>> B = [1 0; 0 7];
```

il programma riconosce che la matrice B ha cambiato le dimensioni da 2×3 a 2×2 . L'elemento della riga i e della colonna j viene denotato con $A(i,j)$. Quindi $A(4,2)$ indica l'elemento che si trova nella quarta riga e in colonna 2. Per fare riferimento a elementi di vettori è sufficiente utilizzare un solo indice. Se si fa riferimento a un elemento di una matrice di dimensione $m \times n$ che non esiste allora il Matlab segnala l'errore con il seguente messaggio:

Index exceeds matrix dimension

Se C è una matrice non ancora inizializzata allora l'istruzione

```
>> C(3,2)= 1
```

fornisce come risposta

```
C =
    0    0
    0    0
    0    1
```

cioè il programma assume come dimensioni per C dei numeri sufficientemente grandi affinché l'assegnazione abbia senso. Se ora si pone

```
>> C(1,3)= 2
```

si ha:

```
C =
    0    0    2
    0    0    0
    0    1    0
```

In Matlab gli indici devono essere strettamente positivi, eventuali indici frazionari sono approssimati al più grande intero minore o uguale mentre se si richiede un elemento di indice negativo oppure uguale a zero si ha sullo schermo il seguente messaggio di errore:

Index into matrix is negative or zero

4.2.1 Sottomatrici e notazione :

Vediamo ora alcuni esempi che illustrano l'uso di : per vettori e matrici. Le istruzioni

```
>> x=[1:5];
```

e

```
>> x=1:5;
```

sono equivalenti all'assegnazione diretta del vettore x :

```
>> x=[1 2 3 4 5];
```

Ciò vale anche per vettori di elementi reali. Infatti l'istruzione

```
>> x=[0.2:0.2:1.2];
```

equivale a scrivere

```
>> x=[0.2 0.4 0.6 0.8 1.0 1.2];
```

Inoltre è possibile anche l'uso di incrementi negativi:

```
>> x=[5:-1:1];
```

è equivalente a

```
>> x=[5 4 3 2 1];
```

L'istruzione

```
>> x=x(n:-1:1);
```

inverte gli elementi del vettore x di dimensione n . La notazione : può essere anche applicata a matrici. Infatti se A è una matrice abbiamo:

```
>> y=A(1:4,3);
```

assegna al vettore colonna y i primi 4 elementi della terza colonna della matrice A ;

```
>> y=A(4,2:5);
```

assegna al vettore riga y gli elementi della quarta riga di A compresi tra il secondo e il quinto;

```
>> y=A(:,3);
```

assegna al vettore colonna y la terza colonna di A ;

```
>> y=A(2,:);
```

assegna al vettore riga y la seconda riga di A ;

```
>> B=A(1:4,:);
```

assegna alla matrice B le prime 4 righe di A ;

```
>> B=A(:,2:6);
```

assegna alla matrice B le colonne A il cui indice è compreso tra 2 e 6;

```
>> B=A(:,[2 4]);
```

assegna alla matrice B la seconda e la quarta colonna di A ;

```
>> A(:,[2 4 5])=B(:,1:3);
```

sostituisce alle colonne 2, 4 e 5 della matrice A le prime 3 colonne della matrice B .

4.3 Operazioni su matrici e vettori

In Matlab sono definite le seguenti operazioni su matrici e vettori:

| | |
|----|---|
| + | addizione |
| - | sottrazione |
| * | moltiplicazione |
| ^ | elevazione a potenza |
| ' | trasposto |
| / | divisione |
| () | specificano l'ordine di valutazione delle espressioni |

Ovviamente queste operazioni possono essere applicate anche a scalari. Se le dimensioni delle matrici coinvolte non sono compatibili allora viene segnalato un errore eccetto nel caso di operazione tra uno scalare e una matrice. Per esempio se A è una matrice di qualsiasi dimensione allora l'istruzione

```
>> C = A+2;
```

assegna alla matrice C gli elementi di A incrementati di 2.

Nel caso del prodotto tra matrici è necessario prestare molta attenzione alle dimensioni delle matrici. Infatti ricordiamo che se $A \in \mathbb{R}^{m \times p}$ e $B \in \mathbb{R}^{p \times n}$ allora la matrice

$$C = A \cdot B, \quad C \in \mathbb{R}^{m \times n}$$

si definisce nel seguente modo:

$$c_{ij} = \sum_{k=1}^p a_{ik} b_{kj}, \quad i = 1, \dots, m \quad j = 1, \dots, n$$

ed è la matrice che viene calcolata scrivendo l'istruzione

