

# Capitolo 1

## Approccio computazionale: alcune sorgenti di errore

### 1.1 Fonti di errore nella risoluzione di un problema mediante calcolatore

Uno dei problemi fondamentali della Matematica Numerica è quello di valutare l'accuratezza del risultato di un calcolo e dunque l'affidabilità<sup>1</sup> del risultato stesso.

Dati due numeri  $x$  e  $\tilde{x}$  e supposto che  $\tilde{x}$  sia un'approssimazione di  $x$ , il primo problema che si pone è quello di stimare la “bontà” di  $\tilde{x}$  come approssimazione di  $x$ .

La maniera più intuitiva per fare ciò si basa sul concetto di *errore assoluto*.

#### Definizione 1.1. (Errore assoluto)

*Dati un numero  $x$  ed una sua approssimazione  $\tilde{x}$ , si dice errore assoluto in  $\tilde{x}$  la quantità*

$$E = |x - \tilde{x}|.$$

Si può ritener che  $\tilde{x}$  sia una buona approssimazione di  $x$  se  $E$  è “sufficientemente piccolo”. La difficoltà consiste ora nel chiarire che cosa si intenda per “sufficientemente piccolo”.

♣ **Esempio 1.1.** Si considerino  $x = 10.1294$  e  $\tilde{x} = 10.1253$ . Le loro parti intere e le loro prime due cifre decimali sono uguali e risulta  $E = 0.0041 < 10^{-2}$ .



In generale vale la seguente:

---

<sup>1</sup>Per ora basta intuire il significato del termine *affidabile*, basandoci su considerazioni empiriche o sull'esperienza che ciascuno di noi ha.

**Proprietà 1.1. (dell'errore assoluto)**

*Se due numeri  $x$  e  $\tilde{x}$  hanno la stessa parte intera e le stesse prime  $m$  cifre decimali allora risulta*

$$E = |x - \tilde{x}| < 10^{-m}.$$

Tale proprietà fornisce un'interpretazione quantitativa dell'errore assoluto e quindi chiarisce, in qualche modo, il significato dell'espressione "sufficientemente piccolo". <sup>2</sup>

L'errore assoluto non sempre basta a valutare la bontà di un'approssimazione.

♣ **Esempio 1.2.** Se si considerano  $x_1 = 10.12$  e  $\tilde{x}_1 = 10.05$ , si ha  $E_1 = |x_1 - \tilde{x}_1| = 0.07$ , ma anche se si considerano  $x_2 = 10000.14$  e  $\tilde{x}_2 = 10000.07$  risulta  $E_2 = |x_2 - \tilde{x}_2| = 0.07$ . L'errore assoluto è dunque lo stesso in entrambi i casi, benché, intuitivamente, nel secondo caso l'approssimazione appaia essere migliore.

Analogamente, un errore di 50 mila euro nelle previsioni di spesa annua di un'industria con un fatturato annuo di 2500 milioni di euro è praticamente irrilevante, mentre lo stesso errore ha certamente effetti notevoli sul bilancio di una piccola impresa familiare, con un fatturato annuo di 200 mila euro.



Infatti, per come è definito, l'errore assoluto non tiene conto della grandezza del valore da approssimare. Un modo per fare questo è rapportare l'errore assoluto a  $|x|$ , ovvero "scalare" l'errore considerando  $|x|$  come unità di misura. A tal fine si introduce il concetto di *errore relativo*.

**Definizione 1.2. (Errore relativo)**

*Dati un numero  $x$  ed una sua approssimazione  $\tilde{x}$ , si dice errore relativo in  $\tilde{x}$  la quantità*

$$E' = \frac{|x - \tilde{x}|}{|x|} \quad (x \neq 0).$$

---

<sup>2</sup> Si noti che tale proprietà non può essere invertita, nel senso che non è vero che se due numeri differiscono in modulo per meno di  $10^{-m}$  allora hanno la stessa parte intera e le stesse prime  $m$  cifre decimali; un esempio significativo è costituito da  $x = 3.000000$  e  $\tilde{x} = 2.999999$ . Questa restrizione, comunque, è dovuta unicamente al criterio di rappresentazione posizionale, cioè al significato che hanno le cifre della rappresentazione posizionale di un numero, e quindi è una restrizione formale. Pertanto, in seguito si assumerà che  $x$  e  $\tilde{x}$  hanno le stesse cifre fino alla  $m$ -ma cifra decimale ogni qualvolta risulti  $|x - \tilde{x}| < 10^{-m}$ .

♣ **Esempio 1.3.** Consideriamo nuovamente  $x_1 = 10.12$ ,  $\tilde{x}_1 = 10.05$ ,  $x_2 = 10000.14$  e  $\tilde{x}_2 = 10000.07$  e calcoliamo gli errori relativi corrispondenti. Otteniamo così  $E'_1 \simeq 0.0069 = 0.69 \times 10^{-2}$  ed  $E'_2 \simeq 0.0000069 = 0.69 \times 10^{-5}$ , cioè, come era prevedibile, l'errore relativo è nel secondo caso molto più piccolo che nel primo. Si osservi che  $x_1$  e  $\tilde{x}_1$  hanno le prime due cifre significative<sup>3</sup> uguali ed è  $E'_1 < 10^{-1}$ , mentre  $x_2$  e  $\tilde{x}_2$  hanno le prime cinque cifre significative uguali ed è  $E'_2 < 10^{-4}$ .



In generale vale la seguente:

### Proprietà 1.2. (dell'errore relativo)

*Se due numeri  $x$  e  $\tilde{x}$  hanno le stesse prime  $m$  cifre significative, allora risulta*

$$E' = \frac{|x - \tilde{x}|}{|x|} < 10^{-m+1}.$$

Pertanto l'errore relativo misura la distanza tra due numeri in termini di cifre significative corrette, piuttosto che di cifre decimali corrette. Si noti che la relazione

$$E' = \frac{|x - \tilde{x}|}{|x|} < 10^{-m+1} \quad (1.1)$$

non implica che  $x$  e  $\tilde{x}$  abbiano le stesse prime  $m$  cifre significative.

♣ **Esempio 1.4.** Considerati  $x = 57.27$  e  $\tilde{x} = 57.3$ , si ha:

$$E' = \frac{|57.27 - 57.3|}{57.27} \simeq 0.52 \times 10^{-3} < 10^{-3} = 10^{-4+1},$$

ma  $x$  e  $\tilde{x}$  non hanno le prime quattro cifre significative uguali, hanno uguali soltanto le prime due.



La proprietà 1.2 non individua il massimo valore di  $m$  per cui vale la relazione (1.1) e non può pertanto essere invertita. Si può invece dimostrare che se

$$E' = \frac{|x - \tilde{x}|}{|x|} < 10^{-m},$$

---

<sup>3</sup>Con l'espressione *cifre significative* si indicano tutte le cifre comprese tra la prima e l'ultima cifra non nulla del numero considerato, incluse la prima e l'ultima. Ad esempio, considerati  $x_1 = 4000000$ ,  $x_2 = 0.000000200$  e  $x_3 = 0.04501010$ , si ha che  $x_1$  e  $x_2$  hanno una sola cifra significativa (rispettivamente la cifra 4 e la cifra 2) e  $x_3$  ha sei cifre significative (le cifre 450101).

allora risulta che  $x$  e  $\tilde{x}$  hanno almeno le prime  $m$  cifre significative uguali, a patto di trascurare la restrizione formale di cui si è parlato nella nota (2). In seguito tale restrizione sarà trascurata.

Non è possibile affermare che, in generale, l'errore relativo sia più efficace dell'errore assoluto; infatti la scelta dell'uno o dell'altro dipende dal particolare problema in esame. Per esempio, quando si progetta il controllo di un missile a lunga gittata, si desidera che l'obiettivo prefissato sia raggiunto con un errore indipendente dalla distanza dell'obiettivo stesso dal punto di lancio. In questo caso, infatti, interessa proprio che l'errore assoluto tra le coordinate del punto di impatto e quelle dell'obiettivo sia tale da consentire la distruzione dell'obiettivo stesso.

Si osservi infine che in molti casi non si ha interesse a conoscere esattamente il valore numerico dell'errore (assoluto o relativo), ma si cerca di avere informazioni sul suo *ordine di grandezza*<sup>4</sup>.

In generale, nella risoluzione di un problema mediante calcolatore (*risoluzione computazionale*), non ci si aspetta di calcolare la soluzione “esatta”, in quanto il processo di risoluzione comporta sempre l'introduzione di errori. Tali errori sono intrinsecamente legati a questo processo; ciò implica che essi non possono essere evitati, ma solo controllati e stimati, se possibile quantitativamente, per esempio attraverso maggiorazioni. A tale proposito è doveroso citare la frase del Prof. B.Parlett, professore emerito all'università della California, Berkeley, apparsa nel SIAM NEWS, vol. 36, N. 9, del Novembre 2003. B.Parlett, riferendosi ad alcune questioni di calcolo numerico a precisione finita, afferma che le difficoltà sono spesso subdole, piuttosto che profonde ([...] My suggestion is that the difficulties in matrix computations may not be *deep* but they are *subtle*. ). Si suddivide il processo di risoluzione computazionale di un problema nelle fasi fondamentali seguenti (Figura 1.1):

1. descrizione del problema mediante un *modello (problema) matematico*;
2. approssimazione del modello matematico mediante un *modello (problema) numerico*;

---

<sup>4</sup>

**Definizione 1.3. (Ordine di grandezza)**

Un numero reale  $x$  ha **ordine di grandezza**  $\beta^m$  se

$$|x| = \mu \times \beta^m \quad \beta^{-1} \leq \mu < 1 , \quad (1.2)$$

dove  $\beta$  è la base di numerazione del sistema aritmetico considerato; in particolare, se  $m = 0$ , ovvero  $\beta^m = 1$ , si dirà che  $x$  è dell'ordine dell'unità. Se vale la (1.2) si dirà anche che  $x$  è di ordine  $m$ , sottintendendo la base di numerazione.

Con tale definizione, se si considera ad esempio  $\beta = 10$ , risulta che  $r = 0.000000000529$  (raggio di Bohr dell'atomo di idrogeno, in cm) ha ordine di grandezza  $10^{-10}$ , ovvero è di ordine  $-10$ , e  $c = 300000000$  (velocità della luce in m/sec) ha ordine di grandezza  $10^9$ , ovvero è di ordine  $9$ .

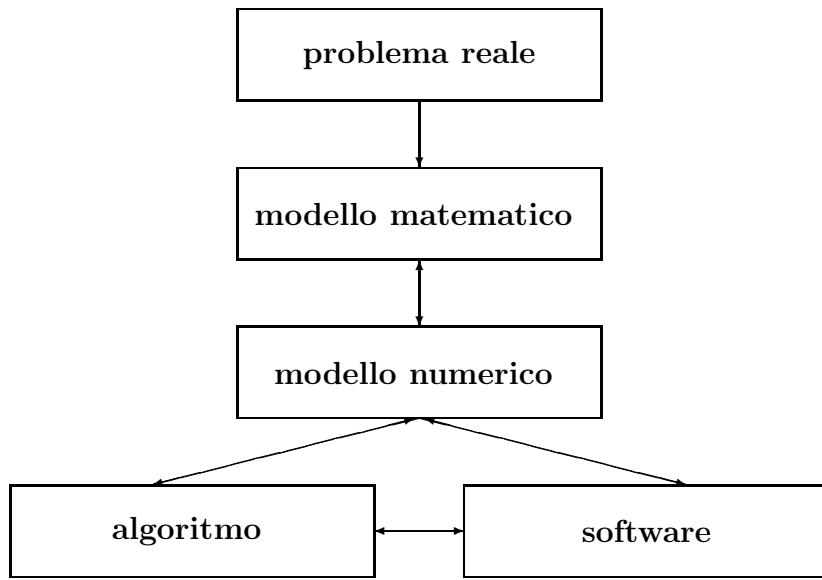


Figura 1.1: Schema del processo di risoluzione computazionale di un problema.

3. descrizione del modello numerico mediante un *algoritmo*;
4. *implementazione* dell'algoritmo in uno specifico ambiente di elaborazione (*software*).

Tale processo presuppone che il problema in esame sia stato già formulato correttamente ed analizzato, individuando chiaramente le informazioni a disposizione per ottenerne la soluzione.

La fase 1 consiste nel passaggio dal problema reale (fisico, chimico, ingegneristico, economico, ...) ad un problema matematico, mediante l'individuazione di relazioni logico-matematiche (equazioni algebriche, equazioni differenziali, disequazioni, ...) che colleghino tra loro le informazioni note, relative al problema in esame, e la soluzione. Si noti che in questa fase devono essere individuate le quantità che si ritiene descrivano esaustivamente il problema (i *dati* del problema) e quelle che sono assunte come *soluzione* del problema stesso.

♣ **Esempio 1.5.** Si consideri una sbarra cilindrica in posizione orizzontale, con un'estremità fissa e l'altra libera. Si vuole determinare la forma assunta dalla sbarra quando all'estremità libera è applicata una forza (deflessione elastica della sbarra), avendo a disposizione le seguenti informazioni: lunghezza e sezione della sbarra, materiale di cui è costituita la sbarra, forza applicata all'estremità libera.

Si vuole costruire un modello matematico per tale problema. Se si suppone che la sezione della sbarra sia piccola rispetto alla sua lunghezza, ci si riconduce ad un problema bidimensionale. Scelto un

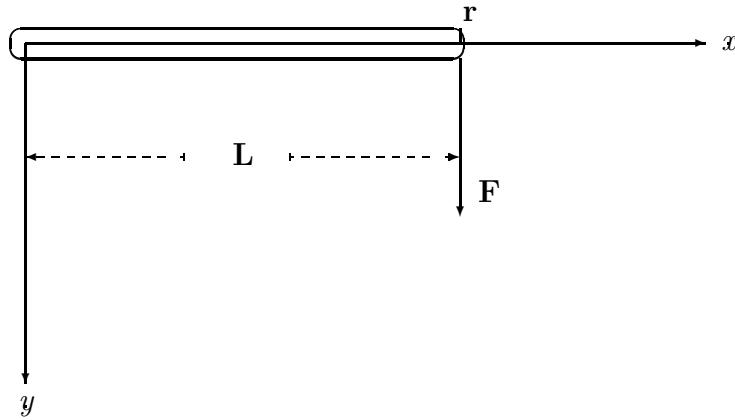


Figura 1.2: Deflessione elastica di una sbarra. Rappresentazione della sbarra prima della flessione, in un riferimento cartesiano.

sistema di riferimento cartesiano ortogonale  $Oxy$ , con l'origine coincidente con l'estremità fissa, l'asse  $x$  tale che la sbarra giaccia su di esso e l'asse  $y$  orientato nella direzione della forza applicata all'estremità libera (Figura 1.2), si ha che la deflessione elastica è la soluzione del seguente problema differenziale (modello A):

$$\begin{aligned} \frac{y''(x)}{(1+y'(x)^2)^{3/2}} &= \frac{F(L-x)}{CI} \\ y(0) &= 0 \\ y'(0) &= 0 \end{aligned}$$

dove  $C$  è una costante dipendente dal materiale (modulo di trazione o di Young),  $I$  il momento di inerzia della sezione trasversale della sbarra,  $L$  la lunghezza della sbarra,  $F$  la forza applicata all'estremità libera. Le condizioni  $y(0) = 0$  e  $y'(0) = 0$  indicano che lo spostamento e la pendenza dell'estremità fissa della sbarra sono nulli. Si noti che  $C$ ,  $I$ ,  $L$ ,  $F$  e  $x$  sono grandezze note e sono dette *dati di input* del modello matematico; la quantità  $y$ , da determinare, è la soluzione del modello ed è denominata *dato di output*.

Si osservi che nella costruzione di tale modello sono stati introdotti degli errori dovuti al fatto che si trascura non solo la sezione della sbarra, ma anche la torsione, la forza di gravità, etc. Un'ulteriore semplificazione del problema si può ottenere supponendo che il peso applicato sia “piccolo e quindi la pendenza  $y'(x)$  sia tale che  $y'(x)^2 \ll 1$ . In tal caso si ha il seguente problema differenziale (modello B):

$$\begin{aligned} y''(x) &= \frac{F(L-x)}{CI} \\ y(0) &= 0 \\ y'(0) &= 0 . \end{aligned}$$

Quest'ultima semplificazione facilita il calcolo della soluzione, ma, ovviamente, introduce in essa ulteriori errori. In Figura 1.3 sono riportati i grafici delle soluzioni relative ai modelli A e B per una sbarra di acciaio ( $C = 200GN/m^2 = 200 \times 10^6 kg/(mm \cdot s^2)$ )<sup>5</sup> di lunghezza  $L = 100$  mm e sezione di raggio

---

<sup>5</sup>  $1GN = 10^9 N = 10^9 kg \cdot m/s^2$



$R = 1.5$  mm, al variare della forza  $F$  ( $F = 10^4 N, 5 \cdot 10^4 N, 10 \cdot 10^4 N, 15 \cdot 10^4 N$ )<sup>6</sup> Si noti che per  $F = 10^4 N$  i grafici relativi alle due soluzioni sono praticamente coincidenti, per  $F = 5 \cdot 10^4 N$  essi differiscono leggermente, per  $F = 10 \cdot 10^4 N$  tale differenza diventa sensibile (le posizioni dell'estremo libero della sbarra secondo il modello A ed il modello B distano circa 10 mm), per  $F = 15 \cdot 10^4 N$  le due soluzioni si discostano significativamente.

In generale, anche se si riesce a determinare un'espressione analitica della soluzione di un problema matematico, spesso non è possibile utilizzarla per calcolare il valore della soluzione in un insieme assegnato di punti. Ciò accade, ad esempio, per il modello A della deflessione elastica di una sbarra<sup>7</sup>, o, più semplicemente, per la serie  $\sum_{n=0}^{\infty} (1/n^3)$ , la funzione integrale  $\int_0^x (\sin t/t) dt$ , la funzione  $\log x$ , etc. La fase 2 consiste nel passare dal modello matematico al *modello numerico*, cioè ad un particolare modello matematico che coinvolge un numero *finito* di numeri reali e relazioni matematiche tra questi, calcolabili con un numero *finito* di operazioni aritmetiche.

♣ **Esempio 1.6.** Si consideri lo sviluppo in serie di Mac Laurin della funzione  $e^x$ :

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots . \quad (1.3)$$

Se si sostituisce alla serie la somma dei primi  $n$  termini si ottiene un'espressione valutabile con un numero finito di operazioni aritmetiche. Naturalmente tale sostituzione comporta l'introduzione di un errore, detto *errore di troncamento analitico*, che dipende dalla scelta di  $n$ . Più precisamente, dato che

$$e^x = 1 + x + \frac{x^2}{2!} + \dots + \frac{x^n}{n!} + R_n(x) , \quad (1.4)$$

dove  $R_n(x)$  è il resto di indice  $n$  della serie considerata, si ha che  $R_n(x)$  rappresenta l'errore di troncamento analitico. A questo punto bisogna affrontare il problema di stimare l'errore di troncamento analitico. In questo caso si ha:

$$R_n(x) = e^{\xi} \frac{x^{n+1}}{(n+1)!} ,$$

---

<sup>6</sup>Se la sezione trasversale della sbarra è circolare di raggio  $R$ , il centro di massa è al centro della circonferenza ed il momento di inerzia lungo ciascun asse che giace nella sezione trasversale e passante per il centro, è  $I = \frac{\pi}{4} R^4$ .

<sup>7</sup>In questo caso la soluzione (e, quindi, la forma della sbarra) ha l'espressione parametrica:

$$\begin{aligned} x &= \sqrt{\frac{2IC}{F}} (\sqrt{\cos \theta_0} - \sqrt{\cos \theta_0 - \cos \theta}) \\ y &= \sqrt{\frac{IC}{2F}} \int_0^{\pi/2} \frac{\cos \theta d\theta}{\sqrt{\cos \theta_0 - \cos \theta}} \end{aligned}$$

dove  $\theta = \theta(l)$  è l'angolo formato dalla tangente alla sbarra nel generico punto a distanza  $l$  dall'estremo fissato, e dall'asse  $y$ . All'estremo fissato ( $l = 0$ ),  $\theta(0) = \frac{\pi}{2}$ , mentre all'estremo libero ( $l = L$ , lunghezza della sbarra) si pone  $\theta(L) = \theta_0$ .

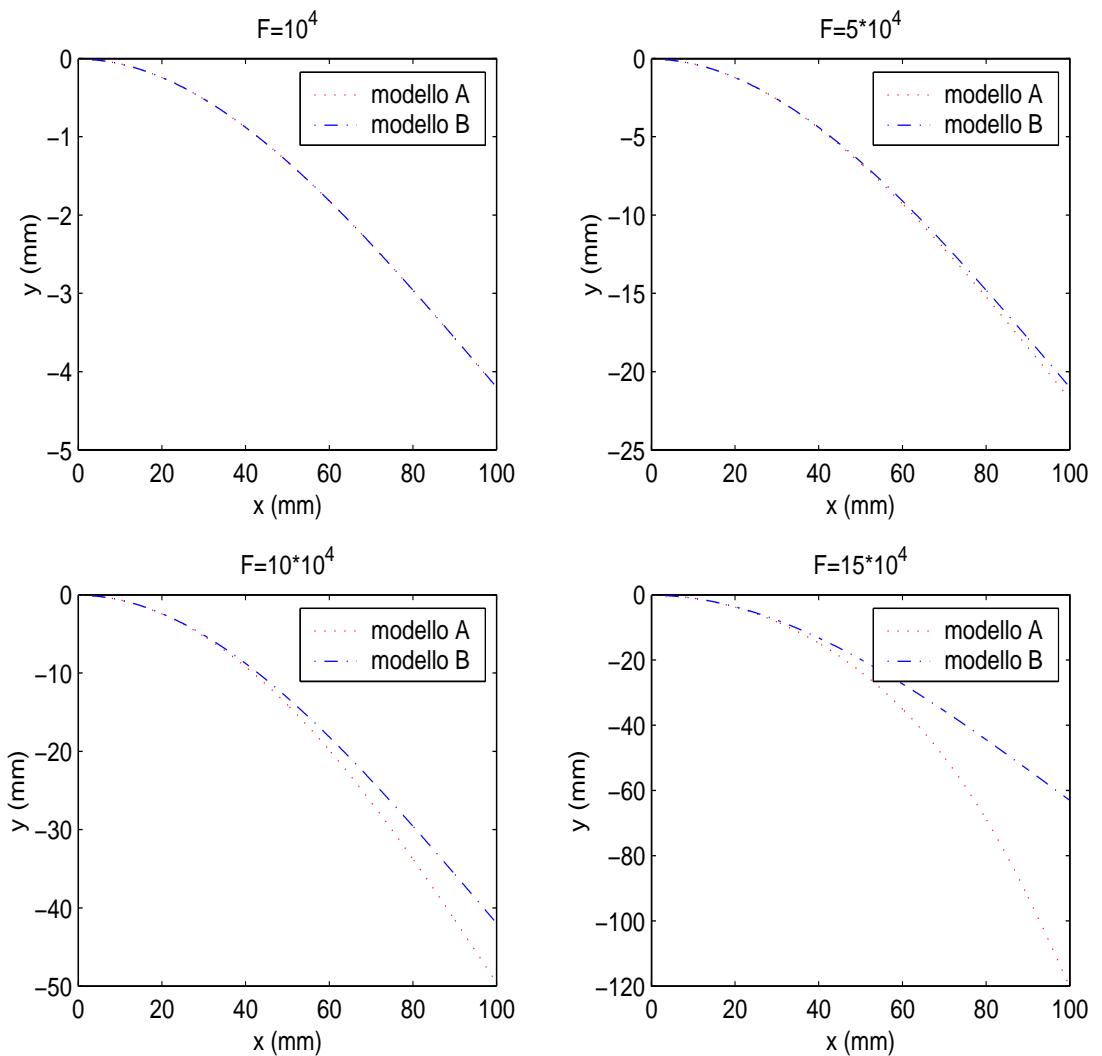


Figura 1.3: Deflessione elastica di una sbarra. Soluzioni relative ai modelli A e B.

dove  $\xi$  appartiene ad un opportuno intervallo del tipo  $[-a, a]$ ; dunque, una stima calcolabile dell'errore di troncamento analitico è la seguente:

$$|R_n(x)| \leq e^a \frac{|x|^{n+1}}{(n+1)!} . \quad (1.5)$$

Si osservi comunque che la maggiorazione (1.5) risolve solo apparentemente il problema della stima di  $R_n(x)$ ; infatti il problema si sposta alla determinazione di  $a$ , o meglio di un valore di  $a$  per cui  $e^a \frac{|x|^{n+1}}{(n+1)!}$  non sia “eccessivamente grande”.



♦ **Esempio 1.7.** Sia  $f(x)$  una funzione reale di variabile reale, derivabile. Se si sostituisce alla derivata di  $f(x)$  il rapporto incrementale:

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

dove  $h$  è scelto opportunamente<sup>8</sup>, si ottiene un'espressione valutabile con un numero finito di operazioni aritmetiche, supposto, ovviamente, che il calcolo di  $f(x)$  in un punto possa essere effettuato con un numero finito di operazioni aritmetiche. Chiaramente l'errore di troncamento analitico, o *errore di discretizzazione*<sup>9</sup>, dipende da  $h$  e, poiché

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

l'errore di troncamento analitico tende a 0 al tendere di  $h$  a 0.



La fase 3 consiste nell'individuare un *algoritmo* risolutivo per il modello numerico, cioè una sequenza finita di operazioni aritmetiche che, se eseguite, consentono di calcolare, a partire dai dati di input, i dati di output.

Il concetto di algoritmo è indissolubilmente legato al suo esecutore, nel senso che le operazioni che vi compaiono devono essere chiaramente specificate ed eseguibili in relazione alle possibilità dell'esecutore. Nel nostro caso l'esecutore di algoritmi è un calcolatore con un *sistema aritmetico a precisione finita*, cioè che opera solo su numeri rappresentati mediante un numero finito di cifre significative.

---

<sup>8</sup>A tale proposito si veda il quesito 16, del §1.10.2

<sup>9</sup>Nella nostra accezione un problema matematico è caratterizzato dal concetto di infinito ed un problema numerico dalla finitezza; un problema matematico può essere inoltre continuo o numerabile. Talvolta il passaggio dal modello matematico al modello numerico richiede prima il passaggio da un processo continuo ad uno numerabile e poi da questo ad un processo finito; in tal caso, spesso l'errore associato al primo passaggio è detto di *discretizzazione* e quello associato al secondo è detto di *troncamento analitico*. Qui si userà la denominazione *troncamento analitico* per indicare l'errore complessivo dovuto alla sostituzione del modello matematico col modello numerico.

Questa fase del processo risolutivo comporta l'introduzione di un errore, detto *errore di roundoff*, dovuto proprio all'utilizzo di un sistema aritmetico a precisione finita. E' a questo tipo di errore che sarà dedicato ampio spazio in seguito.

La fase 4 consiste nell'implementazione dell'algoritmo in uno specifico ambiente di calcolo (*software* o *programma*).

I dati di output del programma dipendono quindi non solo dall'algoritmo, ma anche dalle caratteristiche specifiche dell'ambiente di calcolo, come, ad esempio, i compilatori, il particolare sistema aritmetico, etc. I dati di output del programma sono assunti come la "soluzione" del problema reale nell'approccio computazionale.

## 1.2 I sistemi aritmetici a precisione finita

Il termine *sistema aritmetico a precisione finita* si riferisce in generale ai criteri di rappresentazione in memoria dei dati di tipo numerico ed alle operazioni elementari (dell'unità aritmetica) definite su di essi.<sup>10</sup>

I tipi di dati<sup>11</sup> numerici elementari qui considerati sono il tipo *intero* ed il tipo *reale floating-point* (reale f.p.)<sup>12</sup>, poiché, come si vedrà in seguito, sono quelli che possono generare situazioni computazionali che riducono o annullano l'affidabilità di un algoritmo numerico.

### 1.2.1 Il sistema aritmetico intero

Il criterio di rappresentazione dei dati di tipo intero è quello posizionale in base  $\beta$  ( $\beta \in \mathcal{N}$ ,  $\beta > 1$ )<sup>13</sup>. Usualmente  $\beta = 2$ , cioè si utilizza la rappresentazione binaria; ad esempio:

$$\begin{aligned} 1_{10} &= 1_2, \\ 3_{10} &= 11_2, \\ 16_{10} &= 10000_2, \\ 734_{10} &= 1011011110_2, \end{aligned}$$

e la memorizzazione avviene ponendo in una locazione la stringa di bit, da destra verso sinistra. Il bit più a sinistra è riservato all'informazione (binaria) del segno.

La lunghezza finita  $l$  di una locazione di memoria comporta l'esistenza di un massimo (e di un minimo) intero rappresentabile. Si consideri, ad esempio, un sistema aritmetico intero con  $l = 8$ <sup>14</sup>; si ha allora che il numero  $64_{10} = 1000000_2$  è rappresentabile e viene

---

<sup>10</sup>Si prenderà in considerazione solo la rappresentazione dei numeri in memoria (rappresentazione interna). La rappresentazione dei numeri sui dispositivi di I/O (video, stampante, etc.) è la normale rappresentazione decimale; allo stesso modo si prenderanno in considerazione solo le operazioni aritmetiche eseguibili dall'unità aritmetica.

<sup>11</sup>Si definisce *tipo di dato* l'insieme dei valori che un dato può assumere.

<sup>12</sup>Il tipo di dato *complesso f.p.* è essenzialmente riconducibile a quello reale f.p., a cui appartengono la parte reale ed il coefficiente dell'immaginario.

<sup>13</sup> $\mathcal{N}$  indica l'insieme dei numeri naturali.

<sup>14</sup>Nella maggior parte degli calcolatori  $l = 32, 64, 128$ .

memorizzato nel modo seguente:<sup>15</sup>

+	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

In tale sistema il più grande numero rappresentabile, che sarà detto  $imax$ , è  $127_{10}$ , poiché la sua memorizzazione è

+	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

mentre la rappresentazione binaria di  $128_{10} = 10000000_2$  è costituita da 8 cifre e non può essere memorizzata. Si può concludere che un intero  $n \in \mathbb{Z}$ <sup>16</sup> è un intero *rappresentabile* se e solo se

$$|n| \leq imax$$

e che l'insieme  $I$  degli interi rappresentabili è l'insieme finito

$$\{-imax, -imax + 1, \dots, -1, 0, 1, \dots, imax - 1, imax\}$$

dove

$$imax = 2^{l-1} - 1.$$

Se  $n \in \mathbb{Z}$ , ma  $n \notin I$ , allora  $n$  non è rappresentabile ed un eventuale tentativo di memorizzazione provoca la situazione eccezionale detta *overflow* degli interi. In generale l'insieme  $I$  è individuato univocamente dalla sola lunghezza  $l$ .<sup>17</sup>

Sui dati di tipo intero sono definite le operazioni: *addizione*, *sottrazione*, *moltiplicazione* e *divisione*<sup>18</sup>, denotate qui in modo diverso dalle corrispondenti operazioni in  $\mathbb{Z}$ :  $[+]$ ,  $[-]$ ,  $[ \times ]$ ,  $[ / ]$ . Il risultato  $r$  di un'operazione  $[\#]$  (dove  $\#$  sta per  $+, -, \times, /$ ) si dice *definito* se  $r \in I$ , altrimenti si dice che esso è indefinito e che si è verificata una situazione eccezionale di overflow degli interi.

<sup>15</sup>Per maggiore chiarezza, per rappresentare il bit del segno sono usati i simboli  $+$  e  $-$  al posto delle cifre binarie 0 e 1.

<sup>16</sup> $\mathbb{Z}$  indica l'insieme degli interi relativi.

<sup>17</sup>Oltre alla rappresentazione per segno e modulo, qui utilizzata, esistono altre rappresentazioni degli interi relativi. In particolare, si ricorda quella del *complemento a due* [6]: il primo bit di sinistra ha ancora il ruolo di bit di segno, ed i numeri non negativi hanno la stessa forma fornita dalla rappresentazione in segno e modulo, mentre il valore, in modulo, di un numero negativo rappresentato in complemento a due si ottiene invertendo il valore di ogni singolo bit (complemento a uno), valutando la rappresentazione come intero senza segno, ed aggiungendo uno al risultato; in essa pertanto, il tipo intero non è simmetrico rispetto allo zero, essendo  $imin = -imax - 1$  ( $imin$  = minimo intero rappresentabile). Ad esempio nella rappresentazione in *complemento a due* il numero  $1010_2 = -6_{10}$ , infatti:  $1010 = -(0101 + 1) = -(5 + 1) = -6$ .

<sup>18</sup>Il risultato della divisione intera è la parte intera del quoziente.

♣ **Esempio 1.8.** Siano  $imax = 100$ ,  $n = 50$ ,  $m = 60$ . Si ha:

$$|n + m| = 110 > imax$$

e quindi il risultato dell'operazione  $50 \boxed{+} 60$  è indefinito.



In generale, se  $n$  e  $m$  sono interi tali che

$$|n| < imax, |m| < imax ,$$

allora

$$n \boxed{\#} m = \begin{cases} n\#m & \text{se } |n\#m| \leq imax \\ \text{indefinito} & \text{se } |n\#m| > imax . \end{cases} \quad (1.6)$$

♣ **Esempio 1.9.** In un sistema aritmetico intero con  $\beta = 2$  e  $l = 32$  si vuole calcolare  $n!$ . Per  $n > 13$  tale calcolo provoca overflow in quanto

$$13! > 2^{31} - 1 = imax$$

(si ricordi che uno dei 32 bit a disposizione per la rappresentazione del numero è riservato al segno).



♣ **Esempio 1.10.** Siano  $imax = 100$ ,  $n = 60$ ,  $m = 50$  e  $p = -40$  e si voglia calcolare

$$n + m + p .$$

Il risultato dell'operazione

$$(n \boxed{+} m) \boxed{+} p = (60 \boxed{+} 50) \boxed{+} (-40)$$

è indefinito, essendo

$$|60 + 50| > 100 = imax ,$$

mentre si ha che

$$n \boxed{+} (m \boxed{+} p) = 60 \boxed{+} (50 \boxed{+} (-40)) = 60 \boxed{+} 10 = 70 .$$

Non vale dunque la proprietà associativa dell'addizione.



In generale, la (1.6) comporta che alcune proprietà dell'aritmetica tradizionale (proprietà associativa dell'addizione e proprietà distributiva della moltiplicazione rispetto all'addizione) non siano più valide.

In conclusione, ricapitolando quanto detto precedentemente, un sistema aritmetico intero è costituito da:

- un insieme finito  $I \subset \mathcal{Z}$ , i cui elementi (gli elementi del tipo intero), sono detti gli *interi rappresentabili*;
- un criterio di rappresentazione degli elementi di  $I$ , su cui è basata la memorizzazione degli elementi del tipo;
- un insieme di operazioni, dette operazioni aritmetiche sul tipo intero, in cui sia gli operandi sia i risultati sono elementi del tipo;
- un insieme di algoritmi (cablati nell'unità aritmetica), che consentono l'esecuzione delle operazioni aritmetiche sul tipo intero.

### 1.2.2 Il sistema aritmetico floating-point

Il criterio di rappresentazione degli elementi di tipo reale è quello della rappresentazione *floating-point normalizzata* (f.p.n.) in base  $\beta$ <sup>19</sup>. Tale criterio si basa sull'idea di utilizzare solo le cifre significative di un numero reale, in modo che questo sia rappresentato come prodotto di un numero  $f$ , con  $\beta^{-1} \leq |f| < 1$ , per una potenza intera della base  $\beta$ . Per esempio, il numero  $x = 22.334$  ha la rappresentazione f.p.n. in base 10:

$$x = 0.22334 \times 10^2, \quad \text{con } f = 0.22334, \quad 10^{-1} \leq 0.22334 < 1.$$

Si noti che la limitazione  $\beta^{-1} \leq |f| < 1$  (*normalizzazione*) implica che la prima cifra della parte frazionaria di  $f$  (nella base  $\beta$ ) sia diversa da zero<sup>20</sup> e rende unica la rappresentazione. Qualsiasi numero reale  $x \neq 0$  può essere rappresentato in forma f.p.n. e si ha

$$x = f \times \beta^e, \quad e \in \mathcal{Z}, \quad \beta^{-1} \leq |f| < 1.$$

Detto  $m$  il numero intero formato dalle cifre della parte frazionaria di  $f$  ed avente lo stesso segno di  $f$ , il numero  $x = f \times \beta^e$  è individuato dalla coppia  $(m, e)$ ;  $m$  è detto mantissa ed  $e$  è detto esponente della rappresentazione f.p.n.

♣ **Esempio 1.11.** Considerato  $\beta = 10$ , si ha:

$$\begin{aligned} x = 35.16904 &\xrightarrow{\text{f.p.n.}} x = 0.3516904 \times 10^2 \implies (m, e) = (+3516904, +2) \\ x = 0.00012 &\xrightarrow{\text{f.p.n.}} x = 0.12 \times 10^{-3} \implies (m, e) = (+12, -3) \\ x = -300000000 &\xrightarrow{\text{f.p.n.}} x = -0.3 \times 10^9 \implies (m, e) = (-3, +9) \\ x = -0.00000479 &\xrightarrow{\text{f.p.n.}} x = -0.479 \times 10^{-5} \implies (m, e) = (-479, -5) \end{aligned}$$



<sup>19</sup>La scelta di  $\beta$  è fatta sulla base di considerazioni di efficienza e semplicità di implementazione hardware. L'unico requisito teorico è che  $\beta$  sia un numero intero maggiore dell'unità.

<sup>20</sup>Infatti si ha:

$$|f| = 0.d_1d_2\ldots, \quad \begin{aligned} 1 \leq d_1 &\leq \beta - 1, \\ 0 \leq d_k &\leq \beta - 1, \quad k > 1. \end{aligned}$$

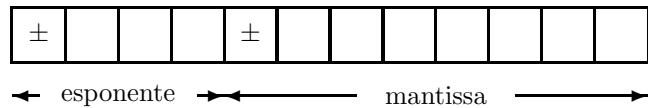
La memorizzazione di un numero reale rappresentato mediante il criterio f.p.n. richiede quindi la memorizzazione delle informazioni seguenti: il segno di  $m$ , le cifre di  $m$ , il segno di  $e$  e le cifre di  $e$ .<sup>21</sup> A causa della lunghezza finita di una locazione di memoria si ha un numero finito di cifre per la rappresentazione di  $m$  ed un numero finito di cifre per la rappresentazione di  $e$ .

Il criterio di rappresentazione f.p.n. è quindi univocamente caratterizzato dalle quantità:

1. la base  $\beta$ ;
2. il massimo numero  $t$  di cifre (nella base  $\beta$ ) della mantissa  $m$ ;
3. il massimo numero di cifre (nella base  $\beta$ ) per l'esponente  $e$ , ovvero un minimo ed un massimo esponente rappresentabile.

Il numero  $t$  è detto *precisione* del sistema aritmetico f.p.n.. Si denoteranno inoltre con  $emin$  ed  $emax$  rispettivamente il minimo ed il massimo esponente rappresentabile.<sup>22</sup> In termini di mantissa ed esponente, l'unicità della rappresentazione  $\pm m \times \beta^e$ , risulta legata all'essere  $emin \leq e \leq emax$  ed alla condizione di normalizzazione per la mantissa,  $\beta^{t-1} \leq m \leq \beta^t - 1$ . Negli esempi seguenti  $\beta$  sarà rappresentata in base 10, mentre  $m$  ed  $e$  saranno in genere rappresentati in base  $\beta$ .

♣ **Esempio 1.12.** Si consideri il criterio f.p.n. a precisione finita caratterizzato da  $\beta = 2$ ,  $t = 7$ ,  $emin = -7$  ed  $emax = 7$ , a cui corrisponde una locazione di memoria (fissata di lunghezza 12) suddivisa come segue:



Si considerino i numeri, per semplicità già in forma binaria,

$$a = 1011.11, \quad b = -101.100111, \quad c = 101110101.1,$$

<sup>21</sup> Si noti che se  $\beta = 2$  la prima cifra di  $m$  è necessariamente 1 e quindi è inutile memorizzarla esplicitamente (memorizzazione con il primo *bit implicito*).

<sup>22</sup> Si noti che, poiché lo zero ha mantissa nulla, la sua rappresentazione f.p.n. è anomala; si assumerà quindi

$$0 \xrightarrow{f.p.n.} 0.00\dots \times \beta^{emin}.$$

che in notazione f.p.n. sono rappresentati come segue:

$$\begin{aligned} a &= 0.101111 \times 2^{100} && \text{cioè } m = +101111, \quad e = +100, \\ b &= -0.101100111 \times 2^{11} && \text{cioè } m = -101100111, \quad e = +11, \\ c &= 0.1011101011 \times 2^{1001} && \text{cioè } m = +1011101011, \quad e = +1001. \end{aligned}$$

Solo  $a$  può essere rappresentato secondo il criterio f.p.n. a precisione finita fissato e quindi memorizzato nella corrispondente locazione di memoria. Infatti,  $b$  ha l'esponente rappresentabile ( $emin \leq 3 \leq emax$ ), ma il numero di cifre della sua mantissa è 9, mentre per  $c$  non sono rappresentabili né l'esponente ( $9 > emax$ ), né la mantissa ( $10 > t$ ).<sup>23</sup>

♣

Il fatto che le locazioni di memoria abbiano lunghezza finita comporta conseguenze notevoli:

1. esistono un massimo numero reale rappresentabile  $rmax$  (come per il tipo intero) ed un minimo numero reale positivo rappresentabile  $rmin$  (ciò discende dalla necessaria finitezza della rappresentazione di  $e$ );
2. non tutti i numeri reali appartenenti all'intervallo  $[rmin, rmax]$  sono rappresentabili secondo il criterio f.p.n. a precisione finita (ciò discende dalla necessaria finitezza della rappresentazione di  $m$ ).

Fissato un particolare criterio f.p.n. a precisione finita, consegue che risulta univocamente determinato l'insieme  $F = F(\beta, t, emin, emax)$  degli elementi del tipo reale che gode delle seguenti proprietà:

- $F$  ha un elemento di massimo modulo,  $rmax$ , ed un elemento di minimo modulo,  $rmin$ ;
- $F$  contiene lo zero;
- $F$  è costituito da un numero finito di elementi;
- gli elementi di  $F$  sono i numeri reali che appartengono all'insieme

$$[-rmax, -rmin] \cup \{0\} \cup [rmin, rmax] \tag{1.7}$$

ed in notazione f.p.n. hanno la mantissa con un numero di cifre minore o uguale a  $t$ .

Gli elementi di  $F$  si dicono *numeri rappresentabili esattamente* o *numeri macchina* e l'insieme (1.7) si dice *insieme di rappresentabilità* (Figura 1.4)

---

<sup>23</sup> La rappresentazione dell'esponente  $e$ , qui considerata, è detta *rappresentazione segno/modulo*. Spesso, però, l'esponente  $e$  non è rappresentato in tale forma, ma nella forma  $e + eb$ , dove  $eb$  è un numero intero (detto *bias*) tale che  $e + eb \geq 0$ . Ad esempio, se  $\beta = 2$  e il numero di cifre per l'esponente (segno incluso) è 8, si considera solitamente  $eb = 127$  e  $0 \leq e + eb \leq 255$  (si noti che l'insieme degli esponenti rappresentabili non è simmetrico rispetto allo zero, in quanto  $emin = -127$  ed  $emax = 128$ ). Tale rappresentazione è detta *eccesso 127*.

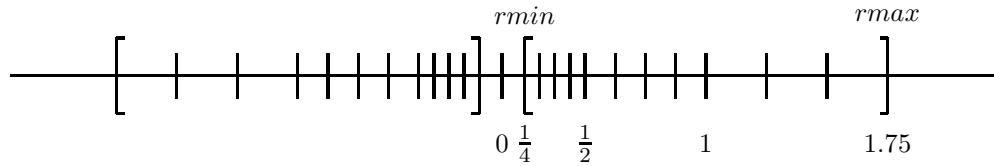


Figura 1.4: Rappresentazione grafica dell'insieme  $F(\beta, t, emin, emax)$ , e del corrispondente insieme di rappresentabilità (cfr. esempio 1.13)

♣ **Esempio 1.13.** Si consideri  $F(2, 3, -1, 1)$ .  $F$  è il sottoinsieme finito di  $\mathbb{R}$ :

$$\begin{aligned} F &= \{0, \pm 0.100 \times 2^{-1}, \pm 0.101 \times 2^{-1}, \dots, \pm 0.110 \times 2^1, \pm 0.111 \times 2^1\} = \\ &= \{0, \pm 0.25_{10}, \pm 0.3125_{10}, \dots, \pm 1.5_{10}, \pm 1.75_{10}\}, \end{aligned}$$

costituito da 25 elementi (Figura 1.4). In questo caso  $rmax = 0.111 \times 2^1 = 1.75_{10}$  e  $rmin = 0.100 \times 2^{-1} = 0.25_{10}$ .

♣

In generale, dato  $F(\beta, t, emin, emax)$ , il numero di elementi di  $F$  è<sup>24</sup>

$$2(\beta - 1)\beta^{t-1}(emax - emin + 1) + 1$$

e risulta

$$\begin{aligned} rmax &= 0.dd\dots d \times \beta^{emax} = \beta^{emax}(1 - \beta^{-t}) \quad (d = \beta - 1); \\ rmin &= 0.10\dots 0 \times \beta^{emin} = \beta^{emin-1} \end{aligned}$$

Gli elementi di  $F$  non sono uniformemente spaziati fra loro, ma solo fra potenze della base successive. Dunque la spaziatura assoluta non è uniforme, ma quella relativa lo è; infatti ogni elemento di  $F$  differisce dagli adiacenti per una sola unità sull'ultima cifra della mantissa.

Per ogni numero reale  $x = f \times \beta^e$  si verifica una ed una sola delle tre situazioni:

1.  $x \in F$  ed è quindi esattamente rappresentabile;
2.  $x$  non è rappresentabile in  $F$ ;
3.  $x$  è rappresentabile, ma non esattamente, in  $F$ .

<sup>24</sup>Fissato un valore dell'esponente, la prima cifra della mantissa di un numero diverso da zero può assumere  $\beta - 1$  valori distinti e ciascuna delle rimanenti  $t - 1$  cifre può assumere  $\beta$  valori distinti, per un totale di  $\beta^{t-1}(\beta - 1)$  possibili combinazioni delle cifre della mantissa. Se si osserva che gli esponenti rappresentabili sono  $emax - emin + 1$  e che per ogni numero rappresentabile positivo il suo opposto è rappresentabile, e si considera anche lo zero, si trova che il numero di elementi di  $F$  è quello specificato.

Il caso 2 si presenta se  $|x| < rmin$ , ed allora si dice che si è verificato un *underflow*, oppure se  $|x| > rmax$ , ed allora si dice che si è verificato un *overflow* ( $rmin$  e  $rmax$  sono chiamati rispettivamente *soglia di underflow* e *soglia di overflow*); tale situazione in pratica si ha quando  $e < emin$  oppure  $e > emax$ . Per la gestione delle cosiddette *situazioni eccezionali* si rimanda al paragrafo 1.7 sul sistema aritmetico standard IEEE.

♣ **Esempio 1.14.** Dato l'insieme  $F(2, 3, -1, 1)$  (cfr. esempio 1.13), si ha underflow per ogni  $x \in \mathbb{R}$  tale che  $x < 0.25$  ed overflow per ogni  $x > 1.75$ .



Il caso 3 si verifica quando  $emin \leq e \leq emax$  e il numero di cifre binarie di  $m$  è maggiore di  $t$ . Si pone dunque il problema di come rappresentare  $x$  mediante un elemento di  $F$ .

♣ **Esempio 1.15.** Si consideri  $F(10, 2, -9, 9)$ . Un numero  $\tilde{x} \in F$  è del tipo

$$\tilde{x} = \pm 0.d_1d_2 \times 10^e, \quad e = -9, -8, \dots, 8, 9.$$

Il numero  $x = 1.26$  non appartiene a  $F$ , ma può essere rappresentato mediante un elemento di  $F$ . Infatti  $x$  è strettamente compreso tra due numeri di  $F$ , cioè

$$1.2 < x < 1.3$$

ed uno dei due si può scegliere come approssimazione di  $x$  mediante un elemento di  $F$ .



Per approssimare un numero reale  $x = 0.d_1d_1 \dots d_t d_{t+1} \dots \times \beta^e$  appartenente all'insieme di rappresentabilità di  $F$ , ma non rappresentabile esattamente, con un numero  $fl(x) \in F$ , si può utilizzare una delle due tecniche seguenti:

- *troncamento*:  $fl(x)$  si ottiene troncando le cifre della mantissa seguenti la  $t$ -ma, cioè

$$fl(x) = 0.d_1d_2 \dots d_t \times \beta^e;$$

- *arrotondamento*:  $fl(x)$  si ottiene da  $x$  sommando  $\frac{1}{2}\beta$  alla  $(t+1)$ -esima cifra della sua mantissa e poi troncando le cifre seguenti la  $t$ -ma, ovvero incrementando di 1  $d_t$  se è  $d_{t+1} \geq \frac{1}{2}\beta$  e poi troncando le cifre della mantissa successive alla  $t$ -ma, cioè:

$$fl(x) = \begin{cases} 0.d_1d_2 \dots d_t \times \beta^e & d_{t+1} < \frac{1}{2}\beta \\ 0.d_1d_2 \dots (d_t + 1) \times \beta^e & d_{t+1} \geq \frac{1}{2}\beta \end{cases}$$

In relazione all'esempio 1.15 si ha:

$$\begin{aligned} \text{troncamento : } & fl(1.26) = 1.2 ; \\ \text{arrotondamento : } & fl(1.26) = 1.3 . \end{aligned}$$

Se invece di  $x = 1.26$  si considera  $x = 1.24$ , si ha:

$$\begin{aligned} \text{troncamento : } & fl(1.24) = 1.2 ; \\ \text{arrotondamento : } & fl(1.24) = 1.2 . \end{aligned}$$

In un sistema aritmetico reale f.p., le caratteristiche legate alla rappresentazione, cioè l'insieme  $F$  e le funzioni di troncamento ed arrotondamento, sono dette le *caratteristiche statiche del sistema*. Si definiscono *caratteristiche dinamiche* le proprietà legate alle operazioni f.p. definite in  $F$  ed agli algoritmi utilizzati dall'unità aritmetica per implementarle.

Le operazioni f.p. sono: addizione f.p.  $\oplus$ , sottrazione f.p.  $\ominus$ , moltiplicazione f.p.  $\otimes$ , divisione f.p.  $\oslash$ , qui denotate in modo diverso dalle corrispondenti usuali operazioni in  $\mathbb{R}$ .<sup>25</sup>

Il risultato  $\tilde{r} = x \# y$  (dove  $\#$  sta per  $+, -, \times, /$ ), con  $x, y \in F$ , si dice *definito* se  $r = x \# y$  appartiene all'insieme di rappresentabilità, altrimenti il risultato è indefinito e si dice che si è verificata una situazione eccezionale di *overflow* se  $|r| > r_{max}$ , o di *underflow* se  $|r| < r_{min}$ . E' opportuno sottolineare che anche nel caso di risultato  $r$  rappresentabile si ha in generale  $\tilde{r} \neq r$ , in quanto  $\tilde{r}$  appartiene sempre a  $F$ , mentre  $r$  può non appartenere a  $F$ .

♣ **Esempio 1.16.** Si considerino  $F(10, 2, -2, 2)$  e la moltiplicazione f.p. tra  $x = 2.8$  e  $y = 7.7$ . E' immediato verificare che  $x, y \in F$ ; infatti in rappresentazione f.p.n. si ha  $x = 0.28 \times 10^1$  e  $y = 0.77 \times 10^1$  e quindi  $x$  e  $y$  sono rappresentabili esattamente. Il risultato  $r = x \times y = 21.56 = 0.2156 \times 10^2$  della moltiplicazione usuale in  $\mathbb{R}$  appartiene all'insieme di rappresentabilità, ma, avendo una mantissa di 4 cifre, non è rappresentabile esattamente, cioè non appartiene a  $F$ .



Il risultato  $\tilde{r}$  di una operazione f.p. dipende dall'algoritmo utilizzato dal sistema aritmetico f.p., cioè, in definitiva, dall'organizzazione e dal progetto (a livello hardware) dell'unità aritmetica. Si considererà ottimale, da un punto di vista dinamico, un sistema aritmetico f.p. per il quale si abbia

$$\tilde{r} = x \# y = fl(r) = fl(x \# y). \quad (1.8)$$

---

<sup>25</sup>Non ci si occuperà dell'operazione di conversione dalla rappresentazione in base  $\beta$  alla rappresentazione decimale e da quella decimale a quella in base  $\beta$ , cioè dalla rappresentazione interna a quella esterna e viceversa.

La (1.8) implica che il risultato  $\tilde{r}$  dell'operazione f.p., che è un numero macchina, è esattamente la rappresentazione f.p.n. a precisione finita del risultato  $r$  della corrispondente operazione in  $\Re$ . Un tale sistema aritmetico garantisce dunque che il risultato di qualsiasi operazione f.p. differisca dal risultato dell'operazione in  $\Re$  corrispondente (il risultato esatto), di una quantità che è il solo errore di rappresentazione di  $r$ , e viene perciò detto *sistema a massima accuratezza dinamica*.

Si consideri, ad esempio, l'addizione f.p.. La sua esecuzione consta essenzialmente di quattro fasi:

1. confronto tra gli esponenti della rappresentazione f.p.n. degli addendi per individuare l'addendo con esponente più piccolo;
2. "shift" delle cifre della mantissa di tale addendo in modo che il relativo esponente risulti uguale a quello dell'altro addendo;
3. addizione delle mantisse degli addendi;
4. normalizzazione ed arrotondamento del risultato.

Usualmente l'unità aritmetica memorizza gli operandi f.p., ed anche i dati intermedi generati durante l'esecuzione dell'operazione f.p., in locazioni dette registri (fisicamente all'interno dell'unità) con precisione  $t_{reg} > t$ . Quindi i passi 2 e 3 sono eseguiti avendo a disposizione per la mantissa un numero di cifre maggiore della precisione del sistema aritmetico considerato.

Se si suppone che  $\beta = 10$  e  $t = 4$  e che i passi 2 e 3 siano eseguiti facendo uso di un registro a doppia precisione, il calcolo di  $\tilde{c} = a \oplus b$ , con  $a = 0.9983 \times 10^2$  e  $b = 0.4652 \times 10^{-1}$ , avviene nel modo seguente:

1. confronto degli esponenti: ovviamente  $2 > -1$ ;
2. shift della mantissa di  $b$ :  $0.4652 \longrightarrow 0.00046520$ ;
3. somma delle mantisse:  $0.99830000 + 0.00046520 = 0.99876520$ ;
4. arrotondamento e normalizzazione del risultato:  $c = 0.9988 \times 10^2$ .

Si noti che durante il passo 4 sono state perse tre cifre significative del risultato e quindi  $c$  non è il risultato esatto dell'addizione di  $a$  e  $b$ , anche se  $a$  e  $b$  sono numeri macchina; tuttavia  $c = fl(0.99876520 \times 10^2)$  e dunque, in questo caso, si ha la massima accuratezza<sup>26</sup>.

Al contrario, se i registri hanno  $t_{reg} = t$  si ottiene:

1. confronto degli esponenti:  $2 > -1$ ;
2. shift della mantissa di  $b$ :  $0.4652 \longrightarrow 0.0004$ ;

---

<sup>26</sup>E' possibile dimostrare che  $t_{reg} = t + 3$ , insieme con accorgimenti detti *sticky bit* e *rounding bit*, è tale da garantire la massima accuratezza in qualsiasi operazione f.p. [11].

3. somma delle mantisse:  $0.9983 + 0.0004 = 0.9987$ ;
  4. arrotondamento e normalizzazione del risultato:  $c = 0.9987 \times 10^2$ ;
- quindi non si ha la massima accuratezza.

## 1.3 L'errore di roundoff

L'errore che si commette approssimando un numero reale  $x$  mediante  $fl(x)$  si dice *errore di roundoff*; più precisamente, si ha:

**Definizione 1.4. (Errore di roundoff)**

Considerato un sistema aritmetico f.p. a precisione finita, con  $F(\beta, t, emin, emax)$ , sia  $x$  un numero reale appartenente all'insieme di rappresentabilità di  $F$  e sia  $fl(x)$  la sua rappresentazione in  $F$ . Si dice errore assoluto di roundoff il numero

$$|fl(x) - x|,$$

e si dice errore relativo di roundoff il numero

$$\frac{|fl(x) - x|}{|x|}.$$

L'errore di roundoff si genera sia quando un numero reale, dato nella usuale rappresentazione decimale, viene rappresentato in  $F$ , cioè nel passaggio dalla rappresentazione esterna a quella interna (*errore di roundoff di rappresentazione*), sia nell'esecuzione di ogni operazione f.p. (*errore di roundoff delle operazioni f.p.*).

### 1.3.1 L'errore di roundoff di rappresentazione

Siano  $\beta = 10$  e  $t = 5$  e si indichino con  $fl_T(x) = m_T \times \beta^e$  e  $fl_A(x) = m_A \times \beta^e$  le rappresentazioni f.p.n. di un numero reale  $x$  ottenute, rispettivamente, per troncamento e per arrotondamento, e con  $\hat{m}$  la mantissa normalizzata di  $x$  in aritmetica ordinaria (cioè ad infinite cifre).<sup>27</sup> Si considerino i numeri riportati in Tabella 1.1.

E' immediato osservare che  $\hat{m}$  e  $m_T$  differiscono di una quantità inferiore a  $10^{-5}$ , mentre  $\hat{m}$  e  $m_A$  differiscono di una quantità non superiore a  $0.5 \times 10^{-5}$ ; l'errore relativo di roundoff è minore di  $10^{-4}$  nel primo caso e di  $0.5 \times 10^{-4}$  nel secondo (Tabella 1.2).

---

<sup>27</sup>Da ora in poi il termine mantissa è utilizzato sia per indicare la mantissa di un numero in rappresentazione f.p.n., sia per indicare la parte frazionaria del numero in tale rappresentazione.