```
>> C = A*B
```

In caso contrario se scriviamo

```
>> C = B*A
```

allora il programma segnala errore a meno che non sia $m = n$.

È importante notare che le operazioni $*$, \wedge , e $/$ operano elemento per elemento se sono precedute da un punto:

$$C=A.*B \quad \Rightarrow \quad c_{ij} = a_{ij}b_{ij}$$

$$C=A./B \quad \Rightarrow \quad c_{ij} = a_{ij}/b_{ij}$$

$$C=A.^B \quad \Rightarrow \quad c_{ij} = a_{ij}^{b_{ij}}$$

4.3.1 Costanti predefinite

Come la maggior parte dei linguaggi di programmazione il MatLab ha alcune costanti predefinite cioè delle variabili che hanno un proprio valore senza che esso venga esplicitamente assegnato:

| | |
|-----|--|
| eps | precisione di macchina ($\simeq 2.2 \cdot 10^{-16}$) |
| pi | π (cioè 3.14159265358979) |
| i | unita' immaginaria ($\sqrt{-1}$) |
| j | unita' immaginaria ($\sqrt{-1}$) |

`realmax` il piu' grande numero floating point ($1.7976e + 308$)
`realmin` il piu' piccolo numero floating point ($2.2251e - 308$)
`inf` infinito (∞)
`NaN` Not a Number.

La costante `inf` è ottenuta come risultato di una divisione per zero oppure il calcolo del logaritmo di zero o se il risultato è un overflow (per esempio $2*\text{realmax}$). La costante `NaN` invece è ottenuta come risultato di operazioni matematicamente non definite come $0/0$ oppure $\infty - \infty$.

Come accade per la maggior parte dei linguaggi di programmazione anche in Matlab è possibile definire variabili il cui nome è una costante predefinita, quindi per esempio è possibile usare la variabile `i` come indice intero.

4.3.2 Operatori relazionali e logici

I seguenti sono gli operatori relazionali

| | |
|--------------------|-------------------|
| <code><</code> | minore |
| <code>></code> | maggiore |
| <code><=</code> | minore o uguale |
| <code>>=</code> | maggiore o uguale |
| <code>==</code> | uguale |
| <code>~=</code> | diverso |

Una relazione di tipo logico assume valore 0 o 1 a seconda del fatto se essa sia rispettivamente falsa o vera. Per esempio scrivendo

```
>> 3<5
```

otterremo

```
>> 3<5
ans =
    1
```

oppure scrivendo

```
>> 1>5
```

la risposta è

```
>> 1>5
ans =
    0
```

Quando un operatore relazionale è applicato a matrici di dimensioni uguali si ottiene come risultato una matrice i cui elementi sono 1 oppure 0. Vediamo il seguente esempio:

```
>> A=[2 1 ; 0 3];
>> B=[2 -1 ; -2 3];
>> A==B
ans =
    1 0
    0 1

>> A>B
ans =
    0 1
    1 0

>> A>=B
ans =
    1 1
    1 1
```

Gli operatori logici che il Matlab consente di utilizzare sono i seguenti:

| | |
|---|-----|
| & | AND |
| | OR |
| ~ | NOT |

4.4 Funzioni predefinite

4.4.1 Funzioni scalari

Alcune funzioni Matlab predefinite operano essenzialmente su scalari ma quando vengono applicate a matrici (e vettori) vengono interpretate come se fossero applicate a ciascun elemento della matrice (o componente del vettore).

| | |
|--------------------|-------------------------|
| <code>sin</code> | seno |
| <code>cos</code> | coseno |
| <code>tan</code> | tangente |
| <code>asin</code> | arcoseno |
| <code>acos</code> | arcocoseno |
| <code>atan</code> | arcotangente |
| <code>sinh</code> | seno iperbolico |
| <code>cosh</code> | coseno iperbolico |
| <code>tanh</code> | tangente iperbolica |
| <code>asinh</code> | arcoseno iperbolico |
| <code>acosh</code> | arcocoseno iperbolico |
| <code>atanh</code> | arcotangente iperbolica |
| <code>exp</code> | esponenziale |
| <code>log</code> | logaritmo naturale |
| <code>log10</code> | logaritmo in base 10 |
| <code>sqrt</code> | radice quadrata |
| <code>abs</code> | valore assoluto |
| <code>rem</code> | resto della divisione |
| <code>sign</code> | segno |
| <code>round</code> | arrotondamento |
| <code>floor</code> | parte intera inferiore |
| <code>ceil</code> | parte intera superiore |

Tra queste funzioni appena nominate le ultime tre meritano un piccolo approfondimento, nella seguente tabella sono riportati i valori di tali funzioni per differenti numeri reali:

| x | $\text{round}(x)$ | $\text{floor}(x)$ | $\text{ceil}(x)$ |
|------|-------------------|-------------------|------------------|
| 3.7 | 4 | 3 | 4 |
| 3.1 | 3 | 3 | 4 |
| -4.7 | -5 | -5 | -4 |
| -4.3 | -4 | -5 | -4 |

Osserviamo come $\text{floor}(x)$ è sempre più piccolo di x mentre $\text{ceil}(x)$ è maggiore di x .

4.4.2 Funzioni vettoriali

Altre funzioni Matlab operano essenzialmente su vettori (riga o colonna), ma possono agire anche su matrici in modo tale da produrre un vettore riga

contenente i risultati della loro applicazione a ciascuna colonna. Per ottenere il risultato della loro azione sulle righe basta applicare la stessa funzione alla matrice trasposta A' . Alcune di queste funzioni sono:

| | |
|---------------------|---------------------------------------|
| <code>max</code> | massimo elemento di un vettore |
| <code>min</code> | minimo elemento di un vettore |
| <code>sum</code> | somma degli elementi di un vettore |
| <code>prod</code> | prodotto degli elementi di un vettore |
| <code>sort</code> | ordinamento di un vettore |
| <code>length</code> | numero di elementi di un vettore |

Per esempio per determinare il massimo elemento di una matrice A si deve scrivere `max(max(A))` piuttosto che `max(A)`. Le funzioni `max` e `min` possono fornire in uscita anche l'indice della componente massima (o minima) del vettore. La sintassi in questo caso è la seguente:

```
>> [massimo,k]=max(x);
>> [minimo,k]=min(x);
```

4.4.3 Funzioni di matrici

Le più utili funzioni di matrici sono le seguenti:

| | |
|-------------------|---|
| <code>eig</code> | autovalori e autovettori |
| <code>inv</code> | inversa |
| <code>det</code> | determinante |
| <code>size</code> | dimensioni |
| <code>norm</code> | norma |
| <code>cond</code> | numero di condizione in norma 2 |
| <code>rank</code> | rango |
| <code>tril</code> | parte triangolare inferiore |
| <code>triu</code> | parte triangolare superiore |
| <code>diag</code> | fornisce in output un vettore colonna dove e' memorizzata la parte diagonale di una matrice. Se la funzione e' applicata invece ad un vettore allora in uscita avremo una matrice diagonale i cui elementi principali sono quelli del vettore di input. |

Le funzioni Matlab possono avere uno o più argomenti di output. Per esempio `y = eig(A)`, o semplicemente `eig(A)`, produce un vettore colonna contenente gli autovalori di A mentre

```
>> [U,D] = eig(A);
```

produce una matrice U le cui colonne sono gli autovettori di A e una matrice diagonale D con gli autovalori di A sulla sua diagonale. Anche la funzione `size` ha due parametri di output:

```
>> [m,n] = size(A);
```

assegna a `m` ed `n` rispettivamente il numero di righe e di colonne della matrice A .

La funzione `norm` se viene applicata ad una matrice calcola la norma 2 della stessa matrice. È tuttavia possibile specificare anche altre norme. Per esempio

```
>> norm(A,'inf');
```

calcola la norma infinito di A mentre

```
>> norm(A,1);
```

calcola la norma 1 di A .

4.5 Le istruzioni `for`, `while`, `if` e `switch`

Il Matlab è dotato delle principali istruzioni che servono a renderlo un linguaggio strutturato. La più importante istruzione per la ripetizione in sequenza delle istruzioni è il `for`, che ha la seguente sintassi:

```
for var=val_0:step:val_1
    lista istruzioni
end
```

La variabile `var` assume come valore iniziale `val_0`, viene eseguita la lista di istruzioni che segue, poi è incrementata del valore `step`, vengono rieseguite le istruzioni che seguono e così via, finché il suo valore non supera `val_1`. Il valore dello `step` può essere negativo, nel qual caso il valore di `val_0` deve essere logicamente superiore a `val_1`.

La sintassi per l'istruzione `while` è la seguente.

```
while espressione logica
    istruzioni
end
```

Le *istruzioni* vengono eseguite fintantochè l'*espressione logica* rimane vera. La sintassi completa dell'istruzione `if` è la seguente:

```
if espressione logica
    istruzioni
elseif espressione logica
    istruzioni
else
    istruzioni
end
```

I rami `elseif` possono essere più di uno come anche essere assenti. Anche il ramo `else` può mancare. Vediamo ora alcuni esempi di come le istruzioni appena descritte possono essere applicate.

Se all'interno delle istruzioni che seguono il `for` o il `while` si verifica la necessità di interrompere il ciclo delle istruzioni allora ciò può essere fatto utilizzando l'istruzione `break`.

Ultima istruzione di questo tipo (e presente solo nell'ultima versione del programma) è l'istruzione `switch` che ha lo stesso ruolo e quasi la stessa sintassi dell'omonima istruzione in linguaggio C:

```
switch variabile
    case valore_0
        istruzioni
    case valore_1
        istruzioni
    case valore_2
        istruzioni
    otherwise
        istruzioni
end
```

che, in funzione del valore assunto dalla variabile, esegue o meno una serie di istruzioni. In particolare se nessuno dei valori previsti è assunto dalla variabile allora viene previsto un caso alternativo (`otherwise`) che li contempla tutti. Vediamo il seguente esempio:

```
switch rem(n,2)
    case 0
        disp('n e'' un numero pari')
```

```
    case 1
        disp('n e'' un numero dispari')
    otherwise
        disp('Caso impossibile')
end
```

4.6 Istruzioni per gestire il Workspace

Il comando

```
>> who
```

elenca le variabili presenti nell'area di lavoro, mentre il comando

```
>> whos
```

elenca, oltre al nome delle variabili, anche il tipo e l'occupazione di memoria. Una variabile può essere cancellata da tale area con il comando

```
>> clear nome variabile
```

mentre il comando

```
>> clear
```

cancella tutte le variabili presenti nell'area di lavoro. Il comando \hat{C} inoltre interrompe l'esecuzione di un file Matlab. L'istruzione

```
>> save
```

salva il contenuto dell'area di lavoro (cioè le variabili e il loro valore) nel file binario `matlab.mat`. Se invece si scrive

```
>> save nomefile
```

allora tutta l'area di lavoro viene salvata nel file `nomefile.mat`. Se invece si vogliono salvare solo alcune variabili e non tutta l'area di lavoro allora è possibile farlo specificando, oltre al nome del file, anche l'elenco di tali variabili. Per esempio

```
>> save nomefile A B x
```

salva nel file `nomefile.mat` solo il contenuto delle variabili `A`, `B` e `x`. Scrivendo

```
>> save nomefile A B x -ascii
```

allora il file `nomefile.mat` non ha il formato binario ma `ascii`, e questo è utile se si vuole verificare il contenuto del file.

Per ripristinare il contenuto dell'area di lavoro dal file `matlab.mat` il comando è

```
>> load
```

mentre è possibile anche in questo caso specificare il file da caricare. Facendo riferimento all'esempio del comando `save` allora scrivendo

```
>> load nomefile
```

ripristina le variabili e il loro valore che erano stati memorizzati nel file `nomefile.mat`.

4.7 M-files

Il Matlab può eseguire una sequenza di istruzioni memorizzate in un file. Questi file prendono il nome di *M-files* perchè la loro estensione è `.m`. Ci sono due tipi di M-files: gli *script files* e i *function files*.

Script files

Uno script file consiste in una sequenza di normali istruzioni Matlab. Se il file ha come nome `prova.m` allora basterà eseguire il comando

```
>> prova
```

per far sì che le istruzioni vengano eseguite. Le variabili di uno script file sono di tipo globale, per questo sono spesso utilizzati anche per assegnare dati a matrici di grosse dimensioni, in modo tale da evitare errori di input. Per esempio se in file `assegna.m` vi è la seguente assegnazione:

```
A=[0 -2 13 4; -5 3 10 -8; 10 -12 14 17; -1 4 5 6];
```

allora l'istruzione `assegna` servirà per definire la matrice `A`.

Function files

Permettono all'utente di definire funzioni che non sono standard. Le variabili definite nelle funzioni sono locali, anche se esistono delle istruzioni che permettono di operare su variabili globali. Vediamo il seguente esempio.