$x = \hat{m} \times \beta^e$	$fl_T(x) = m_T \times \beta^e$	$fl_A(x) = m_A \times \beta^e$
$-0.115237 \times 10^{-1}$	$-0.11523 \times 10^{-1}$	$-0.11524 \times 10^{-1}$
$0.1111248 \times 10^4$	$0.11112 \times 10^4$	$0.11112 \times 10^4$
$0.5723378 \times 10^2$	$0.57233 \times 10^2$	$0.57234 \times 10^2$
$-0.461775 \times 10^1$	$-0.46177 \times 10^1$	$-0.46178 \times 10^1$

Tabella 1.1: Rappresentazione floating-point normalizzata: troncamento ed arrotondamento della mantissa

$x$	$ \hat{m} - m_T $	$ \hat{m} - m_A $	$ fl_T(x) - x / x $	$ fl_A(x) - x / x $
$-0.115237 \times 10^{-1}$	$0.70 \times 10^{-5}$	$0.30 \times 10^{-5}$	$0.61 \times 10^{-4}$	$0.26 \times 10^{-4}$
$0.1111248 \times 10^4$	$0.48 \times 10^{-5}$	$0.48 \times 10^{-5}$	$0.43 \times 10^{-4}$	$0.43 \times 10^{-4}$
$0.5723378 \times 10^2$	$0.78 \times 10^{-5}$	$0.22 \times 10^{-5}$	$0.14 \times 10^{-4}$	$0.38 \times 10^{-5}$
$-0.461775 \times 10^1$	$0.50 \times 10^{-5}$	$0.50 \times 10^{-5}$	$0.11 \times 10^{-4}$	$0.11 \times 10^{-4}$

Tabella 1.2: Rappresentazione floating-point normalizzata: errore di round-off assoluto e relativo

Si vuole ora stabilire in generale il massimo errore relativo che si commette approssimando un numero reale  $x \neq 0$  con  $fl(x)$  in un sistema aritmetico con  $F(\beta, t, emin, emax)$ .

Si consideri, ad esempio, il caso dell'arrotondamento; siano

$$\begin{aligned} x &= \hat{m} \times \beta^e, \\ fl(x) &= m_A \times \beta^e \end{aligned}$$

rispettivamente la rappresentazione f.p.n. di  $x$  (in aritmetica ordinaria) e la rappresentazione f.p.n. a precisione finita (in  $F$ ) corrispondente. Dunque

$$|\hat{m}| \geq \beta^{-1}$$

e  $\hat{m}$  e  $m_A$  differiscono per una quantità non superiore a  $0.5 \times \beta^{-t}$ , cioè

$$|\hat{m} - m_A| \leq \frac{1}{2}\beta^{-t}.$$

Si ha:

$$\frac{|fl(x) - x|}{|x|} = \frac{|m_A - \hat{m}|}{|\hat{m}|} \leq \frac{\frac{1}{2}\beta^{-t}}{\beta^{-1}} = \frac{1}{2}\beta^{1-t},$$

da cui risulta che  $\frac{1}{2}\beta^{1-t}$  è il *massimo errore relativo* che si commette approssimando  $x$  con  $fl(x)$ . Analogi risultati sussiste per il troncamento:

$$\frac{|fl(x) - x|}{|x|} \leq \beta^{1-t}.$$

### Definizione 1.5. (Massima accuratezza relativa)

*Si dice massima accuratezza relativa di un sistema aritmetico f.p. a precisione finita con  $F(\beta, t, emin, emax)$  il massimo errore che si commette nel rappresentare un numero  $x$  nel sistema  $F$*

$$\begin{aligned} u &= \max \frac{|fl(x) - x|}{|x|} = \frac{1}{2}\beta^{1-t}, \quad \text{nel caso dell'arrotondamento}. \\ u' &= \max \frac{|fl(x) - x|}{|x|} = \beta^{1-t}, \quad \text{nel caso del troncamento}. \end{aligned}$$

Come si vedrà in seguito, la massima accuratezza relativa è una delle costanti fondamentali di un sistema aritmetico f.p. a precisione finita.

♣ **Esempio 1.17.** Siano  $\beta = 10$  e  $t = 5$ . La massima accuratezza relativa è

$$\begin{aligned} u &= 0.5 \times 10^{-4} && (\text{arrotondamento}), \\ u' &= 10^{-4} && (\text{troncamento}). \end{aligned}$$



Si osservi che, posto  $\delta = (fl(x) - x)/x$ , sussiste la relazione:

$$fl(x) = x(1 + \delta), \quad |\delta| \leq u \quad (|\delta| \leq u'). \quad (1.9)$$

Da ora in poi si prenderà in considerazione solo l'arrotondamento, perché è lo schema usato nella maggior parte degli calcolatori.

### 1.3.2 L'errore di roundoff delle operazioni floating-point

In un sistema aritmetico f.p. a precisione finita si introduce un errore nel risultato di ogni operazione aritmetica. Come sarà chiarito in seguito, anche la risoluzione di un problema apparentemente semplice, quale ad esempio un'equazione algebrica di secondo grado, può risultare tutt'altro che banale, in quanto nel progetto di un algoritmo è essenziale tenere conto della presenza dell'errore di roundoff, che altrimenti può condurre ad una soluzione completamente errata.

Si supponga che il sistema aritmetico garantisca la massima accuratezza dinamica; allora, indicata con  $\#$  una qualsiasi delle quattro operazioni elementari e con  $\#\#$  la corrispondente operazione f.p. , se  $a$  e  $b$  sono due numeri macchina si ha:

$$a \#\# b = fl(a\#b) = (a\#b)(1 + \delta), \quad (1.10)$$

con  $|\delta| \leq u$ .

♣ **Esempio 1.18.** Si considerino  $\beta = 10$  e  $t = 4$  e si sommino i numeri  $a = 0.5496 \times 10^2$ ,  $b = 0.8714 \times 10^1$  e  $c = 0.1493 \times 10^{-1}$ .

Eseguendo le addizioni secondo l'ordine  $(a \oplus b) \oplus c$ , con un registro a doppia precisione, si ha:

$$\begin{aligned} & (0.5496 \times 10^2 \oplus 0.8714 \times 10^1) \oplus 0.1493 \times 10^{-1} = \\ & = 0.6367 \times 10^2 \oplus 0.1493 \times 10^{-1} = 0.6368 \times 10^2, \end{aligned}$$

calcolando invece  $a \oplus (b \oplus c)$  si ha:

$$\begin{aligned} & 0.5496 \times 10^2 \oplus (0.8714 \times 10^1 \oplus 0.1493 \times 10^{-1}) = \\ & = 0.5496 \times 10^2 \oplus 0.8729 \times 10^1 = 0.6369 \times 10^2. \end{aligned}$$

I risultati ottenuti nei due casi differiscono di 1 unità sull'ultima cifra significativa e quindi non vale la proprietà associativa dell'addizione.



♣ **Esempio 1.19.** Siano  $\beta = 10$  e  $t = 4$  e si considerino i numeri  $a = 0.2240 \times 10^2$ ,  $b = 0.7953 \times 10^1$  e  $c = 0.3329 \times 10^2$ .

Utilizzando un registro a doppia precisione, si ha:

$$\begin{aligned} (a \oplus b) \otimes c &= \\ &= (0.2240 \times 10^2 \oplus 0.7953 \times 10^1) \otimes 0.3329 \times 10^2 = \\ &= 0.3035 \times 10^2 \otimes 0.3329 \times 10^2 = 0.1010 \times 10^4 \end{aligned}$$

e

$$\begin{aligned} (a \otimes c) \oplus (b \otimes c) &= \\ &= (0.2240 \times 10^2 \otimes 0.3329 \times 10^2) \oplus (0.7953 \times 10^1 \otimes 0.3329 \times 10^2) = \\ &= 0.7457 \times 10^3 \oplus 0.2648 \times 10^3 = 0.1011 \times 10^4. \end{aligned}$$

Le due sequenze di operazioni non conducono allo stesso risultato e quindi non è valida la proprietà distributiva della moltiplicazione rispetto all'addizione.



## 1.4 L'epsilon macchina

Supponiamo di eseguire, in un sistema aritmetico f.p. con  $\beta = 10$  e  $t = 3$ , dotato di massima accuratezza dinamica, l'addizione:

$$0.994 \times 10^0 \oplus 0.177 \times 10^{-3} = 0.994 \times 10^0 \oplus 0.000177 \times 10^0 = 0.994 \times 10^0.$$

In generale, quando si effettua un'addizione f.p. tra numeri aventi ordini di grandezza differenti può accadere che il risultato sia uguale all'addendo maggiore (in valore assoluto). Questo fenomeno è dovuto alla precisione finita del sistema aritmetico utilizzato ed esiste anche se il sistema garantisce la massima accuratezza dinamica.

In particolare, è interessante studiare il caso in cui l'addendo maggiore in valore assoluto è il numero 1.

**Definizione 1.6. (Epsilon macchina)**

*In un sistema aritmetico f.p. con  $F(\beta, t, emin, emax)$ , si dice epsilon macchina il più piccolo numero  $\epsilon \in F$  tale che*

$$1 \oplus \epsilon = fl(1 + \epsilon) > 1. \quad (1.11)$$

Si supponga, ad esempio,  $\beta = 10$  e  $t = 3$  e si calcoli il più grande numero  $\eta \in F$  tale che<sup>28</sup>

$$1 \oplus \eta = 1. \quad (1.12)$$

---

<sup>28</sup>E' facile convincersi che la limitatezza del sistema f.p. a precisione finita implica l'esistenza di elementi diversi dallo zero che si comportano come lo zero nella somma f.p. con 1. In realtà esiste un insieme di numeri macchina che godono di questa proprietà.

Chiaramente  $0 < \eta = 0.\alpha_1\alpha_2\alpha_3 \times 10^p$ , con  $p < 0$ . Si supponga inoltre che l'addizione f.p. sia eseguita con un registro a doppia precisione.

Affinché sia valida (1.12), l'operazione  $1 \oplus \eta$  deve essere effettuata come segue:

$$\begin{array}{r} 0. \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \times \quad 10^1 \quad + \\ 0. \quad 0 \quad 0 \quad 0 \quad \alpha_1 \quad \alpha_2 \quad \alpha_3 \quad \times \quad 10^1 \quad = \\ \hline 0. \quad 1 \quad 0 \quad 0 \quad \alpha_1 \quad \alpha_2 \quad \alpha_3 \quad \times \quad 10^1 \end{array} \quad (1.13)$$

e la terza cifra significativa del risultato non deve essere modificata quando si effettua l'arrotondamento a 3 cifre. Quest'ultima condizione equivale a imporre che  $\alpha_1 < 5$ , mentre da (1.13), avendo effettuato uno shift della mantissa di  $\eta$  di tre cifre verso destra, si ricava

$$p + 3 = 1, \quad \text{da cui} \quad p = 1 - 3 = -2.$$

Pertanto  $\eta = 0.499 \times 10^{-2}$  e quindi  $\epsilon = 0.500 \times 10^{-2}$ .

In generale, in un sistema aritmetico f.p. con  $F(\beta, t, emin, emax)$ , si ha:

$$\epsilon = \frac{1}{2}\beta^{1-t},$$

nel caso dell'arrotondamento, mentre,

$$\epsilon = \beta^{1-t}$$

nel caso del troncamento.

L'epsilon macchina coincide con la massima accuratezza relativa  $u$ . Pertanto, nel seguito, i termini epsilon macchina e massima accuratezza relativa saranno considerati sinonimi.

Nota la base di numerazione, conoscere l'epsilon macchina equivale a conoscere la precisione del sistema aritmetico; d'altra parte la definizione stessa di epsilon macchina suggerisce un modo per calcolare tale quantità nel sistema aritmetico di un calcolatore, quando non è nota la precisione. L'algoritmo per il calcolo dell'epsilon macchina è infatti basato sulla definizione 1.6; il valore finale di  $u$  è, appunto, l'epsilon macchina.

```

procedure precrel(out: u)

    /# SCOPO: calcolo dell'epsilon macchina

    /# SPECIFICHE PARAMETRI:
        var: u      : reale   {epsilon macchina}

    /# VARIABILI LOCALI:
        var: u1     : reale   {variabile di appoggio}
        var: u2     : reale   {variabile di appoggio}

    /# INIZIO ISTRUZIONI:
        u1 := 1.;

        repeat
            u := u1;
            u1 := u1/ $\beta$ ;
            u2 := u1 + 1.;

            until(u2 = 1.)

    end precrel

```

Procedura 1.1: Algoritmo per il calcolo dell'epsilon macchina

Implementando l'algoritmo in Fortran su un processore Pentium IV<sup>29</sup> in singola ed

---

<sup>29</sup>Da questo momento in poi, a meno che non specificato diversamente, tutti gli esperimenti numerici saranno eseguiti in Fortran 77, utilizzando la singola o la doppia precisione, e faranno riferimento ad processore Pentium IV, con le seguenti caratteristiche hardware:

- Aritmetica Standard IEEE-754
- 1500 MHz clock
- 256 KB cache
- 512 MB RAM memory
- 40 GB disk

in doppia precisione, si ha rispettivamente:<sup>30</sup>

$$\begin{aligned} u &= 2^{-23} \simeq 0.1192093 \times 10^{-6}, \\ u &= 2^{-52} \simeq 0.2220446 \times 10^{-15}. \end{aligned} \quad (1.14)$$

Con un ragionamento analogo al precedente si verifica che, se  $x, y \in F$  e  $x = m \times \beta^p$ , risulta:

$$x \oplus y \neq x \iff |y| \geq \epsilon_x = \frac{1}{2}\beta^{p-t},$$

cioè, se  $|y| < \epsilon_x$ ,  $y$  non dà contributo alla somma. Quindi, dato un qualunque numero macchina  $x$ , esiste un insieme di numeri macchina che si comportano come lo zero nell'addizione f.p. con  $x$ . Questa osservazione è di importanza fondamentale nel progetto di algoritmi numerici.

Si noti che

$$\epsilon_x = \frac{1}{2}\beta^{p-t} = \beta^{p-1} \times \frac{1}{2}\beta^{1-t} = \beta^{p-1} \times u \leq |x| \times u;$$

in generale, quindi, si assume

$$\epsilon_x = |x| \times u,$$

in quanto tale numero si calcola immediatamente a partire da  $x$  e da  $u$ .

Si supponga di voler calcolare  $e^x$ , con  $x > 0$ . Questo problema matematico può essere approssimato dal problema numerico “calcolo di una somma” del tipo:

$$S_N = \sum_{n=0}^N \frac{x^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \dots + \frac{x^N}{N!}. \quad (1.15)$$

$S_N$  è un'approssimazione di  $e^x$ , la cui “bontà” dipende da  $N$  ed è tanto migliore quanto maggiore è  $N$ <sup>31</sup>.

Da un punto di vista algoritmico il calcolo di  $S_N$  è basato sullo schema:

$$\begin{aligned} S_0 &= 1 \\ S_n &= S_{n-1} + x^n/n!, \quad n = 1, 2, \dots, N. \end{aligned} \quad (1.16)$$

Una formula del tipo della (1.16) sarà detta *formula ricorrente* e la computazione basata su di essa sarà detta *processo iterativo*.

L'aspetto centrale dal punto di vista numerico consiste nella determinazione del numero  $N$  degli addendi da sommare. In Tabella 1.3 sono riportati i valori di  $x^n/n!$  e di  $S_n$  per  $x = 5$  e  $n = 1, 2, \dots, 23$ , calcolati utilizzando 7 cifre decimali significative.<sup>32</sup>

---

<sup>30</sup>E' interessante notare che l'implementazione diretta dell'algoritmo descritto nella Procedura 1.1, può portare a risultati diversi sullo stesso calcolatore se si usano compilatori diversi. Alcuni compilatori (ad es. Microsoft Fortran 6.0) ottimizzano infatti il programma eseguibile mantenendo i valori di  $u1$  ed  $u2$  sempre nei registri. In tal caso l'algoritmo deve essere implementato “costringendo” il compilatore a conservare  $u2$  in memoria centrale; si vuole calcolare l'epsilon macchina relativo alla precisione delle locazioni di memoria, non a quella dei registri!

<sup>31</sup>In questo caso si ha  $\lim_{N \rightarrow \infty} S_N = e^x$

<sup>32</sup>Implementazione utilizzando la singola precisione.

$n$	$x^n/n!$	$S_n$
0	$0.1000000 \times 10^1$	$0.1000000 \times 10^1$
1	$0.5000000 \times 10^1$	$0.6000000 \times 10^1$
2	$0.1250000 \times 10^2$	$0.1850000 \times 10^2$
3	$0.2083333 \times 10^2$	$0.3933334 \times 10^2$
4	$0.2604167 \times 10^2$	$0.6537500 \times 10^2$
5	$0.2604166 \times 10^2$	$0.9141666 \times 10^2$
6	$0.2170139 \times 10^2$	$0.1131180 \times 10^3$
7	$0.1550099 \times 10^2$	$0.1286190 \times 10^3$
8	$0.9688119 \times 10^1$	$0.1383072 \times 10^3$
9	$0.5382288 \times 10^1$	$0.1436895 \times 10^3$
:	:	:
18	$0.5958255 \times 10^{-3}$	$0.1484129 \times 10^3$
19	$0.1567962 \times 10^{-3}$	$0.1484131 \times 10^3$
20	$0.3919905 \times 10^{-4}$	$0.1484132 \times 10^3$
21	$0.9333107 \times 10^{-5}$	$0.1484132 \times 10^3$
22	$0.2121161 \times 10^{-5}$	$0.1484132 \times 10^3$
23	$0.4611219 \times 10^{-6}$	$0.1484132 \times 10^3$

**Tabella 1.3:** Valori di  $x^n/n!$  e di  $S_n$ 

Si noti che a partire da  $n = 21$ ,  $S_n$  non varia, ovvero il termine  $x^n/n!$  non dà contributo alla somma  $S_{n-1} + x^n/n!$ ; i calcoli effettuati a partire da  $n = 21$  sono dunque inutili.

Sulla base dell'osservazione precedente, si può affermare che piuttosto che fissare a priori  $N$ , è più conveniente che sia l'algoritmo stesso a determinare dinamicamente, cioè nel corso della sua esecuzione, il valore più opportuno di  $N$ . Nel caso non sia nota alcuna informazione sull'errore di troncamento analitico, la scelta più naturale è che l'algoritmo continui il processo iterativo (1.16) arrestandosi al più piccolo valore  $\bar{n}$  per cui si abbia

$$\frac{|x|^{\bar{n}}}{\bar{n}!} < \epsilon_{S_{\bar{n}-1}} = |S_{\bar{n}-1}| \times u, \quad (1.17)$$

cioè quando l'addendo  $x^{\bar{n}}/\bar{n}!$  si comporta come lo zero nella somma f.p. con  $S_{\bar{n}-1}$ . Inoltre la (1.17) vale per  $n > \bar{n}$  e quindi se  $n > \bar{n}$  il processo iterativo non modifica alcuna cifra significativa della soluzione calcolata dall'algoritmo. L'algoritmo ha dunque determinato dinamicamente il miglior valore di  $N$  ( $N = \bar{n}$ ). La (1.17) rappresenta un esempio di condizione che, se verificata, comporta la fine del processo iterativo (cfr. Procedura 1.2).

Si dirà *criterio di arresto* di un algoritmo basato su un processo iterativo l'insieme delle condizioni che, se verificate, determinano l'arresto del processo iterativo. L'individua-

zione di un criterio di arresto efficiente (cioè che non comporti iterazioni inutili) ed affidabile (cioè che non arresti prematuramente le iterazioni) è uno degli aspetti principali del progetto degli algoritmi basati su processi iterativi.

L' algoritmo descritto nella Procedura 1.2 usa il criterio di arresto precedente, che si dirà *criterio di arresto naturale*:

```

procedure espo(in:  $x, u$  ; out:  $sum$ )
  /# SCOPO: calcolo del valore della funzione esponenziale
  /# SPECIFICHE PARAMETRI:
  /# PARAMETRI DI INPUT:
    var:  $x$  : reale {argomento della funzione}
          {esponenziale}
    var:  $u$  : reale {epsilon macchina}

  /# PARAMETRI DI OUTPUT:
    var:  $sum$  : reale {valore assunto dalla funzione}
          {esponenziale in  $x$ }

  /# VARIABILI LOCALI:
    var:  $n$  : intero {numero di termini della serie}
    var:  $add$  : reale {termine generale della serie}

  /# INIZIO ISTRUZIONI:
     $sum := 0.;$ 
     $n := 0;$ 
     $add := 1.;$ 
    repeat
       $sum := sum + add;$ 
       $n := n + 1;$ 
       $add := add \times (x/n);$  {calcolo di  $x^n/n!$ }
    until( $|add| < |sum| \times u$ ) {criterio di arresto naturale}
  end espo

```

### Procedura 1.2: Algoritmo per il calcolo di $e^x$

Le considerazioni precedenti si applicano in generale ad un processo iterativo basato su una formula ricorrente del tipo

$$S_{n+1} = S_n + a_n. \quad (1.18)$$

Il criterio di arresto naturale è

$$|a_n| < \epsilon_{S_n},$$

dove

$$\epsilon_{S_n} = |S_n| \times u.$$

E' opportuno notare che anche il calcolo di  $\epsilon_{S_n}$  è sottoposto alle limitazioni del sistema aritmetico f.p. a precisione finita.

♣ **Esempio 1.20.** Si consideri un sistema aritmetico f.p. con  $F(10, 7, -9, 9)$  e si calcolino i valori di  $\epsilon_{S_n}$  in relazione ai numeri  $y = -0.1238452 \times 10^2$  e  $z = 0.2356112 \times 10^{-8}$ . Si ricordi che l'epsilon macchina ed il minimo numero reale positivo rappresentabile in  $F$  sono, rispettivamente,  $u = 0.5 \times 10^{-6}$  e  $rmin = 0.1 \times 10^{-9}$ . Si ha:

$$\begin{aligned}\epsilon_y &= (0.1238452 \times 10^2) \times (0.5 \times 10^{-6}) = 0.6192260 \times 10^{-5}, \\ \epsilon_z &= (0.2356112 \times 10^{-8}) \times (0.5 \times 10^{-6}) = 0.1178056 \times 10^{-14} < rmin\end{aligned}$$

e quindi il calcolo di  $fl(\epsilon_z)$  provoca underflow. ♣

L'algoritmo descritto nella Procedura 1.3 calcola  $\epsilon_x$  per un valore generico  $x \in F$  in modo accurato.

Inoltre, l'algoritmo richiede che sia noto il valore di  $rmin$ , che può essere calcolato con l'algoritmo descritto nella Procedura 1.4. <sup>33</sup>

Si supponga di voler utilizzare la (1.16) per calcolare un'approssimazione di  $e^x$  con  $p$  cifre decimali corrette. Bisogna dunque arrestare il procedimento iterativo quando

$$E = |e^x - S_n| < 10^{-p} = tol.$$

Il valore di  $E$  non è noto e bisogna quindi calcolarne una stima.<sup>34</sup> A tal fine si ricordi che

$$e^x - S_n = R_n(x),$$

e che al divergere di  $n$  il resto della formula di Taylor è infinitesimo

$$\lim_{x \rightarrow 0} \frac{R_n(x)}{x^n} = 0;$$

---

<sup>33</sup>Tale algoritmo si basa sull'ipotesi, generalmente verificata, che, in caso di underflow, ad  $r$  venga assegnato il valore 0.

<sup>34</sup>Una stima di  $E$  è fornita dalla (1.5) (cfr. esempio 1.6):

$$E = |e^x - S_n| = |R_n(x)| \leq e^a \frac{|x|^{n+1}}{(n+1)!},$$

con  $a$  opportunamente scelto. Come già osservato nell'esempio 1.6, il calcolo effettivo di tale stima richiede però la determinazione di  $a$ , o meglio di un valore di  $a$  per cui  $e^a|x|^{n+1}/(n+1)!$  non sia "eccessivamente grande". Tale problema non è di facile risoluzione e quindi in generale non si ricorre alla stima suddetta.

```

procedure epsrel(in:  $x, rmin, u$  ; out:  $\epsilon_x$ )
  /# SCOPO: calcolo dell'epsilon macchina relativo a x
  /# SPECIFICHE PARAMETRI:
  /# PARAMETRI DI INPUT:
    var:  $x$  : reale {numero macchina di cui calcolare}
           {l'epsilon macchina relativo}
    var:  $rmin$  : reale {minimo numero reale positivo}
           {rappresentabile}
    var:  $u$  : reale {epsilon macchina}

  /# PARAMETRI DI OUTPUT:
  var:  $\epsilon_x$  : reale {epsilon macchina relativo a x}

  /# INIZIO ISTRUZIONI:
  if  $|x| > rmin/u$  then
     $\epsilon_x := |x| \times u;$ 
  else
     $\epsilon_x := rmin;$ 
  endif
end epsrel

```

Procedura 1.3: Algoritmo per il calcolo di  $\epsilon_x$ 

concludendo, a meno del fattore costante  $e^a$  (o  $e^\xi$  con  $\xi \in [-a, a]$ ), vale l'approssimazione:

$$E = |e^x - S_n| \simeq \frac{|x|^{n+1}}{(n+1)!} = |S_{n+1} - S_n|.$$

La quantità

$$|S_{n+1} - S_n|$$

si assume pertanto come stima dell'errore assoluto  $E$  al passo  $n$  e si utilizza il criterio d'arresto

$$|S_{n+1} - S_n| < tol. \quad (1.19)$$

Per quanto detto sul criterio d'arresto naturale, non ha senso richiedere che

$$tol < \varepsilon_{S_n} = |S_n| \times u. \quad (1.20)$$

```

procedure rmin(in:  $\beta$  ; out:  $rmin$ )
    /# SCOPO: calcolo del minimo numero reale positivo
        rappresentabile

    /# SPECIFICHE PARAMETRI:
    /# PARAMETRI DI INPUT:
        var:  $\beta$       : intero   {base del sistema aritmetico}

    /# PARAMETRI DI OUTPUT:
        var:  $rmin$   : reale   {minimo num. reale positivo rappr.}

    /# VARIABILI LOCALI:
        var:  $r$       : reale   {variabile di appoggio}

    /# INIZIO ISTRUZIONI:
         $r := 1.;$ 
        repeat
             $rmin := r;$ 
             $r := r/\beta;$ 
        until ( $r = 0.$ )
        end rmin

```

Procedura 1.4: Algoritmo per il calcolo di  $rmin$

Analogamente, se si vuole calcolare un'approssimazione di  $e^x$  con  $p$  cifre significative corrette, si utilizza il criterio d'arresto

$$\frac{|S_{n+1} - S_n|}{|S_n|} < 10^{-p} = tol \quad (1.21)$$

e in questo caso non ha senso richiedere che

$$tol < u. \quad (1.22)$$

In Tabella 1.4 sono riportati i valori di  $e^5$ , calcolati utilizzando la Procedura 1.2 con il criterio d'arresto (1.21)<sup>35</sup>, per valori di  $tol$  differenti, e sono indicati gli errori relativi,

---

<sup>35</sup>Implementazione utilizzando la singola precisione.

$e^5$	$tol$	$E'$
$0.1483096 \times 10^3$	$10^{-3}$	$0.70 \times 10^{-3}$
$0.1484029 \times 10^3$	$10^{-4}$	$0.69 \times 10^{-4}$
$0.1484124 \times 10^3$	$10^{-5}$	$0.51 \times 10^{-5}$
$0.1484131 \times 10^3$	$10^{-6}$	$0.40 \times 10^{-6}$
$0.1484132 \times 10^3$	$10^{-7}$	$0.26 \times 10^{-6}$
$0.1484132 \times 10^3$	$10^{-8}$	$0.26 \times 10^{-6}$
$0.1484132 \times 10^3$	$10^{-9}$	$0.26 \times 10^{-6}$
$0.1484132 \times 10^3$	$10^{-10}$	$0.26 \times 10^{-6}$
$0.1484132 \times 10^3$	$10^{-20}$	$0.26 \times 10^{-6}$
$0.1484132 \times 10^3$	$10^{-30}$	$0.26 \times 10^{-6}$

**Tabella 1.4:** Valori di  $e^5$  ottenuti utilizzando diversi valori di tolleranza

$E'$ , corrispondenti. Si osservi che per  $10^{-6} \leq tol \leq 10^{-3}$  l'errore relativo è dell'ordine di  $tol$ , mentre per  $tol \leq 10^{-7}$  tale errore è sempre  $0.26 \times 10^{-6}$ , in accordo col valore di  $u$  riportato in (1.14).<sup>36</sup>

Pertanto, i criteri d'arresto (1.19) e (1.21) devono essere modificati, per evitare che sia richiesta un'accuratezza che non può essere ottenuta:

$$\begin{aligned}|S_{n+1} - S_n| &< tol + |S_n| \times u, \\ \frac{|S_{n+1} - S_n|}{|S_n|} &< tol + u.\end{aligned}$$

Quanto detto sul criterio di arresto naturale per il calcolo di  $e^x$  si estende in generale al progetto dei criteri di arresto di algoritmi basati su processi iterativi, che approssimano un valore  $x$  come limite di una successione  $\{x_n\}_{n \in \mathcal{N}}$ . In tal caso, si assumono come stime dell'errore assoluto e dell'errore relativo rispettivamente le quantità

$$|x_{n+1} - x_n|$$

e

$$\frac{|x_{n+1} - x_n|}{|x_n|}$$

e si utilizzano i criteri d'arresto<sup>37</sup>

$$|x_{n+1} - x_n| < tol + |x_n| \times u \quad (1.23)$$

<sup>36</sup>Gli errori relativi sono stati calcolati utilizzando un'approssimazione di  $e^5$  con 9 cifre significative corrette ( $e^5 \simeq 0.148413159 \times 10^3$ ).

<sup>37</sup>Si osservi che i criteri d'arresto (1.23) e (1.24) garantiscono che il processo iterativo si arresti quando due approssimazioni successive sono uguali a meno di una tolleranza prefissata, ma non assicurano che l'errore sia minore di tale tolleranza. Per tale motivo essi generalmente si utilizzano in combinazione con

e

$$\frac{|x_{n+1} - x_n|}{|x_n|} < tol + u. \quad (1.24)$$

## 1.5 Il condizionamento di un problema matematico

Nella risoluzione computazionale di un problema, i dati sono affetti dall'errore di roundoff. In generale, essi sono affetti anche da errori sperimentali, cioè dovuti ai procedimenti di misurazione utilizzati per la loro rilevazione; tali errori, inevitabili, non saranno esplicitamente considerati, in quanto assimilabili agli errori di roundoff.

In base a quanto detto, ci si augura che, in un problema matematico, un errore relativo (assoluto) nei dati di un certo ordine di grandezza si traduca in un errore relativo (assoluto) dello stesso ordine nella soluzione.

### Definizione 1.7. (Buon condizionamento di un problema)

*Un problema si dice ben condizionato se l'errore relativo (assoluto) nella soluzione ha al più lo stesso ordine di grandezza dell'errore relativo (assoluto) nei dati.*

In accordo con la definizione precedente, un problema per il quale l'errore relativo nella soluzione ha ordine di grandezza maggiore rispetto all'errore relativo nei dati si dice *mal condizionato*.

♣ **Esempio 1.21.** Il sistema di equazioni lineari:

$$\begin{cases} 2.1 & x + 3.5 & y = 8 \\ 4.19 & x + 7 & y = 15 \end{cases} \quad (1.25)$$

ammette come soluzione  $(100, -57.714\dots)$ . La risoluzione di (1.25) è un problema i cui dati sono i coefficienti della matrice del sistema ed il vettore dei termini noti.

Si consideri ora il sistema:

$$\begin{cases} 2.1 & x + 3.5 & y = 8 \\ 4.192 & x + 7 & y = 15 \end{cases} \quad (1.26)$$

che ammette come soluzione  $(125, -72.714\dots)$ . Il sistema (1.26) può essere visto come una perturbazione del sistema (1.25) (nel coefficiente di  $x$  nella seconda equazione è stato introdotto un errore relativo dell'ordine di  $10^{-3}$ ) e la risoluzione di (1.26) come la risoluzione di un problema perturbato rispetto al problema di riferimento (la risoluzione di (1.25)). La perturbazione introdotta, corrispondente, da un punto di vista geometrico, ad una modifica nell'inclinazione della seconda delle due rette rappresentate

---

altri criteri, che tengono conto, ad esempio, di caratteristiche particolari del processo iterativo in esame. Si osservi inoltre che se  $x_n = 0$  i criteri suddetti risultano inadeguati (il criterio (1.24) è addirittura inutilizzabile!). Un modo per superare tale problema può essere quello di utilizzare un criterio d'arresto del tipo

$$\frac{|x_{n+1} - x_n|}{|x_n| + 1} < tol + u.$$

dalle equazioni, comporta un'amplificazione dell'errore notevole, dovuta al fatto che le due rette sono "quasi parallele" e quindi una lieve perturbazione del coefficiente angolare di una delle due provoca uno spostamento del punto di intersezione. Tale situazione è illustrata in Figura 1.5.

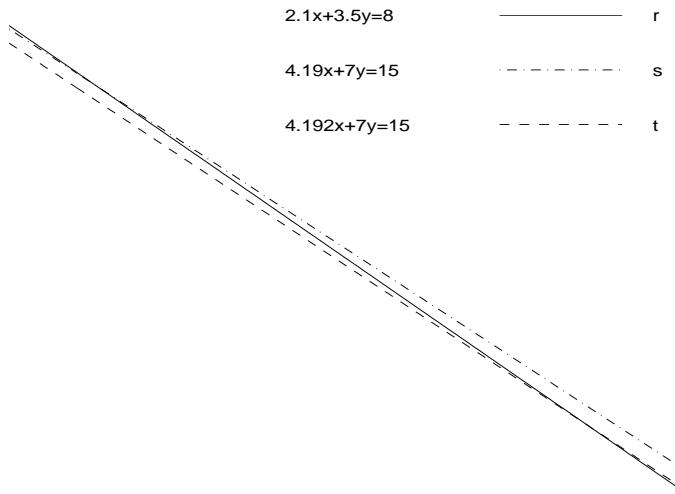


Figura 1.5: Interpretazione geometrica del condizionamento: mal condizionamento di un sistema lineare.

♣ **Esempio 1.22.** Consideriamo ora il sistema di equazioni lineari:

$$\begin{cases} -3.5 & x + 2.1 & y = 15 \\ 2.1 & x + 3.5 & y = 8 \end{cases} \quad (1.27)$$

che ammette come soluzione  $(-0.6723, 2.6891)$ , ed il sistema:

$$\begin{cases} -3.502 & x + 2.1 & y = 15 \\ 2.1 & x + 3.5 & y = 8 \end{cases} \quad (1.28)$$

che ammette come soluzione  $(-0.6720, 2.6889)$ . Come nell'esempio precedente, il sistema (1.28) può essere visto come una perturbazione del sistema (1.27) (anche in questo esempio, nel coefficiente di  $x$  nella seconda equazione è stato introdotto un errore relativo dell'ordine di  $10^{-3}$ ) e la risoluzione di (1.28) come la risoluzione di un problema perturbato rispetto al problema di riferimento (la risoluzione di (1.27)). La perturbazione introdotta non comporta questa volta un'amplificazione dell'errore, dovuta al fatto che in questo caso le due rette sono "quasi ortogonali"<sup>38</sup> e quindi una lieve perturbazione del coefficiente angolare di una delle due non provoca uno spostamento significativo del punto di intersezione. Tale situazione è illustrata in Figura 1.6.



<sup>38</sup>Per due rette ortogonali di coefficiente angolare, rispettivamente,  $m$  e  $m'$  sussiste la relazione:  $m \cdot m' = -1$ .

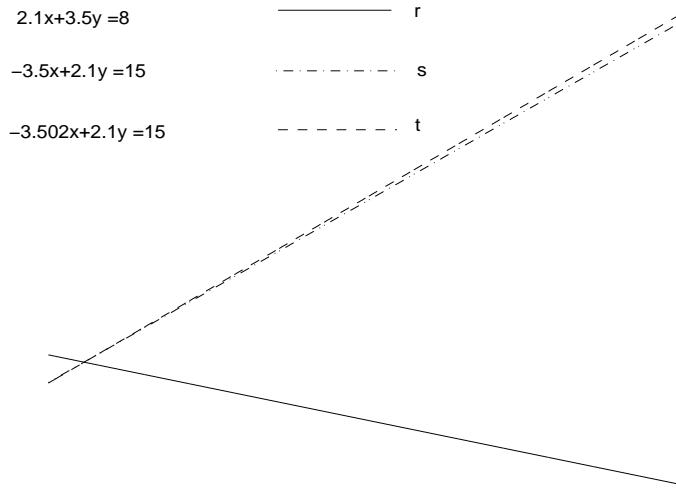


Figura 1.6: Interpretazione geometrica del condizionamento: ben condizionamento di un sistema lineare.

♣ **Esempio 1.23.** Si consideri l'equazione:

$$(x - 2)^4 = 0, \quad (1.29)$$

che ha quattro radici coincidenti uguali a 2. Si consideri poi l'equazione:

$$(x - 2)^4 = 10^{-8}, \quad (1.30)$$

che può essere vista come una perturbazione della precedente, in quanto è ottenuta sommando  $10^{-8}$  al secondo membro di (1.29); è immediato osservare che (1.30) ha due soluzioni reali (1.99 e 2.01) e due soluzioni complesse ( $2 - 0.01i$  e  $2 + 0.01i$ ). La perturbazione introdotta ha quindi modificato il campo di appartenenza delle soluzioni ed ha introdotto nelle radici reali una perturbazione di un ordine di grandezza 4 volte superiore.



♣ **Esempio 1.24.** Si consideri il problema del calcolo della differenza tra due numeri. Siano  $x = 12345678.0$  e  $y = 12345677.0$  e sia  $z$  la differenza

$$z = x - y = 1.00000000;$$

si consideri ora il problema, che si ottiene perturbando il precedente,

$$z' = x' - y', \quad x' = 12345678.1, \quad y' = 12345676.9,$$

la cui soluzione è

$$z' = 1.20000000.$$

Si osservi che una perturbazione nella nona cifra significativa dei dati ha provocato una perturbazione nella seconda cifra significativa della soluzione, dunque l'errore relativo nei dati

$$\frac{|x - x'|}{|x|} = 0.81 \times 10^{-8}, \quad \frac{|y - y'|}{|y|} = 0.81 \times 10^{-8}$$

si è amplificato in

$$\frac{|z - z'|}{|z|} = 0.2.$$

Se si considera il numero delle cifre significative corrette degli addendi e del risultato, si ha che si passa da otto cifre, in  $x'$  e  $y'$ , ad una cifra, in  $z'$ , cioè sono “cancellate”<sup>39</sup> sette cifre significative. Tale fenomeno è detto, appunto, *cancellazione* ed una giustificazione di esso è data nell'esempio 1.27.



Dagli esempi precedenti si deduce che nei problemi mal condizionati “piccoli errori” nei dati possono risultare amplificati nella soluzione, rendendola talvolta inaccettabile.

Detto  $\delta$  l'errore nei dati e  $\sigma$  l'errore corrispondente nella soluzione, e posto

$$\sigma = \mu \cdot \delta,$$

$\mu$  è detto *indice di condizionamento* del problema; risulta, quindi, che:

se  $\mu \leq 1$  il problema è *ben condizionato*;  
se  $\mu > 1$  il problema è *mal condizionato*<sup>40</sup>.

Si ricordi che, essendo interessati alla risoluzione di problemi mediante calcolatore, *l'analisi del condizionamento deve essere fatta sempre*, perché i dati sono affetti almeno dall'errore di roundoff. Ad esempio, il problema (1.30) può essere visto come una versione di (1.29) perturbata dal solo errore di roundoff nel secondo membro.

È quindi importante riuscire a dare una stima quantitativa dell'indice di condizionamento.

Si supponga di valutare una funzione  $f$ , reale di variabile reale, in un punto  $x$ . In generale, nell'approccio computazionale,  $f$  non sarà valutata in  $x$ , ma in  $x + \Delta x$  a causa dell'errore di roundoff. L'errore  $\Delta x$  induce quindi un errore nel risultato. Si noti che la

---

<sup>39</sup>La cancellazione delle cifre significative è dovuta al fatto che nell'effettuare la sottrazione in corrispondenza delle cifre uguali (rispettivamente in  $x$  e in  $y$ ) si ottiene una cifra nulla. A seguito della normalizzazione le cifre nulle vengono traslate verso sinistra e pertanto corrispondono a posizioni che vengono perse.

<sup>40</sup>La maggior parte dei problemi dell'Analisi Numerica risulta, più o meno, mal condizionata ed è sempre  $\delta > 0$ .

funzione  $f$  rappresenta il problema matematico,  $x$  i suoi dati ( $\Delta x$  è la loro perturbazione assoluta) e  $f(x)$  la soluzione. Posto

$$\Delta f = f(x + \Delta x) - f(x), \quad (1.31)$$

se  $f$  è derivabile in  $x$ , si ha:

$$\Delta f = f'(x)\Delta x + R(\Delta x), \quad \lim_{\Delta x \rightarrow 0} \frac{R(\Delta x)}{\Delta x} = 0$$

e quindi

$$|\Delta f| \simeq |f'(x)\Delta x|;$$

$|f'(x)|$  è, allora, il fattore di amplificazione nella soluzione dell'errore assoluto presente nei dati ed è, quindi, l'*indice di condizionamento assoluto*. In molti casi è più interessante considerare l'errore relativo, per il quale vale la relazione

$$\left| \frac{\Delta f}{f} \right| \simeq \left| \frac{f'(x)x}{f(x)} \right| \cdot \left| \frac{\Delta x}{x} \right|. \quad (1.32)$$

La quantità

$$C(f, x) = |f'(x)x|/|f(x)| \quad (1.33)$$

è il fattore di amplificazione nella soluzione dell'errore relativo nei dati ed è, quindi, l'*indice di condizionamento relativo*.  $C(f, x)$  quantifica appunto la sensibilità del problema agli errori nei dati.

E' utile ricordare la seguente regola operativa facilmente deducibile dalla (1.32): se i dati del problema hanno  $p$  cifre significative corrette ( $p \leq t$ , con  $t$  precisione del sistema aritmetico f.p.), allora un indice di condizionamento relativo  $C = 10^q$  indica che la soluzione non può essere ottenuta con più di  $p - q$  cifre significative corrette. Si può concludere allora che un problema mal condizionato è completamente intrattabile se  $q \simeq p$ ; altrimenti esso è trattabile a patto che si sia disposti a sopportare una perdita di  $q$  cifre significative nella soluzione.

♣ **Esempio 1.25.** Sia  $f(x) = x^2$ . Risulta

$$C(f, x) = 2 \quad \forall x \in \mathbb{R}.$$

Più in generale,  $\forall n \in \mathcal{N}$ ,  $C(x^n, x) = n$  e quindi il condizionamento del problema della valutazione della funzione  $x^n$  non dipende da  $x$ ; inoltre il mal condizionamento della funzione cresce al crescere di  $n$ .



♣ **Esempio 1.26.** Sia  $f(x) = \sqrt{1-x}$ ,  $x \leq 1$ . In questo caso

$$C(f, x) = \frac{|x|}{2(1-x)};$$

per  $x = 0.9999$ ,  $C(f, x) = 4999.5$  e quindi il problema è mal condizionato per  $x \simeq 1$ .



Per problemi più complessi della semplice valutazione di una funzione scalare di una sola variabile<sup>41</sup> si può cercare di valutare il fattore di amplificazione dell'errore generalizzando l'approccio qui presentato. Un caso semplice da analizzare è quello della sottrazione tra due numeri.

---

<sup>41</sup> Assegnata una funzione vettoriale ad  $m$  componenti, di  $n$  variabili:

$$\begin{aligned}\underline{f} &= (f_1, \dots, f_m) \\ f_i &= f_i(\underline{x}) \quad i = 1, \dots, m \\ \underline{x} &= (x_1, \dots, x_n)\end{aligned}$$

dallo sviluppo in serie di Taylor ed adottando, inoltre, per due vettori assegnati

$$\begin{aligned}w &= (w_1, \dots, w_k) \\ v &= (v_1, \dots, v_k)\end{aligned}$$

la notazione

$$\frac{w}{v} = \left( \frac{w_1}{v_1}, \dots, \frac{w_k}{v_k} \right)$$

si ha

$$\frac{\Delta \underline{f}}{\underline{f}} \cong \left( \frac{\delta f_i}{\delta x_j} \cdot \frac{x_j}{f_i} \right) \frac{\Delta \underline{x}}{\underline{x}} \quad (1.34)$$

dove con  $\Delta \underline{x}$  e  $\Delta \underline{f}$  si indicano, rispettivamente:

$$\begin{aligned}\Delta \underline{x} &= (\Delta x_1, \dots, \Delta x_n) \\ \Delta \underline{f} &= (\Delta f_1, \dots, \Delta f_m)\end{aligned}$$

da cui segue

$$\Delta f_i = f_i(\underline{x} + \Delta \underline{x}) - f_i(\underline{x}) \quad i = 1, \dots, m.$$

Passando alle norme infinito nella (1.34), si ottiene

$$\left\| \frac{\Delta \underline{f}}{\underline{f}} \right\|_{\infty} \leq \left\| \left( \frac{\delta f_i}{\delta x_j} \cdot \frac{x_j}{f_i} \right) \right\|_{\infty} \cdot \left\| \frac{\Delta \underline{x}}{\underline{x}} \right\|_{\infty}$$

La quantità

$$C(\underline{f}, \underline{x}) = \left\| \left( \frac{\delta f_i}{\delta x_j} \cdot \frac{x_j}{f_i} \right) \right\|_{\infty}$$

è detta *indice di condizionamento relativo* della funzione  $\underline{f}$  nel punto  $\underline{x}$ .

♣ **Esempio 1.27.** Sia  $f(x, y) = x - y$ . Posto  $\Delta f = f(x + \Delta x, y + \Delta y) - f(x, y)$ , si ha:

$$\Delta f = x + \Delta x - y - \Delta y - x + y = \Delta x - \Delta y,$$

e quindi

$$\begin{aligned} \frac{|\Delta f|}{|f|} &\leq \frac{|\Delta x| + |\Delta y|}{|x - y|} = \frac{|x| + |y|}{|x - y|} \cdot \frac{|\Delta x| + |\Delta y|}{|x| + |y|} = \\ &= \frac{|xf_x| + |yf_y|}{|f(x, y)|} \cdot \frac{\|(\Delta x, \Delta y)\|}{\|(x, y)\|}, \end{aligned}$$

dove  $\|\cdot\| = \|\cdot\|_1$  in  $\Re^2$ ,  $f_x = \frac{\partial f}{\partial x}$  e  $f_y = \frac{\partial f}{\partial y}$ . In questo caso l'indice di condizionamento relativo è

$$C(f, x, y) = (|x| + |y|)/|x - y|.$$

E' evidente che  $C(f, x, y)$  cresce al diminuire della distanza tra  $x$  e  $y$ ; dunque il calcolo della differenza tra due numeri è un problema mal condizionato quando i due numeri sono "vicini". Ciò spiega il fenomeno della cancellazione illustrato nell'esempio 1.24.



♣ **Esempio 1.28.** Sia  $f(x) = M \cdot x$ , con  $M \in \Re^{m \times n}$  ed  $x \in \Re^n$ . Approssimando la derivata di  $f$  con il rapporto incrementale, l'indice di condizionamento assoluto del problema del prodotto tra una matrice ed un vettore è:

$$\begin{aligned} C_A(f, x) &= \left| \frac{\Delta f}{\Delta x} \right| = \left\| \frac{M(x + \Delta x) - Mx}{\Delta x} \right\| = \\ &\leq \frac{\|M(x + \Delta x) - Mx\|}{\|\Delta x\|} = \frac{\|M\Delta x\|}{\|\Delta x\|} \leq \\ &\leq \|M\| \frac{\|\Delta x\|}{\|\Delta x\|} = \|M\| \end{aligned}$$

L'indice di condizionamento relativo è:

$$\begin{aligned} C_R(f, x) &= \left| \frac{\Delta f}{\Delta x} \cdot \frac{x}{f(x)} \right| = \\ &= \left\| \frac{M(x + \Delta x) - Mx}{\Delta x} \cdot \frac{x}{Mx} \right\| \leq \\ &\leq \frac{\|M\Delta x\|}{\|\Delta x\|} \cdot \frac{\|x\|}{\|Mx\|} \leq \\ &\leq \frac{\|M\| \cdot \|\Delta x\|}{\|\Delta x\|} \cdot \frac{\|x\|}{\|Mx\|} = \frac{\|M\| \cdot \|x\|}{\|Mx\|} \end{aligned}$$

Si osserva che, essendo

$$\|Mx\|_\alpha \leq \|M\|_\beta \cdot \|x\|_\alpha,$$

dove  $\|\cdot\|_\beta$  è una qualsiasi norma matriciale, compatibile con la norma vettoriale  $\|\cdot\|_\alpha$  (cfr. §B.3), segue:

$$\frac{\|M\| \cdot \|x\|}{\|Mx\|} \geq 1$$

ovvero l'indice di condizionamento relativo del problema del prodotto tra una matrice ed un vettore è almeno 1.



## 1.6 La stabilità di un algoritmo

Anche se il problema da risolvere è ben condizionato, la bontà del risultato computazionale dipende dal “comportamento” dell’algoritmo utilizzato.

♣ **Esempio 1.29.**

$$\begin{cases} 0.0001x + y = 1 \\ x + y = 2 \end{cases}$$

Si può dimostrare che questo problema è ben condizionato; la sua soluzione è il vettore  $(1.0001000\dots, 0.99989999\dots)$ .

Sostituendo alla seconda equazione la prima equazione moltiplicata per  $10^4$  e sottratta dalla seconda, si ha il sistema equivalente

$$\begin{cases} 0.0001x + y = 1 \\ -9999y = -9998 \end{cases}$$

da cui

$$\begin{cases} y = \frac{-9998}{-9999} = 1.0001000\dots \\ x = \frac{1}{0.0001}(1 - \frac{9998}{9999}) = 0.99989998\dots \end{cases}$$

Eseguendo questo algoritmo in un sistema aritmetico f.p. con precisione  $t = 3$  si ottiene:

$$\begin{cases} y = \frac{-0.100 \times 10^5}{-0.100 \times 10^4} = 0.100 \times 10^1 = 1 \\ x = \frac{0.100 \times 10^1}{0.100 \times 10^{-3}}(0.100 \times 10^1 - 0.100 \times 10^1) = 0 \end{cases}$$

cioè il vettore soluzione calcolato dall’algoritmo ha la seconda componente senza alcuna cifra significativa corretta rispetto alla soluzione esatta del problema.

Invertendo l’ordine delle equazioni e moltiplicando per  $10^{-4}$  la prima equazione, si ottiene, nello stesso sistema f.p., la soluzione  $x = 1, y = 1$ , che è un’approssimazione accettabile della soluzione.

Questo fenomeno è dovuto al modo in cui si è propagato l’errore di roundoff durante l’esecuzione dell’algoritmo. Infatti, nel primo caso la divisione per il coefficiente 0.0001 ha prodotto come fattore moltiplicativo  $10^4$  amplificando l’errore di roundoff. Invertendo l’ordine delle equazioni, il fattore moltiplicativo diventa  $10^{-4}$  che, al contrario del primo caso, riduce notevolmente la propagazione dell’errore.



In generale, due algoritmi per la risoluzione di uno stesso problema possono produrre risultati differenti, in quanto gli errori complessivi di roundoff sono diversi. Il concetto di *stabilità* (e quindi di instabilità) si riferisce alla sensibilità di un algoritmo alla precisione finita, cioè agli errori di roundoff dei dati e delle operazioni f.p..

**Definizione 1.8. (Instabilità di un algoritmo)**

*Un algoritmo si dice instabile, se gli errori di roundoff introdotti nei dati si propagano amplificandosi in maniera tale che i risultati siano inaccettabili.*

In altri termini, un algoritmo è instabile se, a causa della propagazione dell'errore di roundoff, il risultato dell'algoritmo differisce sostanzialmente dalla soluzione del problema numerico.

Un altro esempio di algoritmo instabile è illustrato qui di seguito.

**♣ Esempio 1.30.** Si consideri l'integrale

$$I_n = \int_0^1 x^n e^{x-1} dx, \quad n \geq 0.$$

Integrando per parti si ha:

$$\int_0^1 x^n e^{x-1} dx = [x^n e^{x-1}]_0^1 - n \int_0^1 x^{n-1} e^{x-1} dx = 1 - n \int_0^1 x^{n-1} e^{x-1} dx,$$

cioè:

$$I_n = 1 - n I_{n-1}, \quad n \geq 1. \quad (1.35)$$

Dato che

$$I_0 = \int_0^1 e^{x-1} dx = 1 - \frac{1}{e},$$

si può calcolare  $I_n$  mediante un algoritmo basato sulla formula ricorrente

$$\begin{aligned} I_0 &= 1 - 1/e, \\ I_n &= 1 - n I_{n-1}, \quad n \geq 1. \end{aligned} \quad (1.36)$$

Si consideri, ad esempio, il calcolo di  $I_{14}$ . Implementando il procedimento utilizzando la singola precisione, si ottengono i risultati riportati in Tabella 1.5.

Il valore calcolato di  $I_{14}$  è completamente errato; infatti  $I_{14}$  deve essere positivo. Più in generale, i valori ottenuti per  $n > 9$  sono completamente errati.<sup>42</sup> Perché si verifica questa situazione? Per dare una risposta al quesito occorre analizzare la propagazione dell'errore di roundoff dell'unico dato di input dell'algoritmo,  $I_0$ , supponendo, per semplicità, di effettuare le operazioni in aritmetica esatta. L'errore  $\epsilon_0$  introdotto in  $1/e$ , e quindi in  $I_0$ , è tale che:

$$\bar{I}_0 = I_0 + \epsilon_0, \quad |\epsilon_0| \simeq 10^{-6},$$

dove  $\bar{I}_0$  denota il valore calcolato di  $I_0$ . Quindi, per i termini successivi si ha:

$$\begin{aligned} \bar{I}_1 &= 1 - 1 \times \bar{I}_0 = (1 - I_0) - \epsilon_0 = I_1 - \epsilon_0, \\ \bar{I}_2 &= 1 - 2 \times \bar{I}_1 = (1 - 2 \times I_1) + 2 \times \epsilon_0 = I_2 + 2 \times \epsilon_0 \\ &\vdots \\ \bar{I}_n &= 1 - n \times \bar{I}_{n-1} = I_n + (-1)^n n! \times \epsilon_0. \\ &\vdots \end{aligned}$$

---

<sup>42</sup>Ciò è evidente per  $n > 10$ , in quanto, per ogni  $n$ ,  $I_n$  deve essere positivo e tale che  $I_{n+1} < I_n$ . Nella seguente tabella sono riportati i valori corretti di  $I_n$ , per  $n = 0, 1, 2, \dots, 14$ , arrotondati a 7 cifre significative:

$n$	$I_n$
0	$+0.6321206 \times 10^0$
1	$+0.3678795 \times 10^0$
2	$+0.2642411 \times 10^0$
3	$+0.2072767 \times 10^0$
4	$+0.1708932 \times 10^0$
5	$+0.1455340 \times 10^0$
6	$+0.1267958 \times 10^0$
7	$+0.1124296 \times 10^0$
8	$+0.1005630 \times 10^0$
9	$+0.9493256 \times 10^{-1}$
10	$+0.5067444 \times 10^{-1}$
11	$+0.4425812 \times 10^0$
12	$-0.4310974 \times 10^1$
13	$+0.5704266 \times 10^2$
14	$-0.7975973 \times 10^3$

Tabella 1.5: Calcolo di un integrale per ricorrenza: risultati ottenuti implementando la formula (1.36) con  $n = 14$

Indicato con  $\epsilon_n$  l'errore assoluto al passo  $n$ , per  $n = 10$  risulta  $\epsilon_n \simeq 1$  e, a partire da  $n = 10$ ,  $\epsilon_n$  cresce “enormemente”, vanificando i calcoli successivi. In particolare, per  $n = 14$   $\epsilon_n \simeq 14! \times 10^{-6} \simeq 87000$ . L'algoritmo basato sulla formula (1.36) è dunque instabile e non può essere utilizzato.<sup>43</sup> Si deve quindi

$n$	$I_n$
0	$.6321206 \times 10^0$
1	$.3678794 \times 10^0$
2	$.2642411 \times 10^0$
3	$.2072766 \times 10^0$
4	$.1708934 \times 10^0$
5	$.1455329 \times 10^0$
6	$.1268024 \times 10^0$
7	$.1123835 \times 10^0$
8	$.1009320 \times 10^0$
9	$.9161229 \times 10^{-1}$
10	$.8387707 \times 10^{-1}$
11	$.7735223 \times 10^{-1}$
12	$.7177325 \times 10^{-1}$
13	$.6694770 \times 10^{-1}$
14	$.6273216 \times 10^{-1}$

<sup>43</sup>Il valore di  $n$  a partire dal quale i risultati non sono più attendibili dipende dal sistema f.p. utilizzato. Infatti, in un sistema aritmetico a precisione più elevata il valore di  $n$  aumenta; si osservi però che qualunque sia la precisione finita esiste sempre un valore di  $n$  per cui il risultato è inaccettabile.

$n$	$I_n$
20	0.0000000
19	$0.5000000 \times 10^{-1}$
18	$0.5000000 \times 10^{-1}$
17	$0.5277778 \times 10^{-1}$
16	$0.5571895 \times 10^{-1}$
15	$0.5901757 \times 10^{-1}$
14	$0.6273216 \times 10^{-1}$

**Tabella 1.6:** Calcolo di un integrale per ricorrenza: risultati ottenuti implementando la formula (1.37) con  $n = 14$

progettare un algoritmo diverso basato su una formula che non amplifichi l'errore di roundoff.