```
function a = randint(m,n)
% randint(m,n) Fornisce in output una matrice
% di dimensioni m×n di numeri casuali
% interi compresi tra 0 e 9.
a = floor(10*rand(m,n));
```

Tale funzione va scritta in un file chiamato `randint.m` (corrispondente al nome della funzione). La prima linea definisce il nome della funzione e gli argomenti di input e di output. Questa linea serve a distinguere i function files dagli script files. Quindi l'istruzione Matlab

```
>> c=randint(5,4);
```

assegna a c una matrice di elementi interi casuali di 5 righe e 4 colonne. Le variabili m , n e a sono interne alla funzione quindi il loro valore non modifica il valore di eventuali variabili globali aventi lo stesso nome. Se la funzione ammette più di un parametro di output questa allora la prima riga del function file deve essere modificata nel seguente modo:

```
function [var_0,var_1,var_2]= nomefunzione(inp_0,inp_1)
```

Vediamo ora di scrivere una funzione Matlab per calcolare il valore assunto da un polinomio per un certo valore x . Ricordiamo che assegnato un polinomio di grado n

$$p(x) = a_1 + a_2x + a_3x^2 + \dots + a_nx^{n-1} + a_{n+1}x^n$$

la seguente *Regola di Horner* permette di valutare il polinomio minimizzando il numero di operazioni necessarie. Tale regola consiste nel riscrivere lo stesso polinomio in questo modo:

$$p(x) = a_1 + x(a_2 + x(a_3 + \dots + x(a_n + a_{n+1}x) \dots))).$$

In questo modo il numero di moltiplicazioni necessarie passa all'incirca da $O(n^2/2)$ a $O(n)$. Vediamo ora la funzione Matlab che implementa tale regola.

```

function y= horner(a,x,n)
% horner(a,x,n) Fornisce in output il valore
% di un polinomio di grado n nel punto x
% a vettore di n+1 elementi contenente i
% coefficienti del polinomio
% x punto dove si vuol calcolare il polinomio
% n grado del polinomio
p=0;
for i=n+1:-1:1
    p=p*x+a(i);
end
y=p;

```

Per interrompere l'esecuzione di una funzione e tornare al programma chiamante si usa l'istruzione

```
return
```

4.8 Messaggi di errore, Istruzioni di Input

Stringhe di testo possono essere visualizzate sullo schermo mediante l'istruzione `disp`. Per esempio

```
disp('Messaggio sul video')
```

Se la stringa tra parentesi contiene un'apice allora deve essere raddoppiato. La stessa istruzione può essere utilizzata anche per visualizzare il valore di una variabile, è sufficiente scrivere, al posto della stringa, e senza apici, il nome della variabile.

I messaggi di errore possono essere visualizzati con l'istruzione `error`. Consideriamo il seguente esempio:

```

if a==0
    error('Divisione per zero')
else
    b=b/a;
end

```

l'istruzione `error` causa l'interruzione nell'esecuzione del file.

In un M-file è possibile introdurre dati di input in maniera interattiva mediante l'istruzione `input`. Per esempio quando l'istruzione

```
iter = input('Inserire il numero di iterate ')
```

viene eseguita allora il messaggio è visualizzato sullo schermo e il programma rimane in attesa che l'utente digiti un valore da tastiera e tale valore, di qualsiasi tipo esso sia, viene assegnato alla variabile `iter`.

Vediamo ora un modo per poter memorizzare in un file di ascii l'output di un M-file oppure di una sequenza di istruzioni Matlab. Infatti

```
>> diary nomefile
>> istruzioni
>> diary off
```

serve a memorizzare nel file *nomefile* tutte le istruzioni e l'output che è stato prodotto dopo la prima chiamata della funzione e prima della seconda.

4.8.1 Formato di output

In Matlab tutte le elaborazioni vengono effettuate in doppia precisione. Il formato con cui l'output compare sul video può però essere controllato mediante i seguenti comandi.

```
format short
```

È il formato utilizzato per default dal programma ed è di tipo fixed point con 4 cifre decimali;

```
format long
```

Tale formato è di tipo fixed point con 14 cifre decimali;

```
format short e
```

Tale formato è la notazione scientifica (esponenziale) con 4 cifre decimali;

```
format long e
```

Tale formato è la notazione scientifica (esponenziale) con 15 cifre decimali. Vediamo per esempio come i numeri $4/3$ e $1.2345e - 6$ sono rappresentati nei formati che abbiamo appena descritto e negli altri disponibili:

```
format short          1.3333
format short e       1.3333e+000
format short g       1.3333
format long          1.33333333333333
format long e       1.333333333333333e+000
format long g       1.333333333333333
format rat           4/3
```

```
format short          0.0000
format short e       1.2345e-006
format short g       1.2345e-006
format long          0.00000123450000
format long e       1.234500000000000e-006
format long g       1.2345e-006
format rat           1/810045
```

Oltre ai formati appena visti il comando

```
format compact
```

serve a sopprimere le righe vuote e gli spazi dell'output scrivendo sullo schermo il maggior numero di informazione possibile, in modo appunto compatto.

4.9 La grafica con il Matlab

Il Matlab dispone di numerose istruzioni per grafici bidimensionali e tridimensionali e anche di alcune funzioni per la creazione di animazioni. Il comando `plot` serve a disegnare curve nel piano xy . Infatti se x e y sono due vettori di uguale lunghezza allora il comando

```
>> plot(x,y)
```

traccia una curva spezzata che congiunge i punti $(x(i), y(i))$. Per esempio

```
>> x=-4:.01:4;
>> y=sin(x);
>> plot(x,y)
```

traccia il grafico della funzione seno nell'intervallo $[-4, 4]$.

L'istruzione `plot` ammette un parametro opzionale di tipo stringa (racchiuso tra apici) per definire il tipo e il colore del grafico. Infatti è possibile scegliere tra 4 tipi di linee, 5 di punti e 8 colori base. In particolare

| | |
|------|------------------------------|
| '-' | linea continua |
| '--' | linea tratteggiata |
| '-.' | linea tratteggiata e a punti |
| ':' | linea a punti |
| '+' | più |
| 'o' | cerchio |
| 'x' | croce |
| '.' | punto |
| '*' | asterisco |
| 'y' | colore giallo |
| 'r' | colore rosso |
| 'c' | colore ciano |
| 'm' | colore magenta |
| 'g' | colore verde |
| 'w' | colore bianco |
| 'b' | colore blu |
| 'k' | colore nero |

Volendo tracciare per esempio un grafico con linea a puntini e di colore verde L'istruzione è:

```
>> plot(x,y,':g')
```

L'istruzione `plot` consente di tracciare più grafici contemporaneamente. Per esempio

```
>> x=-pi:pi/500:pi;  
>> y=sin(x);  
>> y1=sin(2*x);  
>> y2=sin(3*x);  
>> plot(x,y,'r',x,y1,':g',x,y2,'--b')
```

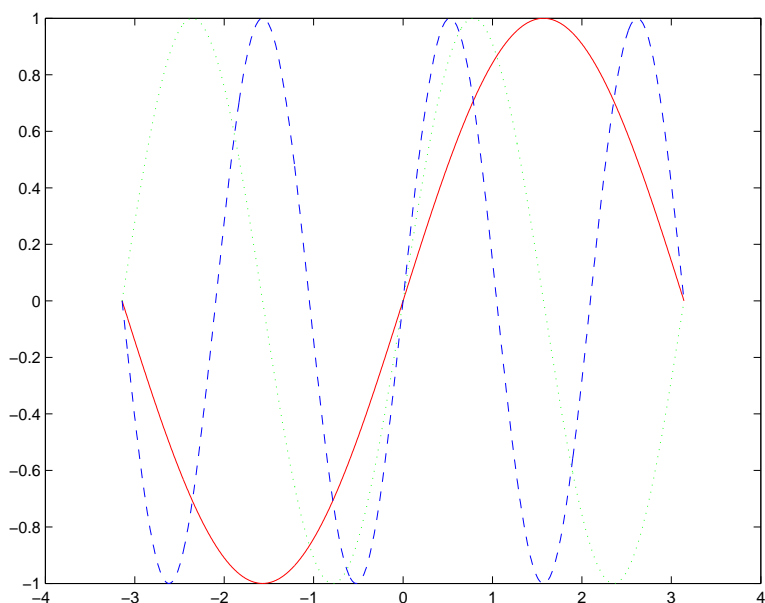


Figura 4.1:

traccia tre grafici nella stessa figura, il primo a tratto continuo (tratto di default) rosso, il secondo verde con tratto a punti ed il terzo tratteggiato e di colore blu. Nella Figura 4.1 è riportato il risultato di tale istruzione. Vediamo ora le altre più importanti istruzioni grafiche:

```
>> title(stringa)
```

serve a dare un titolo al grafico che viene visualizzato al centro nella parte superiore della figura;

```
>> xlabel(stringa)
```

stampa una stringa al di sotto dell'asse delle ascisse;

```
>> ylabel(stringa)
```

stampa una stringa a destra dell'asse delle ordinate (orientata verso l'alto). Per inserire un testo in una qualsiasi parte del grafico esiste il comando

```
>> text(x,y,'testo')
```

che posizione la stringa di caratteri *testo* nel punto di coordinate (x, y) (x e y non devono essere vettori). Di tale comando ne esiste anche una versione che utilizza il mouse:

```
>> gtext('testo')
```

posiziona il testo nel punto selezionato all'interno del grafico schiacciando il pulsante sinistro del mouse.

Il grafico tracciato con il comando `plot` è scalato automaticamente, questo vuol dire che le coordinate della finestra grafica sono calcolate dal programma, tuttavia l'istruzione

```
>> axis([x_min x_max y_min y_max])
```

consente di ridefinire gli assi, e quindi le dimensioni della finestra del grafico corrente.

Una volta tracciato il grafico per poterlo stampare è necessario che venga memorizzato in un file in formato postscript. L'istruzione che consente ciò è

```
>> print -dps nome
```

in questo caso il grafico tracciato in precedenza viene memorizzato nel file *nome.ps* che può essere successivamente stampato utilizzando il comando di stampa del sistema operativo che si sta utilizzando.

A volte può essere utile, una volta tracciato un grafico, ingrandire alcune parti dello stesso. Questo può essere fatto utilizzando il comando `zoom`. Per attivare tale caratteristica è sufficiente il comando

```
>> zoom on
```

mentre per disattivarlo bisogna scrivere:

```
>> zoom off
```

Il funzionamento di tale istruzione è molto semplice. Una volta attivato lo zoom per ingrandire un'area del grafico è sufficiente portare il puntatore del mouse in tale area e cliccare con il tasto sinistro dello stesso. Tale operazione può essere ripetuta alcune volte (non si può ottenere l'ingrandimento un numero molto grande di volte). Per effettuare uno zoom a ritroso bisogna cliccare con il tasto destro del mouse.

Le istruzioni grafiche del Matlab permettono di tracciare curve in tre dimensioni, superfici, di creare delle animazioni e così via. Per approfondire

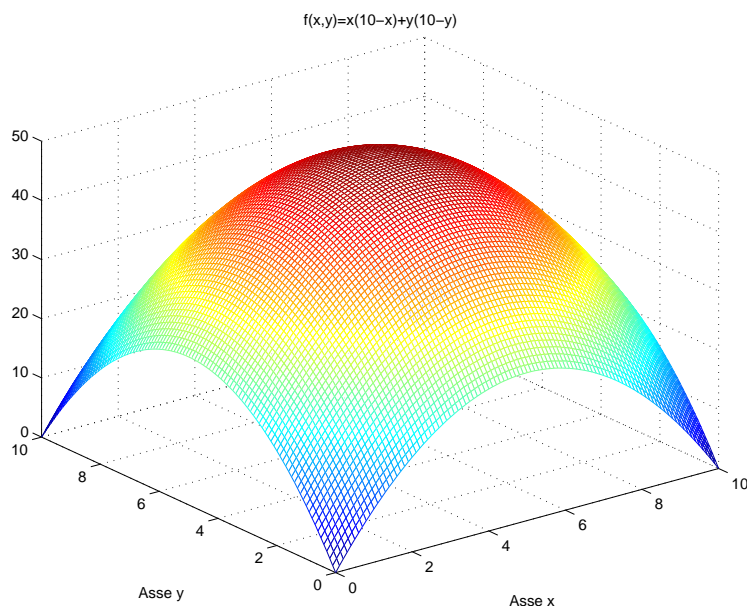


Figura 4.2:

le istruzioni che consentono queste operazioni può essere fatto richiedendo l'help per le istruzioni `plot3`, `mesh` e `movie`.

Nel caso di una superficie, per tracciare il grafico si devono definire due vettori, uno per le ascisse, cioè x , uno per le ordinate, y , e una matrice A per memorizzare le quote, cioè tale che

$$A(i, j) = f(x(j), y(i))$$

Nella Figura 4.2 è tracciato come esempio il grafico della funzione

$$f(x, y) = x(10 - x) + y(10 - y), \quad 0 \leq x, y \leq 10.$$

Le istruzioni per tracciare tale grafico sono le seguenti:

```
x=[0:0.1:10];
y=[0:0.1:10];
n=length(x);
for i=1:n
    for j=1:n
        A(j,i)=x(j)*(10-x(j))+y(i)*(10-y(i));
```

```
        end
    end
    mesh(x,y,A);
    xlabel('Asse x');
    ylabel('Asse y');
    title('f(x,y)=x(10-x)+y(10-y)');
```

4.10 Applicazioni al Calcolo Numerico

Terminiamo il capitolo dedicato al MatLab con l'implementazione di alcune routine di specifici argomenti del Calcolo Numerico, riguardanti in particolare argomenti di Algebra Lineare. La prima riguarda la risoluzione di un sistema triangolare superiore utilizzando il metodo di sostituzione all'indietro.

```
function x=indietro(A,b)
%
% Sintassi x=indietro(A,b)
%
% Risolve un sistema triangolare superiore utilizzando
% il metodo di sostituzione all'indietro
%
% Parametri di input:
% A = Matrice triangolare superiore
% b = Vettore colonna
%
% Parametri di output:
% x = Vettore soluzione
%
n=length(b);
x=zeros(n,1);
if abs(A(n,n))<eps
    error('La matrice A e'' singolare ');
end
x(n)=b(n)/A(n,n);
for k=n-1:-1:1
    x(k)=b(k);
```

```

    for i=k+1:n
        x(k)=x(k)-A(k,i)*x(i);
    end
    if abs(A(k,k))<eps
        error('La matrice A e'' singolare ');
    else
        x(k)=x(k)/A(k,k);
    end
end
end

```

La routine appena descritta risolve un sistema triangolare superiore. Osserviamo innanzitutto che se viene incontrato un elemento diagonale più piccolo, in modulo, della precisione di macchina allora l'algoritmo segnala un errore. Si può inoltre osservare che la routine potrebbe essere scritta in modo più compatto utilizzando la notazione : del MatLab. Infatti il ciclo descritto dalla variabile i si potrebbe sostituire con un'unica istruzione:

$$x(k)=b(k)-A(k,k+1:n)*x(k+1:n);$$

La seconda routine che descriviamo riguarda il metodo di eliminazione di Gauss senza strategie di pivoting per risolvere il sistema lineare $Ax = b$.

```

function x=gauss(A,b);
%
% Sintassi x=gauss(A,b)
%
% Risolve un sistema lineare utilizzando il
% metodo di eliminazione di Gauss
%
% Parametri di input:
% A = Matrice dei coefficienti
% b = Vettore dei termini noti
%
% Parametri di output:
% x = Vettore soluzione
%
[m,n]=size(A);
if m~=n

```

```

        error('Metodo non applicabile');
    end
    if length(b)~=n
        error('Metodo non applicabile');
    end
    for k=1:n
        if abs(A(k,k))<eps
            error('Elemento pivotale nullo ');
        end
        for i=k+1:n
            A(i,k)=A(i,k)/A(k,k);
            for j=k+1:n
                A(i,j)=A(i,j)-A(k,j)*A(i,k);
            end
            b(i)=b(i)-b(k)*A(i,k);
        end
    end
    x=indietro(A,b);

```

Possiamo osservare come la funzione che abbiamo descritto utilizzi la routine `indietro.m` per risolvere il sistema triangolare superiore ottenuto applicando il metodo di Gauss.

Un modo alternativo per implementare il metodo di Gauss è quello di applicarlo alla matrice $A=[A \ b]$ di dimensione $n \times (n + 1)$, inglobando il vettore dei termini noti nella matrice A , modificando il ciclo dell'indice j nel seguente modo:

```
j=k+1:n+1
```

ed evitando di esplicitare la modifica del vettore b ed applicando infine la routine `indietro.m` nel seguente modo:

```
x=indietro(A(:,1:n),A(:,n+1));
```