Si osservi che la relazione di ricorrenza (1.35) può essere riscritta nel modo seguente:

$$I_{n-1} = (1 - I_n)/n \quad n \geq 1. \quad (1.37)$$

Dato che la successione  $\{I_n\}$  è decrescente e tende a 0, si può porre  $I_{20} = 0$  e calcolare poi  $I_{19}, I_{18}, \dots, I_{14}$  utilizzando la (1.37). L'implementazione di questo algoritmo calcola una buona approssimazione di  $I_{14}$  e, più in generale, fornisce risultati accettabili per  $n \leq 17$  (Tabella 1.6).<sup>44</sup>

L'algoritmo è infatti stabile; l'errore  $\epsilon_{20}$  introdotto in  $I_{20}$  si propaga nel modo seguente:

$$\begin{aligned} \bar{I}_{20} &= I_{20} + \epsilon_{20}, \\ \bar{I}_{19} &= (1 - \bar{I}_{20})/20 = I_{19} - \epsilon_{20}/20, \\ \bar{I}_{18} &= (1 - \bar{I}_{19})/19 = I_{18} + \epsilon_{20}/(20 \times 19), \\ &\vdots \\ \bar{I}_{n-1} &= (1 - \bar{I}_n)/n = I_{n-1} + (-1)^{n-1} \epsilon_{20}/(20 \times 19 \times \cdots n), \\ &\vdots \end{aligned}$$

---

<sup>44</sup>I valori corretti di  $I_n$ , per  $n = 20, 19, \dots, 14$ , arrotondati a 7 cifre significative, sono riportati nella tabella seguente:

$n$	$I_n$
20	$.4554488 \times 10^{-1}$
19	$.4772276 \times 10^{-1}$
18	$.5011985 \times 10^{-1}$
17	$.5277112 \times 10^{-1}$
16	$.5571935 \times 10^{-1}$
15	$.5901754 \times 10^{-1}$
14	$.6273216 \times 10^{-1}$

Si noti che i valori calcolati con la (1.37) per  $n = 20, 19$  sono inattendibili solo perché il valore iniziale  $I_{20}$  è arbitrario; a differenza dell'algoritmo precedente, si può calcolare con la massima accuratezza possibile il valore di qualunque  $I_n$ , a patto di considerare come dato iniziale  $I_{\bar{n}} = 0$  con  $\bar{n}$  sufficientemente maggiore di  $n$ .

$n$	$x^n/n!$	$S_n$
0	$+0.1000000 \times 10^1$	$+0.1000000 \times 10^1$
1	$-0.1000000 \times 10^2$	$-0.9000000 \times 10^1$
2	$+0.5000000 \times 10^2$	$+0.4100000 \times 10^2$
3	$-0.1666667 \times 10^3$	$-0.1256667 \times 10^3$
4	$+0.4166666 \times 10^3$	$+0.2910000 \times 10^3$
$\vdots$	$\vdots$	$\vdots$
32	$+0.3800392 \times 10^{-3}$	$+0.3651343 \times 10^{-4}$
33	$-0.1151634 \times 10^{-3}$	$-0.7864997 \times 10^{-4}$
34	$+0.3387159 \times 10^{-4}$	$-0.4477838 \times 10^{-4}$
35	$-0.9677598 \times 10^{-5}$	$-0.5445597 \times 10^{-4}$
$\vdots$	$\vdots$	$\vdots$
41	$-0.2989313 \times 10^{-8}$	$-0.5234286 \times 10^{-4}$
42	$+0.7117412 \times 10^{-9}$	$-0.5234215 \times 10^{-4}$
43	$-0.1655212 \times 10^{-9}$	$-0.5234231 \times 10^{-4}$
44	$+0.3761846 \times 10^{-10}$	$-0.5234228 \times 10^{-4}$

Tabella 1.7: Valori di  $x^n/n!$  e di  $S_n$ 

L'errore iniziale  $\epsilon_{20}$  al generico passo  $n$  è diviso per  $20 \times 19 \times \dots \times (n+1)$  e quindi non è amplificato, anzi è ridotto.



Un altro esempio di algoritmo instabile è l'algoritmo per il calcolo di un'approssimazione di  $e^x$ , con  $x < 0$ , basato sullo sviluppo in serie di Mac Laurin.

♣ **Esempio 1.31.** Si supponga  $t = 7$  e  $x = -10$  e si approssimi  $e^x$  con la somma

$$S_N(x) = \sum_{n=0}^N \frac{x^n}{n!} = \sum_{n=0}^N \frac{(-1)^n |x|^n}{n!}. \quad (1.38)$$

Un algoritmo basato sul calcolo diretto della (1.38) con il criterio d'arresto naturale (cfr. Procedura 1.2) fornisce il risultato  $-0.5234228 \times 10^{-4}$ , mentre il valore corretto di  $e^{-10}$ , arrotondato a 7 cifre significative, è  $0.4539993 \times 10^{-4}$ . L'errore relativo è dunque circa  $0.2152915 \times 10$  ed il risultato dell'algoritmo non è sicuramente una buona approssimazione di  $e^{-10}$ . Il problema risiede nel fatto che i termini dello sviluppo in serie di  $e^{-10}$  hanno segni alterni e la sequenza di somme effettuate non è altro che una sequenza di sottrazioni tra numeri “abbastanza vicini” (Tabella 1.7).

La causa dell'instabilità di questo algoritmo sta nella progressiva perdita di cifre significative e quindi nel conseguente aumento dell'errore relativo; si verificano cioè successivamente vari fenomeni di cancellazione nelle sottrazioni. Si ricordi che la cancellazione è la manifestazione di una caratteristica

intrinseca dell'operazione aritmetica di sottrazione, che è mal condizionata quando gli operandi sono "vicini" (cfr. esempi 1.24 e 1.27). Il fatto che si usi un sistema aritmetico a precisione finita implica solo che vi è certamente un errore (quello di roundoff) negli addendi e quindi il mal condizionamento della sottrazione si manifesta effettivamente. Sono proprio gli errori negli addendi le cause della cancellazione, mentre l'operazione sottrazione, nell'unità aritmetica, è completamente incolpevole.

Un algoritmo stabile per il calcolo di un'approssimazione di  $e^x$  per  $x < 0$ , utilizzando sempre lo sviluppo in serie di Mac Laurin, è basato sull'osservazione che  $e^x = 1/e^{-x}$  ed il problema è trasformato in quello del calcolo di  $e^x$  per  $x > 0$ . L'algoritmo descritto dalla Procedura 1.2 del §1.4, che è stabile se  $x > 0$  (gli addendi sono tutti positivi e quindi non c'è fenomeno di cancellazione), è usato per il calcolo dell'esponenziale positivo ed il reciproco del risultato è il risultato del nuovo algoritmo stabile per  $e^x$  con  $x < 0$ .



E' importante osservare che, a differenza del condizionamento, l'instabilità è una caratteristica dell'algoritmo e non del problema e quindi l'instabilità può essere evitata con opportune modifiche dell'algoritmo, come, appunto, nell'esempio 1.31.

## 1.7 L'Aritmetica Standard IEEE

Nel 1982 il *Floating - point Working Group dell'IEEE*<sup>45</sup> *Computer Society's Microprocessor Standard Committee* propose uno standard di **sistema aritmetico floating point binario**, per il progetto delle unità aritmetiche dei calcolatori<sup>46</sup>.

L'aritmetica Standard IEEE descrive un modello di sistema aritmetico floating point (cioè definisce la rappresentazione dei dati numerici e le operazioni aritmetiche eseguite su tali dati) le cui caratteristiche fondamentali si possono sintetizzare nel modo seguente:

- massima accuratezza statica, per la rappresentazione dei numeri floating point;
- massima accuratezza dinamica, per le operazioni aritmetiche;
- gestione delle eccezioni;
- gestione del *gradual underflow*;
- quattro schemi per l'arrotondamento.

Nel seguito si descrive, brevemente, lo standard IEEE, riassumendo le proprietà della rappresentazione dei numeri floating point, la gestione delle situazioni eccezionali, il gradual underflow e gli schemi dell'arrotondamento.

<sup>45</sup>Institute of Electrical and Electronic Engineering

<sup>46</sup>L'aritmetica standard IEEE è anche nota come aritmetica *KCS* dalle iniziali di W. Kahan, J. Coonen, studente di W. Kahan presso l'Università della California di Berkeley, e del prof. H. Stone. W. Kahan ha ricevuto per questo motivo il premio **TURING** dall'ACM, nel 1989.

1. **Rappresentazione dei numeri floating point:** l’aritmetica IEEE definisce tre formati:

- singola precisione,
- doppia precisione,
- precisione estesa.

Mentre i primi due sono strettamente obbligatori, il terzo formato è facoltativo in quanto viene utilizzato al più nei registri all’interno dell’unità aritmetico-logica per la rappresentazione dei risultati intermedi in una successione di operazioni floating point.

Nella tabella seguente riportiamo le caratteristiche del formato esteso della singola e della doppia precisione:

Tipo	Lunghezza parola	precisione	esponente min	esponente max
singola	$\geq 43$	$\geq 32$	$\leq -1021$	$\geq 1024$
doppia	$\geq 79$	$\geq 64$	$\leq -16381$	$\geq 16384$

La locazione di memoria per i formati in singola e doppia precisione è suddivisa nel modo seguente:

Tipo	Segno	Esponente	Mantissa
singola	$\pm$	8	23
doppia	$\pm$	11	52

### Organizzazione della locazione di memoria per la rappresentazione dei numeri floating point nello standard IEEE

Tenendo conto dell’organizzazione della locazione di memoria, il numero di bit riservati per l’esponente è 8 per la singola precisione e 11 per la doppia; pertanto l’intervallo di rappresentabilità è delimitato dagli esponenti  $-127$  e  $+128$ , in base 2, per la singola precisione e da  $-1023$  a  $+1024$  per la doppia precisione. Questo corrisponde, in base 10, ad una fascia esponenziale che varia da  $-38$  a  $+38$  per la singola precisione e da  $-308$  a  $+308$  per la doppia precisione. Inoltre, un solo bit viene riservato per il segno della mantissa. Per l’esponente si usa la notazione ad *eccesso 127* nella singola precisione e ad *eccesso 1023* nella doppia precisione<sup>47</sup>. Ciò significa che invece di rappresentare l’esponente  $e$  con  $-127 \leq$

<sup>47</sup>Questo concetto è già stato introdotto nella nota (23) del paragrafo 1.2.2.

$e \leq 128$ , il valore che viene memorizzato è un intero senza segno  $e'$  con  $0 \leq e' \leq 255$ , per la singola precisione. Per quanto riguarda la doppia precisione, invece di rappresentare l'esponente  $e$  con  $-1023 \leq e \leq 1024$  viene memorizzato un numero positivo,  $e'$ , con  $0 \leq e' \leq 2047$ . Questo significa considerare anche il bit dedicato al segno come un bit per il campo dell'esponente e calcolare il numero decimale ottenuto dall'intera sequenza di bit, considerato come intero senza segno.

♣ **Esempio 1.32.** Considerati 3 bit, di cui il primo per il segno ed i restanti due per il numero stesso, si ottengono i seguenti numeri naturali:

segno + rapp.bin	rapp. dec.
0 00	-0
0 01	-1
0 10	-2
0 11	-3
1 00	+0
1 01	+1
1 10	+2
1 11	+3

Utilizzando invece la notazione senza segno, la stessa sequenza di bit rappresenta un diverso intervallo di numeri naturali (positivi):

segno + rapp.bin	rapp. dec. senza segno
0 00	0
0 01	1
0 10	2
0 11	3
1 00	4
1 01	5
1 10	6
1 11	7

In questo caso l'intervallo di rappresentabilità viene slittato da  $[-3, +3]$  a  $[0, 7]$ .



L'aritmetica IEEE si avvale della rappresentazione floating point normalizzata, in cui, per la rappresentazione della mantissa, si fa uso del **bit implicito**:<sup>48</sup> il bit

<sup>48</sup>Questo concetto è già stato introdotto nella nota (21) del paragrafo 1.2.2 nell'ambito della descrizione di un qualsiasi sistema aritmetico floating point.

iniziale ha sempre valore 1 e non è esplicitamente rappresentato. Questo comporta la possibilità di rappresentare effettivamente 24 bit per la singola precisione e 53 per la doppia precisione.

Relativamente all'esecuzione delle **operazioni aritmetiche floating point**, l'aritmetica IEEE utilizza, per la rappresentazione delle mantisse degli operandi, registri aritmetici con 2 *guard digits* più uno *sticky-bit* per garantire la massima accuratezza dinamica. I due bit aggiuntivi (i *guard digits*) sono utilizzati per rappresentare i primi due bit significativi della parte di mantissa (degli operandi e del risultato) che viene rimossa a causa dell'arrotondamento. Il terzo bit, detto *sticky bit*, è ottenuto come OR logico di tutti i bit rimanenti. Come anche già detto si può dimostrare che tre bit sono sufficienti a garantire la massima accuratezza dinamica.

## 2. Gestione delle situazioni eccezionali:

Tali situazioni sono classificate in:

- operazione non valida (INVALID),
- divisione per zero,
- overflow,
- underflow.

Ad ogni eccezione sono associati un *flag* ed un *trap*. Il *flag* indica quale eccezione si è verificata. Il *trap* attiva la scelta tra due possibilità: chiamata ad una procedura di gestione dell'eccezione con interruzione dell'esecuzione oppure nessuna interruzione dell'esecuzione.

Un'operazione INVALID produce un **NaN** (Not a Number), la cui rappresentazione in memoria è la seguente:

Segno	Esponente	Mantissa
±	1...1	qualsiasi combinazione di bit

### Rappresentazione di NaN nell'aritmetica IEEE

Un risultato NaN è ottenuto effettuando operazioni non definite come ad esempio  $0 \times \infty$ ,  $0/0$ ,  $\infty/\infty$ ,  $+\infty - \infty$ .

La divisione per zero produce come risultato **Inf**, la cui rappresentazione in memoria è la seguente:

Segno	Esponente	Mantissa
±	1...1	0.....0

### Rappresentazione di Inf nell'aritmetica IEEE

Il segno di **Inf** è l'usuale segno di un quoziente pertanto la divisione per zero è l'unica operazione algebrica che rivela il segno di zero.

Lo standard IEEE introduce alcune funzioni di ambiente per la gestione delle eccezioni; tra queste particolarmente utili sono **FINITE(x)** che restituisce il valore *vero* se l'argomento è rappresentabile, *falso* in caso contrario, e la funzione **ISNAN(x)** che ritorna il valore *vero* se l'argomento è un NaN, *falso* altrimenti.

Sia **NaN** che **Inf** possono essere utilizzati nell'esecuzione delle usuali operazioni aritmetiche. Tipicamente il risultato di un'operazione che coinvolge un **NaN** continua ad essere **NaN**, lo stesso vale per **Inf** con le dovute eccezioni quali, ad esempio, nella divisione. In tal caso se il numeratore è diverso da zero ed il denominatore è **Inf**, il quoziente è zero.

### 3. Gradual underflow:

Lo standard IEEE introduce i cosiddetti *numeri denormalizzati*, aventi, come esponente, l'esponente minimo consentito ed una mantissa non normalizzata. Questo significa che dopo  $rmin = 0.1 \times \beta^{emin}$  sono ancora rappresentabili i numeri del tipo<sup>49</sup>:

$$\begin{aligned} 0.01 \times \beta^{emin} &= 0.1 \times \beta^{emin-1} \\ 0.001 \times \beta^{emin} &= 0.1 \times \beta^{emin-2} \\ 0.0001 \times \beta^{emin} &= 0.1 \times \beta^{emin-3} \\ \dots &\quad \dots \quad \dots \\ 0.00\dots1 \times \beta^{emin} &= 0.1 \times \beta^{emin-t} \end{aligned}$$

Come si può notare in questo caso la soglia di underflow viene ridotta: in particolare, per la singola precisione, essendo  $t = 7$ , essa risulta pari a  $10^{-38-7} = 10^{-45}$ , mentre per la doppia precisione ( $t = 16$ ), si ha il valore  $10^{-308-16} = 10^{-324}$ .

I numeri denormalizzati sono particolarmente utili nell'esecuzione di sommatorie in cui uno degli addendi può andare in underflow. In tal caso l'underflow graduale consente di non perdere completamente il contributo di un tale addendo, fornendo quindi un risultato più accurato.

### 4. Schemi di arrotondamento

Nella rappresentazione dei numeri floating point, usualmente, si utilizza lo schema dell'arrotondamento. Esso è utilizzato per default nello standard IEEE, ed

---

<sup>49</sup>Si osservi che, nel caso particolare in cui  $\beta = 2$ ,

$$rmin = 1.0 \times \beta^{emin+1},$$

in quanto gli esponenti *emin* ed *emax* sono riservati, ovvero sono utilizzati esclusivamente per lo 0, i numeri *denormalizzati*, per Inf e NaN.

è indicato anche come *Round to the Nearest (RN)*<sup>50</sup>. Lo standard IEEE prevede la possibilità di utilizzare tre ulteriori **schemi di arrotondamento**, indicati, rispettivamente, come *Round towards Zero (RZ)*, *Round towards Plus  $\infty$  (RP)* e *Round towards Minus  $\infty$  (RM)*. Il primo dei tre, RZ, è l'usuale **troncamento**<sup>51</sup>, mentre gli altri due scelgono come approssimazione floating point rispettivamente il numero più grande<sup>52</sup> ed il più piccolo<sup>53</sup>.

♣ **Esempio 1.33.** Sia  $F=(10,6,-9,9)$  e  $x = .123456789$ , si ha

$$fl(x) = \begin{cases} .123457 & \text{per RN e RP} \\ .123456 & \text{per RZ e RM} \end{cases}$$

Se consideriamo  $x = -.123456789$ , si ha :

$$fl(x) = \begin{cases} -.123457 & \text{per RN e RM} \\ -.123456 & \text{per RZ e RP} \end{cases}$$



Nella tabella seguente riportiamo le caratteristiche dei numeri floating point nello standard IEEE e la loro tipica rappresentazione:

---

<sup>50</sup> *Round to Nearest (RN)* significa *Approssimazione al più vicino*, principio sul quale si basa la tecnica di arrotondamento per la rappresentazione di un qualsiasi numero reale (appartenente all'insieme di rappresentabilità) mediante il numero macchina ad esso più vicino.

<sup>51</sup> *Round towards Zero (RZ)* significa *Approssimazione verso lo zero*, principio sul quale si basa la tecnica del troncamento, in base alla quale un numero reale rappresentabile viene approssimato con il numero macchina più vicino che, in valore assoluto, risulta più piccolo.

<sup>52</sup> *Round towards Plus  $\infty$  (RP)* significa *Approssimazione verso più infinito*, e consiste nell'approssimare un numero reale rappresentabile con il più vicino numero macchina situato nella direzione positiva dell'asse reale.

<sup>53</sup> *Round towards Minus  $\infty$  (RM)* significa *Approssimazione verso meno infinito*, e consiste nell'approssimare un numero reale rappresentabile con il più vicino numero macchina situato nella direzione negativa dell'asse reale.

Argomento	Singola precisione	Doppia precisione
Bit di segno	1	1
Bit per l'esponente	8	11
Bit per la mantissa	23	52
Bit totali	32	64
Rappr. esponente	Eccesso 127	Eccesso 1023
Fascia esponenziale	da $-127$ a $+128$	da $-1023$ a $+1024$
Minimo normalizzato	$-2^{-126}$	$-2^{-1022}$
Max normalizzato	$(2 - 2^{-23})2^{127}$	$(2 - 2^{-52})2^{1023}$
Fascia decimale	da $10^{-38}$ a $10^{+38}$	da $10^{-308}$ a $10^{+308}$
Minimo denormalizzato	$10^{-45}$	$10^{-324}$

### Lo standard IEEE

## 1.8 Analisi della propagazione dell'errore di roundoff

L'analisi della propagazione dell'errore di roundoff, cioè lo studio dell'errore globale di roundoff inteso come somma (algebrica) di tutti gli errori di roundoff generati dall'algoritmo, riveste un ruolo fondamentale nella progettazione e sviluppo di un algoritmo. E' dunque importante avere a disposizione strumenti che consentano di valutare gli effetti di tale errore sulla soluzione calcolata dall'algoritmo (eseguito in aritmetica a precisione finita).

Nel seguito sono presentati due metodi di analisi della propagazione dell'errore di roundoff:

- la *forward error analysis*;
- la *backward error analysis*.

♣ **Esempio 1.34.** Sia assegnato un sistema aritmetico f. p.

$$F : (\beta = 10, t = 3, \underbrace{t_{reg}}_{max.acc.din.} = 2 \cdot t = 6),$$

in cui  $u = 0.5\beta^{1-t} = 0.5 \times 10^{-2}$ . Se  $x = 3.453$  risulta

$$x^* = fl(x) = 0.345 \times 10^1 = 3.45,$$

Si desidera calcolare il quadrato di  $x$ . In aritmetica esatta  $s = x^2 = 11.923209$ . In  $F$ , invece,

$$s^* = x^* \otimes x^* = 0.345 \times 10 \otimes 0.345 \times 10 = 0.119 \times 10^2 = 11.9$$

L'errore relativo di roundoff su  $s^*$  è, dunque,

$$E'_{s^*} = \frac{|s - s^*|}{s} = 0.195 \times 10^{-2} \leq 3 \times \underbrace{(0.5 \times 10^{-2})}_{u} = 3u = 1.5 \times 10^{-2}.$$

Segue, allora, che

$$E'_{s^*} \simeq ku \quad \text{con } k = 3$$

cioè la stima dell'errore relativo è dello stesso ordine di grandezza dell'errore di roundoff sul dato, per cui la soluzione si ritiene *accettabile*. ♣

Effettuiamo l'**analisi della propagazione dell'errore di roundoff**, ovvero studiamo in che modo gli errori introdotti dal sistema aritmetico a precisione finita si propagano sulla soluzione. Nel calcolo di

$$s^* = x^* \oplus y^*$$

siano:

1.  $\delta_1, \delta_2$ : errori di *rappresentazione* dei dati:

$$\begin{aligned} x^* &= fl(x) = x(1 + \delta_1) \quad \text{con } |\delta_1| \leq u \\ y^* &= fl(y) = y(1 + \delta_2) \quad \text{con } |\delta_2| \leq u \end{aligned}$$

2.  $\rho_1$ : errore sulle *operazioni floating-point* in  $F$ :

$$\begin{aligned} x^* &= fl(x) \in F \\ y^* &= fl(y) \in F \end{aligned}$$

per cui si ricava:

$$\begin{aligned} fl(x) \oplus fl(y) &= fl(fl(x) \# fl(y)) = (fl(x) \# fl(y))(1 + \rho_1) \\ &\text{con } |\rho_1| \leq u \end{aligned}$$

supponendo di essere in presenza di un sistema che realizza la massima accuratezza dinamica. In particolare, volendo eseguire il calcolo del quadrato di un dato  $x$ :

$$x^* = fl(x) = x(1 + \delta_1)$$

eseguendo la moltiplicazione

$$s^* = x^* \otimes x^* = fl(x^* \times x^*) = fl(fl(x) \times fl(x))$$

si ottiene

$$\begin{aligned} fl(fl(x) \times fl(x)) &= (\underbrace{x(1 + \delta_1)}_{err. di rappr.} \cdot \underbrace{x(1 + \delta_1)}_{err. di rappr.}) \cdot \underbrace{(1 + \rho_1)}_{err. sulla moltipl.} = \\ &= x^2(1 + \delta_1)^2(1 + \rho_1) \end{aligned}$$

con  $|\delta_1|, |\rho_1| \leq u$ , da cui:

$$s^* = x^* \otimes x^* = x^2(1 + \delta_1)^2(1 + \rho_1).$$

L'errore relativo risulta:

$$\begin{aligned} E'_{s^*} &= \frac{|s^* - s|}{|s|} = \frac{|x^2(1 + \delta_1)^2(1 + \rho_1) - x^2|}{x^2} = \\ &= |1 + \rho_1 + \delta_1^2 + \rho_1\delta_1^2 + 2\delta_1 + 2\delta_1\rho_1 - 1| \leq \\ &\leq 3u + 3u^2 + u^3 = 3u + \mathcal{O}(u^2) \simeq 3u \end{aligned}$$

ovvero  $E'_{s^*}$  è dello stesso ordine di grandezza della massima accuratezza relativa; la *soluzione* può, dunque, ritenersi *accettabile*.

♣ **Esempio 1.35.** Assegnato un sistema aritmetico f.p.

$$F : (\beta = 10, t = 4, \underbrace{t_{reg}}_{=} = 2 \cdot t = 8),$$

in cui  $u = 0.5\beta^{1-t} = 0.5 \times 10^{-3}$ , ed i dati:

$$\begin{aligned} x &= 7.41330 \Rightarrow x^* = fl(x) = 0.7413 \times 10^1 \\ y &= 3.453749 \Rightarrow y^* = fl(y) = 0.3454 \times 10^1 \end{aligned}$$

calcolare la differenza

$$z^* = x^* - y^*$$

In aritmetica a precisione infinita  $z = x - y = 3.959551$ ; nel sistema f.p. considerato:

$$\begin{aligned} z^* &= x^* - y^* = \\ &= 0.7413 \times 10^1 - 0.3454 \times 10^1 \\ &= 0.3959 \times 10^1 \end{aligned}$$

L'errore relativo di roundoff su  $z^*$  risulta:

$$\begin{aligned} E'_{z^*} &= \frac{|z^* - z|}{|z|} = \frac{|0.3959 \times 10^1 - 3.959551|}{3.959551} = \\ &= 0.1392 \times 10^{-3} \simeq 0.3 \times \underbrace{0.5 \times 10^{-3}}_u \end{aligned}$$

ovvero

$$E'_{z^*} \simeq ku \quad \text{con } k = 0.3$$

La stima ottenuta è, dunque, dello stesso ordine di grandezza dell'errore di roundoff sul dato, per cui la *soluzione* può ritenersi *accettabile*. ♣

♣ **Esempio 1.36.** Assegnato un sistema aritmetico f.p.

$$F : (\beta = 10, t = 4, \underbrace{t_{reg} = 2 \cdot t}_{= 8}),$$

in cui  $u = 0.5\beta^{1-t} = 0.5 \times 10^{-3}$ , ed i dati:

$$\begin{aligned} x &= 3.453749 \Rightarrow x^* = fl(x) = 0.3454 \times 10^1 \\ y &= 3.453432 \Rightarrow y^* = fl(y) = 0.3453 \times 10^1 \end{aligned}$$

calcolare la differenza

$$z^* = x^* - y^*$$

In aritmetica a precisione infinita  $z = x - y = 0.000317$ ; nel sistema f.p. considerato:

$$\begin{aligned} z^* &= x^* - y^* = \\ &= 0.3454 \times 10^1 - 0.3453 \times 10^1 \\ &= 0.1 \times 10^{-2} \end{aligned}$$

L'errore relativo di roundoff su  $z^*$  risulta:

$$\begin{aligned} E'_{z^*} &= \frac{|z^* - z|}{|z|} = \frac{|0.1 \times 10^{-2} - 0.000317|}{0.000317} = \\ &= 0.2155 \times 10^1 = 4.31 \times 10^3 \times \underbrace{0.5 \times 10^{-3}}_u \end{aligned}$$

ovvero

$$E'_{z^*} \simeq ku \quad \text{con} \quad k = 4310$$

La stima ottenuta è, dunque, molto più grande dell'errore di roundoff sul dato (è circa di tre ordini di grandezza più grande), per cui la *soluzione* può ritenersi *non accettabile*. ♣

Riguardando i due esempi precedenti si può dedurre che, nello stesso sistema aritmetico a precisione finita, l'algoritmo per il calcolo di  $x - y$  produce un *risultato accettabile* in corrispondenza di

$$\begin{aligned} x &= 7.41330 \\ y &= 3.453749 \end{aligned}$$

ed un *risultato NON accettabile* per

$$\begin{aligned} x &= 3.453749 \\ y &= 3.453432 \end{aligned}$$

Ha senso, allora, chiedersi se l'algoritmo per il calcolo della differenza sia stabile o non lo sia. Effettuiamo, quindi, l'analisi della propagazione dell'errore di roundoff, nel calcolo della differenza tra due numeri,  $x$  e  $y$ .

Siano:

$$\begin{aligned} x^* &= fl(x) = x(1 + \delta_1) \\ y^* &= fl(y) = y(1 + \delta_2) \end{aligned}$$

si ha:

$$\begin{aligned} z^* &= x^* \ominus y^* = \\ &= fl(x^* - y^*) = \\ &= fl(fl(x) - fl(y)) = \\ &= (x(1 + \delta_1) - y(1 + \delta_2))(1 + \rho_1) \end{aligned}$$

con  $|\delta_1|, |\delta_2|, |\rho_1| \leq u$ . Dall'espressione dell'errore relativo si deduce in che modo, tali errori, si propagano sulla soluzione  $z^*$ :

$$\begin{aligned} E'_{z^*} &= \frac{|z^* - z|}{|z|} = \frac{|(x(1 + \delta_1) - y(1 + \delta_2))(1 + \rho_1) - (x - y)|}{|x - y|} = \\ &= \frac{|x(\delta_1 + \rho_1 + \delta_1\rho_1) - y(\delta_2 + \rho_1 + \delta_2\rho_1)|}{|x - y|} \leq \\ &\leq \frac{|x|(2u + u^2) + |y|(2u + u^2)}{|x - y|} = (2u + \mathcal{O}(u^2)) \frac{|x| + |y|}{|x - y|} \end{aligned}$$

ovvero

$$E'_{z^*} = (2u + \mathcal{O}(u^2)) \frac{|x| + |y|}{|x - y|}$$

Si osservi, dunque, che l'errore relativo nel calcolo della differenza tra due numeri, dipende da  $x$  e  $y$ . In altre parole, non si può concludere se la soluzione sia accettabile o no, indipendentemente dai valori di  $x$  e  $y$ . Posto

$$k = \frac{|x| + |y|}{|x - y|},$$

in corrispondenza di

$$\begin{aligned} x &= 7.41330 \\ y &= 3.453749 \end{aligned}$$

si ottiene  $k = 2.7445$  e, quindi, il risultato della differenza  $x - y$  può ritenersi accettabile. Al contrario, per

$$\begin{aligned} x &= 3.453749 \\ y &= 3.453432 \end{aligned}$$

si ha  $k = 2.17 \times 10^4$ , costante grande per cui il risultato della differenza può ritenersi NON accettabile.

L'analisi della propagazione dell'errore introdotto dalla precisione finita, *durante l'esecuzione delle operazioni richieste dall'algoritmo*, viene detta **forward error analysis (f.e.a.)** (analisi in avanti). Analizziamo, ora, quando un algoritmo si può dire stabile nel senso della forward error analysis.

Si indichino con  $S(d)$  e  $S_c(d)$ , rispettivamente, la soluzione esatta e quella calcolata. La f.e.a. si propone di fornire una stima dell'errore relativo (assoluto):

$$E'_s = \frac{\|S(d) - S_c(d)\|}{\|S(d)\|} \quad (\|S(d) - S_c(d)\|), \quad (1.39)$$

dove  $\|\cdot\|$  è una norma fissata, seguendo, passo dopo passo, l'introduzione di errori nel processo risolutivo, a partire dall'errore nei dati e la loro propagazione. In particolare, se

$$E'_s = \frac{\|S(d) - S_c(d)\|}{\|S(d)\|} \leq k \cdot u$$

qualora  $k$  sia una costante "sufficientemente piccola", l'algoritmo si dice stabile nel senso della f.e.a.; se, invece,  $k$  è una costante "grande", l'algoritmo si dice instabile nel senso della f.e.a.. Ad esempio, si osserva che *l'algoritmo per il calcolo del quadrato di un numero è stabile*, essendo

$$E'_{s^*} = 3u.$$

Nel calcolo della sottrazione, invece, applicando la f.e.a. si è ottenuto:

$$E'_{z^*} = (2u + \mathcal{O}(u^2)) \underbrace{\frac{|x| + |y|}{|x - y|}}_k.$$

Lo stesso algoritmo ha prodotto un risultato accettabile nel primo caso, essendo

$$E'_{z^*} \simeq ku \quad \text{con} \quad k = 0.3$$

ovvero l'algoritmo è stabile nel senso della f.e.a., mentre, nel secondo caso, il risultato NON è accettabile essendo

$$E'_{z^*} \simeq ku \quad \text{con} \quad k = 4310$$

ovvero l'algoritmo è instabile nel senso della f.e.a. . La f.e.a. fornisce, quindi, in relazione al calcolo della differenza, un fattore di amplificazione dell'errore che dipende sia dall'algoritmo che dai dati del problema.

Nasce, quindi, traendo spunto da questo caso, la necessità di isolare l'amplificazione dell'errore introdotto dalle operazioni dell'algoritmo, indipendentemente dai dati del problema. A tale scopo un'idea è quella di ricondurre tutti gli errori introdotti dalle operazioni floating point a precisione finita, ad errori sui dati, assumendo, quindi, che le operazioni dell'algoritmo siano, di fatto, eseguite in aritmetica a precisione *infinita*. In tal modo, infatti, la perturbazione, oramai ricondotta sul dato iniziale, sarà quella indotta dall'algoritmo ed una sua (eventuale) amplificazione sulla soluzione sarà quella indotta dal problema, ovvero dal (mal) condizionamento del problema. L'analisi dell'errore basata sull'idea di considerare la soluzione, calcolata dall'algoritmo in un sistema aritmetico a precisione finita, come soluzione esatta (ovvero ottenuta in aritmetica a precisione infinita) di un problema dello stesso tipo, con dati perturbati, viene detta **backward error analysis (b.e.a.)** (analisi all'indietro).

♦ **Esempio 1.37.** Assegnato un sistema aritmetico f.p.

$$F : (\beta = 10, t = 4, \underbrace{t_{reg}}_{max.acc.din.} = 2 \cdot t = 8),$$

in cui  $u = 0.5\beta^{1-t} = 0.5 \times 10^{-3}$ , ed i dati:

$$\begin{aligned} x &= 3.453749 \Rightarrow x^* = fl(x) = 0.3454 \times 10^1 \\ y &= 3.453432 \Rightarrow y^* = fl(y) = 0.3453 \times 10^1 \end{aligned}$$

calcolare la differenza

$$x^* \ominus y^*$$

In aritmetica esatta  $z = x - y = 0.000317$ ; nel sistema f.p. considerato:

$$\begin{aligned} z^* &= x^* \ominus y^* = \\ &= 0.1 \times 10^{-2} \end{aligned}$$

Volendo analizzare gli errori introdotti nel calcolo di  $z^* = x^* \ominus y^*$ , bisogna ricordare che l'errore sul dato  $x$  risulta:

$$\begin{aligned} \delta_1 &= \left| \frac{x - fl(x)}{x} \right| = \\ &= \left| \frac{0.3453749 \times 10^1 - 0.3454 \times 10^1}{0.3453749 \times 10^1} \right| = \\ &= 0.7 \times 10^{-4} \end{aligned}$$

da cui, in particolare,

$$fl(x) = x(1 + \delta_1) \Leftrightarrow 0.3454 \times 10^1 = 3.453749(1 + 0.7 \times 10^{-4})$$

Analogamente l'errore sul dato  $y$  risulta:

$$\begin{aligned} \delta_2 &= \left| \frac{y - fl(y)}{y} \right| = \\ &= \left| \frac{0.3453432 \times 10^1 - 0.3453 \times 10^1}{0.3453432 \times 10^1} \right| = \\ &= 0.1 \times 10^{-3} \end{aligned}$$

da cui, in particolare,

$$fl(y) = y(1 + \delta_2) \Leftrightarrow 0.3453 \times 10^1 = 3.453432(1 + 0.1 \times 10^{-3})$$

Poiché

$$\begin{aligned} z^* &= fl(x) \ominus fl(y) = \\ &= 0.3454 \times 10^1 \ominus 0.3453 \times 10^1 \\ &= 0.1 \times 10^{-2} \end{aligned}$$

si ha

$$\begin{aligned} \rho_1 &= \frac{|z^* - z|}{|z|} = \left| \frac{0.1 \times 10^{-2} - 0.000317}{0.000317} \right| = \\ &= 0.2 \times 10^1 \end{aligned}$$

e, quindi,

$$z^* = z(1 + \rho_1) \Leftrightarrow 0.1 \times 10^{-2} = 0.000317(1 + 0.2 \times 10^1)$$

Quindi:

$$\begin{aligned} fl(x) &= x(1 + \delta_1) \Rightarrow 0.3454 \times 10 = (0.3453749 \times 10)(1 + 0.7 \times 10^{-4}) \\ fl(y) &= y(1 + \delta_2) \Rightarrow 0.3453 \times 10 = (0.3453432 \times 10)(1 + 0.1 \times 10^{-3}) \\ z^* = fl(fl(x) - fl(y)) &= z(1 + \rho_1) \Rightarrow 0.1 \times 10^{-2} = 0.000317(1 + 0.2 \times 10) \end{aligned} \quad (1.40)$$

Effettuando le operazioni in aritmetica a precisione infinita, dalle (1.40) segue:

$$\begin{aligned} z^* &= (0.3453749 \times 10)(1 + 0.7 \times 10^{-4}) \ominus (0.3453432 \times 10)(1 + 0.1 \times 10^{-3}) = \\ &= [(0.3453749 \times 10)(1 + 0.7 \times 10^{-4}) - (0.3453432 \times 10)(1 + 0.1 \times 10^{-3})] \times (1 + 0.2 \times 10) \\ &\simeq \underbrace{0.000317(1 + 0.1 \times 10^{-3})(1 + 0.7 \times 10^{-4})}_{(x-y)(1+\delta_1)(1+\delta_2)} \underbrace{(1+0.2\times 10)}_{(1+\rho_1)} \end{aligned}$$

da cui,

$$\begin{aligned} z^* &= x^* \ominus y^* = (x - y)(1 + \delta_1)(1 + \delta_2)(1 + \rho_1), \\ \text{con } &|\delta_1| \leq u, |\delta_2| \leq u, |\rho_1| \leq u \end{aligned}$$

Ovvero:

$$z^* = \underbrace{[x(1 + \delta_1)(1 + \delta_2)(1 + \rho_1)]}_{\underline{x}} - \underbrace{[y(1 + \delta_1)(1 + \delta_2)(1 + \rho_1)]}_{\underline{y}}$$

In questo modo gli errori introdotti dalla sottrazione eseguita a precisione finita sono stati ricondotti ad errori su  $x$  e su  $y$ . Per stimare quanto è grande la perturbazione ricondotta sui dati iniziali  $x$  e  $y$  calcoliamo la distanza relativa tra  $(x, y)$  e  $(\underline{x}, \underline{y})$ .

$$\begin{aligned} E'_{(\underline{x}, \underline{y})} &= \frac{\|(x, \underline{y}) - (x, y)\|}{\|(x, y)\|} = \\ &= \frac{|x(\delta_1 + \rho_1 + \delta_1\rho_1) - y(\delta_2 + \rho_1 + \delta_2\rho_1)|}{|x| + |y|} \leq \\ &\leq \frac{|x(2u + u^2)| + |y(2u + u^2)|}{|x| + |y|} = 2u + u^2 = 2u + \mathcal{O}(u^2) \simeq 2u \end{aligned} \quad (1.41)$$

Essendo, dunque,

$$E'_{(\underline{x}, \underline{y})} \simeq k'u \quad \text{con } k' = 2$$

la stima dell'errore relativo ottenuta è dello stesso ordine di grandezza della massima accuratezza relativa e, quindi,  $(\underline{x}, \underline{y})$  è un'approssimazione di  $(x, y)$  che dista da  $(x, y)$  di una quantità dello stesso ordine di grandezza della massima accuratezza relativa. Sostanzialmente  $(\underline{x}, \underline{y})$  è un'approssimazione di  $(x, y)$  *accettabile*. In questo caso, la perturbazione introdotta dalle operazioni eseguite dall'algoritmo e poi ricondotta sul dato iniziale, è dello stesso ordine di grandezza dell'errore di roundoff sul dato iniziale per cui **l'algoritmo della sottrazione è stabile nel senso della b.e.a.**



In generale, considerato un algoritmo, che risolve un problema di dati iniziali  $d$ , se si indicano con  $d^*$  e  $d$ , rispettivamente il dato perturbato ed il dato iniziale, la b.e.a. si propone di fornire una stima dell'errore relativo (assoluto):

$$\frac{\|d - d^*\|}{\|d\|} \quad (\|d - d^*\|),$$

dove  $\|\cdot\|$  è una norma fissata. In dettaglio, se

$$E'_s = \frac{\|d - d^*\|}{\|d\|} \leq k \cdot u$$

qualora  $k$  sia una costante "sufficientemente piccola", l'algoritmo si dice stabile nel senso della b.e.a.; se, invece,  $k$  è una costante "grande", l'algoritmo si dice instabile nel senso della b.e.a. .

Nell'esempio relativo al calcolo della sottrazione:

$$E'_{z^*} \simeq 2u \underbrace{\frac{|x| + |y|}{|x - y|}}_k$$

Ricordiamo che, dall'analisi del condizionamento del problema della sottrazione, l'indice di condizionamento della sottrazione è

$$k = \frac{|x| + |y|}{|x - y|},$$

per cui, in corrispondenza dei dati

$$\begin{aligned} x &= 7.41330 \\ y &= 3.453749 \end{aligned}$$

si deduce  $k = 2.7445$  ed il problema risulta *ben condizionato*; al contrario, per

$$\begin{aligned} x &= 3.453749 \\ y &= 3.453432 \end{aligned}$$

si ha  $k = 2.17 \times 10^4$ , costante "grande" per cui il problema è *mal condizionato*. Quindi, per l'algoritmo della sottrazione, dalla **f.e.a.** si deduce:

$$E'_s \simeq 2 \frac{|x| + |y|}{|x - y|} u$$

ovvero *l'errore si approssima con una costante che dipende dall'algoritmo e dal condizionamento del problema*; dalla **b.e.a.**:

$$E'_d \simeq 2u$$

ovvero *l'errore si approssima con una costante che dipende solo dall'algoritmo*. In conclusione l'algoritmo della sottrazione è stabile nel senso della b.e.a. ma può fornire un risultato inaccettabile se i numeri sono vicini, perché è mal condizionato.

In generale, la f.e.a. fornisce:

$$E'_s \simeq ku$$

ovvero l'errore si approssima con una costante  $k$  che dipende dall'algoritmo e dal condizionamento del problema; per la **b.e.a.**, invece,

$$E'_d \simeq k'u$$

ovvero l'errore si approssima con una costante  $k'$  che dipende solo dall'algoritmo.  
Siano

$$\begin{aligned} d &= \text{dati iniziali} \\ S(d) &= \text{soluzione esatta} \\ \tilde{d} &= \text{dati perturbati} \\ \tilde{S}(d) &= \text{soluzione calcolata} \end{aligned}$$

Vale, dunque, lo schema seguente:

$$\begin{array}{lll} E'_{\tilde{S}(d)} & = E'_{S(\tilde{d})} & \Rightarrow \text{Backward error analysis:} \\ & \downarrow & \text{analisi dell'algoritmo} \\ E'_{S(\tilde{d})} & = \mu E'_{\tilde{d}} & \Rightarrow \text{Stima del condizionamento:} \\ & \downarrow & \text{analisi del problema} \\ E'_{\tilde{d}} & = ku & \Rightarrow \text{risultato della b.e.a.} \\ & \overbrace{\hspace{10em}}^{\downarrow} & \\ & E'_{\tilde{S}(d)} & = \mu ku \end{array}$$

con

- $\mu$  dipendente dal problema (indice di cond. del probl.)
- $k$  dipendente dall'algoritmo (fattore di amplif. dell'errore)
- $u$  dipendente dalla precisione del sistema (max acc. rel.)

Se il problema è ben condizionato ( $\mu \leq 1$ ) l'errore sui dati non viene amplificato ed *un algoritmo stabile nel senso della b.e.a. ( $k < 1$ ) fornisce un risultato accettabile*; se il problema è mal condizionato ( $\mu > 1$ ) l'errore sui dati viene amplificato ed *anche un algoritmo stabile nel senso della b.e.a. ( $k < 1$ ) fornisce un risultato inaccettabile*.

Una rappresentazione grafica degli errori stimati dalla f.e.a e dalla b.e.a e, più in generale, delle relazioni che intercorrono tra i due metodi di analisi dell'errore, è fornita in Figura 1.7.

♣ **Esempio 1.38.** Sia assegnato un sistema aritmetico f. p.

$$F : (\beta = 10, t = 3, \underbrace{t_{reg} = 2 \cdot t}_{max.acc.din.} = 6),$$

in cui  $u = 0.5\beta^{1-t} = 0.5 \times 10^{-2}$ . Se  $x = 3.453$  risulta

$$x^* = fl(x) = 0.345 \times 10^1 = 3.45.$$

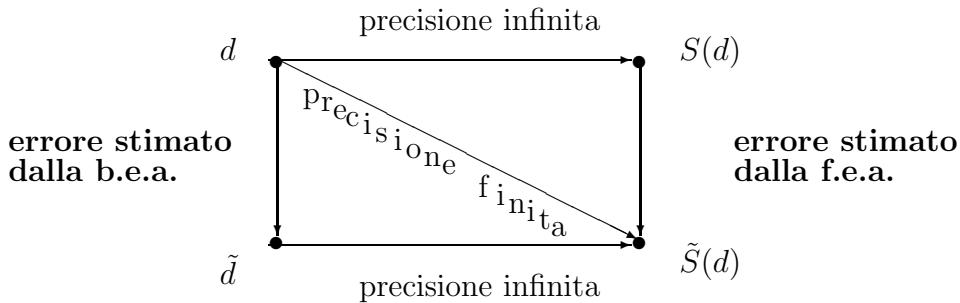


Figura 1.7: Errori stimati con la forward error analysis e con la backward error analysis.

Calcolare il quadrato di  $x$ . In aritmetica esatta  $x^2 = 3.453 \times 3.453 = 11.923209$ . In  $F$ , invece,

$$s^* = x^* \otimes x^* = 0.345 \times 10 \otimes 0.345 \times 10 = 0.119 \times 10^2 = 11.9$$

Per studiare gli errori introdotti nel calcolo di  $x^2$  in  $F$ , si ricorda che, l'errore sul dato è il seguente:

$$\begin{aligned} \delta_1 &= \left| \frac{x - fl(x)}{x} \right| = \\ &= \left| \frac{0.3453 \times 10^1 - 0.345 \times 10^1}{0.3453 \times 10^1} \right| = \\ &= 0.8 \times 10^{-4} \end{aligned}$$

da cui,

$$fl(x) = x(1 + \delta_1) \Leftrightarrow 0.345 \times 10 = 3.453(1 + 0.8 \times 10^{-4}) \quad (1.42)$$

mentre, dall'essere

$$s^* = fl(fl(x) \times fl(x)) = 0.345 \times 10 \otimes 0.345 \times 10 = 0.119 \times 10^2 \quad (1.43)$$

l'errore sulla operazione risulta:

$$\begin{aligned} \delta_2 &= \left| \frac{s - s^*}{s} \right| = \\ &= \frac{0.11923209 \times 10^2 - 0.119 \times 10^2}{0.11923209 \times 10^2} = 0.2 \times 10^{-2} \end{aligned}$$

da cui:

$$s^* = s(1 + \delta_2) \Leftrightarrow 0.119 \times 10^2 = (0.11923209 \times 10^2)(1 + 0.2 \times 10^{-2}) \quad (1.44)$$

Dalla (1.43), tenendo presente le (1.42) e (1.44), eseguendo i calcoli in precisione infinita, si ottiene:

$$\begin{aligned} &= 3.453(1 + 0.8 \times 10^{-4}) \otimes 3.453(1 + 0.8 \times 10^{-4}) = \\ &= [3.453(1 + 0.8 \times 10^{-4}) \times 3.453(1 + 0.8 \times 10^{-4})] \times (1 + 0.2 \times 10^{-2}) = \\ &= \underbrace{11.923209(1 + 0.8 \times 10^{-4})^2}_{x^2(1+\delta_1)^2} \underbrace{(1 + 0.2 \times 10^{-2})}_{(1+\delta_2)} \end{aligned}$$

Sostanzialmente risulta:

$$s^* = x^* \otimes x^* = x^2(1 + \delta_1)^2(1 + \delta_2) \quad \text{con} \quad |\delta_1| \leq u, |\delta_2| \leq u$$

Più in dettaglio:

$$\begin{aligned} s^* &= 11.923209(1 + 0.8 \times 10^{-4})^2(1 + 0.2 \times 10^{-2}) = \\ &= (11.923209)^{\frac{1}{2}}(1 + 0.8 \times 10^{-4})(1 + 0.2 \times 10^{-2})^{\frac{1}{2}} \times \\ &\quad \times (11.923209)^{\frac{1}{2}}(1 + 0.8 \times 10^{-4})(1 + 0.2 \times 10^{-2})^{\frac{1}{2}} \end{aligned}$$

e quest'ultimo è il risultato della moltiplicazione eseguita in aritmetica a precisione infinita:

$$s^* = \underbrace{[x(1 + \delta_1)\sqrt{1 + \delta_2}]}_{\underline{x}} \times \underbrace{[x(1 + \delta_1)\sqrt{1 + \delta_2}]}_{\underline{x}}$$

Con il procedimento descritto, gli errori introdotti dalla moltiplicazione eseguita a precisione finita sono ricondotti ad errori su  $x$ . Per eseguire una stima della perturbazione ricondotta sul dato iniziale  $x$  si può calcolare la distanza relativa tra  $x$  e  $\underline{x}$ , ovvero l'errore relativo su  $x$ :

$$\begin{aligned} E'_{\underline{x}} &= \frac{|x - \underline{x}|}{|x|} = \frac{|x(1 + \delta_1)\sqrt{1 + \delta_2} - x|}{|x|} \\ &\leq |\sqrt{1 + \delta_2} - 1| + |\delta_1\sqrt{1 + \delta_2}| \leq \\ &\leq 2u + u^2 = 2u + \mathcal{O}(u^2) \simeq 2u \end{aligned}$$

da cui si deduce che l'errore relativo in  $\underline{x}$  è maggiorato, a meno di  $\mathcal{O}(u^2)$ , da  $2u$ ; sostanzialmente  $\underline{x}$  è un'approssimazione di  $x$  che dista da  $x$  di una quantità dello stesso ordine di grandezza della massima accuratezza relativa, cioè:

$$E'_{\underline{x}} \simeq k'u \quad \text{con} \quad k' = 2$$

ovvero  $\underline{x}$  è un'approssimazione di  $x$  accettabile. In questo caso la perturbazione introdotta dalle operazioni eseguite dall'algoritmo, e poi ricondotta sul dato iniziale, è dello stesso ordine di grandezza dell'errore di roundoff sul dato iniziale, per cui **l'algoritmo per il calcolo del quadrato è stabile nel senso della b.e.a.**



♣ **Esempio 1.39.** Siano assegnati  $x, y, z \in F$ , dove  $F$  è un opportuno sistema aritmetico f.p., ovvero siano

$$\begin{aligned} \bar{x} &= fl(x) = x(1 + \delta_1) \\ \bar{y} &= fl(y) = y(1 + \delta_2) \\ \bar{z} &= fl(z) = z(1 + \delta_3) \end{aligned}$$

con  $\delta_i = 0$ ,  $i = 1, 2, 3$ . Calcolare, in  $F$ ,

$$\tilde{a} = x \oplus y \otimes z$$

Nel sistema aritmetico  $F$ , si devono eseguire, dunque, una *moltiplicazione f.p.* ed un'*addizione f.p.*.  
Posto

$$\tilde{w} = yz(1 + \rho_1),$$

il calcolo di  $\tilde{a}$  si può ricondurre all'operazione f.p.

$$\tilde{a} = x \oplus \tilde{w} = (x + \tilde{w})(1 + \rho_2)$$

e, quindi,

$$\tilde{a} = (x + (yz(1 + \rho_1)))(1 + \rho_2)$$

Applicando la b.e.a. si ha

$$\begin{aligned}\tilde{a} &= (x + (yz(1 + \rho_1)))(1 + \rho_2) = (x + yz + yz\rho_1)(1 + \rho_2) = \\ &= x + yz + yz\rho_1 + x\rho_2 + yz\rho_2 + yz\rho_1\rho_2 = \\ &= x + x\rho_2 + yz + yz\rho_1 + yz\rho_2 + yz\rho_1\rho_2 = \\ &= x(1 + \rho_2) + y(1 + \rho_2)z(1 + \rho_1)\end{aligned}\tag{1.45}$$

Allora  $\tilde{a}$  si può considerare come il risultato *esatto* delle operazioni eseguite tra

$$\bar{x} = x(1 + \rho_2), \quad \bar{y} = y(1 + \rho_2) \quad \text{e} \quad \bar{z} = z(1 + \rho_1).$$



## 1.9 Esempio di studio: risoluzione di un'equazione di secondo grado

Si consideri la generica equazione di secondo grado:

$$ax^2 + bx + c = 0, \tag{1.46}$$

dove, per semplicità, si suppongono  $a$ ,  $b$  e  $c$  reali. E' noto che, se  $a \neq 0$ , le radici di (1.46) hanno la seguente espressione:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}, \tag{1.47}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}. \tag{1.48}$$

E' naturale scrivere un algoritmo risolutivo che applichi direttamente (1.47) e (1.48):

```

:
q := -b
y := b2
w := 4 × a
v := w × c
p := y - v
d := √p
s1 := q + d
s2 := q - d
r := 2 × a
x̄1 := s1/r
x̄2 := s2/r
:

```

**Procedura 1.5: Algoritmo per la risoluzione di un'equazione di secondo grado**

Fissato il sistema aritmetico f.p. con  $F(10, 8, 50, -50)$ , i quattro esempi seguenti mostrano l'applicazione dell'algoritmo 5 alla risoluzione di alcune equazioni di secondo grado.

1.  $6x^2 + 5x - 4 = 0$ .

Le radici sono:

$$x_1 = 0.5, \quad x_2 = -1.333333\dots,$$

quelle ottenute come risultato dell'algoritmo 5 sono:

$$\bar{x}_1 = 0.50000000, \quad \bar{x}_2 = -1.333333;$$

in questo caso l'algoritmo 5 fornisce la massima accuratezza del risultato.

2.  $x^2 - 10^5x + 1 = 0$ .

L'equazione ha le seguenti radici, rappresentate con undici cifre significative:

$$x_1 = 99999.999990, \quad x_2 = 0.000010000000001.$$

L'esecuzione dell'algoritmo 5 fornisce:

$$\begin{aligned}
q &= -b = 0.10000000 \times 10^6 \\
y &= b^2 = 0.10000000 \times 10^{11} \\
w &= 4 \times a = 0.40000000 \times 10^1 \\
v &= w \times c = 0.40000000 \times 10^1
\end{aligned}$$

$$\begin{aligned}
p &= y - v = 0.10000000 \times 10^{11} - 0.40000000 \times 10^1 = \\
&= 0.10000000 \times 10^{11}
\end{aligned}$$

$$\begin{aligned} d &= \sqrt{p} = 0.10000000 \times 10^6 \\ s_1 &= q + d = 0.20000000 \times 10^6; & s_2 &= q - d = 0 \\ r &= 2 \times a = 0.20000000 \times 10^1 \\ \boxed{\bar{x}_1 = s_1/r = 0.10000000 \times 10^6}; & \quad \boxed{\bar{x}_2 = s_2/r = 0} \end{aligned}$$

Gli errori relativi delle soluzioni calcolate dall'algoritmo sono:

$$E'_1 = \frac{|x_1 - \bar{x}_1|}{|x_1|} \simeq 10^{-7}, \quad E'_2 = \frac{|x_2 - \bar{x}_2|}{|x_2|} = 1.$$

Dunque l'algoritmo fornisce un'ottima approssimazione di  $x_1$  (massima accuratezza), ma fornisce un'approssimazione di  $x_2$  affetta da un errore relativo del 100%. Ciò è dovuto al fatto che nel calcolo di  $p$ , in aritmetica a precisione finita, si perde completamente il contributo di  $v$  e, quindi, il calcolo di  $s_2$  diventa una sottrazione tra due numeri uguali (fenomeno di cancellazione).

3.  $x^2 - 4x + 3.9999999 = 0$ .

Le radici, rappresentate con otto cifre significative, sono:

$$x_1 = 2.0003162, \quad x_2 = -1.9996838.$$

Applicando l'algoritmo 5 si ha:

$$\begin{aligned} q &= -b = 0.40000000 \times 10^1 \\ y &= b^2 = 0.16000000 \times 10^2 \\ w &= 4 \times a = 0.40000000 \times 10^1 \\ \boxed{v = w \times c = (0.40000000 \times 10^1) \times (0.39999999 \times 10^1) =} \\ &\quad = 0.16000000 \times 10^2 \end{aligned}$$

$$\begin{aligned} p &= y - v = 0 \\ d &= \sqrt{p} = 0 \\ s_1 &= q + d = s_2 = q - d = 0.40000000 \times 10^1 \\ r &= 2 \times a = 0.20000000 \times 10^1 \\ \boxed{\bar{x}_1 = s_1/r = \bar{x}_2 = s_2/r = 0.20000000 \times 10^1} \end{aligned}$$

L'algoritmo calcola una radice doppia, invece di due radici semplici, e tale radice ha 4 cifre decimali in comune con  $x_1$ , cioè solo metà della precisione di  $F$ . In questo caso la difficoltà risiede nel fatto che per rappresentare la mantissa di  $v$  si hanno a disposizione solo 8 cifre.

4.  $10^{40}x^2 - 5 \times 10^{40}x + 6 \times 10^{40} = 0$ .

Le radici esatte dell'equazione sono:

$$x_1 = 3, \quad x_2 = 2;$$

applicando l'algoritmo 5 si ha:

$$\begin{aligned} q &= -b = 0.50000000 \times 10^{41} \\ y &= b^2 = 0.25000000 \times 10^{82} > \beta^{emax} = 10^{50} \\ &\vdots \end{aligned}$$

cioè si verifica un overflow. Anche se i coefficienti e le radici sono rappresentabili in  $F$ , l'algoritmo produce la situazione eccezionale di overflow, non fornendo alcuna informazione sulla soluzione. In generale, dato che le formule (1.47) e (1.48) richiedono il calcolo di  $b^2$  e  $4ac$ , si può verificare overflow o underflow anche quando i coefficienti e le radici sono, in modulo, “molto distanti” da  $\beta^{emax}$  e  $\beta^{emin}$  rispettivamente.

Tranne che nel caso 1, l'applicazione diretta delle formule (1.47) e (1.48) non fornisce risultati soddisfacenti.

I principali requisiti di un “buon” algoritmo, per il calcolo delle radici di un polinomio di secondo grado  $ax^2 + bx + c$ , sono i seguenti:

- I. se  $a = b = c = 0$ ,  
l'algoritmo deve segnalare che l'equazione è soddisfatta da ogni numero complesso  $x$ ;
- II. se  $a = b = 0$  e  $c \neq 0$ ,  
deve segnalare che l'equazione non ammette soluzioni;
- III. se  $a = 0$  e  $b \neq 0$ ,  
deve segnalare che l'equazione è di primo grado e calcolarne la soluzione con la formula  $x = -c/b$ ;
- IV. se  $a \neq 0$ ,  
deve calcolare le radici con un'accuratezza adeguata (evitando, quindi, cancellazioni e situazioni di overflow ed underflow nelle operazioni), se queste sono rappresentabili nel sistema aritmetico f.p. utilizzato e deve segnalare il verificarsi di situazioni eccezionali in caso contrario<sup>54</sup>.

Lo sviluppo di un algoritmo che abbia tutti i requisiti I-IV non è affatto semplice. Tralasciando il requisito dell'accuratezza in IV, è possibile tuttavia utilizzare “semplici”

---

<sup>54</sup>Una precisa specificazione di tale requisito è discussa in [4], dove si richiede ad esempio che, utilizzando un sistema aritmetico f.p. con  $F(\beta, t, emin, emax)$ , l'errore sulle ultime 2 cifre di ciascuna radice calcolata  $x^*$  non sia più grande di  $\beta + 1$  unità, cioè

$$\frac{|x - x^*|}{|x|} \leq 2u \frac{\beta + 1}{\beta}.$$

accorgimenti per superare alcune difficoltà come quelle incontrate nei precedenti esempi 2-4.

Le formule (1.47) e (1.48) possono essere riscritte, razionalizzando i numeratori, nel modo seguente:

$$x_1 = \frac{2c}{-b - \sqrt{b^2 - 4ac}}, \quad (1.49)$$

$$x_2 = \frac{2c}{-b + \sqrt{b^2 - 4ac}}. \quad (1.50)$$

Se nell'esempio 2 si utilizza (1.50) per calcolare  $x_2$  si ottiene:

$$x_2^* = \frac{0.20000000 \times 10^1}{0.10000000 \times 10^6 + 0.10000000 \times 10^6} = 0.10000000 \times 10^{-4},$$

cioè si calcola un valore di  $x_2$  con tutte le 8 cifre della mantissa corrette. In generale, supposto  $b \neq 0$  e  $b^2 - 4ac > 0$ , se  $b < 0$  le formule (1.50) e (1.47) evitano la sottrazione  $-b - \sqrt{b^2 - 4ac}$ , che può essere causa di un fenomeno di cancellazione; analogo discorso vale per le formule (1.48) e (1.49) nel caso  $b > 0$ . Si osservi che questo significa calcolare una delle radici con (1.47) o (1.48), a seconda che sia  $b < 0$  o  $b > 0$ , e l'altra mediante la relazione

$$x_1 x_2 = c/a. \quad (1.51)$$

Utilizzando (1.51) per calcolare  $x_2$ , nell'esempio 2, si ha lo stesso valore ottenuto con la formula (1.50):

$$x_2^* = \frac{0.10000000 \times 10^1}{(0.10000000 \times 10^6) \times (0.10000000 \times 10^1)} = 0.10000000 \times 10^{-4}.$$

La difficoltà che si presenta nell'esempio 3 può essere superata eseguendo in doppia precisione le operazioni relative al calcolo di  $v$ :

$$q = -b = 0.40000000 \times 10^1$$

$y = b^2 = 0.1600000000000000 \times 10^2$
$w = 4 \times a = 0.4000000000000000 \times 10^1$
$v = w \times c = 0.1599999960000000 \times 10^2$
$p = y - v = 0.4000000000000000 \times 10^{-6}$

$$d = \sqrt{p} = 0.6325 \times 10^{-3}$$

$$s_1 = q + d = 0.40006325 \times 10^1; \quad s_2 = q - d = 0.39993675 \times 10^1$$

$$r = 2 \times a = 0.20000000 \times 10^1$$

$\bar{x}_1 = s_1/r = 0.20003163 \times 10^1$	$\bar{x}_2 = s_2/r = 0.19996838 \times 10^1$
--	--

Tale tecnica permette di calcolare due soluzioni distinte, commettendo al più un errore sull'ultima cifra significativa.

E' da osservare che le difficoltà relative all'esempio 3 sono legate, più che all'algoritmo 5, all'equazione, ovvero al suo condizionamento. Secondo un approccio di tipo b.e.a., le radici coincidenti  $\bar{x}_1 = \bar{x}_2 = 2$  si possono vedere come soluzioni esatte dell'equazione

$$0.999999992x^2 - 3.99999968x + 3.99999968 = 0,$$

i cui coefficienti differiscono dai coefficienti omologhi dell'altra equazione per meno di 1 unità sull'ottava cifra significativa; quindi, a piccole variazioni nei dati corrispondono sensibili variazioni nei risultati.

Per finire, si consideri nuovamente l'esempio 4. Moltiplicando i coefficienti dell'equazione per  $10^{-40}$  si ha:

$$x^2 - 5x + 6 = 0,$$

le cui soluzioni si possono calcolare esattamente con l'algoritmo 5. In questo caso si dice che è stato effettuato uno *scaling dei coefficienti*. Per evitare gli errori di roundoff connessi con la moltiplicazione f.p., si scelgono fattori di scaling che sono potenze della base del sistema aritmetico utilizzato ( $10^{-40}$  nell'esempio). Si noti, tuttavia, che non esiste una tecnica di scaling applicabile in generale.<sup>55</sup>

Altre tecniche sono utilizzabili per evitare l'overflow e l'underflow. Ad esempio, se il calcolo di  $b^2$  provoca overflow, mentre  $a, b, c$  e  $4ac$  sono rappresentabili, se  $b \neq 0$ , si può calcolare  $\sqrt{b^2 - 4ac}$  secondo la formula:

$$\sqrt{b^2 - 4ac} = |b| \sqrt{1 - a \frac{2}{b} \frac{2}{b} c};$$

se, invece,  $b \neq 0, a, b, c$  sono rappresentabili e  $4ac$  produce underflow, si può calcolare una delle radici con la formula

$$x_i = -b/a \quad (1.52)$$

e l'altra sfruttando (1.51).<sup>56</sup>

---

<sup>55</sup>Lo scaling dei coefficienti non risolve sempre i problemi di overflow od underflow. Si consideri ad esempio l'equazione

$$10^{-30}x^2 - 10^{30}x + 10^{30} = 0,$$

le cui radici  $x_1$  e  $x_2$  sono "prossime" a  $10^{60}$  e ad 1. Un buon algoritmo risolutivo deve essere in grado di calcolare un'approssimazione di  $x_2$  conformemente alle richieste del punto IV; d'altra parte ogni tentativo di scalare i coefficienti dell'equazione moltiplicandoli per uno stesso fattore non migliora la situazione, anzi può provocare un overflow od un underflow. Questa equazione è infatti un difficile test per un algoritmo risolutivo.

In generale, in casi come questo, si può cercare di effettuare uno *scaling dei coefficienti e dell'incognita*, cioè di trasformare (1.46) in un'equazione

$$a'y^2 + b'y + c' = 0,$$

con

$$a' = \beta^h a, \quad b' = \beta^h b, \quad c' = \beta^h c, \quad x = \beta^k y,$$

che sia di più facile risoluzione [11].

<sup>56</sup>In tal modo si trascura  $4ac$  in (1.47) se  $b < 0$  o in (1.48) se  $b > 0$ . D'altra parte, nel sistema aritmetico di un elaboratore, un valore che provoca underflow viene posto, generalmente, uguale a 0 e, quindi, la (1.52) si può vedere come conseguenza immediata dell'applicazione della corrispondente formula risolutiva.

A conclusione di questa discussione, si può, quindi, evidenziare come, anche la risoluzione di un problema matematico semplice, quale è la risoluzione di un'equazione di secondo grado, possa risultare estremamente complesso utilizzando un sistema aritmetico f.p. a precisione finita.

## 1.10 Esercizi sull'aritmetica Floating-Point

Nel seguito si indicherà con  $\mathfrak{S}$  un sistema aritmetico *floating-point* a precisione finita

$$\mathfrak{S} = (\beta, t, emin, emax)$$

con  $\beta$ =base,  $t$ = precisione,  $emin$  = esponente minimo,  $emax$  = esponente massimo; i valori numerici dei parametri che lo caratterizzano saranno specificati, laddove occorre.

### 1.10.1 Alcuni esercizi numerici

**Esercizio 1** Utilizzando il troncamento rappresentare:

- In  $\mathfrak{S}$ :  $\beta = 10$ ,  $t=5$ ,  $emin = -50$ ,  $emax = +50$

- a) 2.718218285
- b) -1073741824
- c) 0.577216
- d)  $-123 \times 10^{-45}$

- In  $\mathfrak{S}$ :  $\beta = 10$ ,  $t=4$ ,  $emin = -9$ ,  $emax = +9$

- a) 989273
- b) 0.00000000000001
- c)  $-0.34 \times 10^5$
- d)  $-23 \times 10^{-2}$

**Esercizio 2** In  $\mathfrak{S}$ :  $\beta = 10$ ,  $t=3$ ,  $emin = -2$ ,  $emax = 2$ , si rappresentino i seguenti numeri e si calcoli *il minimo numero rappresentabile* e *la massima accuratezza relativa*:

$$\begin{aligned}x_1 &= 222.13 \\x_2 &= 0.2 \\x_3 &= 0.056 \\x_4 &= 3.467864\end{aligned}$$

**Esercizio 3** Sia  $\mathfrak{S}$ :

$$\mathfrak{S} : \beta = 10, t = 3, emin = -9, emax = 9$$

si calcolino:

- (a) il massimo numero reale positivo rappresentabile;

- (b) il minimo numero reale positivo rappresentabile;
- (c) l' $\epsilon_{mac}$  in  $\mathfrak{F}$ ;
- (d) si esegua l'addizione seguente e si calcoli l'errore relativo introdotto nel risultato:

$$57.46 + 1.8888$$

**Esercizio 4** Sia

$$\mathfrak{F} : \quad \beta = 10, t = 3, emin = -3, emax = 3.$$

Si rappresentino in  $\mathfrak{F}$  i numeri seguenti e si indichi l'errore relativo che si commette nella loro rappresentazione.

- a)  $x_1 = 12.13$
- b)  $x_2 = 1.2$
- c)  $x_3 = 0.005678$
- d)  $x_4 = 3467.864$

**Esercizio 5** Siano  $I$  ed  $\mathfrak{F}$  rispettivamente un sistema aritmetico intero ed uno floating-point a precisione finita, di parametri rispettivamente:

$$I : \quad \beta = 10 \quad t = 3 \quad imin = -999 \quad imax = 999$$

$$\mathfrak{F} : \quad \beta = 10 \quad t = 3 \quad emin = -9 \quad emax = 9$$

Si eseguano le seguenti operazioni segnalando eventuali "situazioni eccezionali".

1.) **In  $I$ :**

- (a)  $\frac{2}{4}$ ;
- (b)  $-\frac{96}{55}$ ;
- (c)  $50 \times (-600)$ .

2.) **In  $\mathfrak{F}$ :**

- (a)  $\frac{2}{4}$ ;
- (b)  $0.0001 \times 0.000000986$ ;
- (c)  $2.568+355.66$ .

**Esercizio 6** Sia:

$$\mathfrak{F} : \beta = 10, t = 3, emin = -9, emax = 9.$$

Rappresentare x, y, z e w dove:

$$x = \frac{1}{2}, y = -78666666/0.0078949, z = -0.0001 \times 0.000000789$$

$$w = 4.567 + 255.89$$

**Esercizio 7** Siano  $\mathfrak{F}_1$  e  $\mathfrak{F}_2$  due sistemi aritmetici floating-point a precisione finita di parametri:

$$\begin{aligned} \mathfrak{F}_1 : \beta_1 &= 2 & t_1 &= 23 \\ \mathfrak{F}_2 : \beta_2 &= 10 & t_2 &=? \end{aligned}$$

Determinare  $t_2$  in maniera che i due sistemi abbiano la stessa massima accuratezza relativa.

**Esercizio 8** Sia:

$$\mathfrak{F} : \beta = 10, t = 2, emax = 3, emin = -3.$$

quanti numeri macchina contiene  $\mathfrak{F}$ ?

**Esercizio 9** Sia:

$$\mathfrak{F} : \beta = 10, t = 4, emax = 9, emin = -9.$$

dotato della massima accuratezza dinamica. Rappresentare i numeri:

$$x = 764.65 \quad y = 0.000000098701$$

e determinare gli epsilon macchina  $eps_x$  e  $eps_y$  relativi a ciascuno di essi (supponendo  $t_{reg} > t$ ).

**Esercizio 10** Sia:

$$\mathfrak{F} : \beta = 10, t = 2, emax = 9, emin = -9.$$

determinare quanti sono i numeri macchina in  $\mathfrak{F}$  e rappresentare il numero  $x = 2.12$  in tale sistema aritmetico.

**Esercizio 11** Sia:

$$\mathfrak{S} : \beta = 10, t = 6, emax = 20, emin = -20.$$

eseguire in  $\mathfrak{S}$  le seguenti operazioni:

- a)  $1 + 10^7$
- b)  $1 + 10^3$
- c)  $1 + 10^{-7}$

**Esercizio 12** Sia:

$$\mathfrak{S} : \beta = 10, t = 4, emax = 9, emin = -9.$$

si supponga che in tale sistema aritmetico sia presente un *registro* per eseguire i calcoli in doppia precisione con precisione  $t_{reg} = 8$ . Verificare che il sistema aritmetico considerato non gode della proprietà associativa calcolando  $a + b + c$ , con a, b, e c:

$$\begin{aligned} a &= 0.6472 \times 10^0 \\ b &= 0.4685 \times 10^{-4} \\ c &= 0.3297 \times 10^{-4} \end{aligned}$$

**Esercizio 13** Sia:

$$\mathfrak{S} : \beta = 2, t = 7, emax = 7, emin = -7.$$

dotato di massima accuratezza dinamica. Rappresentare  $x = 9.6$  e  $z = 4.2$  in  $\mathfrak{S}$ . Trovare due numeri  $y \in \mathfrak{S}$  e  $t \in \mathfrak{S}$  tali che  $x + y = x$  e  $z + t = z$ .

**Esercizio 14** Dati  $x = 123.4578$  e  $\bar{x} = 123.4599$  calcolare l'errore assoluto  $E_a$  e l'errore relativo  $E_r$ . Contare il numero di cifre significative e decimali.

**Esercizio 15** Dati  $x = 0.000418$  e  $\bar{x} = 0.000412$  calcolare l'errore assoluto  $E_a$  e l'errore relativo  $E_r$ . Contare il numero di cifre significative e decimali.

**Esercizio 16** Sia:

$$\mathfrak{S} : \beta = 10, t = 5, emin = -9, emax = 9.$$

Si determini la massima accuratezza relativa  $\mu$ . Rappresentare, mediante arrotondamento, i numeri  $x_1 = 123.4578$ ,  $x_2 = 0.0215984$ , calcolare inoltre l'errore relativo di round-off e verificare che esso è minore della massima accuratezza relativa.

**Esercizio 17** Sommare le seguenti coppie di numeri in un sistema aritmetico a massima accuratezza dinamica:  $0.54321 \times 10^{-1} + 0.76543 \times 10^2$  e  $0.49854 \times 10^3 + 0.21485 \times 10^{-3}$ .

**Esercizio 18** Assumiamo un sistema f.p. con base  $\beta = 10$ ,  $t = 4$ , precisione relativa della macchina  $\epsilon_{mac} = 10^{-5}$  ed esponente massimo e minimo  $emax = +20$  ed  $emin = -20$ , rispettivamente. Rappresentare il risultato delle operazioni f.p. seguenti

- (a)  $1 + 10^{-7}$
- (b)  $1 + 10^3$
- (c)  $1 + 10^7$
- (d)  $10^{10} + 10^3$
- (e)  $10^{10}/10^{15}$
- (f)  $10^{-10} \times 10^{-15}$

**Esercizio 19** In un sistema aritmetico avente una soglia di  $UFL = 10^{-38}$  si eseguano le operazioni:

$$(I) \quad a = \sqrt{y^2 + z^2} \quad (II) \quad a = \sqrt{w^2 + z^2} \quad (III) \quad u = (v \times y)/(w \times z)$$

con  $y = 1, v = 10^{-15}, w = 10^{-20}$  e  $z = 10^{-25}$

## Risposte

### Esercizio 1

- In  $\mathfrak{S}$ :  $\beta = 10$ ,  $t=5$ ,  $emin = -50$ ,  $emax = +50$

- a)  $0.27182 \times 10^1$   
 b)  $-0.10737 \times 10^{10}$   
 c)  $0.57722 \times 10^0$   
 d)  $-0.123 \times 10^{-42}$

- In  $\mathfrak{S}$ :  $\beta = 10$ ,  $t=4$ ,  $emin = -9$ ,  $emax = +9$

- a)  $0.9893 \times 10^6$   
 b) **Underflow**  
 c)  $-0.3400 \times 10^5$   
 d)  $-0.2300 \times 10^0$

**Esercizio 2** In  $\mathfrak{S}$ :  $\beta = 10$ ,  $t=3$ ,  $emin = -2$ ,  $emax = 2$ :

$\tilde{x}_1$ → <b>Overflow</b>	Non Rappresentabile in $\mathfrak{S}$
$\tilde{x}_2 = 0.200 \times 10^0$	Esattamente rappresentabile in $\mathfrak{S}$
$\tilde{x}_3 = 0.560 \times 10^{-1}$	Esattamente rappresentabile in $\mathfrak{S}$
$\tilde{x}_4 = 0.346 \times 10^1$	Non esattamente rappresentabile in $\mathfrak{S}$

Il minimo numero rappresentabile in  $\mathfrak{S}$  (troncamento) è:

$$\beta^{emin-1} = 10^{-2-1} = 10^{-3}$$

La massima accuratezza relativa è:

$$\beta^{1-t} = 10^{1-3} = 10^{-2}$$

### Esercizio 3

(a) il massimo numero rappresentabile è:

$$rmax = \beta^{emax}(1 - \beta^{-t}) = 0.999 \times 10^9$$

(b) il minimo numero rappresentabile è:

$$r_{min} = \beta^{e_{min}-1} = 0.1 \times 10^{-10}$$

(c) l' $\epsilon_{mac}$  è:

$$\beta^{1-t} = 10^{1-3} = 10^{-2}$$

(d) Eseguiamo l'addizione una volta rappresentati in  $\mathfrak{S}$  i numeri:

$$\begin{aligned} x_1 &= 57.46 & \tilde{x}_1 &= 0.574 \times 10^2 \\ x_2 &= 1.8888 & \tilde{x}_2 &= 0.188 \times 10^1 \end{aligned}$$

$$\tilde{x}_1 + \tilde{x}_2 = 0.574 \times 10^2 + 0.018 \times 10^2 = 0.592 \times 10^2$$

L'errore relativo è:

$$E_{rel} = \frac{|0.593488 \times 10^2 - 0.592 \times 10^2|}{0.593488 \times 10^2} \simeq 0.0025072 \simeq 0.250 \times 10^{-2}$$

#### Esercizio 4

- a)  $\tilde{x}_1 = 0.121 \times 10^3$  Non rappresentabile esattamente in  $\mathfrak{S}$
- b)  $\tilde{x}_2 = 0.120 \times 10^2$  Rappresentabile esattamente in  $\mathfrak{S}$
- c)  $\tilde{x}_3 = 0.567 \times 10^{-2}$  Non rappresentabile esattamente in  $\mathfrak{S}$   
*Overflow*
- d)

Per quanto riguarda gli errori relativi si ha:

$$E_{1rel} = \left| \frac{0.1213 \times 10^2 - 0.121 \times 10^2}{0.1213 \times 10^2} \right| = 0.00003639 = 0.363 \times 10^{-4}$$

$$E_{2rel} = \left| \frac{0.12 \times 10^2 - 0.12 \times 10^2}{0.12 \times 10^2} \right| = 0$$

$$E_{3rel} = \left| \frac{0.5678 \times 10^{-2} - 0.567 \times 10^{-2}}{0.5678 \times 10^{-2}} \right| = 0.00045424 = 0.454 \times 10^{-3}$$

#### Esercizio 5

1.) In  $I$  si ha:

(a)  $\frac{2}{4} = 0$ ;

(b)  $-\frac{96}{55} = -1.745454545 \dots$

(c)  $50 \times (-600) = -30000$  (Overflow).

2.) In  $\mathfrak{S}$  si ha:

(a)  $\frac{2}{4} = 0.5 \times 10^0$ ;

(b)  $(0.1 \times 10^{-3}) \times (0.986 \times 10^{-6}) =$  (Underflow)  $= 0$

(c)  $x_1 = 2.568 \quad \tilde{x}_1 = 0.256 \times 10^1$

$x_2 = 355.66 \quad \tilde{x}_2 = 0.355 \times 10^3$

$$\tilde{x}_1 + \tilde{x}_2 = 0.355 \times 10^3 + 0.00256 \times 10^2 = 0.357 \times 10^3$$

### Esercizio 6

$$\tilde{x} = 0.5 \times 10^0$$

$$\tilde{y} = 0.997 \times 10^{10} \text{ (Overflow)}$$

$$z = -0.789 \times 10^{-10} \text{ (Underflow)}$$

$$\tilde{w} = 0.457 \times 10^1 + 0.256 \times 10^3 =$$

$$= 0.00457 \times 10^3 + 0.256 \times 10^3 =$$

$$= 0.260 \times 10^3$$

**Esercizio 7** Sia  $u_1$  la massima accuratezza relativa di  $\mathfrak{S}_1$  ed  $u_2$  la massima accuratezza relativa di  $\mathfrak{S}_2$ . Si ha:

$$u_1 = \frac{1}{2} \times \beta_1^{1-t_1} \Rightarrow u_1 = \frac{1}{2} \times 2^{1-23} = 2^{-23}$$

Poiché:

$$u_1 = u_2 = \frac{1}{2} \times 10^{1-t_2} \Rightarrow 10^{1-t_2} = 2u_1 = 2^{-22}$$

da cui,

$$t_2 = 1 - \log_{10} 2u_1 \Rightarrow t_2 = 1 - \log_{10} 2^{-22} = 1 + 22 \log_{10} 2$$

quindi

$$t_2 = 1 + 22 \cdot \frac{\log_{10} 10}{\log_{10} 2} \simeq 1 + 6 = 7$$

**Esercizio 8** I numeri macchina presenti in  $\mathfrak{I}$  sono:

$$2(\beta - 1)\beta^{t-1}(emax - emin + 1) + 1$$

e dunque:

$$2 \times 9 \times 10 \times 7 + 1 = 1261$$

**Esercizio 9**

$$\begin{aligned}\tilde{x} &= 0.7646 \times 10^3; \quad \tilde{y} = 0.9870 \times 10^{-8} \\ \varepsilon_{mac} &= \frac{1}{2}\beta^{1-t} = 0.5 \times 10^{1-4} = 0.5 \times 10^{-3} \\ eps_x &= 0.7646 \times 10^3 \times 0.5 \times 10^{-3} = 0.3823 \times 10^0 \\ eps_y &= 0.987 \times 10^{-8} \times 0.5 \times 10^{-3} = UFL \text{ (Underflow)}\end{aligned}$$

**Esercizio 10** I numeri macchina in  $\mathfrak{I}$  sono:

$$2(\beta - 1)\beta^{t-1}(emax - emin + 1) + 1$$

ovvero

$$2 \times 9 \times 10 \times 18 + 1 = 3241$$

inoltre dato  $x = 2.12$ , la sua rappresentazione in  $\mathfrak{I}$  è:

$$\tilde{x} = 0.21 \times 10^1$$

dunque  $x$  non è rappresentabile esattamente in  $\mathfrak{I}$  e

$$E_{rel} = \frac{|0.212 \times 10^1 - 0.21 \times 10^1|}{0.21 \times 10^1} = 0.009433 \approx 0.943 \times 10^{-2}$$

**Esercizio 11** Eseguendo le operazioni, si ottiene:

- a)  $0.1 \times 10^1 + 0.1 \times 10^8 = 0.00000001 \times 10^8 + 0.1 \times 10^8 = 0.1 \times 10^8$
- b)  $0.1 \times 10^4 + 0.1 \times 10^1 = 0.1000 \times 10^4 + 0.0001 \times 10^4 = 0.1001 \times 10^4$
- c)  $0.1 \times 10^1 + 0.1 \times 10^{-6} = 0.1 \times 10^1 + 0.00000001 \times 10^1 = 0.1 \times 10^1$

**Esercizio 12** Sommiamo nell'ordine (a+b)+c:

Dati	Memoria	Registro
$a$	$0.6472 \times 10^0$	
$b$	$0.4685 \times 10^{-4}$	$0.00004685 \times 10^0$
$a + b$		$0.64724685 \times 10^0$
$fl(a + b)$	$0.6472 \times 10^0$	
$c$	$0.3297 \times 10^{-4}$	$0.00003297 \times 10^0$
$fl(a + b) + c$		$0.64723297 \times 10^0$
$fl(fl(a + b) + c)$		$0.6472 \times 10^0$

Il risultato è uguale ad  $a$  perché  $b$  e  $c$  sono entrambi molto più piccoli del primo.  
Sommiamo adesso nell'ordine  $(b+c)+a$ :

Dati	Memoria	Registro
$c$	$0.3297 \times 10^{-4}$	$0.00003297 \times 10^0$
$b + c$		$0.00007982 \times 10^0$
$fl(b + c)$	$0.7982 \times 10^{-4}$	$0.00007982 \times 10^0$
$a + fl(b + c)$		$0.64727982 \times 10^0$
$fl(a + fl(b + c))$		$0.6473 \times 10^0$

Il risultato di  $(b + c) + a$  differisce da  $(a + b) + c$ . Resta dunque provato che il sistema aritmetico considerato non è **associativo**.

**Esercizio 13** Si ha:

$$9 = 1001_2 \quad 0.6 = 0.\overline{1001}_2$$

$$4 = 100_2 \quad 0.2 = 0.\overline{0011}_2$$

per cui:

$$x = 0.1001100 \times 2^{100}$$

$$y = 0.125 = 0.1 \times 2^{-011} \quad x + y = 0.1001100 \times 2^{100} + 0.00000001 \times 2^{100} = x$$

$$x = 0.1001100 \times 2^{100}$$

$$z = 0.0325 = 0.1 \times 2^{-101} \quad z + t = 0.1001100 \times 2^{011} + 0.000000001 \times 2^{011} = z$$

**Esercizio 14** Si ha  $E_a = |123.4599 - 123.4578| = 0.0021 = 0.21 \times 10^{-2}$  da cui si ha conferma che l'approssimazione è corretta a **2 cifre decimali**, mentre  $E_r = E_a/123.4578 = 0.000017.. = 0.17... \times 10^{-4}$ , da cui si ricava che l'approssimazione è corretta a **5 cifre significative**.

**Esercizio 15** Si ha  $E_a = |0.000418 - 0.000412| = 0.000006 = 0.6 \times 10^{-5}$  da cui si ha conferma che l'approssimazione è corretta a **5 cifre decimali**, mentre  $E_r = E_a/0.000418 = 0.01435 = 0.1435... \times 10^{-1}$ , da cui si ricava che l'approssimazione è corretta a **2 cifre significative**.

**Esercizio 16** Si ha  $\varepsilon = 0.5 \times 10^{-4}$ .

Inoltre è  $x_1 = 0.1234578 \times 10^3$  e  $\bar{x}_1 = 0.12346 \times 10^3$ .

L'errore relativo allora è

$$\frac{|0.1234578 \times 10^3 - 0.12346 \times 10^3|}{0.1234578 \times 10^3} = 0.000017... = 0.17... \times 10^{-4} < \mu$$

Si ha poi  $x_2 = 0.215984 \times 10^{-1}$  e quindi  $\bar{x}_2 = 0.21598 \times 10^{-1}$ . L'errore relativo allora è

$$\frac{|0.215984 \times 10^{-1} - 0.21598 \times 10^{-1}|}{0.215984 \times 10^{-1}} = 0.000018.. = 0.18.. \times 10^{-4} < \mu$$

**Esercizio 17** La prima somma è  $0.76597 \times 10^2$ , mentre la seconda è  $0.49854 \times 10^3$ .

**Esercizio 18** Si ottiene

- |   |                               |
|---|-------------------------------|
| (a) $0.1000 \times 10^1$                  | (b) $0.1001 \times 10^4$      |
| (c) $0.1000 \times 10^8$                  | (d) $0.1000 \times 10^{11}$   |
| (e) $0.1 \times 10^{-4} = \epsilon_{mac}$ | (f) $0.1 \times 10^{-24} = 0$ |

**Esercizio 19** Si ottiene:

- (I)  $a = \sqrt{y^2 + z^2} = \sqrt{1 + (10^{-25})^2} = \sqrt{1 + 10^{-50}} = \sqrt{1 + 0} = 1$
- (II)  $a = \sqrt{w^2 + z^2} = \sqrt{10^{-40} + 10^{-50}} = \sqrt{0 + 0} = 0$
- (III)  $u = \frac{10^{-15}}{0} = INF$

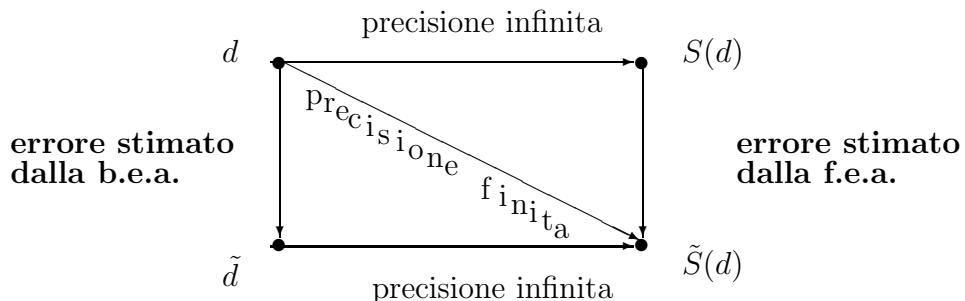
### 1.10.2 Alcuni quesiti

**Quesito 1** Quando la soluzione calcolata di un dato problema può essere considerata accurata nel senso della Backward Error Analysis?

- RISPOSTA

Nella Backward Error Analysis si assume che la soluzione calcolata di un dato problema sia accurata, se essa è la soluzione esatta (ottenuta in un sistema aritmetico a precisione infinita) del medesimo problema perturbato, ovvero dello stesso problema in cui sia stata introdotta una variazione nei dati comparabile con l'errore di *round-off* di rappresentazione dei dati.

Indicata con  $f$  la funzione che descrive il problema da risolvere,  $x$  i dati esatti,  $\hat{x}$  i dati affetti da errore,  $\hat{f}$  la funzione che descrive il problema perturbato risulta:  $\hat{f}(x) = f(\hat{x}) = f(x+\delta)$ , con  $|\delta| \leq u$ , dove  $u$  è la precisione del sistema. In figura è descritto il diagramma operativo della Backward Error Analysis, confrontato con l'altra tecnica di analisi della propagazione dell'errore, cioè la Forward Analysis.



Errori stimati con la forward error analysis e con la backward error analysis.

**Quesito 2** In un sistema aritmetico a precisione finita cosa è più pericoloso durante l'esecuzione di un'operazione: un underflow o un overflow? E perché?

- **RISPOSTA**

Un overflow è un evento più difficile da gestire; può comportare l'interruzione dell'esecuzione dei calcoli; in generale sono utilizzati opportuni indicatori di situazioni eccezionali (l'IEEE utilizza il numero macchina INF) che consentono la prosecuzione dei calcoli fino al termine dell'algoritmo, inficiando, comunque, tutte le operazioni successive e rendendo, pertanto, complessa l'individuazione di errori. Quando si realizza un underflow, la maggior parte dei sistemi aritmetici assegna al risultato del calcolo il **valore 0**. Questo valore può essere plausibile e quindi non inficiare l'esecuzione delle successive operazioni. Tuttavia anche in tal caso va verificato se il numero che è andato in underflow non sia poi utilizzato come divisore di una successiva operazione e creare, dunque, un' ulteriore situazione di overflow.

**Quesito 3** In un sistema aritmetico f.p. a precisione finita, il prodotto di due numeri macchina non è, di solito, rappresentabile esattamente. Perché?

- **RISPOSTA**

Consideriamo il caso in cui non si verifichino situazioni eccezionali, cioè il risultato,  $r$ , dell'operazione sia ancora rappresentabile in  $\mathfrak{F}$ . Supponiamo inoltre che il sistema aritmetico a precisione finita  $\mathfrak{F}$  garantisca la massima accuratezza dinamica, ovvero sia tale che  $\tilde{r}$ , il risultato dell'operazione f.p., differisca dal risultato calcolato in aritmetica a precisione infinita, di una quantità che è il solo errore di rappresentazione di  $r$  in  $\mathfrak{F}$ , cioè:

$$\tilde{r} = x \otimes y = fl(r) = fl(x \times y)$$

con

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

In questo caso, il risultato di un prodotto tra due numeri esattamente rappresentabili, generalmente non è *esattamente rappresentabile*. Se, ad esempio, si moltiplicano due numeri di  $t$  cifre il risultato può essere (al più) composto da  $2t$  cifre ed è dunque necessario un *troncamento* (o un *arrotondamento*) se  $t$  è la precisione del sistema aritmetico.

Fissato un sistema aritmetico a precisione finita di base  $\beta = 10$ , precisione  $t = 2$  ed  $emin = -2$ ,  $emax = 2$ , che utilizzi il troncamento, sia:

$$\begin{aligned} x &= 0.25 \times 10^2 \\ y &= 0.25 \times 10^2 \end{aligned}$$

allora

$$\tilde{r} = fl((0.25 \times 10^2) \times (0.25 \times 10^2)) = 0.62 \times 10^3 \neq r = 0.625 \times 10^3$$

**Quesito 4** Si dia un esempio di operazione f.p. che possa produrre i valori INF e NaN.

- RISPOSTA

I valori indicati sono introdotti in un sistema aritmetico standard IEEE quando si eseguono delle operazioni del tipo:

$$NaN = \begin{cases} \frac{0}{0} \\ 0 * \infty \\ \infty * \infty \end{cases}$$

$$Inf = \frac{k}{0}, \quad k \in \mathbb{R} - \{0\}$$

Se si considera un sistema aritmetico f.p. con soglia di  $UFL=10^{-12}$  ed una costante reale  $\alpha = 0.2 \times 10^{-14}$ , l'operazione seguente produce NaN:

$$\frac{\alpha}{\alpha} = \frac{0}{0} = NaN$$

Mentre il valore INF si ottiene quando si effettua una divisione per zero nel sistema aritmetico f.p. utilizzato:

$$INF = \frac{k}{0}, \quad k \in \mathbb{R} - \{0\}.$$

**Quesito 5** Si spieghi la differenza tra l'epsilon macchina  $\epsilon_{mac}$  ed il livello di underflow  $UFL$  in un sistema aritmetico f.p. Di queste due quantità:

- Quale delle due dipende solo dal numero di cifre della mantissa?
- Quale delle due dipende solo dall'esponente?
- Quale delle due non dipende dal tipo di arrotondamento utilizzato?
- Quale delle due non cambia quando si utilizzano dei numeri denormalizzati?

- RISPOSTA

Il livello di underflow (UFL) è il più piccolo numero macchina positivo, ovvero:

$$UFL = 0.1 \times \beta^{emmin} = \beta^{emmin-1}$$

L'epsilon macchina ( $\epsilon_{mac}$ ) è il più piccolo numero macchina positivo che dà contributo alla somma f.p. con 1; ovvero:

$$1 \oplus \epsilon_{mac} = fl(1 + \epsilon_{mac}) > 1$$

Risulta:

$$\epsilon_{mac} = \beta^{1-t} \text{ con troncamento}$$

- (a) L'  $\epsilon_{mac}$ .
- (b) L'UFL.
- (c) L'UFL.
- (d) La precisione relativa della macchina non varia se si utilizzano numeri denormalizzati. Al contrario l'introduzione dei numeri denormalizzati permette al sistema di avere un *Underflow graduale*, che è implementato riservando valori speciali al campo dell'esponente.

Se si considera un sistema aritmetico a precisione finita

$$\mathfrak{S} = (\beta, t, emin, emax)$$

che adoperi numeri denormalizzati allora è possibile rappresentare:

$$\begin{aligned} 0.\underbrace{01\dots0}_t \times \beta^{emin} &= 0.1 \times \beta^{emin-1} \\ 0.\underbrace{001\dots0}_t \times \beta^{emin} &= 0.1 \times \beta^{emin-2} \\ &\dots && \dots \\ 0.\underbrace{00\dots1}_t \times \beta^{emin} &= 0.1 \times \beta^{emin-t} \end{aligned}$$

dunque la soglia di UFL diventa  $0.1 \times \beta^{emin-t}$ .

**Quesito 6** Si spieghi il fenomeno della cancellazione.

• RISPOSTA

La cancellazione è un fenomeno tipico dei sistemi a *precisione finita*. Esso si verifica a causa del vincolo sul numero di cifre (finito) della mantissa riservato per rappresentare un numero f.p. e può determinare la perdita di cifre significative quando si esegue una sottrazione tra numeri "vicini".

Ad esempio, considerato un sistema aritmetico a precisione finita con  $\beta = 10$ ,  $t = 4$ ,  $emin = -12$ ,  $emax = 12$ , eseguendo la sottrazione f.p. dei due numeri seguenti:

$$0.1234 \times 10^3 - 0.1233 \times 10^3 = 0.0001 \times 10^3 = 0.1000 \times 10^0$$

si ottiene un numero con una sola cifra significativa, in altre parole, si perdono 3 cifre significative nella rappresentazione del risultato.

In generale se

$$f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$$

è la funzione che alla coppia di numeri reali  $(x, y)$  associa il numero reale  $z = x - y$ , è noto (cfr. cap.1) che l'indice di condizionamento del problema, è dato da

$$C(f, x, y) = \frac{|x| + |y|}{|x - y|}$$

Tale quantità cresce al diminuire della distanza tra  $x$  e  $y$ . Cioé la cancellazione è l'effetto del **mal-condizionamento** della sottrazione tra due numeri f.p. "vicini".

**Quesito 7** Per calcolare il punto medio  $m$  di un intervallo  $[x, y]$ , quale delle seguenti due formule è preferibile utilizzare in un sistema aritmetico f.p. a precisione finita? Perché?

$$(a) m = f_1(x, y) = (x + y)/2, \quad (b) m = f_2(x, y) = x + (y - x)/2$$

- RISPOSTA

In base alle considerazioni fatte nel quesito precedente, distinguiamo i casi seguenti:

1.  $x$  e  $y$  di segno discorde;
2.  $x$  e  $y$  di segno concorde.

e, per entrambi,

- \*  $x$  e  $y$  "vicini";
- \*  $x$  e  $y$  "lontani".

Nel caso in cui  $x$  e  $y$  sono *di segno discorde*, conviene utilizzare la formula (b); con essa, infatti, se  $x$  e  $y$  sono "vicini", si evitano gli effetti del mal-condizionamento dell'operazione di sottrazione e, dunque, il verificarsi del fenomeno della cancellazione; se, invece,  $x$  e  $y$  sono "lontani", si evita la perdita di accuratezza dovuta alla fase di shift delle cifre delle mantisse dei numeri da sottrarre ed alle successive fasi di normalizzazione ed arrotondamento (o troncamento) del risultato.

Se, invece,  $x$  e  $y$  sono *di segno concorde* si può utilizzare la formula (a). In effetti, come illustrato dagli esempi descritti di seguito, non mancano casi in cui, in entrambe le circostanze, le due formule producono lo stesso risultato numerico.

Consideriamo un sistema aritmetico  $\mathfrak{S}$  a precisione finita caratterizzato dai seguenti parametri:

$$\mathfrak{S} : \beta = 10, t = 4, \text{emin} = -9, \text{emax} = +9,$$

e supponiamo che in  $\mathfrak{S}$  venga utilizzata, per la rappresentazione dei numeri, la funzione di *troncamento*. Calcoliamo in  $\mathfrak{S}$  il punto medio  $m$  per le seguenti coppie di numeri (in parentesi è riportato il valore di  $m$  che si ottiene in aritmetica a precisione infinita) e distinguiamo due casi:

\*  $x$  e  $y$  "vicini" tra loro in modulo:

1.a)  $x = 3.7678, y = 3.7668, (m = 0.37673 \times 10^1);$

1.b)  $x = 3.7678, y = -3.7668, (m = 0.5 \times 10^{-3});$

\*  $x$  e  $y$  "lontani":

2.a)  $x = 3.7678, y = 2.6211, (m = 0.319445 \times 10^1);$

2.b)  $x = 2.9768, y = -6.7678, (m = -0.18955 \times 10^1);$

2.c)  $x = 3.7678, y = 0.2611 \times 10^{-2}, (m = 0.18852055 \times 10^1);$

2.d)  $x = 2.9775, y = -61.3267, (m = -0.291746 \times 10^2);$

2.e)  $x = -2.7502, y = -6.36852, (m = -0.455936 \times 10^1).$

Indichiamo con  $\tilde{x}$  e  $\tilde{y}$ , la rappresentazione di  $x$  e  $y$  in  $\mathfrak{S}$  ed analizziamo le soluzioni  $\tilde{m}_a$ ,  $\tilde{m}_b$  ottenute in tale sistema rispettivamente con le formule (a) e (b):

	$\tilde{x}$	$\tilde{y}$	$\tilde{m}_a$	$\tilde{m}_b$
1.a)	$0.3767 \times 10^1$	$0.3766 \times 10^1$	$0.3766 \times 10^1$	$0.3767 \times 10^1$
1.b)	$0.3767 \times 10^1$	$-0.3766 \times 10^1$	$0.0 \times 10^0$	$0.1 \times 10^{-3}$
2.a)	$0.3767 \times 10^1$	$0.2621 \times 10^1$	$0.3194 \times 10^1$	$0.3194 \times 10^1$
2.b)	$0.2976 \times 10^1$	$-0.6767 \times 10^1$	$-0.1895 \times 10^1$	$-0.1895 \times 10^1$
2.c)	$0.3767 \times 10^1$	$0.2611 \times 10^{-2}$	$0.1884 \times 10^1$	$0.1885 \times 10^1$
2.d)	$0.2977 \times 10^1$	$-0.6132 \times 10^2$	$-0.2922 \times 10^2$	$-0.2916 \times 10^2$
2.e)	$-0.2750 \times 10^1$	$-0.6368 \times 10^1$	$-0.4559 \times 10^1$	$-0.4559 \times 10^1$

Di seguito sono riportati gli errori relativi  $E_a$  ed  $E_b$ , che si commettono nell'approssimare  $m$ , rispettivamente con  $\tilde{m}_a$  e  $\tilde{m}_b$ :

	$E_a$	$E_b$
1.a)	$0.345 \times 10^{-3}$	$0.769 \times 10^{-4}$
1.b)	$0.1 \times 10^1$	$0.8 \times 10^0$
2.a)	$0.1408 \times 10^{-3}$	$0.1408 \times 10^{-3}$
2.b)	$0.2637 \times 10^{-3}$	$0.2637 \times 10^{-3}$
2.c)	$0.6394 \times 10^{-3}$	$0.109 \times 10^{-3}$
2.d)	$0.1576 \times 10^{-3}$	$0.146 \times 10^{-3}$
2.e)	$0.789 \times 10^{-4}$	$0.789 \times 10^{-4}$

Osservando la tabella, entrambe le formule producono risultati accettabili, in particolare per i casi che abbiamo distinto:

1. se  $x$  e  $y$  hanno segno discordi, siano essi “vicini”, come nell’esempio 1.b), o “lontani”, come nel 2.d), la formula (b) produce risultati più accurati; non mancano, comunque, casi in cui, come per l’esempio 2.b), le due formule producono lo stesso risultato.
2. si osserva, però che anche quando  $x$  e  $y$  hanno segno concorde, la formula (b) produce risultati più accurati, sia se  $x$  e  $y$  sono “vicini”, come nell’esempio 1.a) che “lontani” come nell’esempio 2.c); in particolare, poi, le due formule possono produrre lo stesso risultato, come si verifica nei casi 2.a) e 2.e).

**Quesito 8** Si elenchino due situazioni in cui il calcolo della formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

può dare luogo ad errori, in aritmetica floating-point.

- RISPOSTA

- (I) Se  $b$  è sufficientemente grande il calcolo di  $b^2$  potrebbe comportare un overflow.
- (II) Ci potrebbero essere errori di cancellazione distruttiva quando si effettua il calcolo del discriminante.

Consideriamo, ad esempio, un sistema aritmetico a precisione finita con  $\beta = 10$ ,  $t = 4$ ,  $emin = -10$ ,  $emax = 10$ ; allora se:

$$\begin{aligned} (1) \quad & a = 0.1 \times 10^1 \quad b = 0.2 \times 10^6 \quad c = 0.3 \times 10^{-1} \\ (2) \quad & a = 0.1 \times 10^1 \quad b = 0.1 \times 10^4 \quad c = 0.3 \times 10^1 \end{aligned}$$

si ha:

$$\begin{aligned} (1) \quad & b^2 = 0.4 \times 10^{12} \quad (\text{Overflow}) \\ (2) \quad & \Delta = 0.1 \times 10^7 - 0.12 \times 10^2 = 0.1 \times 10^7 \quad (\text{Cancellazione}) \end{aligned}$$

**Quesito 9** Si supponga di risolvere un’equazione di secondo grado completa del tipo:

$$ax^2 + bx + c = 0$$

utilizzando un sistema aritmetico f.p. (normalizzato) con  $\beta = 10$   $t = 3$  e  $a = 1.22$ ,  $b = 3.34$ ,  $c = 2.28$ .

- (a) Qual è il valore calcolato del discriminante  $b^2 - 4ac$ ?
- (b) Qual è il valore corretto del discriminante in aritmetica a precisione infinita?
- (c) Qual è l’errore relativo nel valore calcolato del discriminante?

• RISPOSTA

Indicando con  $\tilde{\Delta}$  il discriminante calcolato e con  $\Delta$  quello in aritmetica a precisione infinita si ottiene:

- (a)  $\tilde{\Delta} = 0.111556 \times 10^2 - 0.111164 \times 10^2 = 0$
- (b)  $\Delta = 0.292 \times 10^{-1}$
- (c)  $E_{rel} = \frac{|\Delta - \tilde{\Delta}|}{|\Delta|} = 1 = 100\%$

**Quesito 10** Si consideri l'espressione:

$$\frac{1}{1-x} - \frac{1}{1+x},$$

- (a) Per quale intervallo di valori di  $x$  è difficile un calcolo accurato in aritmetica f.p.?
- (b) Si dia un'espressione equivalente della formula in modo che, per i valori di  $x$  del punto (a), il calcolo sia più accurato in aritmetica f.p.

• RISPOSTA

(a) Si possono avere problemi nel calcolo dell'espressione quando  $fl(x)$  non dà contributo nella somma/differenza con 1. Infatti se  $|fl(x)| < \epsilon_{macc}$ :

$$\frac{1}{1-fl(x)} - \frac{1}{1+fl(x)} \equiv 0$$

Ad esempio, se consideriamo un sistema aritmetico a precisione finita di base  $\beta = 10$ ,  $t = 8$   $emin = -20$ ,  $emax = 20$ , se  $x = 10^{-7}$  allora:

$$\frac{1}{1-10^{-7}} - \frac{1}{1+10^{-7}} \equiv 1-1 \equiv 0$$

(b) Se, invece, si considera l'espressione identica:

$$\left( \frac{1}{1-x} - \frac{1}{1+x} \right) = \frac{2x}{1-x^2}$$

in questo caso, nello stesso sistema aritmetico del punto (a) scegliendo ancora  $x = 10^{-7}$ :

$$\frac{2 \cdot 10^{-7}}{1-10^{-14}} = \frac{2 \cdot 10^{-7}}{1-0} = 2 \cdot 10^{-7} \neq 0.$$

**Quesito 11** Se  $x \approx y$  ci si aspetta un errore di cancellazione nel calcolo di:

$$z = \log(x) - \log(y)$$

Tale problema si può risolvere osservando che:

$$z = \log(x) - \log(y) = \log\left(\frac{x}{y}\right).$$

Con questo accorgimento si può affermare che il calcolo di  $\log(x/y)$  è più accurato?

- **RISPOSTA**

Si, perché se

$$x = y \cdot (1 + o(\epsilon))$$

allora

$$\log(x) = \log(y) + \log\left(\underbrace{1 + o(\epsilon)}_{\approx 1}\right) = \log(y) + \delta,$$

e il calcolo di  $z = \log(x) - \log(y)$  può creare problemi di cancellazione distruttiva. Ciò non accade se  $z$  viene valutato nel modo seguente:

1. Calcolo di  $\frac{x}{y}$
2. Calcolo di  $z = \log\left(\frac{x}{y}\right)$ .

Consideriamo il seguente sistema aritmetico a precisione finita,  $\mathfrak{F} : \beta = 10$ ,  $t = 7$ ,  $emin = -10$ ,  $emax = 10$ , siano:

$$x = 0.2509999 \times 10^1, \quad y = 0.2510001 \times 10^1$$

Calcoliamo i logaritmi di  $x$  e  $y$ :

$$\log(x) = 0.3996735 \times 10^0$$

$$\log(y) = 0.3996739 \times 10^0$$

Nel calcolo della differenza fra questi valori, si perdono tutte le cifre significative, a causa della cancellazione distruttiva; infatti si ha:

$$\begin{aligned} \log(x) - \log(y) &= 0.3996735 \times 10^0 - 0.3996739 \times 10^0 = \\ &= -0.0000004 \times 10^0 = -0.4 \times 10^{-6} \end{aligned}$$

Calcoliamo ora il rapporto tra  $x$  e  $y$ :

$$\frac{x}{y} = 0.9999992 \times 10^0$$

da cui:

$$\log\left(\frac{x}{y}\right) = -0.3474357 \times 10^{-8}$$

il risultato ottenuto ha 7 cifre significative ed è più accurato del precedente, in quanto il risultato, in aritmetica a precisione infinita, è:

$$\log\left(\frac{x}{y}\right) = \log(x) - \log(y) = -0.3474357245069005... \times 10^{-8}$$

**Quesito 12** Se  $y$  è un vettore di  $\mathbb{R}^n$ , la norma Euclidea è definita da:

$$\|y\|_2 = \left( \sum_{i=1}^n y_i^2 \right)^{1/2}$$

Come è possibile evitare un overflow nel calcolo di tale norma?

- **RISPOSTA**

Calcolando la norma Euclidea nel modo seguente: posto  $\tilde{y} = \max_{i=1,n} y_i$ ,

$$\|y\|_2 = \tilde{y} \cdot \sqrt{\sum_{i=1}^n \left(\frac{y_i}{\tilde{y}}\right)^2}$$

Dividendo per il più grande degli  $y_i$  riduciamo l'ordine di grandezza delle componenti di  $y$  e, quindi, evitiamo l'overflow.

Consideriamo il sistema aritmetico a precisione finita  $\mathfrak{S}$ :  $\beta = 10, t = 5$ ,  $emin = -4$ ,  $emax = +4$ , ed il vettore di  $\mathbb{R}^3$ :

$$y = (0.1 \times 10^2, 0.2 \times 10^3, 0.6 \times 10^1)$$

Consideriamo il quadrato delle sue componenti:

$$y_1^2 = 0.1 \times 10^3, y_2^2 = 0.4 \times 10^5, y_3^2 = 0.36 \times 10^2$$

la componente  $y_2$  al quadrato genera un overflow.

Scaliamo adesso il vettore per la componente di valor massimo:

$$\tilde{y}_1 = \frac{0.1 \times 10^2}{0.2 \times 10^3} = 0.5 \times 10^{-1}, \tilde{y}_2 = 0.1 \times 10^1, \tilde{y}_3 = \frac{0.6 \times 10^1}{0.2 \times 10^3} = 0.3 \times 10^{-1}$$

Calcoliamo il quadrato delle componenti,  $\tilde{y}_i$ :

$$\tilde{y}_1^2 = 0.25 \times 10^{-3}, \tilde{y}_2^2 = 0.1 \times 10^1, \tilde{y}_3^2 = 0.9 \times 10^{-3}$$

in questo modo viene evitato l'overflow. Calcoliamo la norma, in aritmetica a precisione infinita:

$$\|y\|_2 = 0.20033971 \times 10^3$$

mentre con l'accorgimento proposto, si ottiene:

$$\|\tilde{y}\|_2 = 0.20034 \times 10^3$$

come si può notare, l'errore che si commette è sulla quinta cifra significativa, ed è trascurabile perché compatibile con la precisione del sistema.

**Quesito 13** Assumiamo un sistema aritmetico f.p. normalizzato con  $\beta = 10$ ,  $t = 3$ , e  $emin = -98$ .

- (a) Qual è la soglia di UFL per questo sistema?
- (b) Se  $x = 6.87 \times 10^{-97}$  e  $y = 6.81 \times 10^{-97}$ , qual è il risultato di  $x - y$ ?
- (c) Quale sarebbe il risultato di  $x - y$  se il sistema avesse un underflow graduale?

• RISPOSTA

- (a)  $UFL = 0.1 \times 10^{-97}$
- (b)  $x - y = 0.6 \times 10^{-99} = INF = 0$
- (c) In un sistema che permette l'underflow graduale risulta:

$$x - y = 0.006 \times 10^{-97} = 0.6 \times 10^{-99}$$

**Quesito 14** Sia  $\{x_k\}$  una successione monotona decrescente di numeri positivi ( $x_k > x_{k+1}$  per ogni  $k$ ). In quale ordine gli elementi della successione devono essere sommati per minimizzare l'errore di arrotondamento?

• RISPOSTA

Sommando i numeri in ordine crescente, ovvero dal più piccolo al più grande, si sommano numeri approssimativamente dello stesso ordine di grandezza. Si riducono quindi eventuali errori di cancellazione dovuti all'arrotondamento. Infatti sia  $\mathfrak{S}$  un sistema aritmetico a precisione finita di parametri  $\beta = 10$ ,  $t = 7$ ,  $emin = -10$ ,  $emax = +10$ . Sommiamo i primi 1000 termini nell'ordine **naturale**:

$$T_{1000}^{nat} = \sum_{k=0}^{1000} \frac{5}{(2k+1)^3} = 5 + \frac{5}{3^3} + \cdots + \frac{5}{2001^3}$$

$$T_{1000}^{nat} = 0.5258986 \times 10^1$$

Se invece si sommano i primi 1000 termini in maniera **decrescente** si ha:

$$T_{1000}^{dec} = 0.525899 \times 10^1$$

Si assume che il valore “esatto” sia:

$$T_{1000}^* = 0.525899 \dots \times 10^1$$

dove tutte le cifre indicate sono esatte.

Quindi l'errore assoluto che si commette utilizzando la prima strategia ha un ordine di grandezza di  $10^{-5}$ , ovvero:

$$E = |T_{1000}^{nat} - T_{1000}^*| = 0.12 \times 10^{-5}$$

Mentre l'errore, nel sistema aritmetico  $\mathfrak{F}$ , utilizzando la seconda strategia, risulta dell'ordine della precisione del sistema aritmetico, ovvero

$$E = |T_{1000}^{dec} - T_{1000}^*| < 10^{-7}.$$

**Quesito 15** *Si spieghi perché la serie divergente:*

$$\sum_{n=1}^{\infty} \frac{1}{n}$$

*può avere una somma finita in aritmetica f.p. A partire da quale indice gli addendi non danno più contributo alla sommatoria con i termini precedenti?*

- **RISPOSTA**

Volendo eseguire la somma in un sistema aritmetico a precisione finita, esiste un indice  $n_0$  per cui il termine  $1/n_0$  può non dare contributo alla somma con:

$$\sum_{k=1}^{n_0-1} \frac{1}{k}$$

per due motivi:

- perché  $\frac{1}{n_0} < \text{underflow}$ , e quindi ad esso viene assegnato il valore zero;
- perché:

$$\left| \frac{1}{n_0} \right| < \epsilon_{mac} \left| \sum_{k=1}^{n_0-1} \frac{1}{k} \right|$$

e dunque il termine  $n_0$ -esimo della serie non dà contributo alla somma con i termini precedenti.

**Quesito 16** *Se  $f$  è una funzione reale derivabile, di variabile reale, si ha:*

$$f'(x) = \frac{f(x+h) - f(x)}{h} + E(x), \quad E(x) = \mathcal{O}(h^2)$$

dove l'errore  $E(x)$  tende a zero quando  $h \rightarrow 0$ . Una sua approssimazione è, quindi,

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

Sotto quali condizioni, nel sistema aritmetico a precisione finita  $\mathfrak{S}$ , tale espressione ha significato?

- RISPOSTA

In un sistema aritmetico  $\mathfrak{S}$  a precisione finita i due numeri macchina  $x$  e  $x+h$  sono distinti se

$$h \geq \epsilon_{mac} \cdot |x|$$

cioé, risultano distinti se  $h$  è un numero macchina che dà contributo alla somma con  $x$ . Consideriamo, ad esempio, un sistema aritmetico  $\mathfrak{S}$  a precisione finita caratterizzato dai parametri

$$\mathfrak{S} : \beta = 10, t = 4, emin = -10, emax = +10,$$

e sia

$$f(x) = x^2.$$

Calcoliamo il più piccolo numero  $h$  che dà contributo alla somma con

$$x = 0.3 \times 10^1.$$

In  $\mathfrak{S}$  risulta

$$\epsilon_{mac} = 0.5 \times 10^{-3},$$

eseguendo le operazioni in doppia precisione e cioé  $t_{reg} = 8$ ; si ottiene, dunque:

$$h \geq \epsilon_{mac} \cdot |x| = 0.5 \times 10^{-4} \times 0.3 \times 10^1 = 0.15 \times 10^{-3}$$

per cui è sufficiente che sia

$$h = 0.15 \times 10^{-3}$$

e, di conseguenza,

$$x + h = 0.300015 \times 10^1 \neq x,$$

perché abbia significato la valutazione della funzione  $f$  in due valori distinti. D'altra parte, risulta

$$|f(x+h)| \geq \epsilon_{mac} \cdot |f(x)| \Rightarrow f(x+h) \neq f(x)$$

pertanto deve risultare che:

$$f(x+h) \geq \epsilon_{mac} \cdot f(x)$$

perché abbia significato il numeratore del rapporto incrementale.

**Quesito 17** Illustrare un metodo per evitare il fenomeno della cancellazione distruttiva nel calcolo di una somma finita del tipo:

$$\sum_{k=0}^N \frac{(-1)^k}{2k+1} \quad (1)$$

- RISPOSTA

Per evitare il fenomeno della **cancellazione distruttiva** occorre raggruppare i termini dello stesso segno e sommarli in maniera decrescente.

Ciò significa:

1. **raggruppare** i termini dello stesso segno;
2. **eseguire** la somma parziale dei numeri in maniera decrescente;
3. **eseguire** la differenza tra le somme parziali.

Supponiamo di voler valutare la somma (1) per  $n = 20$

$$\sum_{k=0}^{20} \frac{(-1)^k}{2k+1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots - \frac{1}{41}$$

Nei passi (a) e (b) si calcolano le somme parziali seguenti:

$$\begin{aligned} \sum_{k=pari}^{k=20...0} \frac{(-1)^k}{2k+1} &= \frac{1}{39} + \frac{1}{35} + \cdots + \frac{1}{9} + \frac{1}{5} + 1 \\ - \sum_{k=dispari}^{k=19...1} \frac{(-1)^k}{2k+1} &= \frac{1}{41} + \frac{1}{37} + \cdots + \frac{1}{11} + \frac{1}{7} + \frac{1}{3} \end{aligned}$$

nel passo (c) si calcola la differenza delle somme parziali:

$$\begin{aligned} \sum_{k=0}^{20} \frac{(-1)^k}{2k+1} &= \sum_{k=pari}^{k=20...0} \frac{(-1)^k}{2k+1} + \sum_{k=dispari}^{k=19...1} \frac{(-1)^k}{2k+1} \\ &= \sum_{h=0}^{10} \frac{(-1)^{2h}}{2(2h)+1} + \sum_{h=1}^{10} \frac{(-1)^{(2h-1)}}{2(2h-1)+1} \end{aligned}$$

**Quesito 18** Il condizionamento è un effetto della precisione aritmetica con cui si risolve il problema.

- RISPOSTA [FALSO] . Il condizionamento è una caratteristica intrinseca del problema e dipende solo dalla natura del problema. La precisione aritmetica fa sì che si evidenzino gli effetti del mal condizionamento.

**Quesito 19** Il condizionamento di un problema dipende dall'algoritmo utilizzato per risolverlo.

- RISPOSTA [FALSO]. Il condizionamento dipende solo dal problema.

**Quesito 20** Utilizzando la massima precisione di un sistema aritmetico floating-point a precisione finita, si può migliorare il condizionamento di un problema **malcondizionato**.

- RISPOSTA [FALSO]. Il fattore di amplificazione degli errori sui dati dipende dal problema e non dal sistema aritmetico utilizzato. Pertanto, anche utilizzando una maggiore precisione aritmetica, l'amplificazione dell'errore sulla soluzione non può essere evitata.

**Quesito 21** Un “buon” algoritmo produce una soluzione accurata indipendentemente dal problema che si sta risolvendo.

- RISPOSTA [FALSO]. L'accuratezza della soluzione di un problema dipende oltre che dall'algoritmo anche dal condizionamento del problema. Pertanto, un “buon” algoritmo (cioè un algoritmo stabile), se viene utilizzato per risolvere un problema mal-condizionato, può fornire una soluzione inaccurata.

**Quesito 22** La scelta di un algoritmo per risolvere un problema ha effetto sulla propagazione degli errori nei dati.

- RISPOSTA [VERO]. In generale due algoritmi per la risoluzione di uno stesso problema possono produrre risultati diversi in quanto gli errori di **round-off**, dovuti al modo differente con cui vengono effettuate le operazioni, possono propagarsi in maniera diversa.

**Quesito 23** Se due numeri reali sono esattamente rappresentabili come numeri f.p. in un sistema aritmetico a precisione finita, allora il risultato di un'operazione aritmetica su di loro può non essere esattamente rappresentabile.

- RISPOSTA [VERO]. Adoperando un sistema aritmetico a precisione finita quanto affermato è sicuramente vero.

Ad esempio, fissata la base  $\beta = 10$ , la precisione  $t = 7$  e gli esponenti minimo e massimo rispettivamente  $emin = -2$ ,  $emax = +2$ , considerati i due numeri macchina

$$x_1 = 0.3354110 \times 10 \quad \text{e} \quad x_2 = 0.7666607 \times 10$$

il prodotto risulta

$$x_1 \times x_2 = 25.71464320477000 = 0.2571464320477 \times 10^2$$

e, quest'ultimo è, effettivamente un numero rappresentabile, essendo il suo esponente  $emin \leq 2 \leq emax$ , ma non esattamente, avendo un numero di

cifre significative maggiore di  $t$ ; applicando uno schema di arrotondamento si trova il risultato

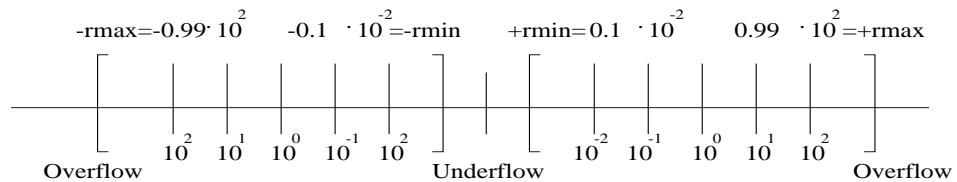
$$fl(x_1 \times x_2) = 0.2571464 \times 10^2.$$

**Quesito 24** I numeri f.p. sono uniformemente distribuiti sulla retta reale  $\mathfrak{R}$ .

- **RISPOSTA [FALSO].** Essi non sono uniformemente distribuiti, ma lo sono solo tra successive potenze della base di numerazione,  $\beta$ , del sistema. Se, ad esempio, fissiamo un sistema aritmetico a precisione finita di base  $\beta = 10$ , precisione  $t = 2$  ed  $emin = -2$ ,  $emax = 2$ , allora tutti e soli i numeri rappresentabili in questo sistema sono quelli appartenenti all'insieme  $\mathfrak{S}$ :

$$\mathfrak{S} := [-0.99 \cdot 10^2, -0.1 \cdot 10^{-2}] \cup \{0\} \cup [0.1 \cdot 10^{-2}, 0.99 \cdot 10^2]$$

Tali numeri, come si può osservare, sono uniformemente distribuiti sull'asse reale  $\mathfrak{R}$ , tra successive potenze di 10.



Rappresentazione grafica dell'insieme  $\mathfrak{S}$

### 1.10.3 Alcuni problemi da risolvere con il calcolatore

**Problema 1** Si scriva un programma in Fortran per il calcolo di un valore approssimato del numero di Nepero  $e$ , utilizzando la definizione:

$$e = \lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n.$$

- (a) Calcolare la quantità  $\left(1 + \frac{1}{n}\right)^n$  al crescere di  $n$  per  $n = 10^k$  con  $k = 1, 2, \dots, 20$
- (b) porre particolare attenzione (effettuando un opportuno controllo) al contributo dato alla somma di  $\frac{1}{n}$  con 1;
- (c) stampare, in corrispondenza di ogni approssimazione,
  - l'indice del passo corrente,
  - il valore stimato,
  - l'errore relativo prodotto nel risultato.

confrontando, ad ogni passo, il valore dedotto con il valore  $e^1$  ottenuto utilizzando la funzione *built-in* del Fortran `exp()`;

- (d) Dopo quanti passi conviene terminare l'esecuzione del programma? Perché?
- (e) In corrispondenza di quale passo si raggiunge il minimo errore relativo?
- (f) Cosa accade nei passi successivi e cosa si può dedurre sull'accuratezza del risultato?

**Problema 2** Si consideri la funzione:

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

Utilizzando la regola de L'Hôpital si prova che:

$$\lim_{x \rightarrow 0} f(x) = 1.$$

- (a) Si provi questo risultato implementando un programma in Fortran per il calcolo di  $f(x)$  per  $x = 10^{-k}$ ,  $k=1,\dots,16$ . I risultati trovati concordano con quelli teorici? Perché?
- (b) Valutare  $f(x)$  utilizzando la formula:  $f(x) = \frac{e^x - 1}{\log(e^x)}$ . Questa formula dà risultati più accurati? Perché?

**Problema 3** Si supponga di generare  $n+1$  punti equidistanziati di  $h = \frac{b-a}{n}$ , nell'intervallo  $[a,b]$ .

- (a) In un sistema aritmetico f.p.  $\mathfrak{F}$  quale dei due algoritmi seguenti è più accurato,

$$x_0 = a, \quad x_k = x_{k-1} + h, \quad k+1, \dots, n$$

oppure

$$x_k = a + hk, \quad k = 0, \dots, n ?$$

- (b) Si scriva un programma che implementi entrambi gli algoritmi e si trovi un esempio, con  $a = 0$  e  $b = 1$ , che ne illustri le differenze.

**Problema 4** Si scriva un programma per il calcolo della derivata di una funzione  $f(x)$  utilizzando l'approssimazione che segue:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}.$$

Si provi il programma usando la funzione  $\ln(x)$ , per  $x = 2$ .

- (a) Si ripeta l'esercizio utilizzando l'approssimazione:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

**Problema 5** Si consideri la serie armonica:

$$\sum_{n=1}^{\infty} \frac{1}{n}$$

- (a) Provare che la serie è divergente.
- (b) Spiegare perché la serie ha somma finita in aritmetica f.p.
- (c) Si dica quando i termini della sommatoria non danno più contributo, nel sistema aritmetico standard IEEE, in singola ed in doppia precisione.
- (d) Si scrivano due programmi per il calcolo della serie armonica uno in singola e l'altro in doppia precisione. Si controlli il processo di calcolo riportando l'indice e la somma parziale ad ogni iterazione. Quale criterio di arresto può essere utilizzato? Si confrontino i risultati con i valori trovati.

**Problema 6** Si scriva un programma per il calcolo della media  $\bar{x}$  e della varianza  $\sigma^2$  di una successione finita  $x_i$ .  $\bar{x}$  è un vettore di dimensione  $n$ , per il calcolo della varianza si utilizzino entrambe le formule:

$$\sigma^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

$$\sigma^2 = \frac{1}{n-1} \left( \sum_{i=1}^n x_i^2 - \frac{1}{n} \left( \sum_{i=1}^n x_i \right)^2 \right)$$

Si confrontino i risultati per le due scelte.

#### 1.10.4 Risposte

Di seguito verranno proposti i programmi in Fortran e le risposte ai relativi problemi illustrati nel paragrafo precedente.

### Problema 1: calcolo del numero di Nepero

```
program calcolo numero di Nepero
```

C Fase di dichiarazione delle variabili

```
integer k,n
real N1,nep,e
real err1,epsilon
```

```
write(*,*)      ,
write(*,*)      , _____ Calcolo del numero di Nepero mediante il limite
write(*,*)      , (1)  $n \rightarrow \inf$  di  $(1 + 1/n)^n$ 
write(*,*)      ,
write(*,*)      , _____
```

C Calcolo del numero di Nepero tramite funzione interna

```
n = 1
e=exp(1.0)
```

C Calcolo dell'epsilon macchina

```
epsilon = epsilon(1.0)
print*, 'Epsilon macchina = ',epsilon
k=0
print*, 'Numero di Nepero mediante funzione interna',e
```

C Fase di calcolo

```
5 continue
k=k+1
n=10**k
N1=1.0/n
if (N1 .ge. epsilon ) then
    N1=1.0+N1
    nep=N1**n
```

C Calcolo dell'errore relativo

```
err1 = abs(e-nep)/abs(e)
```

C Stampa del valore calcolato del numero di Nepero

```
print*, ''
print*, 'iterazione numero',k
print*, 'def(1)=',nep
print*, 'Errore relativo ',err1
print*, ''
else
    print*, '1/n e" minore dell"epsilon macchina'
    print*, 'def(1)=',nep
endif
```

C Terminazione del ciclo di calcolo

```
if( k.lt.21 )goto 5
```

## Prova di esecuzione

---

Calcolo del numero di Nepero mediante il limite  
(1)  $n \rightarrow \infty$  di  $(1 + 1/n)^n$

---

Epsilon macchina = 1.1920929E-07

Numero di Nepero mediante funzione interna 2.71828175

iterazione numero 1

def(1)= 2.59374309

Errore relativo 0.0458152145

iterazione numero 2

def(1)= 2.704813

Errore relativo 0.00495487358

iterazione numero 3

def(1)= 2.71704197

Errore relativo 0.000456088339

iterazione numero 4

def(1)= 2.7180233

Errore relativo 9.50768808E-05

iterazione numero 5

def(1)= 2.72152352

Errore relativo 0.00119258335

iterazione numero 6

def(1)= 2.59460497

Errore relativo 0.0454981439

1/n è minore dell'epsilon macchina

def(1)= 2.59460497

### Osservazione 1

Approssimando il numero di Nepero con la formula (1), l'algoritmo produce l'errore minimo all'iterazione numero 4 dove:

l'approssimazione vale 2.7180233  
l'errore relativo 9.50768808E-05

A partire dalla quinta iterazione l'errore aumenta e quindi l'approssimazione del numero di Nepero diventa sempre meno accurata. Ciò è dovuto alla propagazione dell'errore di round-off durante l'esecuzione dell'algoritmo. Infatti, posto

$$x^{(k)} = 1 + \frac{1}{n},$$

consideriamo l'errore relativo di rappresentazione, nel dato, alla k-esima iterazione:

$$\tilde{x}^{(k)} = x^{(k)}(1 + \delta_k) \quad \text{con} \quad |\delta_k| \leq u.$$

È noto (cfr. cap.1) che l'indice di condizionamento relativo della funzione  $(x^{(k)})^n$  è dato da:

$$\left| \frac{z \cdot nz^{n-1}}{z^n} \right| = n$$

ponendo  $z = x^{(k)}$ , per cui il mal condizionamento del problema cresce con  $n$ . Posto, dunque,  $n = 10^k$  calcolando la potenza n-esima l'errore relativo di rappresentazione, nella soluzione, sarà:

$$|E_{rapp}| = n \cdot |\delta_k| = 10^k \cdot |\delta_k| \quad \text{con} \quad |\delta_k| \leq u$$

Pertanto, alla quinta iterazione cioè per  $k = 5$ , si ottiene:

$$|E_{rapp}| = 10^5 \cdot |\delta_5| \leq 10^5 \cdot u = 10^5 \cdot 10^{-7} = 10^{-2}$$

minore della massima accuratezza relativa, in un sistema aritmetico a precisione finita con base  $\beta = 10$  e precisione  $t = 8$ , per cui  $u = \beta^{1-t} = 10^{-7}$ . Quanto detto, è confermato dal risultato della sperimentazione numerica, che per  $k = 5$  produce:

valore calcolato 2.72152352 =  $0.27215235 \cdot 10$   
errore relativo 0.00119258335 =  $0.11925834 \cdot 10^{-2}$

ovvero quest'ultimo è dello stesso ordine di grandezza del massimo errore di rappresentazione.

L'algoritmo descritto è pertanto **instabile!**

Osservazione 2

Dall'iterazione numero 7 in poi, si ottengono i risultati seguenti:

valore calcolato	2.59460497
errore relativo	0.0454981439

cioè il valore dell'approssimazione non varia in quanto  $1/n$ , non dà più contributo alla somma con 1, perché  $1/n$  risulta minore dell'epsilon macchina.

**Problema 2: approssimazione di un limite**

```
program approssimazione di un limite
```

C Fase di dichiarazione delle variabili

```
integer k,n
```

```
real N1,nep,nep1,aus,e,err1,err2
```

C Stampa a video del problema da risolvere

```
write(*,*) , _____ ,  

write(*,*) , Calcolo del limite ,  

write(*,*) , (1)  $x \rightarrow 0$  di  $(\text{EXP}(x)-1)/x$  ,  

write(*,*) , (2)  $x \rightarrow 0$  di  $(\text{EXP}(x)-1)/\log(\text{EXP}(x))$  ,  

write(*,*) , _____ ,  

write(*,*) , _____ ,
```

C Fase di calcolo

```
k = 0  

5 continue  

k=k+1  

n=10**k  

N1=1.0/n  

aus=exp(N1)  

nep1=aus-1  

if(nep1 .ne. 0.0) then  

    e=nep1/(log(aus))  

    aus=nep1/N1  

    err1 = abs(1-aus)  

    err2 = abs(1-e)
```

C Stampa a video della soluzione calcolata

```
print*,k = ,k  

print*,limite (1)=,aus, errore assoluto = ,err1  

print*,limite (2)=,e, errore assoluto = ,err2  

print*, ,  

else
```

C Risultato dovuto alla precisione finita

```
print*,k = ,k  

print*,limite (1) = limite (2) = 0'  

print*, ,  

endif
```

C Terminazione della fase di calcolo

```
if(k .lt. 16)goto 5  

stop  

end
```

**Prova di esecuzione**

---

Calcolo del limite

- (1)  $x \rightarrow 0$  di  $(\text{EXP}(x)-1)/x$   
(2)  $x \rightarrow 0$  di  $(\text{EXP}(x)-1)/\log(\text{EXP}(x))$
- 

k = 1

limite (1)= 1.05170965 errore assoluto = 0.0517096519  
limite (2)= 1.05170918 errore assoluto = 0.0517091751

k = 2

limite (1)= 1.00501776 errore assoluto = 0.00501775742  
limite (2)= 1.0050168 errore assoluto = 0.00501680374

k = 3

limite (1)= 1.00052357 errore assoluto = 0.0005235672  
limite (2)= 1.0005002 errore assoluto = 0.000500202179

k = 4

limite (1)= 1.00016594 errore assoluto = 0.000165939331  
limite (2)= 1.00004995 errore assoluto = 4.99486923E-05

k = 5

limite (1)= 1.00135803 errore assoluto = 0.00135803223  
limite (2)= 1.00000501 errore assoluto = 5.00679016E-06

k = 6

limite (1)= 0.953674316 errore assoluto = 0.0463256836  
limite (2)= 1.00000048 errore assoluto = 4.76837158E-07

k = 7

limite (1)= 1.1920929 errore assoluto = 0.192092896  
limite (2)= 1.00000012 errore assoluto = 0.0

k = 8

limite (1) = limite (2) = 0

---

**Osservazione 1**

L'errore minimo che si commette approssimando il limite assegnato mediante la formula (1), si ottiene all'iterazione numero 4:

valore calcolato del limite	1.00016594
errore assoluto	0.000165939331

ottenendo un'accuratezza fino alla quarta cifra decimale.

Al crescere del numero di iterazioni il risultato diventa sempre meno accurato, a causa della propagazione dell'errore di round-off. Inoltre, dall'ottava iterazione in poi il risultato è del tutto **errato**, infatti l'algoritmo produce:

$$\frac{e^x - 1}{x} \xrightarrow{x \rightarrow 0} 0$$

Analizziamo nel dettaglio cosa avviene.

Nel sistema aritmetico IEEE, utilizzando la singola precisione, il valore di  $e^x$  con  $x = 10^{-8}$ , calcolato mediante la routine *built-in* del FORTRAN restituisce il valore 1:

$$e^{10^{-8}} \equiv 1$$

quindi:

$$e^{10^{-8}} - 1 = 1 - 1 = 0$$

pertanto, a partire dall'iterazione  $k = 8$ , il valore del numeratore della formula (1), calcolato nel sistema aritmetico del calcolatore, è 0, mentre il denominatore ha un valore non nullo, ovvero  $x = 10^{-8}$ .

La formula (2) fornisce un risultato più accurato della formula (1) fino all'iterazione numero 7. Infatti, sia

$$f(x) = \frac{e^x - 1}{\log e^x}$$

indicando con  $\tilde{y} = e^x(1 + \delta)$  con  $|\delta| < u$ , il valore calcolato di  $e^x$ , ed essendo  $fl(\log \tilde{y}) = x(1 + \eta)$  con  $|\eta| < u$ , si ha:

$$\begin{aligned} \tilde{f}(x) &= fl\left(\frac{fl(\tilde{y} - 1)}{fl(\log \tilde{y})}\right) = fl\left(\frac{(\tilde{y} - 1)(1 + \epsilon_1)}{x(1 + \eta)}\right) = \\ &= \frac{(\tilde{y} - 1)(1 + \epsilon_1)}{x(1 + \eta)}(1 + \epsilon_2) = \\ &= \frac{(e^x(1 + \delta) - 1)}{x}(1 + \epsilon_1)(1 + \eta)^{-1}(1 + \epsilon_2) \approx \\ &\approx \frac{e^x - 1}{x}(1 + \epsilon_1)(1 + \eta)^{-1}(1 + \epsilon_2) \simeq f(x)(1 + \theta_3) \end{aligned}$$

con  $|\epsilon_i| < u$  e  $|\theta_3| \leq \frac{3u}{1-3u} = \mathcal{O}(u)$ . Segue che l'errore relativo che si commette nell'approssimare  $f(x)$  con  $\tilde{f}(x)$  è:

$$\frac{|f(x) - \tilde{f}(x)|}{|f(x)|} \approx \mathcal{O}(u).$$

All'aumentare del numero di iterazioni, per la formula (1) il numeratore risulta numericamente nullo, per cui, per  $k \geq 8$ , (1) restituisce valore zero, essendo non nullo il denominatore,  $x$ , mentre essendo il denominatore della (2) nullo, la (2) restituisce un NaN, che denota la presenza di una situazione eccezionale, dovuta alla divisione 0/0.

### Problema 3: generazione di punti equidistanziati

```
program punti equidistanziati
```

C Fase di dichiarazione delle variabili

```
integer k,n,i,j
```

```
real a,b,h,x(0:100000),y(0:100000)
```

C Stampa a video del problema da risolvere

```
write(*,*) , _____ ,  

write(*,*) , Formule che generano i punti ,  

write(*,*) , (1) x0 = a x(k) = x(k - 1) + h k = 1, ..., n ,  

write(*,*) , (2) x(k) = a + h*k k = 0, ..., n ,  

write(*,*) , _____ ,  

write(*,*) , _____ ,
```

C Fase di calcolo

```
print*, 'Inserire gli estremi a,b'
```

```
read*,a,b
```

```
print*,a,b
```

```
print*, 'Inserire il numero dei punti desiderati'
```

```
read*,n
```

```
print*,n
```

```
x(0)=a
```

```
y(0)=a
```

```
h=(b-a)/n
```

C Generazione dei numeri mediante le due formule

```
do 10 k=1,n
```

```
  x(k)=x(k-1)+h
```

```
  y(k)=y(0)+h*k
```

10      continue

C Stampa a video della soluzione calcolata

```
do 20 k=0,n
```

```
  print*, ,
```

```
  print*, 'Punto ',k
```

```
  print*, 'Punti con la formula (1)='
```

```
  write(6,*)x(k)
```

```
  print*, 'Punti con la formula (2)='
```

```
  write(6,*)y(k)
```

```
  print*, ,
```

<sup>20</sup>A. Murli, Matematica Numerica: metodi, algoritmi e software  
Versione in corso di stampa, solo per uso personale, soggetta ad errori.  
Non è autorizzata la diffusione. Tutti i diritti riservati.

## Prova di esecuzione

---

Formule che generano i punti

- (1)  $x_0 = a \quad x(k) = x(k-1) + h \quad k = 1, \dots, n$
  - (2)  $x(k) = a + h*k \quad k = 0, \dots, n$
- 

Inserire gli estremi a,b

0. 1.

Inserire il numero dei punti desiderati

1001

Punto 0

Punti con la formula (1)= 0.

Punti con la formula (2)= 0.

Punto 1

Punti con la formula (1)= 0.000999000971

Punti con la formula (2)= 0.000999000971

Punto 2

Punti con la formula (1)= 0.00199800194

Punti con la formula (2)= 0.00199800194

Punto 3

Punti con la formula (1)= 0.00299700303

Punti con la formula (2)= 0.00299700303

Punto 4

Punti con la formula (1)= 0.00399600388

Punti con la formula (2)= 0.00399600388

Punto 5

Punti con la formula (1)= 0.00499500474

Punti con la formula (2)= 0.00499500474

:

Punto 996

Punti con la formula (1)= 0.994992673

Punti con la formula (2)= 0.995004952

Punto 997

Punti con la formula (1)= 0.995991647

Punti con la formula (2)= 0.996003985

Punto 998

Punti con la formula (1)= 0.996990621

Punti con la formula (2)= 0.997002959

Punto 999

Punti con la formula (1)= 0.997989595

Punti con la formula (2)= 0.998001993

Punto 1000

Punti con la formula (1)= 0.998988569

Punti con la formula (2)= 0.999000967

Punto 1001

Punti con la formula (1)= 0.999987543

Punti con la formula (2)= 1.

### Osservazione 1

Supponiamo di generare 1001 punti nell'intervallo [0,1]. Per semplicità e brevità di scrittura riportiamo solo i primi cinque e gli ultimi cinque elementi da generare:

$$(0 \ 0.001 \ 0.002 \ 0.003 \ 0.004 \dots \ 0.996 \ 0.997 \ 0.998 \ 0.999 \ 1).$$

Sempre per brevità di scrittura, abbiamo riportato i risultati relativi alle due formule solo delle prime cinque iterazioni dell'algoritmo.

Come si può osservare dalla prova di esecuzione, la formula (2) è più accurata della (1). Effettuiamo  $k$  passi dell'algoritmo utilizzando la formula (1) e facciamo un'analisi della

propagazione dell'errore di round-off, indichiamo con  $\epsilon_a$  e  $\epsilon_h$ , l'errore che si commette nel rappresentare rispettivamente  $a$  ed  $h$  in un sistema aritmetico a precisione finita, mentre con  $\delta_{S_k}$  l'errore che si commette eseguendo la somma al passo  $k - esimo$ . Si ha:

$$\begin{aligned}\tilde{x}_0 &= fl(a) = a(1 + \epsilon_a) \\ \tilde{x}_1 &= fl(\tilde{x}_0 + \tilde{h}) = (a(1 + \epsilon_a) + h(1 + \epsilon_h))(1 + \delta_{S_1}) \\ &\vdots \\ \tilde{x}_k &= fl(\tilde{x}_{k-1} + \tilde{h}) = (\tilde{x}_{k-1} + h(1 + \epsilon_h))(1 + \delta_{S_k}) = \\ &= a(1 + \epsilon_a + \delta_{S_1} + \dots + \delta_{S_{k-1}}) + h(1 + \epsilon_h + \delta_{S_1} + \dots + \delta_{S_{k-1}})\end{aligned}$$

con  $|\epsilon_a| \leq u |\epsilon_h| \leq u |\delta_{S_k}| \leq u$   $k = 1, \dots, n$

Si noti che al passo  $k - esimo$  su  $a$  e su  $h$  si accumulano tutti gli errori di round-off  $\delta_{S_k}$  dovuti alle somme effettuate nei  $k - 1$  passi precedenti. Eseguiamo gli stessi  $k$  passi, utilizzando invece la formula (2) e effettuiamo l'analisi della propagazione dell'errore di round-off:

$$\begin{aligned}\tilde{x}_0 &= fl(a) = a + \epsilon_0 \\ \tilde{x}_k &= fl(\tilde{a} + \tilde{h} \cdot k) = (a(1 + \epsilon_0) + (h(1 + \epsilon_h) \cdot k(1 + \epsilon_k))(1 + \delta_P))(1 + \delta_{S_k})\end{aligned}$$

abbiamo indicato con  $\epsilon_a$ ,  $\epsilon_h$ ,  $\epsilon_k$  gli errori che si commettono nel rappresentare rispettivamente  $a$ ,  $h$  e  $k$  in un sistema aritmetico a precisione finita, mentre con  $\delta_P$  l'errore che si commette eseguendo il prodotto  $h \cdot k$  e con  $\delta_{S_k}$  l'errore che si commette eseguendo la somma al passo  $k - esimo$ . Si osserva facilmente che utilizzando la formula (2), al generico passo  $k$  sono presenti sia  $\delta_P$  che  $\delta_{S_k}$ , ma non si ha l'accumulo degli errori  $\delta_{S_i}$   $i = 1, \dots, k - 1$  delle somme effettuate nei  $k - 1$  passi precedenti, che si verifica invece utilizzando la formula (1).

**Problema 4: calcolo della derivata**

```
program calcolo della derivata
```

C Fase di dichiarazione delle variabili

```
integer n,i,j
real x,h,d1,d2,k
```

C Stampa a video del problema da risolvere

```
write(*,*) ,
write(*,*) , _____ Formule per il calcolo della derivata ,
write(*,*) , (1)  $(f(x + h) - f(x))/h$  ,
write(*,*) , (2)  $(f(x + h) - f(x - h))/2h$  ,
write(*,*) ,
write(*,*) , _____ ,
```

C Fase di calcolo

```
print*, 'Inserire il numero di passi ed il punto di valutazione'
```

```
read*,n,x
```

```
print*,n,x
```

```
k=2.
```

10      continue

```
h=1./k
```

```
call der(x,h,d1,d2)
```

```
print*,'
```

```
print*, 'incremento h =',h
```

```
print*, 'Derivata con la formula (1)=',d1
```

```
print*, 'Derivata con la formula (2)=',d2
```

```
print*,'
```

```
k=2*k
```

```
if(k.le.n) goto 10
```

```
stop
```

```
end
```

C Routine per il calcolo della derivata della funzione logaritmo

```
subroutine der(x,h,d1,d2)
```

```
real x,h,d1,d2
```

```
d1=(log(x+h)-log(x))/h
```

```
d2=(log(x+h)-log(x-h))/(2*h)
```

```
return
```

```
end
```

### Prova di esecuzione

---

Formule per il calcolo della derivata

- (1)  $(f(x + h) - f(x))/h$
  - (2)  $(f(x + h) - f(x - h))/2h$
- 

Inserire il numero di passi ed il punto di valutazione  
10000 2.

incremento h = 0.5

Derivata con la formula (1)= 0.446287155

Derivata con la formula (2)= 0.510825634

incremento h = 0.25

Derivata con la formula (1)= 0.47113204

Derivata con la formula (2)= 0.502628803

incremento h = 0.125

Derivata con la formula (1)= 0.484996796

Derivata con la formula (2)= 0.500652552

incremento h = 0.0625

Derivata con la formula (1)= 0.492346764

Derivata con la formula (2)= 0.500163078

incremento h = 0.03125

Derivata con la formula (1)= 0.496133804

Derivata con la formula (2)= 0.500041008

incremento h = 0.015625

Derivata con la formula (1)= 0.498058319

Derivata con la formula (2)= 0.500011444

incremento h = 0.0078125

Derivata con la formula (1)= 0.499511719

Derivata con la formula (2)= 0.5

incremento h = 0.00390625

Derivata con la formula (1)= 0.499023438

Derivata con la formula (2)= 0.5

incremento h = 0.001953125

Derivata con la formula (1)= 0.499755859

Derivata con la formula (2)= 0.5

incremento h = 0.0009765625

Derivata con la formula (1)= 0.49987793

Derivata con la formula (2)= 0.5

incremento h = 0.00048828125

Derivata con la formula (1)= 0.49987793

Derivata con la formula (2)= 0.5

incremento h = 0.000244140625

Derivata con la formula (1)= 0.5

Derivata con la formula (2)= 0.5

incremento h = 0.000122070312

Derivata con la formula (1)= 0.5

Derivata con la formula (2)= 0.5

### Osservazioni

Sia la formula (1) che la formula (2), costituiscono un'approssimazione della derivata. Per entrambe è possibile stimare il massimo errore di discretizzazione che si commette. La formula di Taylor al secondo ordine, di punto iniziale  $\xi \in (x, x + h)$ , è:

$$f(x) = f(\xi) + f'(\xi)(x - \xi) + f''(\xi) \frac{(x - \xi)^2}{2!} + \mathcal{O}((x - \xi)^3)$$

sottraendo ad ambo i membri  $f(\xi)$  e  $f'(\xi)(x - \xi)$  otteniamo:

$$f(x) - f(\xi) - f'(\xi)(x - \xi) = f''(\xi) \frac{(x - \xi)^2}{2!} + \mathcal{O}((x - \xi)^3)$$

Trascurando gli infinitesimi di ordine superiore al secondo e dividendo ambo i membri per  $(x - \xi)$ :

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

passando ai valori assoluti:

$$\left| \frac{f(x) - f(\xi)}{(x - \xi)} - f'(\xi) \right| = \left| f''(\xi) \frac{(x - \xi)}{2} \right|$$

Maggioriamo il secondo membro, sostituendo a  $(x - \xi)$  l'ampiezza  $h$  dell'intervallo  $(x, x + h)$ , otteniamo:

$$\left| \frac{f(x) - f(\xi)}{(x - \xi)} - f'(\xi) \right| \leq \left| f''(\xi) \frac{h}{2} \right|$$

ovvero l'errore assoluto che si commette nell'approssimare la derivata prima  $f'(x)$  con la formula (1), nota come formula della *differenza in avanti su due punti*, è  $\mathcal{O}(h)$ . Deriviamo, ora, la formula (2). La formula di Taylor al terzo ordine, di punto iniziale  $x$  è:

$$f(x \pm h) = f(x) \pm f'(x)h + f''(x) \frac{h^2}{2!} \pm f'''(x) \frac{h^3}{3!} + \mathcal{O}(h^4) \quad (1.53)$$

da cui si ha la formula della *differenza in avanti*:

$$\frac{f(x + h) - f(x)}{h} \simeq f'(x) + f''(x) \frac{h}{2!} + f'''(x) \frac{h^2}{3!} + \mathcal{O}(h^3)$$

e quella della *differenza all'indietro*:

$$\frac{f(x) - f(x - h)}{h} \simeq f'(x) - f''(x) \frac{h}{2!} + f'''(x) \frac{h^2}{3!} - \mathcal{O}(h^3)$$

Sommindo membro a membro e dividendo per 2 le espressioni in avanti ed all'indietro, trascurando gli infinitesimi di ordine superiore al terzo, si ottiene:

$$\frac{f(x + h) - f(x - h)}{2h} \cong f'(x) + f'''(x) \frac{h^2}{6}$$

Sottraendo  $f(x)$  ad ambo i membri e passando ai valori assoluti si ha:

$$\left| \frac{f(x + h) - f(x - h)}{2h} - f'(x) \right| \cong \left| f'''(x) \frac{h^2}{6} \right|.$$

ovvero l'errore assoluto che si commette nell'approssimare la derivata prima  $f'(x)$  con la formula (2), nota come *formula della differenza centrale*, è  $\mathcal{O}(h^2)$ .

In generale le formule per approssimare le derivate che coinvolgono più punti sono più accurate.

La funzione test utilizzata per approssimare il calcolo della derivata è  $f(x) = \ln(x)$ , la cui derivata è  $f'(x) = \frac{1}{x}$ ; nel punto  $x = 2$  risulta  $f'(2) = \frac{1}{2} = 0.5$ . Riportiamo per ogni incremento  $h$  l'errore relativo che si commette utilizzando la formula (1) o la (2):

$h$	$\frac{f(x+h)-f(x)}{h}$	Err. relativo	$\frac{f(x+h)-f(x-h)}{2h}$	Err. relativo
0.5	0.446287155	$0.1074 \times 10^0$	0.510825634	$0.2170 \times 10^{-1}$
0.25	0.47113204	$0.577 \times 10^{-1}$	0.502628803	$0.5257 \times 10^{-2}$
0.125	0.484996796	$0.3006 \times 10^{-1}$	0.500652552	$0.1305 \times 10^{-2}$
0.0625	0.492346764	$0.1531 \times 10^{-1}$	0.500163078	$0.3260 \times 10^{-3}$
0.03125	0.496133804	$0.7732 \times 10^{-2}$	0.500041008	$0.8200 \times 10^{-4}$
0.015625	0.498058319	$0.3883 \times 10^{-2}$	0.500011444	$0.228 \times 10^{-4}$
0.0078125	0.499023438	$0.1953 \times 10^{-2}$	0.5	0.
0.00390625	0.499511719	$0.9766 \times 10^{-3}$	0.5	0.
0.001953125	0.499755859	$0.4884 \times 10^{-3}$	0.5	0.
0.0009765625	0.49987793	$0.2442 \times 10^{-3}$	0.5	0.
0.00048828125	0.49987793	$0.2442 \times 10^{-3}$	0.5	0.
0.000244140625	0.5	0.	0.5	0.
0.0001220703125	0.5	0.	0.5	0.

Come si può notare la formula (2) è più accurata della (1) e, al tendere di  $h$  a zero, converge più velocemente alla derivata in  $x = 2$ .

**Problema 5: calcolo della Serie Armonica**

```
program calcolo Serie Armonica
```

C Fase di dichiarazione delle variabili

```
integer i,flag
real e,epsilon,Nmax
real add,Sum,Sumold,n
```

C Stampa a video del problema

```
write(*,*)      ,
write(*,*)      , _____ Calcolo della Serie Armonica   ,
write(*,*)      , n → inf di S(1/n)           ,
write(*,*)      , _____
write(*,*)      , _____ ,
```

```
write(*,*)"Inserire massimo numero di iterazioni"
read(*,*)Nmax
write(*,*)Nmax
```

C Inizializzazione dell'addendo n-esimo

```
n = 1.0
add = n
i = 0
flag = 0
```

C Inizializzazione della Somma

```
Sum = 1.0
```

C Calcolo dell'epsilon macchina

```
epsilon = eps()
print*, 'Epsilon macchina = ',epsilon
```

C Fase di Calcolo

```
5 continue
i = i + 1
n= n + 1.0
add=1.0/n
if (add .ge. Sum*epsilon ) then
    Sumold = Sum
    Sum = Sum + add
```

```

C Stampa del valore calcolato della Serie
print*,'
print*,'Approssimazione della serie =',Sum
print*,'Iterazioni eseguite = ',i
print*,'
else
  print*, ' 1/n è minore di (S_n-1)*epsilon macchina'
  print*,'Approssimazione della serie =',Sum
  flag = 1
endif
C Terminazione del ciclo di calcolo
if( i .le. Nmax .and. flag .eq. 0 ) goto 5
print*,'Iterazioni eseguite = ',i
stop
end

```

Prima di descrivere una prova di esecuzione dell'algoritmo descritto, si riportano le risposte ai quesiti proposti nel paragrafo 1.10.3:

(a) La serie armonica:

$$\sum_{n=1}^{\infty} \frac{1}{n}$$

è una serie a termini positivi, pertanto la successione delle somme parziali  $\{s_n\}$ , con  $s_n = \sum_{k=1}^n \frac{1}{k}$  è strettamente crescente, e quindi la serie diverge positivamente:

$$\lim_{n \rightarrow \infty} s_n = +\infty$$

(b) In un sistema aritmetico a precisione finita la serie ha somma finita in quanto:

$$s_{k+1} = s_k + \frac{1}{k+1} \neq s_k \iff \frac{1}{k+1} \geq s_k \cdot \epsilon_{mac}$$

dove  $\epsilon_{mac}$  è l'epsilon macchina.

(c) Per il punto (b), in un sistema aritmetico a precisione finita, la somma non cambia all'iterazione  $k - esima$ , per cui si verifica la disegualanza:

$$\frac{1}{k+1} < s_k \cdot \epsilon_{mac}$$

e per ogni  $n \geq k$  risulta:

$$s_n = s_k$$

(d) Poiché la serie è divergente, il calcolo della serie può essere arrestato o quando  $k$  raggiunge un massimo numero di iterazioni assegnato o quando  $k$  raggiunge il valore

che verifica il punto (c).

Riportiamo di seguito alcune prove di esecuzione dell'algoritmo che calcola il valore della serie armonica in precisione finita. Per semplicità di scrittura, dato l'elevato numero di iterazioni eseguite dall'algoritmo, vengono riportate le somme parziali relative alle ultime 4 iterazioni eseguite.

### Prova di esecuzione

---

Calcolo della Serie Armonica  
 $n \rightarrow inf$  di  $S(1/n)$

---

Inserire massimo numero di iterazioni

1000000

Epsilon macchina = 1.192029E-07

Approssimazione della serie = 13.8890009

Iterazioni eseguite = 603970

Approssimazione della serie = 13.8890028

Iterazioni eseguite = 603971

Approssimazione della serie = 13.8890047

Iterazioni eseguite = 603972

Approssimazione della serie = 13.8890066

Iterazioni eseguite = 603973

$1/n$  e' minore di  $(S_{n-1}) * \text{epsilon macchina}$

Approssimazione della serie = 13.8890066

Iterazioni eseguite = 603974

### Osservazione

L'algoritmo si arresta all'iterazione numero  $k + 1 = 603974 + 1 = 603975$ , cioè quando risulta:

$$\frac{1}{k+1} < s_k \cdot \epsilon_{mac}$$

Quindi il punto (c) si realizza all'iterazione numero 603975:

Approssimazione della serie	13.8890066
Errore relativo	1.90734863E-06

Riportiamo di seguito lo stesso algoritmo per il calcolo della serie armonica, sviluppato in doppia precisione.

```
program calcolo Serie Armonica
```

C Fase di dichiarazione delle variabili

```
integer i,flag
double precision e,epsilon,Nmax
double precision add,Sum,Sumold,n
```

```
write(*,*)      ,
write(*,*)      , _____ Calcolo della Serie Armonica ,
write(*,*)      ,   n → inf  di  S(1/n) ,
write(*,*)      ,
write(*,*)      , _____ ,
```

```
write(*,*)"Inserire massimo numero di iterazioni"
read(*,*)Nmax
write(*,*)Namx
```

C Inizializzazione dell'addendo n-esimo

```
n = 1.0
add = n
i = 0
flag = 0
```

C Inizializzazione della Somma

```
Sum = 1.0
```

C Calcolo dell'epsilon macchina precisona doppia

```
epsilon = epsdouble()
print*, 'Epsilon macchina = ',epsilon
```

C Fase di calcolo

```
5  continue
i = i + 1
n= n + 1.0
add=1.0/n
if (add .ge. Sum*epsilon ) then
    Sumold = Sum
    Sum = Sum + add
```

C Calcolo dell'errore assoluto

```
err1 = abs(Sum - Sumold)
```

```

C Stampa del valore calcolato della Serie
    print*, ''
    print*, 'Approssimazione della serie =',Sum
    print*, 'Iterazioni eseguite ',i
    print*, ''
else
    print*, ' 1/n e" minore di S_(n-1)*epsilon macchina'
    print*, 'Approssimazione della serie =',Sum
    flag = 1
endif

C Terminazione del ciclo di calcolo
if( i .le. Nmax .and. flag .eq. 0 ) goto 5
print*, 'Iterazioni eseguite = ',i
stop
end

```

### Prova di esecuzione

Si descrive, di seguito, un esempio di esecuzione di cui si riportano, per semplicità, dato l'elevato numero di iterazioni eseguite, solo i risultati relativi alle ultime cinque iterazioni.

---

Calcolo della Serie Armonica  
 $n \rightarrow inf$  di  $S(1/n)$

---

Inserire massimo numero di iterazioni  
10000000  
Epsilon macchina = 2.22044605E-16

Approssimazione della serie = 16.6953111  
Iterazioni eseguite 9999996

Approssimazione della serie = 16.6953112

Iterazioni eseguite 9999996

Approssimazione della serie = 16.6953113

Iterazioni eseguite 9999997

Approssimazione della serie = 16.6953114

Iterazioni eseguite 9999998

Approssimazione della serie = 16.6953115

Iterazioni eseguite 9999999

Iterazioni eseguite = 10000000

#### Osservazione

Eseguendo i calcoli in doppia precisione l'algoritmo si arresta all'iterazione numero 10000000, fornendo come risultato: 16.6953115. Anche in precisione doppia valgono le stesse considerazioni fatte per i risultati ottenuti dall'algoritmo in singola precisione. In questo caso l'algoritmo si arresta perché è stato raggiunto il massimo numero di iterazioni.

**Problema 6: calcolo della Varianza**

```
program calcolo della Varianza
```

C Fase di dichiarazione delle variabili

```
integer n1,k
real x(100),sigma1,sigma,med,n
```

C Stampa a video del problema da risolvere

```
write(*,*) , _____ ,
write(*,*) , Calcolo della Varianza ,
write(*,*) , (1) 1/(n-1) [sum_{i=1,n} (x(i)-med)^ 2] ,
write(*,*) , (2) 1/(n-1) [sum_{i=1,n} x(i)^ 2 +
write(*,*) , -1/n (sum_{i=1,n} x(i))^ 2] ,
write(*,*) , _____ ,
```

C Fase di calcolo

```
print*,'Inserire la lunghezza del vettore'
read*,n1
print*,n1
print*,'Inserire le componenti del vettore'
do 10 i=1,n1
    read*,x(i)
    print*,x(i)
10    continue
med=0
do 20 k=1,n1
    med=med+x(k)
20    continue
n=n1
med=med/n
sigma=0
sigma1=0
do 30 k=1,n
    sigma= sigma+((x(k)-med)**2)
    sigma1=sigma1+(x(k)**2)
30    continue
sigma= (1./(n-1))*sigma
sigma1=(1./(n-1))*(sigma1-n*(med**2))
```

C Stampa a video dei risultati

```
print*,'
print*,'Media = ',med
print*,'Varianza con la formula (1)= ',sigma
print*,'Varianza con la formula (2)= ',sigma1
print*,'
```

A. Murli, Matematica Numerica: metodi, algoritmi e software  
**Versione in corso di stampa, solo per uso personale, soggetta ad errori.**  
 end

Non è autorizzata la diffusione. Tutti i diritti riservati.

**Prova di esecuzione**

Calcolo della Varianza

$$(1) \frac{1}{(n-1)} \left[ \sum_{i=1}^n (x(i)-\text{med})^2 \right]$$

$$(2) \frac{1}{(n-1)} \left[ \sum_{i=1}^n x(i)^2 + \frac{1}{n} \left( \sum_{i=1}^n x(i) \right)^2 \right]$$

Inserire la lunghezza del vettore

5

Inserire le componenti del vettore

2.

4.

7.

8.

10.

Media = 6.19999981

Varianza con la formula (1)= 10.1999998

Varianza con la formula (2)= 10.2000008

Calcolo della Varianza

$$(1) \frac{1}{(n-1)} \left[ \sum_{i=1}^n (x(i)-\text{med})^2 \right]$$

$$(2) \frac{1}{(n-1)} \left[ \sum_{i=1}^n x(i)^2 + \frac{1}{n} \left( \sum_{i=1}^n x(i) \right)^2 \right]$$

Inserire la lunghezza del vettore

3

Inserire le componenti del vettore

10000.

10001.

10002.

Media = 10001.

Varianza con la formula (1)= 1.

Varianza con la formula (2)= 0.

### Osservazioni

Le formule utilizzate dall'algoritmo sono:

$$(1) \sigma^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

$$(2) \sigma^2 = \frac{1}{n-1} \left( \sum_{i=1}^n x_i^2 - \frac{1}{n} \left( \sum_{i=1}^n x_i \right)^2 \right)$$

analizziamo le due prove di esecuzione separatamente.

Nella prima prova il vettore è  $\bar{x} = (2, 4, 7, 8, 10)$ . In aritmetica a precisione infinita  $\bar{x}$  ha media  $m = 6.2$  e varianza  $\sigma^2 = 10.2$ . L'errore relativo che si commette utilizzando la formula (1) è:

$$\frac{|\sigma^2 - \tilde{\sigma}^2|}{|\sigma^2|} = \frac{|10.2 - 10.1999998|}{10.2} = 0.2 \times 10^{-8}$$

utilizzando invece la seconda formula

$$\frac{|\sigma^2 - \tilde{\sigma}^2|}{|\sigma^2|} = \frac{|10.2 - 10.2000008|}{10.2} = 0.7 \times 10^{-8}$$

in entrambi i casi il risultato è accettabile, si ottiene un' accuratezza fino alla settima cifra significativa.

Analizziamo ora la seconda prova. Il vettore è  $\bar{x} = (10000, 10001, 10002)$ . In aritmetica a precisione infinita  $\bar{x}$  ha media  $m = 10001$  e varianza  $\sigma^2 = 1$ . L'errore relativo che si commette utilizzando la formula (1) è:

$$\frac{|\sigma^2 - \tilde{\sigma}^2|}{|\sigma^2|} = \frac{|1.0 - 1.0|}{1.0} = 0$$

utilizzando invece la seconda formula

$$\frac{|\sigma^2 - \tilde{\sigma}^2|}{|\sigma^2|} = \frac{|1.0 - 0.0|}{1.0} = 0.1 \times 10^1$$

Questo secondo esempio, mette in luce il fatto che in aritmetica a precisione finita la formula (2) può generare effetti di cancellazione distruttiva e condurre ad un risultato del tutto scorretto.

# Bibliografia

- [1] Cody W.J. et al. - *A Proposed Radix-and Wordlength-Independent Standard for Floating-Point Arithmetic* - IEEE Micro, 84, 1984.
- [2] Dew P.M., James K.R. - *Introduction to Numerical Computation in Pascal* - Springer-Verlag, 1985.
- [3] Forsythe A.I., Keenan T.A., Organick E.I., Stenberg W. - *Computer Science: a First Course*. - John Wiley & Sons, 1969.
- [4] Forsythe G.E. - *What is a Satisfactory Quadratic Equation Solver?* - in *Constructive Aspects of the Fundamental Theorem of Algebra* - Proceedings of a Symposium conducted at the IBM Research Laboratory, Zürich-Rüschlikon, Switzerland, June 5-7, 1967. Edited by Bruno Dejon and Peter Henrici, pp. 53–61.
- [5] Forsythe G.E., Malcom M.A., Moler C.B. - *Computer methods for mathematical computations* - Prentice-Hall, 1977.
- [6] Gear C.W. - *Computer Organization and Programming* - McGraw-Hill, IV edition, 1985.
- [7] Higham N.J. - *Accuracy and stability of numerical algorithms* - II ed., SIAM 2002.
- [8] Kahan W., Coonen J.T., Stone H. - *A Proposed Standard Floating Point Arithmetic* - ACM SIGNUM Newsletter, special issue, Oct. 1979.
- [9] Landau L., Lifchitz E. - *Phisique Théorique. Théorie de l'Élasticité* - tome VII, Mir, 1967.
- [10] Lax P., Burstein S., Lax A. - *Analisi matematica con applicazioni e calcolo numerico* - Zanichelli, 1986.
- [11] Sterbentz P. - *Floating point computation* - Prentice-Hall (1978)
- [12] Wallis P.J.L. - *Improving Floating-Point Programming* - John Wiley & Sons, 1990.
- [13] Wilkinson J.H. - *Rounding Errors in Algebraic Processes* - Prentice-Hall, 1963.

# Capitolo 2

## Calcolo matriciale: metodi diretti

### 2.1 Introduzione al calcolo numerico matriciale

L'obiettivo ultimo della Matematica Numerica è quello di fornire strumenti software per la risoluzione effettiva di problemi concreti. Nel processo di risoluzione di un problema, la fase di risoluzione numerica del modello matematico che descrive il problema in esame, gioca un ruolo fondamentale, intendendo con ciò la fase che va dalla formulazione del modello numerico fino allo sviluppo di software. I modelli matematici più semplici ed i più utilizzati sono di tipo lineare e sono frequentemente rappresentati da sistemi di equazioni lineari. Essi sono di fondamentale importanza in quanto l'analisi e lo studio di problemi concreti, anche estremamente complessi, sono in primo luogo effettuati su un modello lineare semplificato. Tali sistemi derivano direttamente dall'applicazione di leggi che regolano il fenomeno in esame come accade in molti campi della scienza e della tecnica, oppure compaiono nel processo di risoluzione numerica di alcuni classici problemi, quali interpolazione, fitting, ottimizzazione e approssimazione di problemi differenziali. In particolare, è stato stimato che il calcolo della soluzione di un sistema lineare costituisce il nucleo principale del processo di risoluzione di almeno il 70% di tutti i problemi scientifici. Ciò motiva la necessità di avere a disposizione metodi, algoritmi e software affidabili ed efficienti per la risoluzione di tali problemi.

### 2.2 Introduzione ai sistemi di equazioni lineari

♣ **Esempio 2.1.** Assegnate due rette del piano, rispettivamente di equazione:

$$\begin{aligned}4x - y &= -10; \\2x - y &= -20,\end{aligned}$$

si vogliono determinare le coordinate del punto di intersezione. Se  $P$  è un tale punto, allora le coordinate  $(x, y)$  di  $P$  devono soddisfare entrambe le equazioni ovvero **costituiscono una soluzione del sistema**

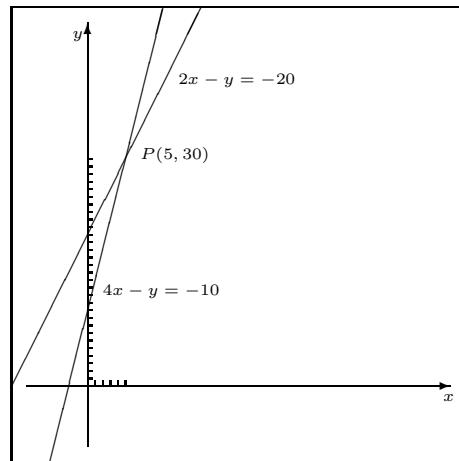


Figura 2.1: Sistema lineare di ordine  $n = 2$ : interpretazione geometrica.

**di equazioni lineari formato dalle equazioni delle due rette.** Il sistema coinvolge **due** equazioni in **due** incognite e viene perciò detto **sistema di ordine  $n = 2$** .

È chiaro che solo alcune coppie di valori  $(x, y)$  soddisfano la prima equazione (ad es. la coppia  $(2, 18)$  soddisfa la prima equazione mentre la coppia  $(3, 20)$  non la soddisfa) e che soltanto alcune coppie  $(x, y)$  soddisfano la seconda equazione. Il problema che ci si pone è determinare se esiste almeno una coppia di valori  $(x, y)$  che soddisfi ambedue le equazioni e, quindi, il sistema di due equazioni in due incognite formato dalle equazioni delle due rette.

Tale sistema può essere risolto sottraendo la seconda equazione dalla prima, ottenendo

$$2x = 10$$

da cui si ricava

$$x = 5;$$

e sostituendo il valore ottenuto per  $x$  nella prima equazione, cioè:

$$20 - y = -10$$

da cui si ricava

$$y = 30.$$

Così la coppia di valori  $(5, 30)$  soddisfa il sistema di equazioni e quindi il punto  $P$  del piano di coordinate  $(5, 30)$  è punto di intersezione delle due rette, come mostrato in Figura 2.1, rappresentando nel piano  $(x, y)$  le due rette assegnate. ♣

La Figura 2.1 mostra il significato geometrico di un sistema di equazioni lineari di ordine  $n = 2$ . Molti problemi, appartenenti ai più svariati settori applicativi e quindi in apparenza completamente differenti, sono descritti in modo naturale da un modello matematico basato su un sistema di equazioni lineari.

♣ **Esempio 2.2.** Si considerino 2 industrie con produzioni diverse, ma con la condizione che ciascuna debba disporre di una certa quantità del prodotto dell'altra. Ad esempio un'industria automobilistica compra pneumatici da un'industria di pneumatici, mentre l'industria di pneumatici acquista camion dall'industria automobilistica per il trasporto del suo prodotto.

Si supponga che l'industria automobilistica per produrre un camion necessiti di 5 pneumatici, mentre l'industria di pneumatici abbia bisogno di 1 camion per ogni 50 pneumatici prodotti. Indicate rispettivamente con  $x$  e  $y$  la quantità di camion e la quantità di pneumatici prodotte nelle 2 industrie al termine del ciclo di produzione, si ha che per soddisfare almeno le esigenze di entrambe le industrie, deve verificarsi

$$\begin{aligned} x &\geq \frac{1}{50}y \\ y &\geq 5x \end{aligned}$$

È chiaro che le due industrie devono soddisfare anche le esigenze dei consumatori, includendo, tra questi, eventuali altre industrie.

Supponendo che siano richiesti, dai consumatori, 3000 camion e 30000 pneumatici, si ha che la quantità totale di camion da produrre è:

$$x = \frac{1}{50}y + 3000$$

e la quantità totale di pneumatici da realizzare è:

$$y = 5x + 30000$$

Quindi, per determinare i valori  $x$  e  $y$  in modo che siano soddisfatte le esigenze dei consumatori e dei processi di produzione, è necessario risolvere il seguente sistema di equazioni lineari:

$$\begin{cases} x - \frac{1}{50}y = 3000 \\ -5x + y = 30000 \end{cases}$$

Moltiplicando per 50 la prima equazione si ottiene il sistema:

$$\begin{cases} 50x - y = 150000 \\ -5x + y = 30000 \end{cases}$$

Addizionando le due equazioni si ha:

$$45x = 180000$$

da cui si ricava:

$$x = 180000/45 = 4000;$$

sostituendo il valore ottenuto per  $x$  nella seconda equazione si ha:

$$-5 \times 4000 + y = 30000$$

da cui si ottiene:

$$y = 50000.$$



♣ **Esempio 2.3.** La determinazione di un possibile schema di reazione chimica è il punto di partenza della maggior parte degli studi cinetici. Più in dettaglio il problema è il seguente: in un esperimento sono identificate alcune specie di molecole (intendendo per specie di molecola *una* molecola di un particolare composto); si vuole determinare quali possono essere le molecole reagenti e quali quelle prodotte e le relative quantità, tenendo presente che il vincolo imposto nella reazione chimica è costituito dalla *Legge di conservazione della massa* (Legge di Lavoisier).

L'applicazione di tale legge conduce ad un sistema di equazioni lineari. In tale sistema ciascun coefficiente rappresenta il numero di atomi di una certa sostanza presente in una delle molecole identificate ( $a_{i,j}$  indica quanti atomi della sostanza  $i$  sono presenti nella molecola  $j$ ) e ciascuna incognita rappresenta un coefficiente (detto *stechiometrico*) che sarà il coefficiente associato ad una molecola nell'equazione finale (da ricavare) che descrive la trasformazione totale. In particolare il valore assoluto del coefficiente stochiometrico fornisce il numero di molecole di un certo composto che entra nella reazione finale ed il segno indica se la molecola associata ha il ruolo di reagente (segno -) o di prodotto (segno +).

Supponendo che in un esperimento campione siano state identificate le seguenti specie di molecole:



si vuole determinare quali di queste molecole possono essere reagenti e quali prodotto e in che quantità ciascuna molecola interviene nella reazione (ovvero qual'è una possibile equazione chimica del tipo:  $\sigma_1CH_2Cl_2 + \sigma_2CH_4 + \sigma_3CH_3Cl = 0$ ).

I coefficienti del sistema lineare sono riportati nella seguente tabella:

atomi	molecole		
	$CH_2Cl_2$	$CH_4$	$CH_3Cl$
C	1	1	1
H	2	4	3
Cl	2	0	1

Le incognite da determinare sono quindi, in numero di 3 (essendo 3 le molecole);  $\sigma_1$  per  $CH_2Cl_2$ ,  $\sigma_2$  per  $CH_4$ ,  $\sigma_3$  per  $CH_3Cl$ . Il sistema si scrive tenendo presente la *Legge di Lavoisier* e quindi

$$\begin{cases} \sigma_1 + \sigma_2 + \sigma_3 = 0 \\ 2\sigma_1 + 4\sigma_2 + 3\sigma_3 = 0 \\ 2\sigma_1 + \sigma_3 = 0 \end{cases}$$

In questo caso si è in presenza di 3 equazioni (quanti sono gli atomi) in 3 incognite. Supponendo, ad esempio, che  $\sigma_1 = -1$  che equivale a sapere che nella reazione  $CH_2Cl_2$  è una molecola reagente, si ottengono i seguenti valori per le rimanenti incognite:  $\sigma_2 = -1$ ,  $\sigma_3 = 2$ . Pertanto la reazione finale, nell'ipotesi considerata, è:



♣ **Esempio 2.4.** Un circuito elettrico è una combinazione di conduttori progettato per il trasporto di energia elettrica o la trasmissione di segnali elettrici. In un circuito RLC, ohmico, induttivo e capacitivo (resistenza R, induttanza L, capacità C collegate in serie e attraversate da corrente alternata) trascorsa

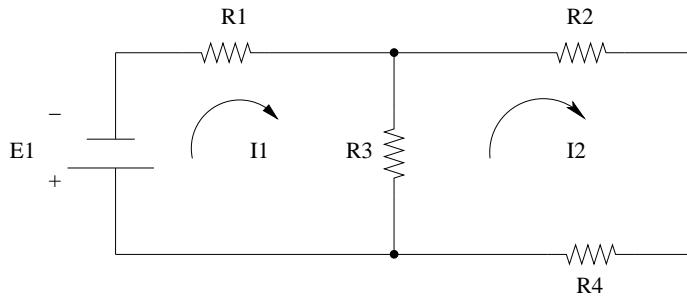


Figura 2.2: Primo circuito elettrico.

una fase transitoria, si rileva il passaggio di una corrente di tipo sinusoidale avente la stessa frequenza della forza elettromotrice (f.e.m.).

Si consideri, ad esempio, il circuito riportato in Figura 2.2 in cui sono presenti 4 resistenze ( $R_1, R_2, R_3, R_4$ ) ed un generatore di f.e.m. ( $E_1$ ). Si supponga che la f.e.m. sia costante e che siano state raggiunte condizioni di stazionarietà nel circuito cosicché le correnti ( $I_1, I_2$ ) siano anch'esse costanti. Generalmente il problema consiste nel trovare le correnti in funzione delle f.e.m. e delle resistenze. Per risolvere questo problema si applicano le *Leggi di conservazione della carica elettrica e dell'energia*<sup>1</sup>.

L'applicazione di tali leggi richiede di chiarire alcune convenzioni. Si considerino

- positive le correnti che si allontanano dal nodo, negative quelle dirette verso il nodo;
- positiva una caduta di potenziale attraverso una resistenza se ci si muove nello stesso verso della corrente, negativa se ci si muove nel verso opposto;
- positiva la caduta di potenziale se si passa attraverso una sorgente di f.e.m., nella direzione in cui agisce la sorgente (aumento di potenziale), negativa se si passa nella direzione opposta (diminuzione di potenziale).

L'applicazione delle *Leggi di Kirchhoff* al circuito in Figura 2.2 permette di determinare le correnti  $I_1$  e  $I_2$ . In particolare la *seconda Legge* applicata alle due maglie in cui circolano le correnti, entrambe orientate in senso orario, fornisce il sistema:

$$\begin{cases} -E_1 = R_1 \cdot I_1 + R_3 \cdot I_1 - R_3 \cdot I_2 \\ 0 = R_3 \cdot I_2 + R_2 \cdot I_2 + R_4 \cdot I_2 - R_3 \cdot I_1 \end{cases}$$

da cui è possibile ricavare le due correnti in funzione delle resistenze e della f.e.m. presenti nel circuito:

$$\begin{cases} I_1 = \frac{-E_1 \cdot R_3}{R_1 \cdot (R_2 + R_3 + R_4) + R_3 \cdot (R_2 + R_4)} \\ I_2 = \frac{-E_1 \cdot (R_2 + R_3 + R_4)}{R_1 \cdot (R_2 + R_3 + R_4) + R_3 \cdot (R_2 + R_4)} \end{cases}$$

Nel caso in cui le correnti siano negative si deduce che il loro verso effettivo è opposto a quello da noi scelto.

<sup>1</sup>Si tratta delle due *Leggi di Kirchhoff* secondo cui:

1. la somma di tutte le correnti in un nodo di un circuito vale zero;
2. la somma di tutte le differenze di potenziale lungo un qualsiasi percorso chiuso nel circuito vale zero.

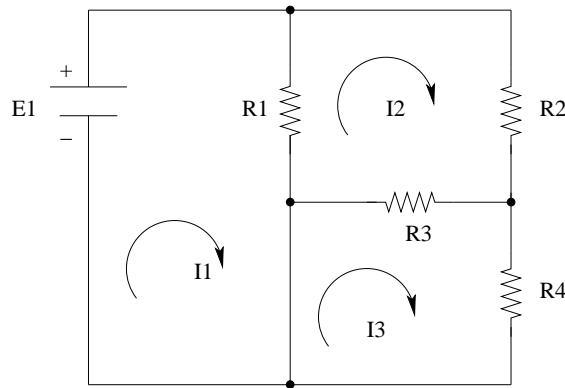


Figura 2.3: Secondo circuito elettrico.

Si consideri, ora, il circuito riportato in Figura 2.3.  
L'applicazione delle *Leggi di Kirchhoff* permette di determinare le tre correnti presenti nel circuito:  $I_1$ ,  $I_2$ ,  $I_3$ ; anche in questo esempio supponiamo che siano tutte orientate in senso orario. In particolare, la *seconda Legge* applicata alle tre maglie in cui rimane suddiviso il circuito conduce al sistema:

$$\begin{cases} E_1 = R_1 \cdot I_1 - R_1 \cdot I_2 \\ 0 = R_1 \cdot I_2 + R_2 \cdot I_2 + R_3 \cdot I_2 - R_3 \cdot I_3 - R_1 \cdot I_1 \\ 0 = R_3 \cdot I_3 + R_4 \cdot I_3 - R_3 \cdot I_2 \end{cases}$$

Dalla prima equazione si ricava:

$$I_1 = \frac{E_1}{R_1} + I_2$$

mentre dalla terza si ricava:

$$I_3 = \frac{R_3}{R_4 + R_3} \cdot I_2$$

Sostituendo tali espressioni nella seconda equazione del sistema, si ricava:

$$I_2 = \frac{(R_3 + R_4) \cdot E_1}{R_2 \cdot R_3 + R_2 \cdot R_4 + R_3 \cdot R_4}$$

I valori di  $I_1$  e di  $I_3$  si ricavano, quindi, dalla loro espressione in funzione di  $I_2$ .



Gli esempi ora descritti mostrano che problemi di natura completamente diversa possono essere descritti in modo naturale da sistemi lineari e la loro risoluzione consiste nella risoluzione di sistemi di equazioni lineari.

## 2.3 Introduzione alla risoluzione numerica di sistemi lineari

♣ **Esempio 2.5.** Si voglia risolvere il sistema di equazioni lineari:

$$\begin{cases} 2x + 5y = 31 \\ 3x + 2y = 19 \end{cases}$$

che può essere indicato nella forma compatta:

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

con

$$A = \begin{pmatrix} 2 & 5 \\ 3 & 2 \end{pmatrix}; \quad \mathbf{z} = \begin{pmatrix} x \\ y \end{pmatrix}; \quad \mathbf{b} = \begin{pmatrix} 31 \\ 19 \end{pmatrix}.$$

Una possibilità per la risoluzione del sistema assegnato è l'utilizzo della regola di Cramer. Tale metodo, applicato al sistema in esame, consiste nei passi seguenti:

1. calcolo del determinante di  $A$ :

$$\det(A) = \begin{vmatrix} 2 & 5 \\ 3 & 2 \end{vmatrix} = 2 \times 2 - 5 \times 3 = -11$$

2. calcolo di  $x$  e  $y$ :

$$x = \frac{\begin{vmatrix} 31 & 5 \\ 19 & 2 \end{vmatrix}}{-11} = \frac{31 \times 2 - 5 \times 19}{-11} = 3;$$

$$y = \frac{\begin{vmatrix} 2 & 31 \\ 3 & 19 \end{vmatrix}}{-11} = \frac{2 \times 19 - 31 \times 3}{-11} = 5$$

Le operazioni aritmetiche effettuate sono:

- passo 1 2 moltiplicazioni e 1 sottrazione;
- passo 2 4 moltiplicazioni, 2 divisioni e 2 sottrazioni.

Quindi, il calcolo della soluzione di un sistema  $2 \times 2$  mediante il metodo di Cramer richiede, in totale

11 flop.

avendo indicato con *flop* una delle quattro operazioni aritmetiche floating point<sup>2</sup>.



<sup>2</sup>Invero, in generale, nello studio del condizionamento di un problema di algebra lineare, si assume la seguente

**Definizione 2.1. (Flop)**

*Si definisce flop (floating-point operation) l'operazione in virgola mobile costituita da una moltiplicazione e da un'addizione.*

♣ **Esempio 2.6.** Si consideri il sistema lineare seguente:

$$\begin{cases} 10x_1 + x_2 - 5x_3 = 1 \\ -20x_1 + 3x_2 + 20x_3 = 2 \\ 5x_1 + 3x_2 + 5x_3 = 6 \end{cases}$$

La risoluzione di tale sistema mediante il metodo di Cramer consiste dei passi seguenti:

- calcolo del determinante della matrice del sistema:

$$\begin{vmatrix} 10 & 1 & -5 \\ -20 & 3 & 20 \\ 5 & 3 & 5 \end{vmatrix} = 10 \times \begin{vmatrix} 3 & 20 \\ 3 & 5 \end{vmatrix} + 20 \times \begin{vmatrix} 1 & -5 \\ 3 & 5 \end{vmatrix} + 5 \times \begin{vmatrix} 1 & -5 \\ 3 & 20 \end{vmatrix} = \\ = 10 \times (3 \times 5 - 20 \times 3) + 20 \times (1 \times 5 + 5 \times 3) + 5 \times (1 \times 20 + 5 \times 3) = 125$$

- calcolo di  $x_1$ ,  $x_2$  ed  $x_3$ :

$$x_1 = \frac{\begin{vmatrix} 1 & 1 & -5 \\ 2 & 3 & 20 \\ 6 & 3 & 5 \end{vmatrix}}{125} = 1;$$

$$x_2 = \frac{\begin{vmatrix} 10 & 1 & -5 \\ -20 & 2 & 20 \\ 5 & 6 & 5 \end{vmatrix}}{125} = -2;$$

$$x_3 = \frac{\begin{vmatrix} 10 & 1 & 1 \\ -20 & 3 & 2 \\ 5 & 3 & 6 \end{vmatrix}}{125} = 1.4.$$

Le operazioni aritmetiche effettuate sono:

- passo 1 14 flop per il calcolo di un determinante di ordine 3;
- passo 2 45 flop.

---

Il concetto di flop appena definito è oramai ampiamente utilizzato in Analisi Numerica quando si analizza la complessità computazionale degli algoritmi. Ciò è dovuto al fatto che, nei calcoli su vettori e matrici, l'usuale operazione di base è rappresentata da una moltiplicazione seguita da una addizione. Si pensi ad esempio al calcolo del prodotto tra una matrice ed un vettore, cioè al calcolo di  $y = Ax$ , con  $x$  e  $y$  vettori di dimensione  $n$  ed  $A$  matrice quadrata di dimensione  $n$ . L'operazione fondamentale per il calcolo della  $i$ -ma componente del vettore  $y$  è  $y_i = a_{ij} \cdot x_j + y_i$ , che deve essere eseguita per tutti i valori di  $j$  da 1 a  $n$ . Tale operazione è costituita, quindi, da una moltiplicazione e da un'addizione (tipicamente indicata come **saxpy**, "scalar" " $\alpha$ " times vector "x", "plus" vector "y").

In totale, sono richieste:

59 flop

Si osservi che per calcolare la soluzione del sistema  $3 \times 3$  in esame è stato necessario valutare 4 determinanti di ordine 3, ciascuno dei quali richiede a sua volta la valutazione di 3 determinanti di ordine 2.



La complessità computazionale di un algoritmo dipende, essenzialmente, da alcuni parametri, tra i quali ricordiamo  $\mathcal{T}(n)$  e  $\mathcal{S}(n)$  che rappresentano, rispettivamente, la complessità di tempo e di spazio dell'algoritmo in funzione di  $n$ , dimensione del problema. La complessità di tempo si misura attraverso il numero di operazioni floating point, *flops*, eseguite dall'algoritmo. La complessità di spazio, invece, dipende dal numero di variabili utilizzate dall'algoritmo. In generale, sia  $\mathcal{T}(n)$  sia  $\mathcal{S}(n)$  sono espresse come funzioni **asintotiche** di  $n$ , ed a tal fine si usa il *simbolo di Landau*,  $\mathcal{O}$ <sup>3</sup>

---

<sup>3</sup>Per il simbolo di Landau si assume la seguente definizione

### Definizione 2.2. (Simbolo di Landau: $\mathcal{O}$ )

Date due funzioni  $f(x)$  e  $g(x)$ , tali che:

$$\lim_{x \rightarrow \infty} f(x) = \infty$$

e:

$$\lim_{x \rightarrow \infty} g(x) = \infty$$

si pone:

$$f(x) = \mathcal{O}(g(x))$$

se:

$$\lim_{x \rightarrow \infty} \frac{f(x)}{g(x)} = C \quad (C = cost \neq 0)$$

In altre parole, nel dare l'andamento asintotico di  $\mathcal{T}(n)$  e  $\mathcal{S}(n)$  si evidenzia il termine dominante in  $\mathcal{T}(n)$  e  $\mathcal{S}(n)$ , al crescere di  $n$ .

Nel risolvere un sistema di equazioni lineari di ordine  $n$  con il metodo di Cramer è necessario valutare  $n + 1$  determinanti di ordine  $n$ , ciascuno dei quali richiede la somma di  $n$  termini ciascuno costruito mediante  $1 + (n - 1)!$  moltiplicazioni.

Quindi, il numero totale di moltiplicazioni richieste per la valutazione degli  $n + 1$  determinanti di ordine  $n$  è dato da:

$$T_{Cramer}(n) = (n + 1)n(n - 1)! + (n + 1)n.$$

Da ciò si ricava che la complessità computazionale del metodo di Cramer applicato ad un sistema di ordine  $n$  è:

$$T_{Cramer}(n) = \mathcal{O}((n + 1)!). \quad (2.1)$$

Nella Tabella 2.1 sono riportati, relativamente ad un sistema lineare di ordine  $n$  (con  $n = 5$ ,  $n = 10$  e  $n = 20$ ), il numero di operazioni richiesto per la sua risoluzione mediante il metodo di Cramer ed il tempo di calcolo necessario per eseguire tali operazioni su calcolatori aventi velocità operative differenti. In particolare sono presi in esame: un personal computer con processore (Intel Pentium 4 a 2.2 GHz), una workstation (IBM pseries 655 a 1.3 GHz), un mainframe (HP server rx5670), un supercomputer (CRAY C90 a 16 processori) ed un processore Intel Pentium IV a 1.5 GHz. Inoltre, si suppone che nell'esecuzione dell'algoritmo la velocità operativa massima di ciascun calcolatore (riportata nella tabella), misurata in milioni di operazioni floating-point al secondo (Mflops), sia effettivamente raggiunta<sup>4</sup>.

Si osservi che, considerato un sistema lineare di ordine 20, disponendo di un calcolatore che effettua oltre 15 miliardi di operazioni floating-point al secondo, il tempo di calcolo richiesto per la sua risoluzione mediante il metodo di Cramer è di 106 anni!! La Tabella 2.1 mette in luce quindi la impraticabilità del metodo di Cramer a causa

---

<sup>4</sup>

### Definizione 2.3. (Mflops, Gflops, Tflops)

**1 Mflops** (*1 milion of floating-point operations per second*) = *1 milione di operazioni floating-point al secondo*,

è una delle unità di misura utilizzate per indicare la velocità operativa di un calcolatore. Altre unità sono:

**1 Gflops** = *1 miliardo di operazioni floating-point al secondo*;

**1 Tflops** = *1000 miliardi di operazioni floating-point al secondo*.

Ad esempio, 100 Mflops era, negli anni Novanta, la velocità operativa tipica di una workstation *IBM RISC/6000 43P modello 240* con processore *166MHz Power PC 640e*.

L'evoluzione ha condotto, inevitabilmente, allo sviluppo di processori con più elevate velocità operative, alcuni dei quali sono elencati in tabella con, in particolare, dettagli relativi al numero di operazioni floating point eseguite al secondo.

(I dati riportati in tabella sono aggiornati agli inizi del 2001 e reperibili dal sito internet

[http://web.maths.unsw.edu.au/~rsw/MATH5315/processors.shtml. \)](http://web.maths.unsw.edu.au/~rsw/MATH5315/processors.shtml.)

<i>n</i>	# op.	Pers. Comp. 4400 Mflops	Workstation 5200 Mflops	Mainframe 4000 Mflops	Supercomp. 15238 Mflops	Pentium IV 3 Gflops
5	720	$1.6 \times 10^{-7}$ sec.	$1.4 \times 10^{-7}$ sec.	$1.8 \times 10^{-7}$ sec.	$4.7 \times 10^{-8}$ sec.	$2.4 \times 10^{-7}$ sec.
10	$3.99 \times 10^7$	$9.1 \times 10^{-3}$ sec.	$7.7 \times 10^{-3}$ sec.	$9.9 \times 10^{-3}$ sec.	$2.6 \times 10^{-3}$ sec.	0.0133 sec.
20	$5.1 \times 10^{19}$	367 anni	310 anni	404 anni	106 anni	539 anni

**Tabella 2.1: Tempo richiesto per risolvere un sistema di equazioni con il metodo di Cramer su calcolatori con differenti velocità operative.**

dell'elevata complessità computazionale<sup>5</sup>.

Appare evidente che il metodo di Cramer non costituisce uno strumento effettivo per la risoluzione di un sistema lineare, ovvero non consente di ottenere una risposta al problema in esame, in quanto l'espressione della soluzione fornita dal metodo non può essere valutata nella pratica. Di conseguenza anche la valutazione del determinante della matrice dei coefficienti per stabilire l'unicità della soluzione non è applicabile nella

Processore	Marca	Clock (MHz)	Peak (Mflops)
Alpha 21264b	Compaq	833	1666
UltraSPARC III	SUN	900	1800
MIPS R12000	SGI	400	800
PA-8700	HP	750	3000
Power3-II	IBM	450	1800
Pentium IV	Intel	1500	1500
Pentium III	Intel	1000	1000
Athalon	AMD	1330	1330
PowerPC G4	Apple	733	2932

Nella tabella si è indicato, in particolare, con **peak**, il *peak performance* (picco di prestazione) di un processore, ovvero il numero massimo operazioni floating point al secondo, che il processore può raggiungere in circostanze "ideali", mentre **clock** indica la *frequenza di clock*. In particolare i dati in tabella fanno riferimento ad un *flop* come alla somma di un'addizione e di una moltiplicazione f.p.; per cui, ad esempio, il *peak* di un Pentium IV è 3000 Mflops (3 Gflops), se si considera che il processore esegue due operazioni floating point per tempo di clock.

Nuovi processori sono, costantemente, in evoluzione; per aggiornamenti si può fare riferimento, ad esempio, ai siti:

<http://www.aceshardware.com>

<http://www.a1-electronics.co.uk/index.html>

<http://www.chipanalyst.com/>

<sup>5</sup>In particolare si parla di complessità non polinomiale (NP)

pratica.

In conclusione, la regola di Cramer fornisce una rappresentazione della soluzione e non uno strumento effettivo di calcolo! Ciò che invece si sta cercando è un metodo “*costruttivo*”, ovvero che consenta di risolvere in maniera effettiva un sistema lineare, intendendo con questo che:

- si possa fare uso, nella sua applicazione, degli strumenti di calcolo disponibili (calcoli a mano, calcolatrici, calcolatori, ...);
- abbia una complessità di tempo “*accettabile*”, ovvero coerente con lo strumento di calcolo utilizzato ed adeguata sia al problema che si sta risolvendo sia allo scopo per cui si sta risolvendo il problema;
- richieda una complessità di spazio “*ammissibile*”, sempre in relazione allo strumento di calcolo.

Per arrivare a determinare un tale metodo, si può seguire un ragionamento che parte dalla individuazione di metodi per la risoluzione di sistemi di equazioni lineari “*semplici*”.

## 2.4 Metodi di back e forward substitution

Quali sistemi sono “semplici da risolvere”?

♣ **Esempio 2.7.** Si vuole risolvere il sistema:

$$\begin{cases} 7x_1 &= 3 \\ 6.5x_2 &= 2 \\ -8x_3 &= 1.4 \end{cases}$$

In questo caso la forma del sistema suggerisce in modo evidente il metodo di risoluzione che consiste nel:

1. calcolo di  $x_1$  dalla I equazione

$$x_1 = \frac{3}{7};$$

2. calcolo di  $x_2$  dalla II equazione

$$x_2 = \frac{2}{6.5} = \frac{4}{13};$$

3. calcolo di  $x_3$  dalla III equazione

$$x_3 = \frac{1.4}{-8} = -\frac{7}{40}.$$



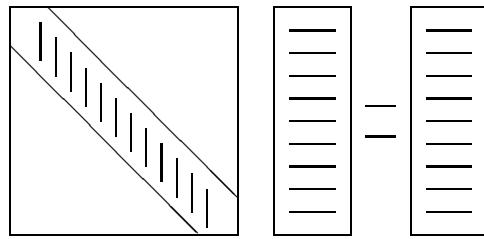


Figura 2.4: Sistema diagonale

**Definizione 2.4. (Sistema Diagonale)**

Un sistema lineare del tipo:

$$\left\{ \begin{array}{lll} a_{1,1}x_1 & & = b_1 \\ a_{2,2}x_2 & & = b_2 \\ a_{3,3}x_3 & & = b_3 \\ \vdots & & \vdots \\ a_{n,n}x_n & & = b_n \end{array} \right.$$

in cui  $a_{i,j} = 0$ , se  $i \neq j$  e con  $i, j = 1, \dots, n$ , definisce un sistema diagonale.

La soluzione di un sistema diagonale è data da:

$$x_i = b_i/a_{i,i}, \quad i = 1, \dots, n$$

se  $a_{i,i} \neq 0$  per  $i = 1, \dots, n$ . Il numero di operazioni richieste per il calcolo della soluzione è, evidentemente,

$$T_{diag}(n) = n \text{ flop}$$

ovvero:

$$T_{diag}(n) = \mathcal{O}(n)$$

Un sistema diagonale può essere rappresentato graficamente nel modo rappresentato in Figura 2.4.

Si osservi che una matrice avente la struttura della matrice dei coefficienti di un sistema diagonale, è detta *matrice diagonale*. In generale, la struttura di una matrice diagonale è la seguente:

$$\begin{pmatrix} a_{1,1} & 0 & \dots & \dots & 0 \\ 0 & a_{2,2} & \dots & \dots & 0 \\ 0 & \dots & a_{3,3} & \dots & 0 \\ \vdots & \dots & \dots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & a_{n,n} \end{pmatrix}.$$

Ad esempio la matrice:

$$A = \begin{pmatrix} 5 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & -7.4 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}; \quad A = \text{diag}(5, 2, -7.4, 1)$$

è una matrice diagonale.

Vi sono altri sistemi “semplici” da risolvere.

♣ **Esempio 2.8.** Si voglia risolvere il sistema di equazioni lineari:

$$\left\{ \begin{array}{l} 2x_1 + 2x_2 + 4x_3 = 5 \\ 7x_2 + 11x_3 = 8 \\ 2x_3 = 2 \end{array} \right.$$

Anche in questo caso la forma del sistema suggerisce direttamente un metodo di risoluzione che consiste nei passi seguenti:

1. calcolo di  $x_3$  dalla III equazione

$$x_3 = 2/2 = 1;$$

2. calcolo di  $x_2$  dalla II equazione (utilizzando per  $x_3$  il valore calcolato al passo 1)

$$x_2 = \frac{8 - 11x_3}{7} = \frac{8 - 11 \times 1}{7} = \frac{-3}{7};$$

3. calcolo di  $x_1$  dalla prima equazione (utilizzando per  $x_3$  e  $x_2$  i valori calcolati ai passi precedenti)

$$x_1 = \frac{5 - 2x_2 - 4x_3}{2} = \frac{5 - 2 \times (-3/7) - 4 \times 1}{2} = \frac{13}{14}.$$

Tale metodo è detto *metodo di sostituzione all’indietro (back-substitution)* in virtù del modo e dell’ordine in cui le incognite sono calcolate. Il numero di operazioni effettuate per risolvere il sistema in esame è:

- passo 1    1 flop
- passo 2    3 flop
- passo 3    5 flop

e quindi, in totale:

$$9 \text{ flop}.$$

In maniera analoga il sistema:

$$\left\{ \begin{array}{l} 2x_1 = 4 \\ 3x_1 + 2x_2 = 5 \\ x_1 + 2x_2 - 3x_3 = 1 \end{array} \right.$$

può essere risolto applicando il metodo seguente:

1. calcolo di  $x_1$  dalla I equazione

$$x_1 = \frac{4}{2} = 2;$$

2. calcolo di  $x_2$  dalla II equazione (utilizzando per  $x_1$  il valore calcolato al passo 1)

$$x_2 = \frac{5 - 3x_1}{2} = \frac{5 - 3 \times 2}{2} = -\frac{1}{2};$$

3. calcolo di  $x_3$  dalla III equazione (utilizzando per  $x_1$  e  $x_2$  i valori calcolati ai passi precedenti)

$$x_3 = \frac{1 - x_1 - 2x_2}{-3} = \frac{1 - 2 - 2 \times (-1/2)}{-3} = 0.$$

Il metodo precedente è detto *metodo di sostituzione in avanti (forward-substitution)*. Anche in questo caso il numero totale di operazioni effettuate è:

9 flop.



In generale è possibile dare la seguente:

### Definizione 2.5. (Sistema triangolare superiore)

*Un sistema lineare del tipo:*

$$\left\{ \begin{array}{l} u_{1,1}x_1 + u_{1,2}x_2 + u_{1,3}x_3 + \dots + u_{1,n}x_n = b_1 \\ u_{2,2}x_2 + u_{2,3}x_3 + \dots + u_{2,n}x_n = b_2 \\ u_{3,3}x_3 + \dots + u_{3,n}x_n = b_3 \\ \vdots \\ u_{n,n}x_n = b_n \end{array} \right. \quad (2.2)$$

in cui  $u_{i,j} = 0$  se  $i > j$ , con  $i, j = 1, \dots, n$ , è detto triangolare superiore.

Un sistema lineare triangolare superiore del tipo (2.2) può essere anche indicato nella forma compatta:

$$Ux = b$$

con:

$$U = \begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \dots & u_{1,n} \\ u_{2,2} & u_{2,3} & \dots & u_{2,n} \\ u_{3,3} & \dots & u_{3,n} \\ \vdots & & \vdots \\ u_{n,n} & & & & \end{pmatrix} \quad (2.3)$$

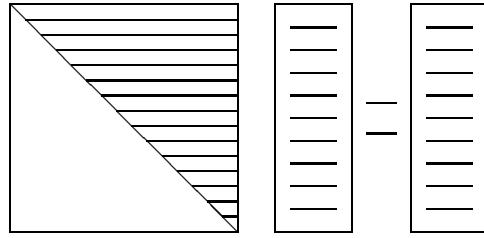


Figura 2.5: Sistema triangolare superiore

Un sistema triangolare superiore è rappresentato graficamente in Figura 2.5.

Una matrice del tipo (2.3), in cui sono nulli tutti gli elementi del triangolo inferiore, ovvero:

$$u_{i,j} = 0 \text{ se } i > j$$

è detta *triangolare superiore*.

Ad esempio la matrice:

$$U = \begin{pmatrix} 1 & 3 & 7 & -4 \\ 0 & 5.3 & 8 & 2.6 \\ 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

è triangolare superiore.

La soluzione di un sistema triangolare superiore può essere ottenuta mediante il metodo di *back-substitution* che calcola:

- dalla  $n$ -ma equazione

$$x_n = \frac{b_n}{u_{n,n}};$$

- dalla  $i$ -ma equazione, per  $i = n - 1, n - 2, \dots, 1$

$$\begin{aligned} x_i &= (b_i - u_{i,i+1} \cdot x_{i+1} - u_{i,i+2} \cdot x_{i+2} - \dots - u_{i,n} \cdot x_n) / u_{i,i} = \\ &= (b_i - \sum_{k=i+1}^n u_{i,k} \cdot x_k) / u_{i,i}. \end{aligned}$$

È evidente che il metodo di *back-substitution* risulta applicabile se  $u_{i,i} \neq 0$ , per  $i = 1, \dots, n$ .

Il numero di operazioni effettuate è:

$$\begin{aligned} &1 \text{ flop per il calcolo di } x_n \\ &3 \text{ flop per il calcolo di } x_{n-1} \\ &5 \text{ flop per il calcolo di } x_{n-2} \\ &2(n-i) + 1 \text{ flop per il calcolo di } x_i, \quad i = n-3, \dots, 1 \end{aligned}$$

e quindi, sommando tutti i precedenti termini si ha:

$$1 + 3 + 5 + \dots + 2(n-1) + 1 \text{ flop} = n^2 \text{ flop}$$

da cui possiamo concludere che la complessità computazionale del metodo di *back-substitution*, applicato ad un sistema triangolare superiore, è:

$$\text{Complessità di tempo } T_{bs}(n) = n^2$$

$$\text{Complessità di spazio } S_{bs}(n) = n^2 + 2n$$

o, in termini asintotici:

$$\begin{aligned} T_{bs}(n) &= \mathcal{O}(n^2) \\ S_{bs}(n) &= \mathcal{O}(n^2) \end{aligned} \tag{2.4}$$

Analogamente è possibile dare la seguente:

### Definizione 2.6. (Sistema triangolare inferiore)

*Un sistema lineare del tipo:*

$$\left\{ \begin{array}{l} l_{1,1}x_1 = b_1 \\ l_{2,1}x_1 + l_{2,2}x_2 = b_2 \\ l_{3,1}x_1 + l_{3,2}x_2 + l_{3,3}x_3 = b_3 \\ \quad \quad \quad \ddots \quad \quad \quad \vdots \\ l_{n,1}x_1 + l_{n,2}x_2 + l_{n,3}x_3 + \dots + l_{n,n}x_n = b_n \end{array} \right. \tag{2.5}$$

in cui  $l_{i,j} = 0$  se  $i < j$ , con  $i, j = 1, \dots, n$ , è detto triangolare inferiore.

Analogamente, considerato un sistema triangolare inferiore del tipo (2.5), esso può essere scritto nella forma:

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

con:

$$L = \begin{pmatrix} l_{1,1} & & & & \\ l_{2,1} & l_{2,2} & & & \\ l_{3,1} & l_{3,2} & l_{3,3} & & \\ \vdots & \dots & & \ddots & \\ l_{n,1} & l_{n,2} & l_{n,3} & \dots & l_{n,n} \end{pmatrix} \tag{2.6}$$

Un sistema triangolare inferiore è rappresentato graficamente in Figura 2.6.

Una matrice del tipo (2.6), in cui sono nulli tutti gli elementi del triangolo superiore, ovvero

$$l_{i,j} = 0 \quad \text{se} \quad i < j$$

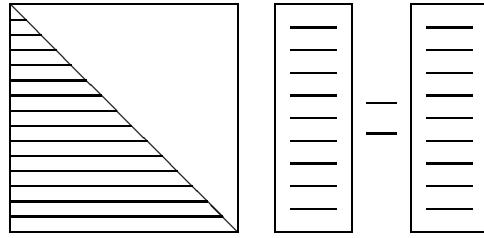


Figura 2.6: Sistema triangolare inferiore

è detta *triangolare inferiore*.

Ad esempio la matrice:

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 6 & 3.3 & 0 & 0 \\ 2.4 & -3 & 6 & 0 \\ -1 & 3.5 & 2 & 0 \end{pmatrix}$$

è una matrice triangolare inferiore.

La soluzione di un sistema triangolare inferiore può essere calcolata mediante il metodo di *forward-substitution* che calcola:

- dalla 1 equazione

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

- dalla  $i$ -ma equazione, per  $i = 2, 3, \dots, n$

$$\begin{aligned} x_i &= (b_i - l_{i,1} \cdot x_1 - l_{i,2} \cdot x_2 - \dots - l_{i,i-1} \cdot x_{i-1}) / l_{i,i} = \\ &= (b_i - \sum_{k=1}^{i-1} l_{i,k} \cdot x_k) / l_{i,i}. \end{aligned}$$

Anche in questo caso il metodo è applicabile nell'ipotesi che  $l_{i,i} \neq 0$  per  $i = 1, 2, \dots, n$ .

Con un procedimento analogo a quello utilizzato per valutare il numero di operazioni aritmetiche richieste dal metodo di *back-substitution*, si ricava che il metodo di *forward-substitution*, applicato ad un sistema triangolare inferiore di ordine  $n$ , ha una complessità computazionale:

**Complessità di tempo**  $T_{fs}(n) = n^2$

**Complessità di spazio**  $S_{fs}(n) = n^2 + 2n$

o, in termini asintotici

$$T_{fs}(n) = \mathcal{O}(n^2)$$

$$S_{fs}(n) = \mathcal{O}(n^2)$$

Gli algoritmi di *back-substitution* e *forward-substitution* sono riportati rispettivamente nella Procedura 2.1 e nella Procedura 2.2.

```

procedure backsub (in:  $u, b, n$ ; out:  $x$ )
  /* SCOPO: Risoluzione di un sistema triangolare superiore.
  /* SPECIFICHE DEI PARAMETRI:
  /* PARAMETRI DI INPUT:
var:  $n$  : intero { dimensione del sistema }
var:  $u(n, n)$  : reale { matrice del sistema }
var:  $b(n)$  : reale { vettore dei termini noti }

  /* PARAMETRI DI OUTPUT:
var:  $x(n)$  : reale { vettore delle soluzioni }
                           { del sistema }

  /* VARIABILI LOCALI:
var:  $i, j$  : interi

  /* INIZIO ISTRUZIONI:
 $x(n) := b(n)/u(n, n)$ 
for  $i = n - 1$  to  $1$  step  $-1$ 
   $x(i) := b(i)$ 
  for  $j = i + 1$  to  $n$ 
     $x(i) := x(i) - u(i, j) * x(j)$ 
  end for
   $x(i) := x(i)/u(i, i)$ 
end for
end backsub

```

Procedura 2.1: Backward-substitution - schema generale

```

procedure forwsub (in:  $l, b, n$ ; out:  $x$ )
  /* SCOPO: Risoluzione di un sistema triangolare inferiore.
  /* SPECIFICHE DEI PARAMETRI:
  /* PARAMETRI DI INPUT:
  var:  $n$  : intero { dimensione del sistema }
  var:  $l(n, n)$  : reale { matrice del sistema }
  var:  $b(n)$  : reale { vettore dei termini noti }

  /* PARAMETRI DI OUTPUT:
  var:  $x(n)$  : reale { vettore delle soluzioni }
    { del sistema }

  /* VARIABILI LOCALI:
  var:  $i, j$  : interi

  /* INIZIO ISTRUZIONI:
   $x(1) := b(1)/l(1, 1)$ 
  for  $i = 2$  to  $n$ 
     $x(i) := b(i)$ 
    for  $j = 1$  to  $i - 1$ 
       $x(i) := x(i) - l(i, j) * x(j)$ 
    end for
     $x(i) := x(i)/l(i, i)$ 
  end for
end forwsub

```

Procedura 2.2: Forward-substitution - schema generale

Si osservi che sia il metodo di *back-substitution* sia quello di *forward-substitution* richiedono, per la valutazione dell'incognita generica  $x_i$ , rispettivamente per  $i = n - 1, \dots, 1$  e per  $i = 2, \dots, n$ , il calcolo di una somma di termini, ciascuno dei quali è il prodotto di un coefficiente della  $i$ -ma equazione per una incognita il cui valore è stato già calcolato. Il risultato di tale somma costituisce il prodotto scalare di due vettori. In particolare:

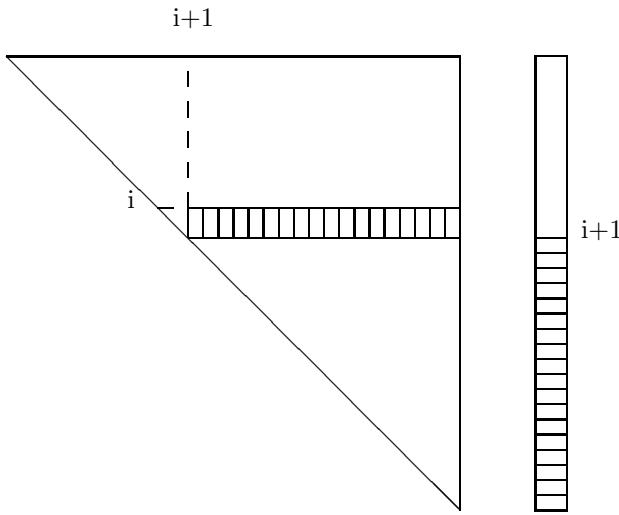


Figura 2.7: Sistema triangolare inferiore

- per il metodo di *back-substitution* ( Figura 2.7 ) si ha:

$$\sum_{k=i+1}^n u_{i,k} x_k = (u_{i,i+1} \ u_{i,i+2} \ \dots \ u_{i,n})(x_{i+1} \ x_{i+2} \ \dots \ x_n)^T$$

- per il metodo di *forward-substitution* ( Figura 2.8 ) si ha:

$$\sum_{k=1}^{i-1} l_{i,k} x_k = (l_{i,1} \ l_{i,2} \ \dots \ l_{i,i-1})(x_1 \ x_2 \ \dots \ x_{i-1})^T$$

Quindi, il prodotto scalare di due vettori costituisce l'operazione di calcolo matriciale alla base dei metodi di *back* e *forward-substitution*.

L'applicazione dei metodi di *back* e *forward-substitution* ad un sistema triangolare, rispettivamente superiore ed inferiore, richiede, ad ogni passo, una divisione per un elemento della diagonale, per cui è necessario chiedersi se tali elementi siano tutti distinti da zero. La risposta a tale quesito si ottiene osservando che il determinante di una matrice  $T = (t_{i,j})$ , con  $i, j = 1, \dots, n$ , triangolare (inferiore o superiore) è espresso da:  $\det(T) = t_{1,1} \cdot t_{2,2} \cdots t_{n,n}$ .

Poiché, assegnata una matrice  $T$  di dimensione  $n$ , si ha:

$$T \text{ non singolare} \iff \det(T) \neq 0$$

se  $T$  è una matrice triangolare, si ottiene:

$$T \text{ triangolare non singolare} \iff t_{i,i} \neq 0 \quad \forall i = 1, \dots, n.$$

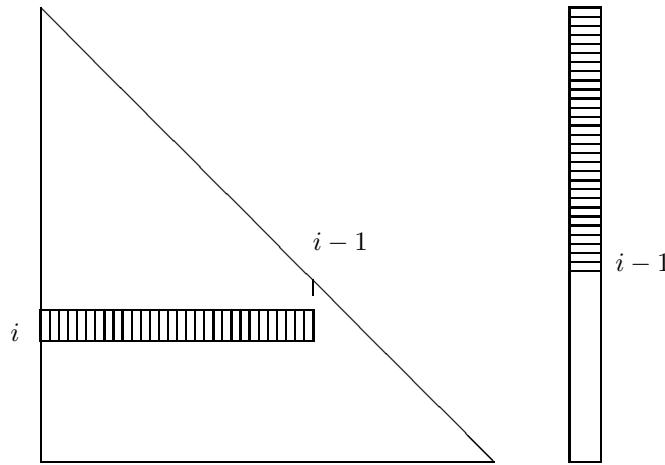


Figura 2.8: Sistema triangolare inferiore

Quindi, dato un sistema di ordine  $n$ ,  $Tx = b$ , con  $T$  triangolare non singolare, si ha che gli elementi diagonali di  $T$  sono non nulli, e ciò permette di applicare il metodo di *back* o *forward-substitution*<sup>6</sup>.

Si supponga, ora, di avere un sistema  $Tx = b$  di ordine  $n$ , con  $T$  triangolare superiore singolare. In tal caso si ha che  $\det(T) = t_{1,1}t_{2,2}\dots t_{n,n} = 0$  e pertanto  $T$  ha almeno un elemento diagonale nullo, ovvero si ha almeno per un indice  $i$  tale che  $t_{i,i} = 0$ . Allora la  $i$ -ma equazione del sistema risulta:

$$0x_i + t_{i,i+1}x_{i+1} + t_{i,i+2}x_{i+2} + \dots + t_{i,n}x_n = b_i$$

Da ciò si ricava che:

- se il sistema è compatibile ma indeterminato, e quindi ammette infinite soluzioni, si ha:

$$r_i = b_i - \sum_{k=i+1}^n t_{i,k}x_k = 0$$

(  $r_i$  si dice *resto della i-ma equazione* ) cioè la  $i$ -ma equazione è verificata per qualsiasi valore di  $x_i$ . Una delle infinite soluzioni si ottiene assegnando un valore a scelta a  $x_i$  ( ad esempio  $x_i = 0$  ) e proseguendo nell'applicazione del metodo di sostituzione;

- se il sistema è incompatibile risulta:

$$r_i \neq 0$$

e l'applicazione del metodo di sostituzione termina.

---

<sup>6</sup>Analoghe considerazioni sussistono per un sistema diagonale.

Occorre osservare, comunque, che la prima proprietà descritta, non si inverte. Più in dettaglio, può verificarsi che la matrice sia singolare, con, in particolare,  $t_{i,i} = 0$  e  $r_i = 0$ , per un generico  $i$ , ma il sistema risulti incompatibile piuttosto che compatibile ma indeterminato. Ad esempio, se

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & \mathbf{0} & 0 & 0 & 0 \\ 4 & 3 & 2 & 1 & 0 & 0 \\ 5 & 4 & 3 & 2 & 1 & 0 \\ 6 & 5 & 4 & 3 & 2 & \mathbf{0} \end{pmatrix} \quad \text{e} \quad b = \begin{pmatrix} 1 \\ 3 \\ 2 \\ 10 \\ 15 \\ 10000 \end{pmatrix}$$

il sistema

$$Tx = b$$

è caratterizzato da  $T$  singolare, ed inoltre, per  $i = 3$ ,  $t_{3,3} = 0$  e  $r_3 = 0$ ; scegliendo  $x_3 = 0$ , procedendo nei conti si trova

$$\begin{aligned} 1 \cdot x_1 &= 1 \\ 2 \cdot x_1 + x_2 &= 3 \Rightarrow x_2 = 1 \\ x_1 + x_2 + 0 \cdot x_3 &= 2 \Rightarrow r_3 = 0; \\ &\vdots \\ 6x_1 + 5x_2 + 4x_3 + 3x_4 + 2x_5 + 0 \cdot x_6 &= 10000 \Rightarrow 0 \neq 10000 \end{aligned}$$

il sistema non ha soluzione!!

Quindi, concludendo, le condizioni  $t_{i,i} = 0$  e  $r_i = 0$  non sono sufficienti per affermare che il sistema è compatibile ma indeterminato. Per queste considerazioni, allora, risulta preferibile terminare, utilizzando un opportuno indicatore di errore, l'esecuzione dell'algoritmo, per la risoluzione di un sistema che presenta tali caratteristiche, potendo, quest'ultimo, risultare incompatibile.

Pertanto, l'applicazione di un metodo di sostituzione non consente di stabilire se un sistema triangolare risulta compatibile ma indeterminato oppure incompatibile. Ci si aspetta, quindi, di ricavare una soluzione solo nel caso in cui questa sia unica.

Per quanto descritto è possibile modificare la Procedura 2.1 ottenendo la Procedura 2.3 che effettua un controllo sulla singolarità del sistema 2.2. Considerazioni analoghe sussistono per il sistema triangolare inferiore (2.6), per il quale è possibile modificare la Procedura 2.2 nella Procedura 2.4.

```

procedure backsub (in:  $u, b, n$  ; out:  $x, comp$ )
  /# SCOPO: Risoluzione di un sistema triangolare superiore.
  /# SPECIFICHE DEI PARAMETRI:
  /# PARAMETRI DI INPUT:
var:  $n$  : intero { dimensione del sistema }
var:  $u(n, n)$  : reale { matrice del sistema }
var:  $b(n)$  : reale { vettore dei termini noti }

  /# PARAMETRI DI OUTPUT:
var:  $x(n)$  : reale { vettore delle soluzioni }
                    { del sistema }
var:  $comp$  : intera { =1 se il sistema è determinato}
                    { =2 se il sistema è}
                    { indeterminato }
                    { =3 se il sistema è }
                    { incompatibile }

  /# VARIABILI LOCALI:
var:  $i, j$  : interi
var:  $r$  : reale

  /# VARIABILI LOCALI:
var:  $i, j$  : interi
var:  $r$  : reale

  /# INIZIO ISTRUZIONI:
 $comp := true$  { sistema compatibile }
if ( $u(n, n) \neq 0$ ) then
   $x(n) := b(n)/u(n, n)$  { calcolo di  $x(n)$  }
else if ( $b(n) = 0$ ) then
   $x(n) := 0$  { il sistema potrebbe essere incompatibile }
   $comp := 2$  { o compatibile ma indeterminato }
else

```

Procedura 2.3: Backward-substitution con controllo della singolarità - continua

```

    comp := 3           {sistema incompatibile}
end if
    i := n - 1
while (i ≠ 0 and comp ≠ 3)           {ciclo sugli x(i)}
        r := b(i)
        for j = i + 1 to n           {calcolo del resto}
            r := r - u(i, j) * x(j)
        end for
        if (u(i, i) ≠ 0) then
            x(i) := r/u(i, i)           {calcolo di x(i)}
        else if (r = 0) then
            x(i) := 0 {il sistema potrebbe essere incompatibile}
            comp := 2           {o compatibile ma indeterminato}
        else
            comp := 3           {sistema incompatibile}
        end if
        i := i - 1
    end while
end backsub

```

Procedura 2.3: Backward-substitution con controllo della singolarità - fine

```

procedure forwsub(in:  $l, b, n$  ; out:  $x, comp$ )
  /# SCOPO: Risoluzione di un sistema triangolare inferiore.
  /# SPECIFICHE DEI PARAMETRI:
  /# PARAMETRI DI INPUT:
    var:  $n$  : intero { dimensione del sistema }
    var:  $l(n, n)$  : reale { matrice del sistema }
    var:  $b(n)$  : reale { vettore dei termini noti }

  /# PARAMETRI DI OUTPUT:
    var:  $x(n)$  : reale { vettore delle soluzioni }
                           { del sistema }
    var:  $comp$  : intera { =1 se il sistema è determinato}
                           { =2 se il sistema è }
                           { indeterminato }
                           { =3 se il sistema è }
                           { incompatibile }

  /# VARIABILI LOCALI:
    var:  $i, j$  : interi
    var:  $r$  : reale

  /# INIZIO ISTRUZIONI:
   $comp := true$  {sistema compatibile}
  if ( $l(1, 1) \neq 0$ ) then
     $x(1) := b(1)/l(1, 1)$  {calcolo di  $x(1)$ }
  else if ( $b(1) = 0$ ) then
     $x(1) := 0$  {il sistema potrebbe essere incompatibile}
     $comp := 2$  {o compatibile ma indeterminato}
  else
     $comp := 3$  {sistema incompatibile}
  end if
   $i := 2$ 
  while ( $i \leq n$  and  $comp \neq 3$ ) {ciclo sugli  $x(i)$ }
     $r := b(i)$ 

```

Procedura 2.4: Forward-substitution con controllo della singolarità - continua

```

for  $j = 1$  to  $i - 1$  {calcolo del resto}
     $r := r - l(i, j) * x(j)$ 
end for

if ( $l(i, i) \neq 0$ ) then
     $x(i) := r / l(i, i)$  {calcolo di x(i)}
else if ( $r = 0$ ) then
     $x(i) := 0$  {il sistema potrebbe essere incompatibile}
     $comp := 2$  {o compatibile ma indeterminato}
else
     $comp := 3$  sistema incompatibile
end if
     $i := i + 1$ 
end while
end forwsub

```

Procedura 2.4: Forward-substitution con controllo della singolarità - fine

## 2.5 Metodo di Eliminazione di Gauss

### 2.5.1 Derivazione elementare del metodo di Gauss

Nel paragrafo precedente si è visto che per la risoluzione di un sistema diagonale o triangolare superiore o inferiore si dispone di algoritmi “*semplici*”. Purtroppo, nella maggior parte dei casi è necessario risolvere un sistema lineare che non è triangolare superiore o inferiore.

Un approccio, frequentemente utilizzato in Analisi Numerica, per la risoluzione di un dato problema è di trasformarlo in uno o più problemi equivalenti di più semplice risoluzione.

Nel nostro caso l’idea è di operare sul sistema in modo da trasformarlo in un sistema triangolare superiore (o inferiore) equivalente (ovvero che ammette la stessa soluzione).

♣ **Esempio 2.9.** Si consideri il sistema di equazioni:

$$\begin{cases} 2x - y = -1 \\ 3x + y = 3 \end{cases} \quad (2.7)$$

La soluzione di tale sistema è data da  $x = 0.4, y = 1.8$ , coordinate del punto di intersezione delle due rette ( Figura 2.9 ) . Come mostrato in Figura 2.10, il punto  $P(0.4,1.8)$  è anche punto di intersezione

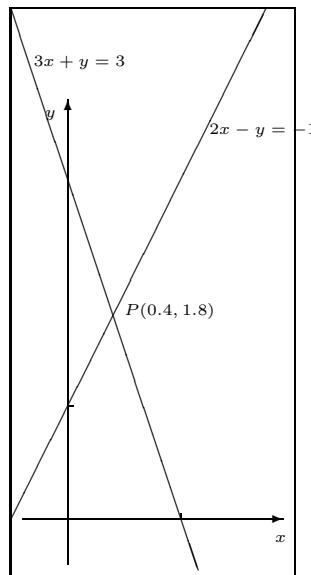


Figura 2.9: Rappresentazione grafica del sistema (2.7): la soluzione è data da  $x = 0.4$ ,  $y = 1.8$ , coordinate del punto di intersezione delle due rette

di altre due rette, ovvero è soluzione di un sistema equivalente al (2.7):

$$\begin{cases} 2x - y = -1 \\ y = 9/5, \end{cases} \quad (2.8)$$

che è un sistema triangolare! Ci si chiede quindi con quale metodo, a partire dal sistema (2.7) sia possibile arrivare ad un sistema triangolare equivalente (2.8).

Lo scopo di tale metodo è l'eliminazione dell'incognita  $x$  dalla II equazione, ovvero la sostituzione della II equazione con un'altra nella sola incognita  $y$ . Ciò può essere ottenuto combinando opportunamente le due equazioni date. Per realizzare ciò si eseguono le operazioni seguenti:

1. si moltiplica per  $3/2$  la I equazione:

$$\frac{3}{2}(2x - y) = \frac{3}{2} \cdot (-1) \implies 3x - \frac{3}{2}y = -\frac{3}{2};$$

in tal modo si ottengono due equazioni aventi ugual coefficiente della  $x$ ;

2. si sottrae la II equazione dalla I e l'equazione che così si ottiene,

$$-\frac{5}{2}y = -\frac{9}{2},$$

è una equazione nella sola  $y$  che si considera come II equazione del sistema in luogo di quella data.

Pertanto al termine delle operazioni suddette si ottiene il sistema:

$$\begin{cases} 2x - y = -1 \\ y = 9/5; \end{cases}$$

che è un sistema triangolare superiore equivalente a quello dato; a tale sistema è applicabile l'algoritmo di back-substitution.



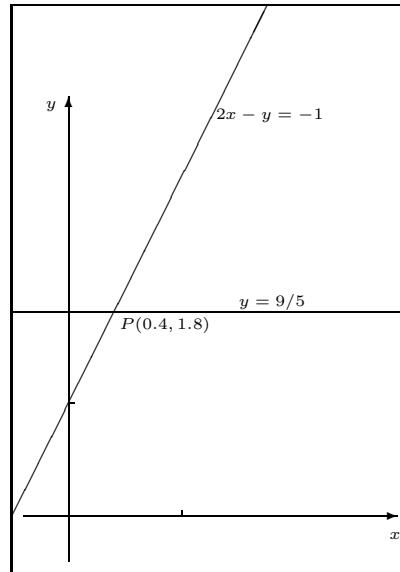


Figura 2.10: Rappresentazione grafica del sistema (2.8), equivalente al (2.7)

Per illustrare l'idea alla base del metodo di Gauss, si consideri l'esempio che segue.

♣ **Esempio 2.10.** Dato il sistema di equazioni:

$$\begin{cases} 2x_1 + 4x_2 - 2x_3 = 6 \\ x_1 - x_2 + 5x_3 = 0 \\ 4x_1 + x_2 - 2x_3 = 2 \end{cases}$$

si ha:

**passo 1.**

**scopo:**

eliminare l'incognita  $x_1$  dalla II e III equazione (ovvero trasformare le ultime due equazioni che contengono tre incognite, in due equazioni che contengono due incognite).

Per realizzare ciò si sottrae da ciascuna di tali equazioni la I moltiplicata per un opportuno fattore. In particolare si opera nel seguente modo:

- si moltiplica per  $1/2$  la prima equazione ottenendo:

$$x_1 + 2x_2 - x_3 = 3$$

e si sottrae tale equazione dalla seconda ottenendo

$$0x_1 - 3x_2 + 6x_3 = -3;$$

- si moltiplica per 2 la prima equazione ottenendo:

$$4x_1 + 8x_2 - 4x_3 = 12$$

e si sottrae tale equazione dalla terza ottenendo

$$0x_1 - 7x_2 + 2x_3 = -10.$$

Pertanto al termine del passo 1. si ottiene il seguente sistema equivalente a quello assegnato:

$$\left\{ \begin{array}{rcl} 2x_1 + 4x_2 - 2x_3 & = 6 \\ -3x_2 + 6x_3 & = -3 \\ -7x_2 + 2x_3 & = -10 \end{array} \right.$$

## passo 2.

### scopo:

*eliminare  $x_2$  dalla III equazione del sistema ottenuto* (ovvero trasformare l'ultima equazione che contiene due incognite, in una equazione che contiene una sola incognita).

Per realizzare ciò si applica al sistema costituito dalle ultime due equazioni, lo stesso procedimento effettuato al passo 1. sull'intero sistema. Pertanto:

- si moltiplica per  $7/3$  la seconda equazione, ottenendo:

$$-7x_2 + 14x_3 = -7,$$

e si sottrae questa equazione dalla terza ottenendo

$$0x_2 - 12x_3 = -3.$$

Al termine di tale passo si ottiene, quindi, il seguente sistema equivalente ai precedenti:

$$\left\{ \begin{array}{rcl} 2x_1 + 4x_2 - 2x_3 & = 6 \\ -3x_2 + 6x_3 & = -3 \\ -12x_3 & = -3. \end{array} \right.$$

Il sistema ottenuto è triangolare superiore e quindi può essere risolto applicando l'algoritmo di back-substitution.

Si osservi che i fattori  $1/2$ ,  $2$  e  $7/3$ , in virtù dell'uso che di essi viene fatto, sono detti *moltiplicatori* e sono dati dal rapporto tra il coefficiente dell'incognita che deve essere eliminata in una certa equazione ed il coefficiente della stessa incognita nell'equazione che viene utilizzata per effettuare il procedimento di eliminazione (nell'esempio, la I equazione al passo 1., la II equazione al passo 2.).

Chiaramente, la fase di trasformazione del sistema eseguita mediante il metodo di eliminazione di Gauss richiede un certo numero di operazioni. Nell'esempio considerato le operazioni effettuate sono le seguenti:

- passo 1.

$2 \text{ flop}$  per il calcolo dei moltiplicatori;  
 $6 \text{ flop}$  per trasformare la II equazione;  
 $6 \text{ flop}$  per trasformare la III equazione.

- passo 2.

$1 \text{ flop}$  per il calcolo del moltiplicatore;  
 $4 \text{ flop}$  per trasformare la III equazione.

In totale quindi, il metodo di eliminazione di Gauss, applicato ad un sistema di ordine 3, richiede:

$$19 \text{ flop}.$$

Per ottenere la soluzione del sistema bisogna aggiungere, a tali operazioni, quelle richieste dal metodo di back-substitution che, in particolare, per l'esempio in esame, sono:

$$9 \text{ flop}.$$

Pertanto, per la risoluzione del sistema assegnato mediante il metodo di Gauss e di back-substitution sono effettuate in totale:

$$19 \text{ flop} + 9 \text{ flop} = 28 \text{ flop}.$$

La risoluzione dello stesso sistema con il metodo di Cramer richiederebbe  $20A + 39M$ , ciò evidenzia i vantaggi computazionali derivanti dall'applicazione del metodo di Gauss, già per sistemi di ordine  $n = 3$ . ♣

### 2.5.2 Descrizione generale del metodo di Gauss

Si consideri il caso generale della risoluzione del sistema:

$$Ax = b$$

con  $A$  matrice dei coefficienti del sistema:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \dots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \dots & a_{n-1,n} \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{pmatrix}$$

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \text{ vettore delle incognite; } b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \text{ vettore dei termini noti.}$$

Lo scopo del metodo di eliminazione di Gauss è quindi quello di trasformare il sistema di partenza, attraverso opportune combinazioni lineari delle equazioni, in un sistema triangolare superiore (oppure triangolare inferiore)<sup>7</sup>.

Il metodo opera sulla matrice  $A$  e sul vettore  $b$  e consiste nei passi seguenti:

---

<sup>7</sup>Si illustra qui il caso di sistema triangolare superiore perché è di uso prevalente nelle librerie di software matematico.

**passo 1.****scopo:**

annullamento di tutte le componenti della I colonna di  $A$ , tranne l'elemento  $a_{1,1}$ .

**procedimento:**

si sottrae dalla riga  $i$ -ma, per  $i = 2, 3, \dots, n$ , la I riga della matrice  $A$  moltiplicata per:

$$m_{i,1} = \frac{a_{i,1}}{a_{1,1}},$$

dove il numeratore costituisce il coefficiente dell'incognita da eliminare ed il denominatore l'elemento diagonale della colonna su cui si opera l'eliminazione. In tal modo si ottiene

$$\begin{aligned} A^{(1)} &= \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ 0 & a_{2,2} - \frac{a_{2,1}}{a_{1,1}}a_{1,2} & \cdots & a_{2,n} - \frac{a_{2,1}}{a_{1,1}}a_{1,n} \\ \vdots & \vdots & \cdots & \vdots \\ 0 & a_{n-1,2} - \frac{a_{n-1,1}}{a_{1,1}}a_{1,2} & \cdots & a_{n-1,n} - \frac{a_{n-1,1}}{a_{1,1}}a_{1,n} \\ 0 & a_{n,2} - \frac{a_{n,1}}{a_{1,1}}a_{1,2} & \cdots & a_{n,n} - \frac{a_{n,1}}{a_{1,1}}a_{1,n} \end{pmatrix} = \\ &= \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ 0 & a_{2,2}^{(1)} & \cdots & a_{2,n}^{(1)} \\ \vdots & \vdots & \cdots & \vdots \\ 0 & a_{n-1,2}^{(1)} & \cdots & a_{n-1,n}^{(1)} \\ 0 & a_{n,2}^{(1)} & \cdots & a_{n,n}^{(1)} \end{pmatrix} \end{aligned}$$

dove, relativamente agli elementi di  $A^{(1)}$ , l'indice in alto indica che essi hanno subito una prima trasformazione; in particolare

$$a_{i,j}^{(1)} = a_{i,j} - m_{i,1}a_{1,j}, \quad i, j = 2, 3, \dots, n.$$

Analoga operazione va effettuata sul vettore dei termini noti. Pertanto si ha:

$$\begin{aligned} b^{(1)} &= (b_1 \ b_2^{(1)} \ \dots \ b_{n-1}^{(1)} \ b_n^{(1)})^T = \\ &= (b_1, \ b_2 - \frac{a_{2,1}}{a_{1,1}}b_1, \ \dots, \ b_{n-1} - \frac{a_{n-1,1}}{a_{1,1}}b_1, \ b_n - \frac{a_{n,1}}{a_{1,1}}b_1)^T, \end{aligned}$$

ovvero:

$$b_i^{(1)} = b_i - m_{i,1}b_1, \quad i = 2, 3, \dots, n.$$

A partire dalla matrice così ottenuta si ripete il procedimento.

**passo 2.****scopo:**

annullamento di tutte le componenti della II colonna di  $A^{(1)}$  al di sotto dell'elemento  $a_{2,2}^{(1)}$ .

**procedimento:**

si sottrae dalla riga  $i$ -ma, per  $i = 3, 4, \dots, n$ , la II riga della matrice  $A^{(1)}$  moltiplicata per:

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

Fatto ciò si ottiene la matrice:

$$A^{(2)} = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \dots & a_{1,n} \\ 0 & a_{2,2}^{(1)} & a_{2,3}^{(1)} & \dots & a_{2,n}^{(1)} \\ 0 & 0 & a_{3,3}^{(2)} & \dots & a_{3,n}^{(2)} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & a_{n-1,3}^{(2)} & \dots & a_{n-1,n}^{(2)} \\ 0 & 0 & a_{n,3}^{(2)} & \dots & a_{n,n}^{(2)} \end{pmatrix}$$

dove:

$$a_{i,j}^{(2)} = a_{i,j}^{(1)} - m_{i,2} a_{2,j}^{(1)}, \quad i, j = 3, 4, \dots, n.$$

Operando allo stesso modo sul vettore  $b^{(1)}$  si ha:

$$b^{(2)} = (b_1 \ b_2^{(1)} \ b_3^{(2)} \ \dots \ b_{n-1}^{(2)} \ b_n^{(2)})^T,$$

dove:

$$b_i^{(2)} = b_i^{(1)} - m_{i,2} b_2^{(1)}, \quad i = 3, 4, \dots, n.$$

Procedendo analogamente, al generico passo  $k$  si ottiene la matrice:

$$A^{(k)} = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,k} & a_{1,k+1} & \dots & a_{1,n} \\ 0 & a_{2,2}^{(1)} & \dots & a_{2,k}^{(1)} & a_{2,k+1}^{(1)} & \dots & a_{2,n}^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & a_{k,k}^{(k-1)} & a_{k,k+1}^{(k-1)} & \dots & a_{k,n}^{(k-1)} \\ 0 & 0 & \dots & 0 & a_{k+1,k+1}^{(k)} & \dots & a_{k+1,n}^{(k)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 0 & a_{n-1,k+1}^{(k)} & \dots & a_{n-1,n}^{(k)} \\ 0 & 0 & \dots & 0 & a_{n,k+1}^{(k)} & \dots & a_{n,n}^{(k)} \end{pmatrix}$$

ed il vettore

$$b^{(k)} = (b_1 \ b_2^{(1)} \ \dots \ b_k^{(k-1)} \ b_{k+1}^{(k)} \ \dots \ b_{n-1}^{(k)} \ b_n^{(k)})^T$$

dove

$$\begin{aligned} a_{i,j}^{(k)} &= a_{i,j}^{(k-1)} - m_{i,k} a_{k,j}^{(k-1)}, \quad i, j = k+1, \dots, n \quad (a_{i,j}^{(0)} = a_{i,j}) \\ b_i^{(k)} &= b_i^{(k-1)} - m_{i,k} b_k^{(k-1)}, \quad i = k+1, \dots, n \end{aligned} \quad (2.9)$$

con:

$$m_{i,k} = \frac{a_{i,k}^{(k-1)}}{a_{k,k}^{(k-1)}}, \quad i = k+1, \dots, n. \quad (2.10)$$

Il procedimento si ripete fino ad ottenere la matrice

$$A^{(n-1)} = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \dots & a_{1,n} \\ 0 & a_{2,2}^{(1)} & a_{2,3}^{(1)} & \dots & a_{2,n}^{(1)} \\ 0 & 0 & a_{3,3}^{(2)} & \dots & a_{3,n}^{(2)} \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & \dots & 0 & a_{n,n}^{(n-1)} \end{pmatrix}.$$

e, di conseguenza, il vettore

$$b^{(n-1)} = (b_1 \ b_2^{(1)} \ b_3^{(2)} \ \dots \ b_n^{(n-1)})^T,$$

con  $A^{(n-1)}$  matrice triangolare superiore.

In Figura 2.11 è mostrata una rappresentazione della matrice dei coefficienti e del vettore dei termini noti al passo  $k$ .

Si osserva che gli elementi della diagonale principale di  $A^{(n-1)}$  sono, nell'ordine, gli  $a_{k,k}^{(k-1)}$  di tutti i passi di eliminazione. Pertanto il metodo di eliminazione di Gauss per un sistema di equazioni di ordine  $n$  termina dopo  $n-1$  passi. Il sistema triangolare superiore  $A^{(n-1)}x = b^{(n-1)}$  ottenuto all'ultimo passo è equivalente al sistema di partenza e può quindi essere risolto mediante l'algoritmo di back-substitution. Un primo raffinamento dell'algoritmo di eliminazione di Gauss è riportato nella Procedura 2.5.

```

procedure gauss (in: a, b, n ; out: a)
var integer: k, n
var real: a[n, n], b[n]
for k = 1 to n - 1
    passo di eliminazione
    in colonna k
end for
end procedure gauss

```

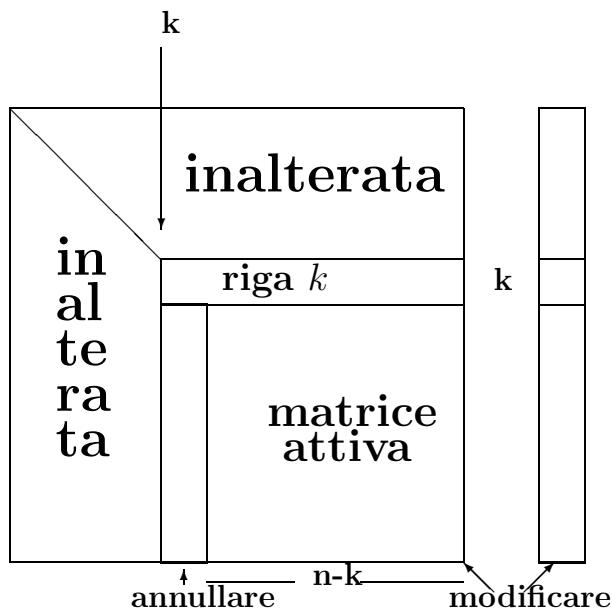


Figura 2.11: Algoritmo di eliminazione di Gauss: rappresentazione della matrice al generico passo  $k$

#### Procedura 2.5: Algoritmo di eliminazione di Gauss - I raffinamento

Per effettuare *il passo di eliminazione in colonna k* nella Procedura 2.5 è necessario calcolare, per ogni riga  $i$ , i moltiplicatori  $m_{i,k}$  (equazione (2.10)) e trasformare la riga  $i$ -ma della matrice  $A^{(k-1)}$  (equazione (2.9)). Quindi un raffinamento dell'algoritmo nella Procedura 2.5 è dato dall'algoritmo nella Procedura 2.6.

```

procedure gauss (in: a, b, n; out: a)
  var integer: i, k, n
  var real: a[n, n], b[n]
  for k = 1 to n - 1
    for i = k + 1 to n
      calcolo del moltiplicatore  $m_{i,k}$ 
      trasformazione  $i$ -ma riga di  $A^{(k-1)}$ 
      trasformazione  $i$ -mo elemento di  $b^{(k-1)}$ 
    end for
  end for
end procedure gauss

```

#### Procedura 2.6: Algoritmo di eliminazione di Gauss - II raffinamento

A questo punto si osservi che, per effettuare le operazioni al passo  $k$  (costruzione di  $A^{(k)}$ ), il metodo esposto richiede solo il risultato del passo precedente (la matrice  $A^{(k-1)}$ ). Discende da ciò che l'algoritmo può operare su una unica struttura dati che al passo  $k$  accoglierà solo  $A^{(k)}$ . Questa proprietà dell'algoritmo, importante dal punto di vista della complessità di spazio, viene sintetizzata dicendo che l'algoritmo è *in place*. In pratica, gli elementi modificati al passo  $k$ , ovvero gli  $a_{i,j}^{(k)}$  con  $i, j = k+1, \dots, n$  e  $k = 1, \dots, (n-1)$ , sono memorizzati al posto degli elementi corrispondenti  $a_{i,j}^{(k-1)}$ , in quanto, questi ultimi sono utilizzati solo per il calcolo degli  $a_{i,j}^{(k)}$ . Inoltre, tenendo presente che al passo  $k$  gli elementi della  $k$ -ma colonna al di sotto della diagonale vengono annullati, è possibile memorizzare su tali elementi i moltiplicatori  $m_{i,k}$ ,  $i = k+1, \dots, n$ , calcolati al passo  $k$ . La versione finale dell'algoritmo di eliminazione di Gauss è quindi riportata nella Procedura 2.7.

```

procedure gauss (in:  $a, b, n$ ; out:  $a, b$ )
  /# SCOPO: trasforma il sistema assegnato in uno
  triangolare superiore equivalente.

  /# SPECIFICHE DEI PARAMETRI:
  /# PARAMETRI DI INPUT:
    var:  $n$  : intero { dimensione del sistema }
    var:  $a(n, n)$  : reale { matrice del sistema }
    var:  $b(n)$  : reale { vettore dei termini noti }

  /# PARAMETRI DI OUTPUT:
    var:  $a(n, n)$  : reale { matrice del sistema }
                           { triangolare superiore }
    var:  $b(n)$  : reale { vettore dei termini noti }
                           { del sistema triangolare }
                           { superiore }

  /# VARIABILI LOCALI:
  var:  $i, j, k$  : interi

  /# INIZIO ISTRUZIONI:
  for  $k = 1$  to  $n - 1$  {ciclo sui passi}
    for  $i = k + 1$  to  $n$ 
       $a(i, k) := a(i, k)/a(k, k)$  {calcolo moltiplicatori}
      for  $j = k + 1$  to  $n$ 
         $a(i, j) := a(i, j) - a(i, k) * a(k, j)$  {trasf. mat. attiva}
      end for
       $b(i) := b(i) - a(i, k) * b(k)$  {trasf. termini noti}
    end for
  end for
end gauss

```

Procedura 2.7: Algoritmo di eliminazione di Gauss - versione finale

Si osservi infine che la modifica della  $i$ -ma riga è effettuata mediante un'operazione tra vettori del tipo *saxpy*

$$y = y + \alpha x$$

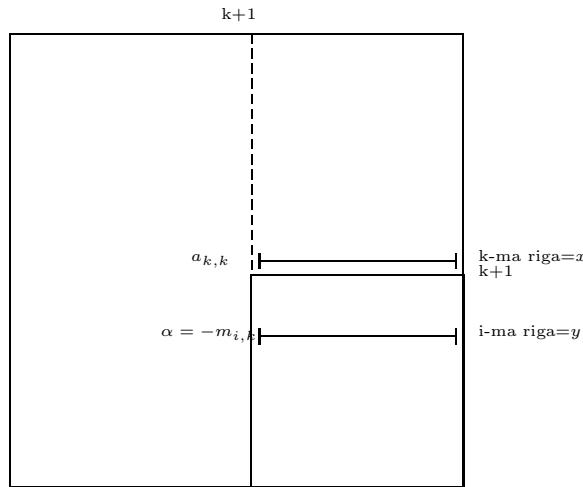


Figura 2.12: Operazione *saxpy* tra la  $k$ -ma e la  $i$ -ma riga di  $A^{(k)}$

Nel caso in esame il vettore  $y$  è costituito dagli ultimi  $(n - k)$  elementi della  $i$ -ma riga di  $A^{(k-1)}$ , il vettore  $x$  dagli ultimi  $(n - k)$  elementi della  $k$ -ma riga di  $A^{(k-1)}$  e  $\alpha$  è il moltiplicatore  $m_{i,k}$  cambiato di segno, come illustrato nella Figura 2.12.

### 2.5.3 Complessità di tempo e di spazio dell'algoritmo di eliminazione di Gauss

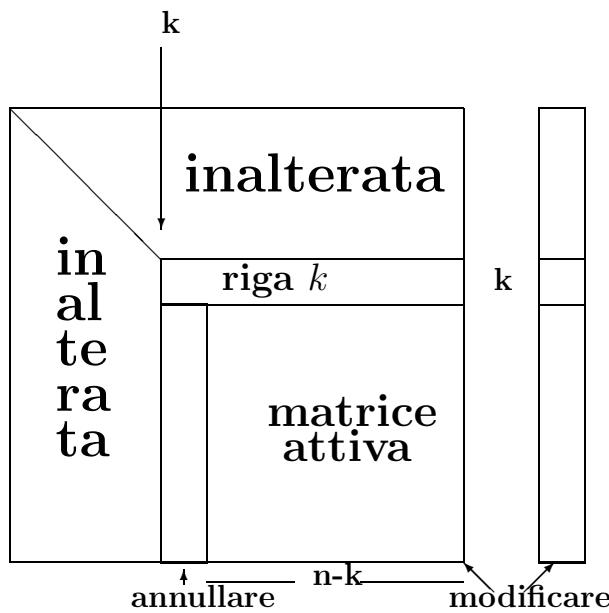
Il metodo di eliminazione di Gauss consente, come si è visto, di trasformare il sistema di partenza in un sistema triangolare superiore di più semplice risoluzione.

L'applicazione di tale metodo realizza quindi una fase di *preprocessing* del problema, intendendo con ciò che la sua applicazione ha lo scopo di trasformare il problema dato, mediante opportune manipolazioni, in uno o più problemi più semplici<sup>8</sup>. È necessario, quindi, in generale, valutare il costo computazionale della fase di preprocessing e il costo richiesto dal metodo (o dai metodi) successivamente applicati ai problemi più semplici ottenuti, in modo da verificare sia la validità dell'utilizzo del preprocessing, sia la convenienza dell'utilizzo di tale strategia considerando anche lo scopo per il quale è richiesta la soluzione del problema assegnato.

Per stimare la complessità di tempo dell'algoritmo di eliminazione di Gauss si cominci con l'osservare, ad esempio, che al passo 1. per annullare gli elementi della I colonna di  $A$ , escluso  $a_{1,1}$ , bisogna effettuare le operazioni seguenti:

$$m_{i,1} = a_{i,1}/a_{1,1}, \quad i = 2, \dots, n;$$

<sup>8</sup>Tale tecnica si è sempre più diffusa per la risoluzione di una gran parte di problemi dell'Analisi Numerica.

Figura 2.13: Schema della matrice  $A$  dopo  $k$  passi dell'algoritmo di Gauss.

$$a_{i,j} = a_{i,j} - m_{i,1}a_{1,j}, \quad i = 2, \dots, n, \quad j = 2, \dots, n;$$

$$b_i = b_i - m_{i,1}b_1, \quad i = 2, \dots, n.$$

Pertanto il passo 1. dell'algoritmo di Gauss richiede:

- $(n - 1)$  flop per valutare i moltiplicatori;
- $2n(n - 1)$  flop per modificare la sottomatrice ed il vettore dei termini noti.

I passi successivi dell'algoritmo eseguono le stesse operazioni ma su matrici (e vettori) che diventano via via di dimensioni sempre più piccole (vedi Figura 2.13).

In particolare (vedi la Procedura 2.7), al passo  $k$  è necessario calcolare:

1.  $m_{i,k} = a_{i,k}/a_{k,k}, \quad i = k + 1, \dots, n;$
2.  $a_{i,j} = a_{i,j} - m_{i,k}a_{k,j}, \quad i = k + 1, \dots, n, \quad j = k + 1, \dots, n;$
3.  $b_i = b_i - m_{i,k}b_k, \quad i = k + 1, \dots, n.$

Posto, quindi,:  

$$l = n - k$$

si ha che al passo  $k$  sono richieste:

- $l$  flop per il calcolo dei moltiplicatori;

- $l[2(l+1) \text{ flop}]$  per il calcolo degli  $a_{i,j}$  e dei  $b_i$ .

Pertanto la complessità di tempo dell'intero algoritmo di eliminazione di Gauss si ottiene sommando il numero di operazioni aritmetiche richieste a ciascun passo  $k$ , per  $k = 1, \dots, n-1$ , e quindi:

$$\sum_{k=1}^{n-1} [(l + 2l(l+1)) \text{ flop}] = \sum_{k=1}^{n-1} [(l + 2(l^2 + l)) \text{ flop}].$$

Essendo

$$\sum_{k=1}^{n-1} l \text{ flop} = \sum_{k=1}^{n-1} (n-k) \text{ flop} = \sum_{i=1}^{n-1} i \text{ flop} = \frac{n(n-1)}{2} \text{ flop} = \frac{n^2 - n}{2} \text{ flop},$$

e

$$\begin{aligned} \sum_{k=1}^{n-1} 2(l^2 + l) \text{ flop} &= \sum_{i=1}^{n-1} 2(i^2 + i) \text{ flop} = \\ &= 2 \left[ \frac{n(n-1)(2n-1)}{6} + \frac{n^2 - n}{2} \right] \text{ flop} = \frac{2(n^3 - n)}{3} \text{ flop} \end{aligned}$$

si ha che la complessità di tempo dell'algoritmo di eliminazione di Gauss, applicato ad un sistema di ordine  $n$  è:

$$T_{Gauss}(n) = \frac{2(n^3 - n)}{3} \text{ flop} + \frac{n^2 - n}{2} \text{ flop} = \mathcal{O}(n^3).$$

La risoluzione del sistema triangolare superiore che si ottiene al termine dell'algoritmo di eliminazione di Gauss richiede l'applicazione della back-substitution che ha complessità di tempo  $\mathcal{O}(n^2)$ . In definitiva, dunque, la risoluzione di un sistema di equazioni lineari di ordine  $n$  mediante l'algoritmo di eliminazione di Gauss e di back-substitution ha una complessità di tempo:

$$T_{Gauss+Back}(n) = \mathcal{O}(n^3).$$

Il confronto tra la complessità del metodo ora esposto e del metodo di Cramer mostra la sostanziale differenza esistente tra i due metodi in termini di numero di operazioni e, quindi di tempo di esecuzione richiesto.

Per quanto riguarda la stima dello spazio di memoria richiesto dall'algoritmo di eliminazione di Gauss, si è già osservato, nel paragrafo precedente, che gli elementi calcolati al passo  $k$  possono essere memorizzati sugli elementi di ugual posto ottenuti al passo  $k-1$ . Pertanto la complessità di spazio dell'algoritmo di eliminazione di Gauss, per una matrice di ordine  $n$  è:

$$S_{Gauss}(n) = n(n+1) = \mathcal{O}(n^2).$$

$n = 100$	#oper. = 681550	
	Tempo di esec.	Mflops ottenuti
Pers. Comp.	$3.5 \times 10^{-4}$ sec.	1911 (4400)
Workstation	$2.3 \times 10^{-4}$ sec.	2899 (5200)
Mainframe	$1.9 \times 10^{-4}$ sec.	3534 (4000)
Supercomp.	$6.3 \times 10^{-5}$ sec.	10780 (15238)
Pentium IV	$2.3 \times 10^{-4}$ sec.	$\leq 3000$

**Tabella 2.2: Tempi di esecuzione e Mflops ( $10^6$  oper. floating-point al sec.) ottenuti per la risoluzione di un sistema di equazioni lineari di ordine  $n = 100$  con il metodo di Gauss e di back-substitution su vari calcolatori. Il numero di operazioni effettuate è 681550.**

Infine, nella Tabella 2.2 sono riportati, per il calcolo della soluzione di un sistema lineare di ordine 100 mediante l'algoritmo di eliminazione di Gauss e l'algoritmo di *back-substitution* implementati nella libreria di software matematico *Linpack*<sup>9</sup> il tempo di esecuzione richiesto e la velocità operativa raggiunta sui cinque calcolatori già considerati nella Tabella 2.1 (per una valutazione dell'efficienza del metodo di Cramer). Si noti che la massima velocità operativa in realtà è puramente teorica (o indicativa delle potenziali prestazioni del calcolatore) e nella pratica solo in alcuni casi ci si avvicina ad essa. Si ricordi che il numero totale di operazioni richiesto dall'implementazione degli algoritmi suddetti è :

$$2 \cdot \frac{n^3}{3} + 3 \cdot \frac{n^2}{2} - 7 \cdot \frac{n}{6} \text{ flop}$$

I tempi riportati nella tabella precedente confermano che, a differenza del metodo di Cramer, il metodo di *eliminazione di Gauss* unitamente al metodo di *back-substitution* costituisce uno strumento effettivo, dal punto di vista computazionale, per la risoluzione di un sistema di equazioni lineari.

---

<sup>9</sup>Benché tale libreria si sia evoluta nella libreria LAPACK alla luce delle nuove architetture vettoriali e più in generale a memoria gerarchica, essa rimane comunque un punto di riferimento per misurare le prestazioni dei calcolatori, al punto che le routine della libreria Linpack sono ancora utilizzati per misurare le prestazioni dei calcolatori. Tali test sono noti come *Linpack benchmark* e sono diventati uno vero e proprio standard de facto. Un'ulteriore evoluzione ha condotto alla libreria ScaLAPACK (or Scalable LAPACK), che include un sottoinsieme di routines LAPACK ridisegnate per macchine parallele a memoria distribuita MIMD.

### 2.5.4 Strategia del pivoting nell'eliminazione di Gauss

Un problema da affrontare nell'algoritmo di eliminazione di Gauss riguarda la possibilità che ad un certo passo  $k$ , per  $k = 1, \dots, n-1$ , l'elemento  $a_{k,k}^{(k-1)}$  sia nullo. Se una tale eventualità si verifica non risulta possibile definire i moltiplicatori  $m_{i,k}$  in quanto ci si troverebbe ad effettuare una divisione per zero.

♣ **Esempio 2.11.** Si consideri il seguente sistema:

$$\left\{ \begin{array}{l} -2x_1 + 4x_2 - 2x_3 - 6x_4 = 4 \\ 3x_1 - 6x_2 + 6x_3 + 10x_4 = -1 \\ -2x_1 + 6x_2 - x_3 + x_4 = 1 \\ 2x_1 - 5x_2 + 4x_3 + 8x_4 = -3 \end{array} \right.$$

la cui soluzione è il vettore  $x = (1 \ 1 \ 2 \ -1)$ .

Applicando il passo 1. dell'algoritmo di eliminazione di Gauss al sistema in esame si ottiene:

- $m_{2,1} = -3/2$ ;  $m_{3,1} = 1$ ;  $m_{4,1} = -1$ , e

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

Si osservi che l'elemento  $a_{2,2}^{(1)}$  vale zero, pertanto non è possibile definire i moltiplicatori  $m_{3,2}$  e  $m_{4,2}$  che richiederebbero una divisione per un elemento nullo. Quindi anche se il sistema assegnato ammette soluzione (la matrice  $A$  è non singolare), essa non può essere calcolata applicando l'algoritmo di eliminazione di Gauss nella formulazione proposta nel precedente paragrafo. Come si può ovviare a tale inconveniente? La “naturale” strategia consiste nello scambiare la II riga di  $A^{(1)}$  con la III o con la IV riga in modo che l'elemento diagonale della II riga sia diverso da zero ed applicare il passo 2. dell'algoritmo di eliminazione di Gauss alla matrice così ottenuta; ciò è lecito in quanto scambiare le righe equivale a scrivere le equazioni in ordine differente e questo, ovviamente, non altera la soluzione del sistema. Infatti, scambiando la II e la III riga si ottiene:

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

Il nuovo elemento diagonale della seconda riga è ora diverso da zero, e quindi si può applicare il passo 2 dell'algoritmo di eliminazione di Gauss ottenendo:

- $m_{3,2} = 0$ ;  $m_{4,2} = -1/2$ , e

$$A^{(2)} = \begin{pmatrix} -2 & 4 & -2 & -6 \\ 0 & 2 & 1 & 7 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 5/2 & 11/2 \end{pmatrix}; \quad b^{(2)} = \begin{pmatrix} 4 \\ -3 \\ 5 \\ -1/2 \end{pmatrix}$$

Poiché  $a_{3,3}^{(2)} = 3 \neq 0$  si esegue l'ultimo passo dell'algoritmo di eliminazione di Gauss che conduce a:

- $m_{4,3} = 5/6$ ;

$$A^{(3)} = \begin{pmatrix} -2 & 4 & -2 & -6 \\ 0 & 2 & 1 & 7 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 14/3 \end{pmatrix}; \quad b^{(3)} = \begin{pmatrix} 4 \\ -3 \\ 5 \\ -14/3 \end{pmatrix}$$

Pertanto al termine del procedimento ora descritto si ottiene il sistema triangolare superiore equivalente a quello dato

$$\left\{ \begin{array}{rcl} -2x_1 + 4x_2 - 2x_3 - 6x_4 = 4 \\ 2x_2 + x_3 + 7x_4 = -3 \\ 3x_3 + x_4 = 5 \\ 14/3x_4 = -14/3 \end{array} \right.$$

La soluzione di tale sistema, e quindi del sistema di partenza, calcolata con l'algoritmo di *back-substitution* è  $x = (1 \ 1 \ 2 \ -1)$ . ♣

L'esempio considerato mostra che è sempre “*possibile*” e qualche volta “*necessario*” scambiare le equazioni del sistema quando si applica l'algoritmo di eliminazione di Gauss.

Una questione fondamentale da affrontare nella valutazione della effettiva applicabilità di un algoritmo, oltre alla sua efficienza computazionale, riguarda il suo comportamento rispetto alla propagazione degli errori di round-off derivanti dall'utilizzo di un sistema aritmetico floating-point a precisione finita. In altre parole, si pone il problema di determinare se l'algoritmo è *stabile* (o *instabile*), ovvero se le operazioni che lo costituiscono *non amplificano* (o *amplificano*) gli *inevitabili* errori di round-off della computazione.

♣ **Esempio 2.12.** Il sistema:

$$\left\{ \begin{array}{rcl} 0.0001x + y = 1 \\ x + y = 2, \end{array} \right. \quad (2.11)$$

ha la seguente soluzione (arrotondata a 5 cifre significative):

$$x = 1.0001; \quad y = .99990.$$

Utilizzando un sistema aritmetico floating-point caratterizzato dai parametri:

$$\beta = 10; \quad t = 3; \quad E_{min} = -6; \quad E_{max} = 6,$$

l'algoritmo di eliminazione di Gauss applicato al sistema dato fornisce il risultato

$$x = 0; \quad y = 1$$

che è evidentemente errato!

Si osservi che se si scambiano le equazioni del sistema, ottenendo

$$\left\{ \begin{array}{rcl} x + y = 2 \\ 0.0001x + y = 1, \end{array} \right.$$

e si applica l'algoritmo di eliminazione di Gauss, utilizzando il sistema aritmetico floating-point a precisione finita precedente, si ha il risultato:

$$x = 1 ; \quad y = 1,$$

che è una approssimazione accettabile della soluzione.

Per spiegare il verificarsi di tali situazioni si osservi che nel sistema aritmetico floating-point a precisione finita considerato, l'applicazione del passo 1. dell'algoritmo di eliminazione di Gauss al sistema (2.11) conduce a:

- $m_{2,1} = a_{2,1}/a_{1,1} = (.100 \times 10^1)/(.100 \times 10^{-3}) = .100 \times 10^5;$
- $a_{2,2}^{(1)} = a_{2,2} - m_{2,1}a_{1,2} = .100 \times 10^1 - (.100 \times 10^5) \times (.100 \times 10^1) = .0000\underline{1} \times 10^5$   
 $- .100 \times 10^5 \simeq -.100 \times 10^5;$
- $b_2^{(1)} = b_2 - m_{2,1}b_1 = .200 \times 10^1 - (.100 \times 10^5) \times (.100 \times 10^1) = .0000\underline{2} \times 10^5 - .100 \times 10^5 \simeq$   
 $-.100 \times 10^5.$

Quindi:

$$A^{(1)} = \begin{pmatrix} .100 \times 10^{-3} & .100 \times 10^1 \\ 0 & -.100 \times 10^5 \end{pmatrix}; \quad b^{(1)} = \begin{pmatrix} .100 \times 10^1 \\ -.100 \times 10^5 \end{pmatrix}.$$

Si noti che il sistema  $A^{(1)}x = b^{(1)}$  non è equivalente al sistema (2.11), infatti la sua soluzione è  $x = 0$ ,  $y = .100 \times 10^1$ . Se si scambiano le equazioni del sistema

$$\begin{cases} x + y = 2 \\ 0.0001 x + y = 1 \end{cases}$$

e si applica l'algoritmo di eliminazione di Gauss si ha:

- $m_{2,1} = a_{2,1}/a_{1,1} = (.100 \times 10^{-3})/(.100 \times 10^1) = .100 \times 10^{-3};$
- $a_{2,2}^{(1)} = a_{2,2} - m_{2,1}a_{1,2} = .100 \times 10^1 - (.100 \times 10^{-3}) \times (.100 \times 10^1) = .100 \times 10^1 - .0000\underline{1} \times$   
 $10^1 \simeq .100 \times 10^1;$
- $b_2^{(1)} = b_2 - m_{2,1}b_1 = .100 \times 10^1 - (.100 \times 10^{-3}) \times (.200 \times 10^1) = .100 \times 10^1 - .0000\underline{2} \times 10^1 \simeq .100 \times 10^1;$

quindi

$$A^{(1)} = \begin{pmatrix} .100 \times 10^1 & .100 \times 10^1 \\ 0 & .100 \times 10^1 \end{pmatrix}, \quad b^{(1)} = \begin{pmatrix} .200 \times 10^1 \\ .100 \times 10^1 \end{pmatrix}$$

e la soluzione di  $A^{(1)}x = b^{(1)}$  è  $x = 1$ ,  $y = 1$ , ed è un'approssimazione accettabile della soluzione del sistema (2.11).



Si noti anche che nella matrice  $A^{(1)}$ , prodotta dall'applicazione dell'algoritmo di eliminazione di Gauss al sistema dato, gli elementi sono di ordini di grandezza diversi tra loro, mentre nella matrice  $A^{(1)}$ , generata dall'algoritmo di eliminazione di Gauss applicato al sistema dopo lo scambio delle equazioni, essi sono omogenei tra loro, evitando così moltiplicatori molto grandi in modulo. Inoltre, l'elemento di massimo modulo di quest'ultima matrice è molto più piccolo di quello della prima matrice  $A^{(1)}$  considerata. Da quanto detto, quindi, si può concludere che l'algoritmo di eliminazione di Gauss è

*instabile* a causa della possibilità di avere divisori “*molto piccoli*”.

L'esempio precedente mostra che l'algoritmo di eliminazione di Gauss “*fallisce*” anche quando un elemento diagonale è “*molto piccolo*” (nel sistema floating-point considerato) rispetto agli altri elementi della matrice; infatti l'algoritmo in presenza di elementi diagonali “*piccoli*” amplifica gli errori di round-off. Tale affermazione può essere derivata, in maniera intuitiva<sup>10</sup>, dall'osservazione che nell'espressione per il calcolo degli  $a_{i,j}^{(k)}$ ,

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

viene effettuata una moltiplicazione tra il moltiplicatore ed un elemento della matrice ottenuta al passo precedente su cui sono state già effettuate  $k-1$  trasformazioni. Più è grande il moltiplicatore, ovvero più è piccolo l'elemento diagonale che lo ha generato, maggiore è l'amplificazione dell'errore di round-off presente nell'elemento  $a_{k,j}^{(k-1)}$ . Ciò si può constatare dall'esempio considerato, non scambiando le equazioni, in cui il moltiplicatore  $m_{2,1}$  è uguale a  $.100 \times 10^5$ , in virtù della divisione per  $a_{1,1} = .100 \times 10^{-3}$ , viene moltiplicato per  $a_{1,2} = 1$  e per la sua grandezza provoca un fenomeno di perdita di cifre nella sottrazione con  $a_{2,2}$ .

L'esempio ora considerato suggerisce anche una strategia per controllare l'amplificazione degli errori di round-off nell'algoritmo di Gauss.

L'obiettivo è avere ad ogni passo moltiplicatori  $m_{i,k}$  che siano in valore assoluto minori di uno, in maniera tale che nella moltiplicazione di ciascun moltiplicatore per il relativo  $a_{k,j}^{(k-1)}$  non si abbia amplificazione dell'errore di round-off presente in questo elemento. L'idea di base è quindi riorganizzare, ad ogni passo  $k$ , la matrice  $A^{(k)}$  in modo da avere un elemento diagonale “*non nullo*”<sup>11</sup> e tale che i moltiplicatori  $m_{i,k}$  che si calcolano mediante esso siano tutti, in valore assoluto, minori o uguali ad uno.

La strategia generale ora descritta viene detta *pivoting*. Il problema, a questo punto, è stabilire come selezionare al passo  $k$  un elemento che soddisfi i requisiti suddetti. Tale elemento è detto *pivot*.

♣ **Esempio 2.13.** Si consideri il seguente sistema:

$$\begin{cases} 0.001x_1 - 7x_2 &= 7 \\ -3x_1 + 2.099x_2 + 6x_3 &= 3.901 \\ 5x_1 - x_2 + 5x_3 &= 6 \end{cases}$$

la cui soluzione esatta è:

$$x_1 = 0; \quad x_2 = -1; \quad x_3 = 1.$$

Se si risolve il sistema dato mediante l'algoritmo di eliminazione di Gauss e di back-substitution utilizzando un sistema aritmetico floating-point caratterizzato dai seguenti parametri:

$$\beta = 10, \quad t = 5, \quad E_{min} = -6, \quad E_{max} = 6,$$

---

<sup>10</sup>Una trattazione rigorosa della stabilità dell'algoritmo di eliminazione di Gauss verrà data nei paragrafi successivi.

<sup>11</sup>Se  $A$  è non singolare esiste sempre un  $a_{i,k}^{(k-1)}$ , per  $i = k, \dots, n$ , non nullo.

si ha la soluzione calcolata:

$$x_1 = 0.1, \quad x_2 = -0.99986, \quad x_3 = 0.40008,$$

evidentemente inaccettabile. È necessario, quindi, applicare la strategia di *pivoting*.

Per stabilire quale elemento selezionare al primo passo, ovvero qual è il pivot al primo passo, si ricordi che l'espressione dei moltiplicatori è la seguente:

$$m_{i,1} = a_{i,1}/\text{pivot}, \quad \text{per } i = 2, 3$$

Poiché lo scopo della strategia di *pivoting* è riorganizzare le equazioni in modo che  $|m_{i,1}| \leq 1$  per  $i = 2, 3$ , il pivot va selezionato in maniera tale che i numeratori delle frazioni siano tutti in valore assoluto minori o uguali al valore assoluto del pivot. Ciò si può ottenere scegliendo come elemento pivot il più grande in valore assoluto tra gli elementi della prima colonna. Quindi, nel caso in esame, l'elemento pivot così selezionato è dato da  $a_{3,1} = 5$ . Poiché tale elemento si trova nella III equazione del sistema bisogna scambiare la I e la III equazione, ottenendo:

$$\left\{ \begin{array}{rcl} 5x_1 & - & x_2 + 5x_3 = 6 \\ -3x_1 & + & 2.099x_2 + 6x_3 = 3.901 \\ 0.001x_1 & - & 7x_2 = 7 \end{array} \right.$$

e a tale sistema si applica il passo 1. dell'algoritmo di Gauss.

Quindi:

- $m_{2,1} = -0.6$ ;  $m_{3,1} = 0.0002$ , e

$$A^{(1)} = \begin{pmatrix} 5 & -1 & 5 \\ 0 & 1.499 & 9 \\ 0 & -6.9998 & -0.001 \end{pmatrix}; \quad b^{(1)} = \begin{pmatrix} 6 \\ 7.501 \\ 6.9988 \end{pmatrix}.$$

Prima di applicare il passo 2. dell'algoritmo bisogna selezionare il pivot sempre con l'obiettivo che il moltiplicatore

$$m_{3,2} = a_{3,2}^{(1)}/\text{pivot}$$

sia in valore assoluto minore di uno. Analogamente a prima ciò si verifica scegliendo come pivot il più grande in valore assoluto tra l'elemento  $a_{2,2}^{(1)} = 1.499$  e  $a_{3,2}^{(1)} = -6.9998$ . La scelta, quindi, cade sull'elemento  $a_{3,2}^{(1)}$  ed è richiesto di conseguenza uno scambio tra la II e III riga di  $A^{(1)}$ , ottenendo

$$\begin{pmatrix} 5 & -1 & 5 \\ 0 & -6.9998 & -0.001 \\ 0 & 1.499 & 9 \end{pmatrix}; \quad \begin{pmatrix} 6 \\ 6.9988 \\ 7.501 \end{pmatrix}.$$

Effettuando il passo 2. dell'algoritmo di Gauss si ha:

- $m_{3,2} = 1.499/(-6.9998) = -0.21415$  e

$$A^{(2)} = \begin{pmatrix} 5 & -1 & 5 \\ 0 & -6.9998 & -0.001 \\ 0 & 0 & 8.9998 \end{pmatrix}, \quad b^{(2)} = \begin{pmatrix} 6 \\ 6.9988 \\ 8.9998 \end{pmatrix}$$

Applicando la back-substitution al sistema così ottenuto, si ha:

$$x_1 = 0; \quad x_2 = -1; \quad x_3 = 1.$$



La strategia di *pivoting* seleziona, come elemento pivot al passo  $k$ , l'elemento di massimo modulo tra quelli della  $k$ -ma colonna a partire dall'elemento diagonale. Tale strategia di *pivoting*, detta **pivoting parziale** si può descrivere nel modo seguente: (vedi anche Figura 2.14):

- seleziona  $r$ , il più piccolo intero tale che

$$|a_{r,k}^{(k-1)}| = \max_{k \leq i \leq n} |a_{i,k}^{(k-1)}|,$$

- scambia la riga  $k$  con la riga  $r$ .

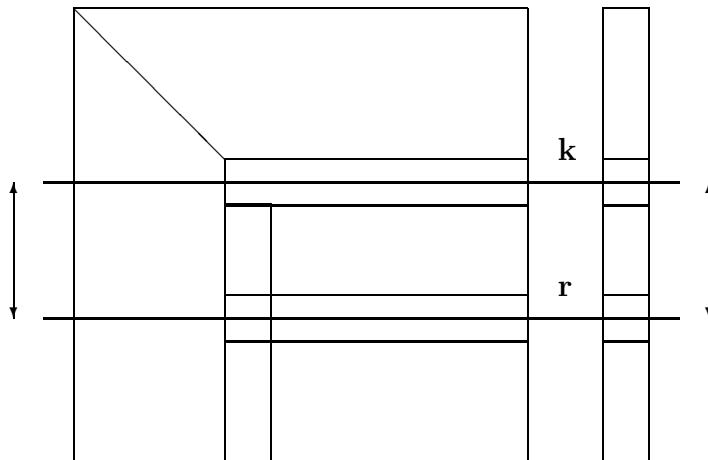


Figura 2.14: Il pivoting parziale nell'algoritmo di eliminazione di Gauss

In tal modo nel calcolo di  $m_{i,k}$  si divide per il massimo elemento in valore assoluto della  $k$  colonna e quindi risulta:

$$|m_{i,k}| \leq 1,$$

raggiungendo così l'obiettivo posto ( riduzione dell'amplificazione dell'errore di round-off ). Una procedura per effettuare sulla matrice  $A$  il pivoting al passo  $k$  dell'algoritmo di eliminazione di Gauss è data nella Procedura 2.8, mentre l'algoritmo di eliminazione di Gauss con pivoting parziale è dato nella Procedura 2.9.

```

procedure pivoting (in:  $a, b, n, k$ ; out:  $a, b$ )
  /* SCOPO: Effettua il pivoting parziale lungo le righe
  /* del sistema assegnato.

  /* SPECIFICHE DEI PARAMETRI:
  /* PARAMETRI DI INPUT:
var:  $n$  : intero { dimensione del sistema }
var:  $k$  : intero { indice della prima riga }
  { da scambiare }
var:  $a(n, n)$  : reale { matrice del sistema }
var:  $b(n)$  : reale { vettore dei termini noti }

  /* PARAMETRI DI OUTPUT:
var:  $b(n)$  : reale { vettore dei termini noti }
  { modificato }
var:  $a(n, n)$  : reale { matrice del sistema }
  { modificata }

```

Procedura 2.8: Procedura di pivoting per l'algoritmo di eliminazione di Gauss - continua

```

 $\text{/}\# VARIABILI LOCALI:$ 
var:  $i, j, r$  : interi
var:  $t$  : reale

 $\text{/}\# INIZIO ISTRUZIONI:$ 
 $mas := abs(a(k, k))$ 
 $r := k$ 
for  $i = k + 1$  to  $n$  {ricerca del massimo}
    if ( $abs(a(i, k)) > mas$ ) then
         $mas := abs(a(i, k))$ 
         $r := i$ 
    end if
endfor
if ( $r \neq k$ ) then
    for  $j = k$  to  $n$  {scambio delle righe}
         $t := a(r, j)$ 
         $a(r, j) := a(k, j)$ 
         $a(k, j) := t$ 
    endfor
     $t := b(r)$  {scambio dei termini noti}
     $b(r) := b(k)$ 
     $b(k) := t$ 
endif
end pivoting

```

Procedura 2.8: Procedura di pivoting per l'algoritmo di eliminazione di Gauss - fine

```

procedure gausspiv (in:  $a, b, n$ ; out:  $a, b$ )
  /# SCOPO: applica il metodo di eliminazione di Gauss.
  /# SPECIFICHE DEI PARAMETRI:
  /# PARAMETRI DI INPUT:
    var:  $n$  : intero { dimensione del sistema }
    var:  $a(n, n)$  : reale { matrice del sistema }
    var:  $b(n)$  : reale { vettore dei termini noti }

  /# PARAMETRI DI OUTPUT:
    var:  $b(n)$  : reale { vettore dei termini noti }
                           { modificato }
    var:  $a(n, n)$  : reale { matrice del sistema }
                           { modificata }

  /# VARIABILI LOCALI:
  var:  $i, j, k$  : interi

  /# INIZIO ISTRUZIONI:
  for  $k = 1$  to  $n - 1$  { ciclo sui passi }
    pivoting(in:  $a, b, n, k$ ; out:  $a, b$ ) { pivoting }
    for  $i = k + 1$  to  $n$ 
       $a(i, k) := a(i, k) / a(k, k)$  { calcolo moltiplicatori }
      for  $j = k + 1$  to  $n$ 
         $a(i, j) := a(i, j) - a(i, k) * a(k, j)$  { trasf. mat. attiva }
      end for
       $b(i) := b(i) - a(i, k) * b(k)$  { trasf. termini noti }
    end for
  end for
end gausspiv

```

Procedura 2.9: Algoritmo di eliminazione di Gauss con pivoting parziale

Si osservi che, al passo  $k$ , se nella fase di *pivoting* parziale risulta

$$\max_{k \leq i \leq n} |a_{i,k}^{(k-1)}| = 0,$$

gli elementi della  $k$ -ma colonna a partire dall'elemento diagonale sono già nulli, cioè:

$$A^{(k-1)} = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,k-1} & a_{1,k} & a_{1,k+1} & \dots & \dots & a_{1,n} \\ 0 & a_{2,2}^{(1)} & \dots & a_{2,k-1}^{(1)} & a_{2,k}^{(1)} & a_{2,k+1}^{(1)} & \dots & \dots & a_{2,n}^{(1)} \\ \vdots & \vdots \\ 0 & 0 & \dots & a_{k-1,k-1}^{(k-2)} & a_{k-1,k}^{(k-2)} & a_{k-1,k+1}^{(k-2)} & \dots & \dots & a_{k-1,n}^{(k-2)} \\ 0 & 0 & \dots & 0 & a_{k,k}^{(k-1)} & a_{k,k+1}^{(k-1)} & \dots & \dots & a_{k,n}^{(k-1)} \\ 0 & 0 & \dots & 0 & a_{k+1,k}^{(k-1)} & a_{k+1,k+1}^{(k-1)} & \dots & \dots & a_{k+1,n}^{(k-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & a_{n-1,k}^{(k-1)} & a_{n-1,k+1}^{(k-1)} & \dots & \ddots & a_{n-1,n}^{(k-1)} \\ 0 & 0 & \dots & 0 & a_{n,k}^{(k-1)} & a_{n,k+1}^{(k-1)} & \dots & \dots & a_{n,n}^{(k-1)} \end{pmatrix}.$$

Pertanto, l'algoritmo può proseguire con l'annullamento degli elementi della  $(k+1)$ -ma colonna al di sotto dell'elemento diagonale. In tal caso al termine dell'algoritmo si otterrà una matrice triangolare superiore  $U$  dove  $a_{k,k}^{(k-1)} = 0$ , e poiché, in questo caso,

$$\det(U) = a_{1,1}^{(0)} \cdot a_{2,2}^{(1)} \cdots a_{n,n}^{(n-1)} = 0$$

la matrice è singolare e il sistema è incompatibile o indeterminato. Al sistema triangolare superiore ottenuto con l'algoritmo di eliminazione di Gauss con pivoting parziale è infine possibile applicare l'algoritmo descritto nella Procedura 2.3 di `back substitution` con controllo della singolarità.

Riesaminando i passi del metodo di eliminazione di Gauss con pivoting, se, al passo  $k$ , si seleziona, come pivot, l'elemento di massimo modulo *in tutta la matrice attiva*, piuttosto che *nella k-esima colonna*, si realizza la tecnica denominata **pivoting totale**. Tale strategia si può descrivere, più in dettaglio, come segue: (vedi anche Figura 2.15):

- si scelgono  $r$  ed  $s$  come gli interi più piccoli per cui

$$|a_{r,s}^{(k-1)}| = \max_{k \leq i, j \leq n} |a_{i,j}^{(k-1)}|,$$

- si scambiano le righe  $k$  e  $r$  e le colonne  $k$  e  $s$ .

♣ **Esempio 2.14.** Si consideri il sistema

$$\begin{cases} 0.00141x_1 + 0.04004x_2 = 0.01142 \\ 0.20000x_1 + 4.91200x_2 = 1.428 \end{cases}$$

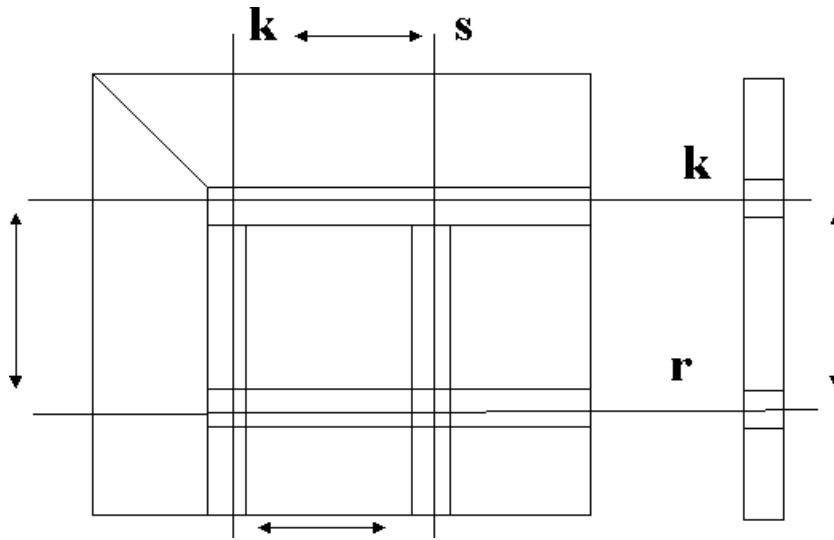


Figura 2.15: Il pivoting totale nell'algoritmo di eliminazione di Gauss

la cui soluzione è il vettore:

$$s = (1, 0.25)^T$$

In un sistema aritmetico floating point a precisione finita:

$$\beta = 10, \quad t = 4, \quad t_{reg} = 8$$

si applichi l'algoritmo di eliminazione di Gauss con pivoting *parziale*. Individuato 0.2 come elemento massimo in modulo, nella prima colonna, si scambiano la prima e la seconda riga:

$$\begin{pmatrix} 0.2000 \times 10^0 & 0.4912 \times 10^1 \\ 0.1410 \times 10^{-2} & 0.4004 \times 10^{-1} \end{pmatrix}, \quad \begin{pmatrix} 0.1428 \times 10^1 \\ 0.1142 \times 10^{-1} \end{pmatrix}.$$

I passi dell'algoritmo di eliminazione di Gauss risultano:

$$\begin{aligned} m_{2,1} &= \frac{a_{2,1}}{a_{1,1}} = \frac{0.1410 \times 10^{-2}}{0.2000 \times 10^0} = \\ &= 0.00705 = 0.7050 \times 10^{-2} \end{aligned}$$

$$\begin{aligned} a_{2,2}^{(1)} &= a_{2,2} - m_{2,1}a_{1,2} = \\ &= 0.4004 \times 10^{-1} - (0.7050 \times 10^{-2})(0.4912 \times 10^1) = \\ &= 0.4004 \times 10^{-1} - 0.3463 \times 10^{-1} = 0.5410 \times 10^{-2} \end{aligned}$$

$$\begin{aligned} b_2^{(1)} &= b_2 - m_{2,1}b_1 = \\ &= 0.1142 \times 10^{-1} - (0.7050 \times 10^{-2})(0.1428 \times 10^1) = \\ &= 0.1142 \times 10^{-1} - 0.1007 \times 10^{-1} = 0.1350 \times 10^{-2} \end{aligned}$$

Quindi la matrice dei coefficienti ed il vettore dei termini noti diventano, rispettivamente:

$$A^{(1)} = \begin{pmatrix} 0.2000 \times 10^0 & 0.4912 \times 10^1 \\ 0 & 0.5410 \times 10^{-2} \end{pmatrix}, \quad b^{(1)} = \begin{pmatrix} 0.1428 \times 10^1 \\ 0.1350 \times 10^{-2} \end{pmatrix}.$$

Applicando la back-substitution si ha:

$$\begin{aligned}\tilde{x}_2 &= \frac{b_2^{(1)}}{a_{2,2}^{(1)}} = \frac{0.1350 \times 10^{-2}}{0.5410 \times 10^{-2}} = 0.2495 \times 10^0 = \boxed{0.2495} \\ \tilde{x}_1 &= \frac{b_1^{(1)} - a_{1,2}^{(1)}\tilde{x}_2}{a_{1,1}^{(1)}} = \\ &= \frac{0.1428 \times 10^1 - (0.4912 \times 10^1)(0.2495 \times 10^0)}{0.2000 \times 10^0} = \\ &= \frac{0.1428 \times 10^1 - 0.1226 \times 10^1}{0.2000 \times 10^0} = \frac{0.0202 \times 10^1}{0.2000 \times 10^0} = \\ &= \frac{0.2020 \times 10^0}{0.2000 \times 10^0} = 0.1010 \times 10^1 = \boxed{1.010}\end{aligned}$$

Quindi:

$$\tilde{s} = (\tilde{x}_1, \tilde{x}_2)^T = (1.010, 0.2495)^T \quad \text{mentre} \quad s = (1, 0.25)^T$$

da cui

$$E' = \frac{\|s - \tilde{s}\|_\infty}{\|s\|_\infty} = 0.01 = 0.1 \times 10^{-1}$$

ovvero l'errore relativo risulta dell' 1% !!! Applicando allo stesso esempio il metodo di eliminazione di Gauss con *pivoting totale* si individua, come elemento massimo in modulo,  $0.4912 \times 10^1$  e si effettua, di conseguenza, lo scambio tra la prima e la seconda riga e tra la prima e la seconda colonna, ottenendo, dunque:

$$\left( \begin{array}{cc} 0.4912 \times 10^1 & 0.2000 \times 10^0 \\ 0.4004 \times 10^{-1} & 0.1410 \times 10^{-2} \end{array} \right), \quad \left( \begin{array}{c} 0.1428 \times 10^1 \\ 0.1142 \times 10^{-1} \end{array} \right).$$

Applicando l'algoritmo di eliminazione di Gauss si calcolano:

$$\begin{aligned}m_{2,1} &= \frac{a_{2,1}}{a_{1,1}} = \frac{0.4004 \times 10^{-1}}{0.4912 \times 10^1} = 0.8151 \times 10^{-2} \\ a_{2,2}^{(1)} &= a_{2,2} - m_{2,1}a_{1,2} = \\ &= 0.1410 \times 10^{-2} - (0.8151 \times 10^{-2})(0.2000 \times 10^0) = \\ &= 0.1410 \times 10^{-2} - 0.1630 \times 10^{-2} = -0.2200 \times 10^{-3} \\ b_2^{(1)} &= b_2 - m_{2,1}b_1 = \\ &= 0.1142 \times 10^{-1} - (0.8151 \times 10^{-2})(0.1428 \times 10^1) = \\ &= 0.1142 \times 10^{-1} - 0.1164 \times 10^{-1} = -0.2200 \times 10^{-3}\end{aligned}$$

Quindi la matrice dei coefficienti ed il vettore dei termini noti diventano, rispettivamente:

$$A^{(1)} = \left( \begin{array}{cc} 0.4912 \times 10^1 & 0.2000 \times 10^0 \\ 0 & -0.2200 \times 10^{-3} \end{array} \right), \quad b^{(1)} = \left( \begin{array}{c} 0.1428 \times 10^1 \\ -0.2200 \times 10^{-3} \end{array} \right).$$

Applicando la back-substitution si ha:

$$\begin{aligned}\tilde{x}_2 &= \frac{b_2^{(1)}}{a_{2,2}^{(1)}} = 0.1000 \times 10^1 = \boxed{1.000} \\ \tilde{x}_1 &= \frac{b_1^{(1)} - a_{1,2}^{(1)}\tilde{x}_2}{a_{1,1}^{(1)}} = \\ &= \frac{0.1428 \times 10^1 - (0.2000 \times 10^0)(0.1000 \times 10^1)}{0.4912 \times 10^1} = \\ &= \frac{0.1428 \times 10^1 - 0.2000 \times 10^0}{0.4912 \times 10^1} = 0.2500 \times 10^0 = \\ &= \boxed{0.25}\end{aligned}$$

Quindi:

$$\tilde{s} = (\tilde{x}_1, \tilde{x}_2)^T = (1, 0.25)^T \quad \text{e} \quad s = (1, 0.25)^T$$

ovvero la soluzione può ritenersi accurata. ♣

Si sottolinea, a proposito della tecnica del pivoting totale, che il

$$(k-1)\text{-esimo pivot: } \max_{k \leq i,j \leq n} |a_{i,j}^{(k-1)}|$$

viene determinato come l'elemento di massimo modulo *tra tutti gli elementi della matrice attiva*, per cui, dividendo opportunamente per esso, si ottiene il moltiplicatore  $|m_{i,k}|$  che risulta minore o uguale del moltiplicatore ottenuto col pivoting parziale; si può dedurre, allora, che **rispetto al pivoting parziale, il pivoting totale garantisce una maggiore riduzione dell'errore di round off**. In effetti, con il pivoting parziale, l'errore relativo di round off nella soluzione è, in generale, al più il doppio rispetto a quello risultante dal pivoting totale. Per quanto riguarda il *costo del pivoting totale*, si effettuano

$(n-k+1)^2$  confronti ad ogni, generico, passo  $k$  ( $k = 1, \dots, n-1$ ),

per un totale di

$$\mathcal{O}\left(\frac{n^3}{3}\right) \quad \text{confronti}$$

e questo è il costo, in termini di numero di confronti, che il pivoting totale aggiunge alla complessità computazionale dell'algoritmo di eliminazione di Gauss. Il *costo del pivoting parziale*, invece, richiede

$(n-k+1)$  confronti ad ogni, generico, passo  $k$  ( $k = 1, \dots, n-1$ ),

per un totale di

$$\mathcal{O}\left(\frac{n^2}{2}\right) \quad \text{confronti}$$

e questo è il costo, in termini di numero di confronti, che il pivoting parziale aggiunge alla complessità computazionale dell'algoritmo di eliminazione di Gauss. Sostanzialmente si può concludere che il pivoting parziale risulta caratterizzato da:

- errore relativo di round off sulla soluzione accettabile;
- minor numero di confronti del pivoting totale

ed è per queste ragioni che conviene, in generale, utilizzare la strategia del pivoting parziale, piuttosto che quella del pivoting totale.

## 2.6 Fattorizzazione LU

♣ **Esempio 2.15.** Si consideri il sistema

$$\begin{cases} 2x_1 + 4x_2 - 2x_3 = 6 \\ x_1 - x_2 + 5x_3 = 0 \\ 4x_1 + x_2 - 2x_3 = -2 \end{cases}$$

o, equivalentemente, in forma compatta:

$$Ax = b$$

con

$$A = \begin{pmatrix} 2 & 4 & -2 \\ 1 & -1 & 5 \\ 4 & 1 & -2 \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}, \quad b = \begin{pmatrix} 6 \\ 0 \\ -2 \end{pmatrix}$$

L'algoritmo di eliminazione di Gauss senza pivoting trasforma il sistema in uno triangolare superiore:

$$Ux = c, \quad (A^{(n-1)}x = b^{(n-1)}),$$

con

$$U = \begin{pmatrix} 2 & 4 & -2 \\ 0 & -3 & 6 \\ 0 & 0 & -12 \end{pmatrix}, \quad c = \begin{pmatrix} 6 \\ -3 \\ -7 \end{pmatrix}$$

I moltiplicatori sono:

- al passo 1:  $m_{2,1} = 1/2$ ,  $m_{3,1} = 2$ ;
- al passo 2:  $m_{3,2} = 7/3$ .

Si consideri, ora, la matrice:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ m_{2,1} & 1 & 0 \\ m_{3,1} & m_{3,2} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 2 & 7/3 & 1 \end{pmatrix};$$

$L$  è una matrice triangolare inferiore con:

- elementi diagonali uguali ad 1;
- elementi della  $k$ -ma colonna al di sotto dell'elemento diagonale uguali ai moltiplicatori calcolati al passo  $k$  ( $k = 1, \dots, n-1$ );

Se si calcola il prodotto  $LU$  si ha:

$$\begin{aligned} LU &= \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 2 & 7/3 & 1 \end{pmatrix} \begin{pmatrix} 2 & 4 & -2 \\ 0 & -3 & 6 \\ 0 & 0 & 12 \end{pmatrix} = \\ &= \begin{pmatrix} 2 & 4 & -2 \\ 1 & -1 & 5 \\ 4 & 1 & -2 \end{pmatrix} = A \end{aligned}$$

ovvero:

$$LU = A$$



Il risultato dell'esempio precedente sussiste in generale, ovvero l'algoritmo di eliminazione di Gauss può essere visto come un *metodo di fattorizzazione* della matrice  $A$  nel prodotto di due matrici:

$$A = LU$$

con:

- $L$  triangolare inferiore,
- $U$  triangolare superiore.

In particolare, per un sistema di dimensione  $n$ , abbiamo:

$$L = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ m_{2,1} & 1 & 0 & \dots & 0 \\ m_{3,1} & m_{3,2} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{n,1} & m_{n,2} & m_{n,3} & \dots & 1 \end{pmatrix}$$

cioé  $L$  ha:

- elementi diagonali uguali ad 1;
- elementi della I colonna al di sotto dell'elemento diagonale uguali ai moltiplicatori calcolati al passo 1;
- elementi della II colonna al di sotto dell'elemento diagonale uguali ai moltiplicatori calcolati al passo 2, e così via.

$$U = A^{(n-1)} = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \dots & a_{1,n} \\ 0 & a_{2,2}^{(1)} & a_{2,3}^{(1)} & \dots & a_{2,n}^{(1)} \\ 0 & 0 & a_{3,3}^{(2)} & \dots & a_{3,n}^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & a_{n,n}^{(n-1)} \end{pmatrix}$$

cioè  $U$  è la matrice ottenuta all'ultimo passo dell'algoritmo di eliminazione di Gauss.

♣ **Esempio 2.16.** Si riprenda l'esempio precedente e si osservi che, definita la matrice:

$$M_1 = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ -2 & 0 & 1 \end{pmatrix}$$

costruita a partire dai moltiplicatori relativi al primo passo dell'algoritmo di eliminazione di Gauss, si può osservare che:

$$M_1 A = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ -2 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 4 & -2 \\ 1 & -1 & 5 \\ 4 & 1 & -2 \end{pmatrix} = \begin{pmatrix} 2 & 4 & -2 \\ 0 & -3 & 6 \\ 0 & -7 & 2 \end{pmatrix} = A^{(1)}$$

Cioè la matrice che si ottiene da  $A$  dopo un passo dell'algoritmo di eliminazione di Gauss può essere ottenuta moltiplicando a sinistra tale matrice per una opportuna matrice  $M_1$ . Analogamente al secondo passo, definendo la matrice:

$$M_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{7}{3} & 1 \end{pmatrix}$$

a partire dall'unico moltiplicatore  $m_{3,2}$ , è possibile ottenere:

$$M_2 M_1 A = M_2 A^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{7}{3} & 1 \end{pmatrix} \begin{pmatrix} 2 & 4 & -2 \\ 0 & -3 & 6 \\ 0 & -7 & 2 \end{pmatrix} = \begin{pmatrix} 2 & 4 & -2 \\ 0 & -3 & 6 \\ 0 & 0 & -12 \end{pmatrix} = A^{(2)}$$

da cui si ha che  $L = M_1^{-1} M_2^{-1}$  e  $U = A^{(2)}$ ,cioè la matrice che si ottiene all'ultimo passo dell'algoritmo di eliminazione di Gauss.



In generale sussiste la seguente:

**Definizione 2.7. (Matrice triangolare unitaria elementare inferiore)**  
*Dato un vettore  $m$  del tipo:*

$$m^T = (0, 0, \dots, m_{k+1}, \dots, m_n)$$

la matrice quadrata  $M_k$  di ordine  $n$  definita come:

$$M_k = I - m e_k^T \quad \text{dove} \quad e_k^T = (\underbrace{0, 0, \dots, 0}_{k-1}, 1, 0, \dots, 0)$$

è detta **triangolare unitaria elementare inferiore** di indice  $k$ .

La matrice  $M_k$  è del tipo:

$$M_k = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & -m_{k+1} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & -m_n & 0 & \cdots & 1 \end{pmatrix}$$

È possibile ora dimostrare il seguente:

**Teorema 2.1.** *Sia  $x$  un vettore di ordine  $n$ :*

$$x = (\xi_1, \xi_2, \dots, \xi_n)^T \quad \xi_k \neq 0$$

*esiste una e una sola matrice  $M_k$  triangolare elementare unitaria inferiore di indice  $k$  tale che:*

$$M_k x = (\xi_1, \xi_2, \dots, \xi_k, 0, \dots, 0)^T$$

**Dimostrazione** Posto  $M_k = I - m e_k^T$ , affinché tale matrice sia triangolare inferiore di indice  $k$  deve essere:

$$m_i = 0 \quad i = 1, \dots, k \tag{2.12}$$

Inoltre, da:

$$M_k x = (I - m e_k^T) x = x - (e_k^T x) m$$

le ultime  $n - k$  componenti di  $M_k x$  sono nulle ovvero:

$$\xi_i - \xi_k m_i = 0 \quad i = k + 1, \dots, n$$

ma, poiché  $\xi_k \neq 0$  si ha:

$$m_i = \frac{\xi_i}{\xi_k} \quad i = k + 1, \dots, n \tag{2.13}$$

La matrice  $M_k$  è quindi univocamente determinata. ■

Tale teorema assicura l'esistenza di una matrice triangolare unitaria inferiore di indice  $k$  univocamente determinata da (2.12) e (2.13), che annulla le ultime  $n - k$  componenti di un vettore  $x$ , lasciando inalterate le prime  $k$  componenti. Se, inoltre  $\xi_k = 0$  la matrice

$M_k$  non esiste.

Quindi, al generico passo  $k$  dell'algoritmo di eliminazione di Gauss, detta:

$$x = \left( a_{1,k}^{(0)}, \dots, a_{k-1,k}^{(k-1)}, \dots, a_{n,k}^{(k-1)} \right)^T$$

la  $k$ -ma colonna della matrice  $A^{(k-1)}$ , dalle (2.12) e (2.13) il vettore  $m$  che definisce la matrice  $M_k$  è:

$$m^T = \left( 0, \dots, 0, \frac{a_{k+1,k}^{(k-1)}}{a_{k,k}^{(k-1)}}, \dots, \frac{a_{n,k}^{(k-1)}}{a_{k,k}^{(k-1)}} \right) = (0, \dots, 0, m_{k+1,k}, \dots, m_{n,k})$$

costituito cioè dai moltiplicatori calcolati nel  $k$ -mo passo del processo di eliminazione.

Per annullare le ultime  $n-k$  componenti della  $k$ -ma colonna della matrice lasciando inalterate le prime  $k$ , è sufficiente moltiplicare a sinistra la matrice  $A^{(k-1)}$  per la matrice elementare unitaria inferiore  $M_k$ , cioè:

$$A^{(k)} = M_k A^{(k-1)}$$

Si osservi inoltre che le matrici  $M_k$  esistono se e solo se gli elementi  $a_{k,k}^{(k-1)} \neq 0$ . L'esistenza delle matrici  $M_k$  equivale quindi alla condizione di applicabilità dell'algoritmo di eliminazione di Gauss.

Da quanto detto si ha che le  $n-1$  trasformazioni dell'algoritmo di eliminazione di Gauss equivalgono a moltiplicare a sinistra la matrice  $A$  per le matrici elementari triangolari inferiori  $M_k$  con  $k = 1, \dots, n-1$ , cioè:

$$A^{(n-1)} = M_{n-1} M_{n-2} \cdots M_1 A$$

da cui:

$$A = M_1^{-1} M_2^{-1} \cdots M_{n-1}^{-1} A^{(n-1)}$$

e quindi, definendo  $L = M_1^{-1} M_2^{-1} \cdots M_{k-1}^{-1}$  e  $U = A^{(n-1)}$  si ottiene che  $A = LU$ .

Si osservi infine che, data la matrice triangolare elementare unitaria inferiore  $M_k = I - me_k^T$  si ha:

$$M_k^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & m_{k+1,k} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & m_n & 0 & \cdots & 1 \end{pmatrix} = I + me_k^T$$

e che:

$$L = M_1^{-1} M_2^{-1} \cdots M_{n-1}^{-1} = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ m_{2,1} & 1 & 0 & \cdots & 0 \\ m_{3,1} & m_{3,2} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{n,1} & m_{n,2} & m_{n,3} & \cdots & 1 \end{pmatrix}$$

cioè  $L$  è una matrice triangolare inferiore, costituita da tutti i moltiplicatori calcolati durante il processo di eliminazione con, inoltre,  $l_{k,k} = 1$  per  $k = 1, \dots, n$ .

Quanto detto assicura quindi che l'algoritmo di eliminazione di Gauss calcola una fattorizzazione della matrice  $A$  del tipo  $A = LU$ , con  $L$  e  $U$  rispettivamente matrici triangolari inferiore e superiore tali che:

$$l_{i,k} = \begin{cases} 1 & \text{se } i = k \\ m_{i,k} = a_{i,k}^{(k-1)} / a_{k,k}^{(k-1)} & \text{se } i > k \end{cases} \quad e \quad u_{i,k} = a_{i,k}^{(k-1)} \quad i \leq k$$

Ci si chiede ora se esistano altre fattorizzazioni con le medesime proprietà. La risposta a tale quesito è data dal seguente:

**Teorema 2.2.** *Sia data una matrice quadrata  $A$  di ordine  $n$ , e siano  $A_k = (a_{i,j})$   $i, j = 1, \dots, k$  i minori principali di ordine  $k$ . Se tali matrici  $A_k$  sono non singolari esiste una unica matrice triangolare inferiore  $L$  e un'unica matrice triangolare superiore  $U$  tali che:*

$$A = LU$$

con  $\text{diag}(L) = (1, 1, \dots, 1)$  e  $\det(A_k) = u_{1,1}u_{2,2} \cdots u_{k,k}$ .

**Dimostrazione** La dimostrazione procede per induzione. Per  $n = 1$  si ha banalmente che la fattorizzazione  $A = LU$  risulta essere  $a_{1,1} = 1u_{1,1}$ . Si supponga quindi il teorema vero per  $n = k - 1$ . Si ha quindi che esiste una fattorizzazione  $A_{k-1} = L_{k-1}U_{k-1}$ . Siano ora le matrice  $A_k$ ,  $L_k$  e  $U_k$  così partizionate:

$$A_k = \begin{pmatrix} A_{k-1} & b \\ c^T & a_{k,k} \end{pmatrix} \quad L_k = \begin{pmatrix} L_{k-1} & 0 \\ m^T & 1 \end{pmatrix} \quad U_k = \begin{pmatrix} U_{k-1} & u \\ \underline{0} & u_{k,k} \end{pmatrix} \quad (2.14)$$

dove  $b$  e  $c$  sono vettori costituiti rispettivamente dalle prime  $k - 1$  componenti della  $k$  colonna e dalla  $k$  riga di  $A$ , mentre l'elemento diagonale  $u_{k,k}$  e i vettori  $m$  e  $u$ , di ordine  $k - 1$ , sono da determinare.

Imponendo che  $A_k = L_kU_k$  si ottengono le relazioni seguenti:

$$\begin{aligned} 1 \quad & A_{k-1} = L_{k-1}U_{k-1} \\ 2 \quad & L_{k-1}u = b \\ 3 \quad & m^T U_{k-1} = c \\ 4 \quad & m^T u + u_{k,k} = a_{k,k} \end{aligned} \quad (2.15)$$

Inoltre si ha che  $\det(A_k) = \det(L_k)\det(U_k)$ . La prima delle (2.15) è vera per l'ipotesi di induzione. La seconda è invece un sistema di equazioni lineari con matrice triangolare inferiore non singolare, in quanto  $\det(L_{k-1}) = 1$  la cui soluzione è il vettore  $u$ . Analogamente la terza delle (2.15) è un sistema triangolare superiore con matrice non singolare, in quanto  $\det(U_{k-1}) = \det(A_{k-1}) = u_{1,1} \cdots u_{k-1,k-1} \neq 0$  e che ha soluzione  $m$ . Dalla quarta relazione è possibile poi determinare  $u_{k,k}$ . Infine, dalle posizioni fatte in (2.14) si ha anche che  $\text{diag}(L_k) = (1, 1, \dots, 1)$  e  $\det(A_k) = u_{1,1}u_{2,2} \cdots u_{k,k}$ . ■

Tale teorema assicura l'esistenza di un'unica fattorizzazione del tipo  $A = LU$  con le proprietà della fattorizzazione calcolata dall'algoritmo di eliminazione di Gauss. È possibile quindi affermare che l'algoritmo di eliminazione di Gauss è il metodo costruttivo per il calcolo dell'unica fattorizzazione  $A = LU$  tale che:

$$L = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ m_{2,1} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ m_{n,1} & m_{n,2} & \cdots & 1 \end{pmatrix} \quad U = \begin{pmatrix} a_{1,1}^{(0)} & a_{1,2}^{(0)} & \cdots & a_{1,n}^{(0)} \\ 0 & a_{2,2}^{(1)} & \cdots & a_{2,n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n,n}^{(n-1)} \end{pmatrix}$$

Una volta calcolati, mediante l'algoritmo di Gauss, i fattori  $L$  e  $U$  di  $A$  ( ovvero la "fattorizzazione  $LU$ " della matrice  $A$  ), risolvere il sistema

$$Ax = b$$

è equivalente a risolvere il sistema:

$$LUx = b$$

Per avere la soluzione di tale sistema, si risolve dapprima il sistema

$$Ly = b \quad (\text{posto } Ux = y)$$

mediante la *forward-substitution*, e poi il sistema

$$Ux = y$$

mediante la *back-substitution*.

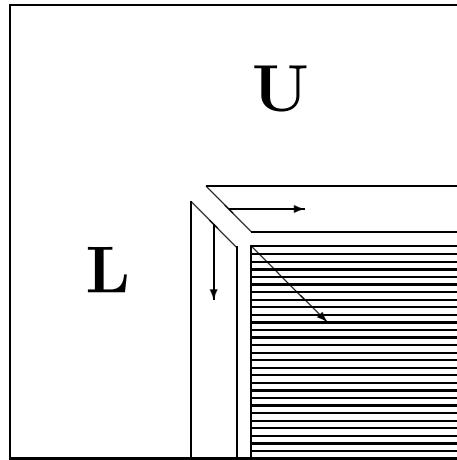
Si osservi, inoltre, che risolvere il sistema  $Ly = b$  equivale ad effettuare sul vettore  $b$  gli  $(n - 1)$  passi dell'algoritmo di Gauss, ovvero  $y = b^{(n-1)}$ , come mostrato nell'esempio che segue:

♣ **Esempio 2.17.** Se si risolve il sistema:

$$\begin{pmatrix} 1 & 0 & 0 \\ m_{2,1} & 1 & 0 \\ m_{3,1} & m_{3,2} & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

con la *forward substitution* si ha:

- $y_1 = b_1$ ;
- $y_2 = b_2 - m_{2,1}y_1 = b_2^{(1)}$ ;
- $y_3 = b_3 - m_{3,1}y_1 - m_{3,2}y_2 = (b_3 - m_{3,1}y_1) - m_{3,2}b_2^{(1)} = b_3^{(1)} - m_{3,2}b_2^{(1)} = b_3^{(2)}$ ;

Figura 2.16: Rappresentazione della fattorizzazione  $LU$  della matrice  $A$ 

ovvero:

$$y = (b_1 \ b_2^{(1)} \ b_3^{(2)})^T = b^{(2)}.$$



In conclusione, quindi, si ha che l'algoritmo di Gauss è equivalente a:

- fattorizzazione  $LU$  di  $A$  ( triangolarizzazione di  $A$  ),
- risoluzione del sistema  $Ly = b$  ( calcolo del vettore  $b^{(n-1)}$  ).

e che tale equivalenza sussiste anche dal punto di vista della complessità computazionale.

La possibilità di fattorizzare la matrice  $A$  nella forma  $A = LU$  risulta particolarmente vantaggiosa, per la risoluzione di quei problemi nei quali si devono risolvere più sistemi aventi la stessa matrice dei coefficienti e diversi termini noti.

**Problema 1:** risoluzione di più sistemi lineari con uguale matrice dei coefficienti.

Molte applicazioni richiedono la risoluzione di più sistemi lineari del tipo:

$$Ax_1 = b_1, \quad Ax_2 = b_2, \quad \dots, \quad Ax_m = b_m,$$

cioè sistemi lineari aventi la stessa matrice dei coefficienti ma termini noti differenti che, ad esempio, non sono dati a priori ma dipendono dalla soluzione di uno dei sistemi precedenti (ad es.  $b_2$  dipende in qualche maniera da  $x_1$ ). In tal caso applicando l'algoritmo di eliminazione di Gauss e la *back-substitution* a ciascun sistema, il costo computazionale sarebbe uguale a:

$$m(T_{Gauss}(n) + T_{bs}(n)) \simeq \mathcal{O}(mn^3)$$

Tale costo computazionale è invece notevolmente ridotto utilizzando la fattorizzazione  $LU$  di  $A$ . Infatti, una volta calcolati i fattori  $L$  ed  $U$  basta risolvere le coppie di sistemi:

$$\begin{aligned} Ly_1 &= b_1, & Ux_1 &= y_1; \\ Ly_2 &= b_2, & Ux_2 &= y_2; \\ Ly_3 &= b_3, & Ux_3 &= y_3; \\ &\vdots && \vdots \\ Ly_m &= b_m, & Ux_m &= y_m; \end{aligned}$$

In sintesi per risolvere il problema in esame basta:

- calcolare **una sola volta** la fattorizzazione  $LU$  di  $A$ , con un costo computazionale di  $\mathcal{O}(\frac{n^3}{3})$ ;
- risolvere  $m$  coppie di sistemi triangolari (uno inferiore ed uno superiore), con un costo computazionale di  $\mathcal{O}(2\frac{n^2}{2})$  per ciascuna di esse.

Di conseguenza il costo computazionale totale è ora

$$\mathcal{O}\left(\frac{n^3}{3} + mn^2\right)$$

**Problema 2:** calcolo dell'inversa di una matrice.

Per calcolare l'inversa  $A^{-1}$  di una assegnata matrice  $A$  di dimensione  $n$ , occorre risolvere  $n$  sistemi lineari:

$$Ax_i = e_i, \quad i = 1, \dots, n,$$

dove  $x_i$  è la  $i$ -ma colonna di  $A^{-1}$  e  $e_i$  è l'  $i$ -mo vettore unitario.

Questo problema è analogo al *Problema 1*, pertanto, utilizzando l'approccio descritto precedentemente si ha che la complessità computazionale del calcolo dell'inversa è:

$$\mathcal{O}\left(\frac{n^3}{3}\right) + 2n\mathcal{O}\left(\frac{n^2}{2}\right) = \mathcal{O}\left(\frac{4}{3}n^3\right)$$

**Problema 3:** calcolo del determinante di una matrice.

Un altro problema in cui la fattorizzazione  $LU$  risulta notevolmente vantaggiosa è il calcolo del determinante di una matrice  $A$  di dimensione  $n$ . Infatti, dalla relazione

$$A = LU$$

si ricava che:

$$\det(A) = \det(LU) = \det(L) \cdot \det(U).$$

Poiché  $L$  ed  $U$  sono matrici triangolari, il loro determinante è dato dal prodotto dei rispettivi elementi diagonali. Pertanto, poiché  $L$  ha tutti gli elementi diagonali uguali ad 1 si ha che:

$$\det(A) = \det(U) = u_{1,1} \cdot u_{2,2} \cdots u_{n,n}$$

In conclusione, la complessità computazionale del calcolo del determinante di  $A$ , utilizzando la fattorizzazione  $LU$ , è:

$$\mathcal{O}\left(\frac{n^3}{3}\right)$$

Ricordando che la complessità computazionale del calcolo del determinante, è  $\mathcal{O}(n!)$ , risulta evidente il vantaggio computazionale ottenuto.

Si consideri ora il seguente:

♣ **Esempio 2.18.** Si applichi l'algoritmo di eliminazione di Gauss con pivoting parziale al sistema:

$$\begin{cases} 2x_1 + 4x_2 - 2x_3 = 6 \\ x_1 - x_2 + 5x_3 = 0 \\ 4x_1 + x_2 - 2x_3 = -2. \end{cases}$$

• passo 1

- pivoting in I colonna:  $\max_{1 \leq i \leq 3} |a_{i,1}| = 4 = a_{1,1}$ , per cui si scambiano la I e la III equazione:

$$\begin{cases} 4x_1 + x_2 - 2x_3 = -2 \\ x_1 - x_2 + 5x_3 = 0 \\ 2x_1 + 4x_2 - 2x_3 = 6; \end{cases}$$

- calcolo moltiplicatori:  $m_{2,1} = 1/4$  e  $m_{3,1} = 1/2$ ;
- eliminazione in I colonna:

$$\begin{cases} 4x_1 + x_2 - 2x_3 = -2 \\ -\frac{5}{4}x_2 + \frac{11}{2}x_3 = \frac{1}{2} \\ \frac{7}{2}x_2 - x_3 = \frac{7}{2}; \end{cases}$$

• passo 2

- pivoting in II colonna  $\max_{2 \leq i \leq 3} |a_{i,2}| = 7/2 = a_{3,2}$ , per cui si scambiano la II e la III equazione:

$$\begin{cases} 4x_1 + x_2 - 2x_3 = -2 \\ -\frac{7}{2}x_2 - x_3 = \frac{7}{2} \\ -\frac{5}{4}x_2 + \frac{11}{2}x_3 = \frac{1}{2}; \end{cases}$$

- calcolo moltiplicatore:  $m_{3,2} = (-\frac{5}{4})/(\frac{7}{2}) = -\frac{5}{14}$ ;
- eliminazione in II colonna:

$$\begin{cases} 4x_1 + x_2 - 2x_3 = -2 \\ \frac{7}{2}x_2 - x_3 = \frac{7}{2} \\ \frac{36}{7}x_3 = 3 \end{cases}$$

Se si considerano ora, le due matrici:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1/4 & -5/14 & 1 \end{pmatrix}; \quad U = \begin{pmatrix} 4 & 1 & -2 \\ 0 & 7/2 & -1 \\ 0 & 0 & 36/7 \end{pmatrix}$$

dove  $L$  è la matrice triangolare inferiore avente elementi diagonali uguali ad 1 ed elementi della  $i$ -ma colonna al di sotto dell'elemento diagonale uguali ai moltiplicatori calcolati al passo  $i^{12}$ , ed  $U$  è la matrice triangolare superiore ottenuta all'ultimo passo dell'algoritmo, e si calcola il prodotto  $LU$ , si ha:

$$LU = \begin{pmatrix} 4 & 1 & -2 \\ 2 & 4 & -2 \\ 1 & -1 & 5 \end{pmatrix}$$

Confrontando tale matrice con la matrice  $A$  si ha che  $LU$  ha le stesse righe di  $A$  ma *in ordine diverso*. Ciò deriva dal fatto che l'applicazione del pivoting parziale modifica l'ordine delle righe. Si effettuino ora su  $A$  gli scambi fatti durante le fasi di pivoting:

- al passo 1 è stato effettuato uno scambio tra la I e la III riga, cioè:

$$\begin{pmatrix} 2 & 4 & -2 \\ 1 & -1 & 5 \\ 4 & 1 & -2 \end{pmatrix} \Rightarrow \begin{pmatrix} 4 & 1 & -2 \\ 1 & -1 & 5 \\ 2 & 4 & -2 \end{pmatrix};$$

- al passo 2 è stato effettuato uno scambio tra la II e la III riga, cioè:

$$\begin{pmatrix} 4 & 1 & -2 \\ 1 & -1 & 5 \\ 2 & 4 & -2 \end{pmatrix} \Rightarrow \begin{pmatrix} 4 & 1 & -2 \\ 2 & 4 & -2 \\ 1 & -1 & 5 \end{pmatrix} = LU$$



L'esempio precedente mostra che l'algoritmo di eliminazione di Gauss con pivoting parziale calcola i fattori  $L$  ed  $U$  di una matrice avente le stesse righe di  $A$  ma nell'ordine determinato dagli scambi di righe effettuati nell'applicazione del pivoting parziale, ovvero  $A$  si ottiene da  $LU$  operando su essa gli scambi di righe effettuati nell'applicazione dell'algoritmo di eliminazione di Gauss con pivoting parziale.

♣ **Esempio 2.19.** Nell'esempio 2.18 a partire dalla matrice:

$$A = \begin{pmatrix} 2 & 4 & -2 \\ 1 & -1 & 5 \\ 4 & 1 & -2 \end{pmatrix}$$

si è calcolata la matrice:

$$LU = \begin{pmatrix} 4 & 1 & -2 \\ 2 & 4 & -2 \\ 1 & -1 & 5 \end{pmatrix} = \overline{A}.$$

Si consideri la seguente matrice:

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

---

<sup>12</sup>Si osservi che sulla II riga c'è il moltiplicatore relativo alla riga che è stata modificata una sola volta, mentre sulla III riga ci sono i due moltiplicatori relativi alla riga modificata due volte.

calcolando il prodotto di  $P$  per  $A$  si ha

$$\begin{aligned} PA &= \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 2 & 4 & -2 \\ 1 & -1 & 5 \\ 4 & 1 & -2 \end{pmatrix} = \\ &= \begin{pmatrix} 4 & 1 & -2 \\ 2 & 4 & -2 \\ 1 & -1 & 5 \end{pmatrix} = \bar{A} = LU \end{aligned}$$

♣

L'effetto della moltiplicazione di  $P$  per  $A$  è stato di modificare l'ordine delle righe di  $A$ . Per tale motivo  $P$  è detta matrice di *permutazione*. La matrice  $P$  considerata è un esempio di matrice di permutazione (cfr. §B.1.1, Appendice B).

Si può, quindi, concludere che i fattori  $L$  ed  $U$  prodotti dall'algoritmo sono i fattori della matrice  $PA$  ovvero:

$$PA = LU$$

dove  $P$  è una matrice di permutazione che “conserva” l'informazione degli scambi effettuati. Una volta effettuata la fattorizzazione  $PA = LU$ , ed osservando che  $PAx = Pb$ , la risoluzione del sistema  $Ax = b$  può essere effettuata mediante la risoluzione del sistema triangolare inferiore  $Ly = Pb$  mediante la *forward substitution* e del sistema triangolare superiore  $Ux = y$  mediante la *backward substitution*. Si osservi comunque che la risoluzione del sistema  $Ly = Pb$  richiede che vengano effettuate sul vettore dei termini noti  $b$  tutte le permutazioni di righe fatte sulla matrice  $A$  nel corso dell'algoritmo di eliminazione di Gauss con pivoting parziale.

#### ♣ Esempio 2.20.

Nell'esempio 2.18

- al passo 1 è stato effettuato uno scambio tra la I e III riga di  $A$ . A questo punto si osservi che se si effettuano gli stessi scambi sulla matrice  $I$ , si ottiene la matrice:

$$P_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

mediante la quale è possibile ottenere:

$$\begin{pmatrix} 4 & 1 & -2 \\ 1 & -1 & 5 \\ 2 & 4 & -2 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 2 & 4 & -2 \\ 1 & -1 & 5 \\ 4 & 1 & -2 \end{pmatrix} = P_1 A$$

cioè la fase di pivoting è equivalente a moltiplicare a sinistra la matrice  $A$  per la matrice  $P_1$ . La successiva fase di trasformazione della matrice produce poi la matrice:

$$A^{(1)} = M_1 P_1 A$$

- Analogamente al passo 2: se si effettua uno scambio tra la II e III riga della matrice  $I$  si ottiene:

$$P_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

mediante la quale è possibile ottenere:

$$\begin{pmatrix} 4 & 1 & -2 \\ 0 & \frac{7}{2} & -1 \\ 0 & -\frac{5}{4} & \frac{11}{2} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 4 & 1 & -2 \\ 0 & -\frac{5}{4} & \frac{11}{2} \\ 0 & \frac{7}{2} & -1 \end{pmatrix} = P_2 A^{(1)}$$

cioè la fase di pivoting è equivalente a moltiplicare a sinistra la matrice  $A^{(1)} = M_1 P_1 A$  per la matrice  $P_2$ . La successiva fase di trasformazione della matrice produce poi la matrice:

$$U = A^{(2)} = M_2 P_2 A^{(1)} = M_2 P_2 M_1 P_1 A = M_2 P_2 M_1 P_1^T P_2 P_1 A$$

Posto quindi  $P = P_2 P_1$  si ha:

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

cioè la matrice  $P$  si ottiene applicando a  $I$  tutti gli scambi di righe effettuati durante l'algoritmo di eliminazione di Gauss.



In generale sussiste il

**Teorema 2.3.** *Data una matrice  $A$  non singolare, l'algoritmo di eliminazione di Gauss con pivoting parziale calcola le matrici  $L$  e  $U$  tali che:*

$$PA = LU$$

dove:

$$P = P_{n-1} \cdots P_1$$

$$\overline{M}_k = P_{n-1} \cdots P_{k+1} M_k P_{k+1} \cdots P_{n-1} \quad \text{per } k \leq n-2$$

$$\overline{M}_{n-1} = M_{n-1}$$

$$L = \overline{M}_1^{-1} \cdot \overline{M}_2^{-1} \cdots \overline{M}_{n-1}^{-1}$$

$$U = \overline{M}_{n-1} \cdot \overline{M}_{n-2} \cdots \overline{M}_1 A$$

**Dimostrazione:** posto  $A^{(0)} = A$ , al generico passo  $k$  dell'algoritmo di eliminazione di Gauss si ha:

$$A^{(k)} = M_k P_k A^{(k-1)} \quad \text{con } k = 1, \dots, n-1$$

dove  $P_k$  e  $M_k$  sono le matrici che effettuano la fase di pivoting e di trasformazione della matrice attiva. All'ultimo passo si ha quindi:

$$U = A^{(n-1)} = M_{n-1} P_{n-1} M_{n-2} P_{n-2} \cdots M_2 P_2 M_1 P_1 A \tag{2.16}$$

Ora, poiché  $\forall k$  si ha  $P_k^T P_k = I$  è possibile scrivere:

$$\begin{aligned} M_{n-2} &= M_{n-2} P_{n-1}^T P_{n-1} \\ M_{n-3} &= M_{n-3} P_{n-2}^T P_{n-1}^T P_{n-1} P_{n-2} \\ &\vdots \\ M_k &= M_k P_{k+1}^T \cdots P_{n-1}^T P_{n-1} \cdots P_{k+1} \quad k \geq 1 \end{aligned}$$

e quindi dalla (2.16) si ottiene:

$$U = M_{n-1} P_{n-1} M_{n-2} P_{n-1}^T P_{n-1} P_{n-2} M_{n-3} P_{n-2}^T P_{n-1} \cdots P_{n-1} P_{n-2} \cdots P_1 A$$

Posto, infine:

$$P = P_{n-1} P_{n-2} \cdots P_1 \quad \overline{M}_k = P_{n-1} \cdots P_{k+1} M_k P_{k+1}^T \cdots P_{n-1}^T$$

si ha la tesi. ■

Poiché  $PA = LU$ , una volta calcolati, mediante l'algoritmo di eliminazione di Gauss con pivoting parziale, i fattori  $L$  ed  $U$  della matrice  $PA$  si ha

$$LUx = (PA)x = P(Ax) = Pb$$

ovvero prima di procedere nella applicazione della *back* e *forward-substitution* per il calcolo di  $x$  è necessario effettuare sul vettore  $b$  gli stessi scambi effettuati sulla matrice  $A$ .

♣ **Esempio 2.21.** Nell'esempio 2.18, considerato

$$b = (6 \ 0 \ 2)^T;$$

- al passo 1 viene effettuato uno scambio tra la I e III riga di  $A$ . È necessario, quindi operare tale scambio sul vettore  $b$  ottenendo:

$$(2 \ 0 \ 6)^T;$$

- al passo 2 viene effettuato uno scambio tra la II e III riga della matrice ottenuta dopo il passo 1. Tale scambio va operato anche sul vettore dei termini noti e, quindi si ha il vettore:

$$(2 \ 6 \ 0)^T.$$

Tale vettore si può ottenere anche moltiplicando la matrice di permutazione  $P$  per il vettore  $b$ . Infatti:

$$Pb = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 6 \\ 0 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 6 \\ 0 \end{pmatrix}$$



Pertanto per risolvere il sistema  $Ax = b$  mediante l'algoritmo di eliminazione di Gauss con pivoting parziale:

1. si applica l'algoritmo di eliminazione di Gauss con pivoting parziale alla matrice  $A$  (Procedura 2.10) e si ottengono i fattori  $L$  ed  $U$  della matrice  $PA$ , dove  $P$  è la matrice di permutazione che si ottiene dalla matrice identica effettuando su essa gli scambi di righe che sono derivati dall'applicazione del pivoting parziale;
2. si effettuano sul vettore  $b$  gli stessi scambi (moltiplicando  $P$  per  $b$ );
3. si risolve il sistema  $LUx = Pb$  mediante *forward* e *back substitution*.

### 2.6.1 Aspetti implementativi

Si è visto, nel paragrafo precedente, che, per operare sul vettore  $b$  dei termini noti gli scambi effettuati durante il processo di eliminazione con pivoting sulle righe, bisogna conoscere la matrice  $P$ . In realtà *non è necessario costruire*  $P$ , ma è sufficiente costruire un vettore che *ricordi gli scambi effettuati*. Come si costruisce tale vettore?

♣ **Esempio 2.22.** Sia

$$A = \begin{pmatrix} 1 & 3 & 5 & 7 \\ -1 & 4 & 0 & 2 \\ 8 & 3 & 0 & 5 \\ 4 & 2 & 0 & 1 \end{pmatrix}$$

Si costruisca un vettore, indicato con  $ipiv$ , di dimensione 4, che contiene inizialmente gli indici di riga in ordine naturale, ovvero:

$$ipiv_i = i, \quad i = 1, \dots, 4$$

$$A = \begin{pmatrix} 1 & 3 & 5 & 7 \\ -1 & 4 & 0 & 2 \\ 8 & 3 & 0 & 5 \\ 4 & 2 & 0 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} = ipiv.$$

Se si opera uno scambio tra la II e IV riga di  $A$ , tale scambio è “memorizzato” in  $ipiv$ , scambiando il contenuto di  $ipiv_2$  ed  $ipiv_4$ :

$$\begin{pmatrix} 1 & 3 & 5 & 7 \\ 4 & 2 & 0 & 1 \\ 8 & 3 & 0 & 5 \\ -1 & 4 & 0 & 2 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 4 \\ 3 \\ 2 \end{pmatrix} = ipiv.$$

Analogamente, se si opera su tale matrice uno scambio tra la II e III riga, esso è memorizzato in  $ipiv$  scambiando il contenuto di  $ipiv_2$  ed  $ipiv_3$ :

$$\begin{pmatrix} 1 & 3 & 5 & 7 \\ 8 & 3 & 0 & 5 \\ 4 & 2 & 0 & 1 \\ -1 & 4 & 0 & 2 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 3 \\ 4 \\ 2 \end{pmatrix} = ipiv$$

Verificando il contenuto degli elementi del vettore  $ipiv$  è possibile stabilire come è stato alterato l'ordine delle righe di  $A$  (nell'esempio si ha che la I riga è rimasta al suo posto, la II riga di  $A$  costituisce ora la IV, la III riga di  $A$  è ora la II e la IV è la III).

Il vettore  $ipiv$  è un *vettore di puntatori*, nel senso che l'elemento di posto  $i$  di  $ipiv$  contiene (“*punta a*”) l'indice della riga di  $A$  che ora costituisce la  $i$ -ma riga ( $ipiv_i = j \Rightarrow$  la  $j$ -ma riga della matrice originaria  $A$  è ora la  $i$ -ma riga).

Una volta costruito il vettore  $ipiv$ , durante il processo di eliminazione con pivoting sulla matrice  $A$ , per effettuare gli opportuni scambi sul vettore  $b$  dei termini noti, basta “*consultarlo*”.

Sia quindi:

$$b = \begin{pmatrix} 3.4 \\ 0 \\ 1.5 \\ 8 \end{pmatrix}; \quad ipiv = \begin{pmatrix} 1 \\ 3 \\ 4 \\ 2 \end{pmatrix}$$

di conseguenza per effettuare gli opportuni scambi su  $b$ , bisogna spostare la III componente al posto della II, la II al posto della IV e la IV al posto della III, ottenendo:

$$\begin{pmatrix} 3.4 \\ 1.5 \\ 8 \\ 0 \end{pmatrix}$$

♣

Il vettore  $ipiv$  è un *vettore di puntatori*, nel senso che l'elemento di posto  $i$  di  $ipiv$  contiene (“*punta a*”) l'indice della riga di  $A$  che ora costituisce la  $i$ -ma riga ( $ipiv_i = j \Rightarrow$  la  $j$ -ma riga della matrice originaria  $A$  è ora la  $i$ -ma riga).

L'utilizzo che si può fare del vettore  $ipiv$  nell'ambito di una procedura per la risoluzione di un sistema lineare  $Ax = b$  basata sull'algoritmo di fattorizzazione  $LU$  con pivoting parziale e di *back* e *forward-substitution* è in realtà più ampio. Infatti, oltre ad essere necessario per effettuare gli opportuni scambi sul vettore  $b$ , esso può essere utilizzato anche **per non scambiare fisicamente** le righe di  $A$  e gli elementi di  $b$ .

### ♣ Esempio 2.23.

$$b = \begin{pmatrix} 2.4 \\ 0 \\ 6 \\ 5 \end{pmatrix}; \quad P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

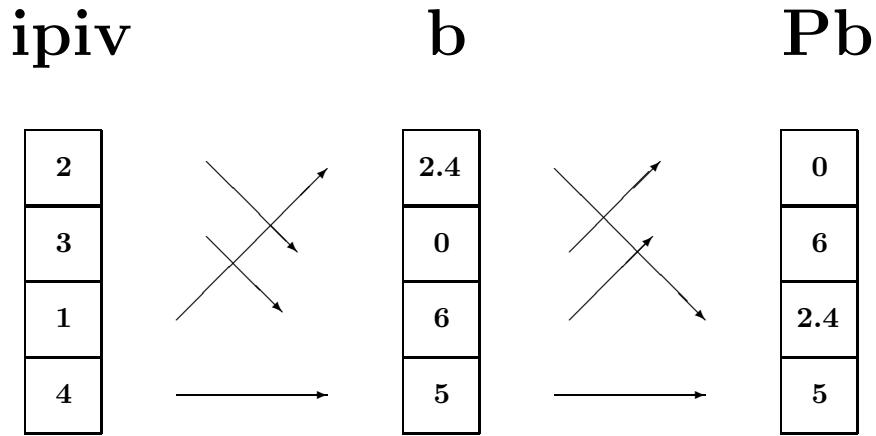
e quindi:

$$Pb = \begin{pmatrix} 0 \\ 6 \\ 2.4 \\ 5 \end{pmatrix}$$

Il vettore  $ipiv$ , che memorizza gli scambi operati su  $b$ , è:

$$ipiv = \begin{pmatrix} 2 \\ 3 \\ 1 \\ 4 \end{pmatrix}$$

Si osservi che

Figura 2.17: Utilizzo dell'array  $ipiv$  per l'accesso alle componenti di  $b$ .

- I componente di  $Pb = 0$  e  $b(ipiv(1)) = b_2 = 0$ ;
- II componente di  $Pb = 6$  e  $b(ipiv(2)) = b_3 = 6$ ;
- III componente di  $Pb = 2.4$  e  $b(ipiv(3)) = b_1 = 2.4$ ;
- IV componente di  $Pb = 5$  e  $b(ipiv(4)) = b_4 = 5$ .

Ciò significa che:

$$b(ipiv(i)) = (Pb)_i, \quad i = 1, 2, \dots, 4,$$

ovvero  $b(ipiv(i))$  è il valore contenuto nell'elemento di posto  $i$  del vettore  $Pb$ .



Quindi, per accedere correttamente alle componenti del vettore  $Pb$ , anziché scambiare fisicamente gli elementi di  $b$ , si possono utilizzare le componenti del vettore  $ipiv$  come indici del vettore  $b$  (Procedura 2.11), ovvero si può accedere alle componenti del vettore  $Pb$  attraverso  $b$ :

$$b(ipiv(i)) = (Pb)_i, \quad i = 1, \dots, n.$$

In maniera analoga si può operare per la matrice  $A$  come mostrato nella Procedura 2.10.

♣ **Esempio 2.24.**

$$A = \begin{pmatrix} 1 & 3 & 5 & 7 \\ -1 & 4 & 0 & 2 \\ 8 & 3 & 0 & 5 \\ 4 & 2 & 0 & 1 \end{pmatrix}; \quad P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

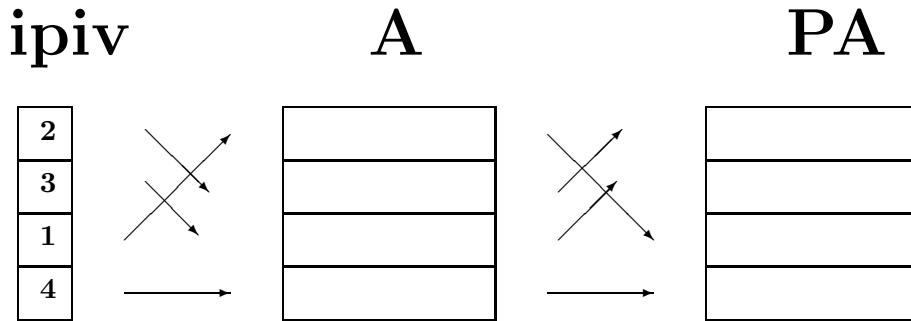


Figura 2.18: Utilizzo dell’array  $ipiv$  per l’accesso alle componenti di  $A$ .

e quindi:

$$PA = \begin{pmatrix} -1 & 4 & 0 & 2 \\ 8 & 3 & 0 & 5 \\ 1 & 3 & 5 & 7 \\ 4 & 2 & 0 & 1 \end{pmatrix}$$

da cui:

$$ipiv = \begin{pmatrix} 2 \\ 3 \\ 1 \\ 4 \end{pmatrix}$$

Osservando la Figura 2.24 si nota che gli elementi

$$a(ipiv(i), j), \quad j = 1, 2, \dots, 4,$$

sono gli elementi della  $i$ -ma riga di  $PA$ . Di conseguenza per accedere correttamente alle righe (e quindi agli elementi) di  $PA$ , anziché scambiare fisicamente la righe di  $A$ , basta utilizzare le componenti del vettore  $ipiv$  come indici di riga della matrice  $A$ .

♣

In conclusione,

- nell’algoritmo di fattorizzazione  $PA = LU$  con pivoting parziale (Procedura 2.10) l’utilizzo di un vettore di puntatori che memorizzi, come illustrato, gli scambi richiesti sulla matrice  $A$  consente di:
  1. non costruire la matrice  $P$  per memorizzare gli scambi;
  2. non effettuare fisicamente lo scambio di righe.
- nell’algoritmo di forward substitution per la risoluzione di  $Ly = Pb$  (Procedura 2.11) l’utilizzo del vettore di puntatori consente di:
  1. conoscere gli scambi da effettuare su  $b$ ;

2. non effettuare fisicamente gli scambi.

Si osservi infine che, data l'equivalenza dell'algoritmo di eliminazione di Gauss con pivoting parziale con la fase di fattorizzazione  $PA = LU$  seguita dalla risoluzione di  $Ly = Pb$ , nell'algoritmo di *back-substitution* per la risoluzione di  $Ux = y$  non è necessario l'utilizzo delle componenti dell'array *ipiv* come indice di componente per il vettore *y*, in quanto la fase di *forward substitution* già ha predisposto gli elementi di *y* nell'ordine opportuno.

```
procedure paeqlu (in: a, ipiv, n; out: a, ipiv)  

  /* SCOPO: fattorizzazione LU con pivoting parziale.  

  /* SPECIFICHE DEI PARAMETRI:  

  /* PARAMETRI DI INPUT:  

var: n : intero { dimensione del sistema }  

var: ipiv(n) : intero { vettore delle permutazioni }  

var: a(n, n) : reale { matrice del sistema }  

               { da fattorizzare }
```

Procedura 2.10: Algoritmo di fattorizzazione  $PA = LU$  con pivoting parziale - continua

```

 $\# PARAMETRI DI OUTPUT:$ 
var:  $ipiv(n)$  : intero { vettore delle permutazioni }
{ effettuate }

var:  $a(n, n)$  : reale { matrice del sistema fattorizzata }

 $\# VARIABILI LOCALI:$ 
var:  $i, j, k, t, r$  : interi
var:  $mas$  : reale

 $\# INIZIO ISTRUZIONI:$ 
for  $i = 1$  to  $n$  {inizializzazione array  $ipiv$ }
 $ipiv(i) := i$ 
end for
for  $k = 1$  to  $n - 1$  {ciclo sui passi}
 $mas := abs(a(ipiv(k), k))$  {pivoting}
 $r := k$ 
for  $i = k + 1$  to  $n$  {ricerca del massimo}
if ( $abs(a(ipiv(i), k)) > mas$ ) then
 $mas := abs(a(ipiv(i), k))$ 
 $r := i$ 
end if
endfor
if ( $r \neq k$ ) then
 $t := ipiv(r)$  {scambio virtuale delle righe}
 $ipiv(r) := ipiv(k)$ 
 $ipiv(k) := t$ 
endif
for  $i = k + 1$  to  $n$  {calcolo degli elementi di  $L$  e  $U$ }
 $a(ipiv(i), k) := a(ipiv(i), k)/a(ipiv(k), k)$ 
for  $j = k + 1$  to  $n$ 
 $a(ipiv(i), j) := a(ipiv(i), j) - a(ipiv(i), k) * a(ipiv(k), j)$ 
end for
end for
end for
end paeqlu

```

Procedura 2.10: Algoritmo di fattorizzazione  $PA = LU$  con pivoting parziale - fine

```

procedure lyeqpb (in:  $a, b, n, ipiv$  ; out:  $x$ )
  /* SCOPO: Risoluzione di un sistema triangolare inferiore,
  con matrice dei coefficienti risultante da
  una fattorizzazione LU, con pivoting parziale e
  scambio virtuale delle righe.

  /* SPECIFICHE DEI PARAMETRI:
  /* PARAMETRI DI INPUT:
var:  $n$  : intero { dimensione del sistema }
var:  $ipiv(n)$  : intero { vettore delle permutazioni }
var:  $a(n, n)$  : reale { matrice del sistema }
  { fattorizzata }
var:  $b(n)$  : reale { vettore dei termini noti }

  /* PARAMETRI DI OUTPUT:
var:  $y(n)$  : reale { vettore delle soluzioni }
/* VARIABILI LOCALI:
var:  $i, j$  : interi
var:  $r$  : reale

  /* INIZIO ISTRUZIONI:
 $y(1) = b(ipiv(1))$  { calcolo di  $y(1)$ }
 $i := 2$ 

while ( $i \leq n$ ) { ciclo sugli  $y(i)$ }
   $r := b(ipiv(i))$ 
  for  $j = 1$  to  $i - 1$  { calcolo del resto}
     $r := r - a(ipiv(i), j) * y(j)$ 
  end for
   $i := i + 1$ 
end while
end lyeqpb

```

Procedura 2.11: Forward-substitution per  $Ly = Pb$  - fine

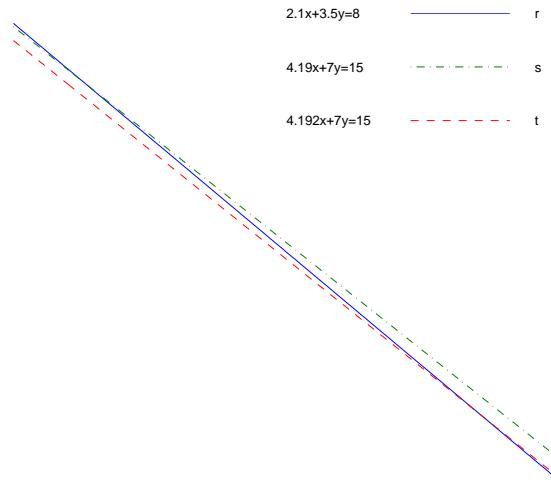


Figura 2.19: Rappresentazione grafica del sistema dell'esempio 2.25

## 2.7 Condizionamento di sistemi lineari

È noto che il *condizionamento* è una caratteristica del problema e non dell'algoritmo risolutivo. Pertanto, la risoluzione di un sistema mal condizionato risulta inaccurata anche utilizzando algoritmi stabili.

È chiaro che, essendo interessati alla risoluzione di sistemi lineari mediante calcolatore, è necessario analizzare il condizionamento del sistema in esame, in quanto i dati sono affetti almeno dagli errori di round-off.

♣ **Esempio 2.25.** Il sistema di equazioni lineari:

$$\begin{cases} 2.1x + 3.5y = 8 \\ 4.19x + 7y = 15, \end{cases} \quad (2.17)$$

ha soluzione  $x = 100$ ,  $y = -57.714 \dots$ . Si consideri ora, il sistema:

$$\begin{cases} 2.1x + 3.5y = 8 \\ 4.192x + 7y = 15, \end{cases} \quad (2.18)$$

che può essere riguardato come una perturbazione del sistema (2.17), essendo stato introdotto un errore sulla quarta cifra significativa del coefficiente della  $x$  nella seconda equazione (un errore relativo di  $10^{-3}$ ). La soluzione del sistema (2.18) è  $x = 125$ ,  $y = -72.714\dots$ . Se si considera la soluzione di (2.18) come la soluzione del sistema (2.17) perturbato, si osserva che l'errore introdotto ha condotto ad un soluzione completamente diversa. Il motivo per cui si è verificato tale fenomeno si desume facilmente dalla rappresentazione geometrica dei due sistemi (Figura 2.19).

La perturbazione introdotta corrisponde ad una modifica della inclinazione della retta rappresentata dalla seconda equazione del sistema (2.17). Tale modifica provoca una grande perturbazione sulla soluzione in quanto le due rette relative al sistema (2.17) sono “quasi parallele” e, quindi una lieve perturbazione del coefficiente angolare di una delle due rette determina un notevole spostamento del punto

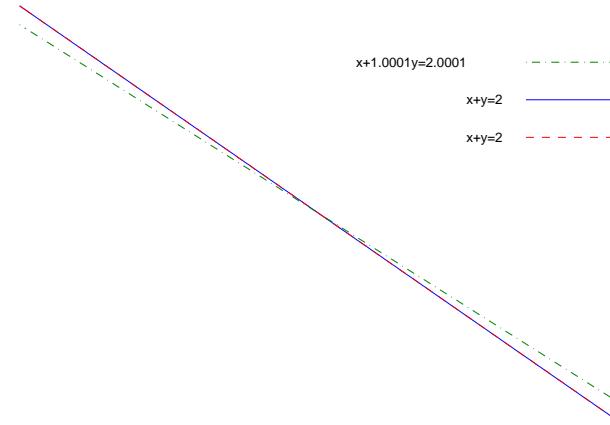


Figura 2.20: Rappresentazione grafica del sistema dell'esempio 2.26

di intersezione. Il sistema (2.17), pertanto, è tale che ad una piccola variazione nei dati corrisponde una grande perturbazione nella soluzione. Il sistema (2.17) è un sistema mal condizionato.



♣ **Esempio 2.26.** Il sistema:

$$\begin{cases} x + y = 2 \\ x + 1.0001y = 2.0001 \end{cases} \quad (2.19)$$

ha soluzione  $x = 1, y = 1$ . Utilizzando un sistema aritmetico floating-point con base  $\beta = 10$  e precisione  $t = 4$ , si ottiene il sistema:

$$\begin{cases} x + y = 2 \\ x + y = 2, \end{cases} \quad (2.20)$$

che è singolare!

Quindi l'errore di round-off nei dati ha alterato il coefficiente angolare della retta rappresentata dalla seconda equazione del sistema (2.19) in maniera tale da rendere coincidenti le due rette (Figura 2.20) e di conseguenza, singolare il relativo sistema.



Come si può stimare quantitativamente il condizionamento di un sistema lineare?  
Dato un sistema lineare:

$$Ax = b, \quad (2.21)$$

$$A = (a_{i,j}), \quad i, j = 1, \dots, n, \quad b = (b_i) \quad i = 1, \dots, n,$$

i coefficienti ed il termine noto sono affetti almeno dall'errore di round-off di rappresentazione. Ciò vuol dire che in luogo del sistema assegnato si risolve *sempre* un sistema perturbato; ad esempio, utilizzando il sistema aritmetico indicato nell'esempio 2.26, in luogo del sistema (2.19) si risolve il sistema (2.20).

In generale, si risolve un sistema con coefficienti perturbati:

$$a_{i,j} + \Delta a_{i,j}, \quad i, j = 1, \dots, n,$$

con  $\Delta a_{i,j}$  = errore di round-off nella rappresentazione di  $a_{i,j}$  e componenti del termine noto perturbate:

$$b_i + \Delta b_i, \quad i = 1, \dots, n,$$

con  $\Delta b_i$  = errore di round-off nella rappresentazione di  $b_i$ .

Naturalmente anche la soluzione,  $x$ , del sistema (2.21), risulterà perturbata. Pertanto, indicata con:

$$\Delta A = (\Delta a_{i,j})_{i,j=1,\dots,n}$$

la matrice delle perturbazioni relativa ai coefficienti del sistema, con:

$$\Delta b = (\Delta b_i)_{i=1,\dots,n}$$

il vettore delle perturbazioni relative alle componenti del termine noto e, detta  $\Delta x$  la perturbazione introdotta sulla soluzione, in luogo del sistema (2.21) si risolverà il sistema perturbato:

$$(A + \Delta A)(x + \Delta x) = b + \Delta b \quad (2.22)$$

♣ **Esempio 2.27.** Utilizzando un sistema aritmetico floating-point con base  $\beta = 10$  e precisione relativa  $t = 3$ , con il metodo dell'arrotondamento, in luogo del sistema

$$\begin{cases} 4.124x + 3.02y = 5.246 \\ 1.34x + 2.563y = 7.31, \end{cases}$$

si risolve il sistema:

$$\begin{cases} 4.12x + 3.02y = 5.25 \\ 1.34x + 2.56y = 7.31 \end{cases}$$

Pertanto,

$$A = \begin{pmatrix} 4.124 & 3.02 \\ 1.34 & 2.563 \end{pmatrix}; \quad b = \begin{pmatrix} 5.246 \\ 7.31 \end{pmatrix}$$

e

$$\Delta A = \begin{pmatrix} -0.004 & 0 \\ 0 & -0.003 \end{pmatrix}; \quad \Delta b = \begin{pmatrix} 0.004 \\ 0 \end{pmatrix}$$



Prima di risolvere il sistema perturbato è necessario valutare come gli errori  $\Delta A$  e  $\Delta b$  influenzino la soluzione, ovvero determinare una stima dell'errore  $\Delta x$  in funzione degli errori nei dati.

### 2.7.1 Indice di Condizionamento

Per stimare il condizionamento è necessario valutare l'errore relativo nella soluzione del sistema:

$$\frac{\|\Delta x\|}{\|x\|},$$

in funzione degli errori relativi nei dati:

$$\frac{\|\Delta A\|}{\|A\|} \text{ e } \frac{\|\Delta b\|}{\|b\|},$$

cioè bisogna determinare quanto l'errore nei dati si amplifichi nella soluzione.

Inizialmente, per semplicità, si supponga che ci sia una perturbazione solo sul termine noto del sistema, ovvero anziché

$$Ax = b, \tag{2.23}$$

si sta risolvendo il sistema perturbato:

$$A(x + \Delta x) = b + \Delta b, \tag{2.24}$$

nella ipotesi che  $\Delta b$  sia “piccolo”<sup>13</sup> rispetto a  $b$ .

Per stimare  $\Delta x$  in funzione di  $\Delta b$ , si osservi che da (2.24) discende che:

$$Ax + A\Delta x = b + \Delta b, \quad \text{con } Ax = b$$

e quindi:

$$A\Delta x = \Delta b$$

da cui:

$$\Delta x = A^{-1}\Delta b$$

nell'ipotesi che  $A$  sia invertibile.

Introdotta, dunque, una norma matriciale compatibile con una vettoriale (cfr. Definizione B.5), la soluzione soddisfa la diseguaglianza

$$\|\Delta x\| \leq \|A^{-1}\| \|\Delta b\|.$$

---

<sup>13</sup>Per “piccolo” si intende dell'ordine dell'errore di round-off che si commette nel rappresentare  $b$  in un assegnato sistema aritmetico floating-point a precisione finita.

Questa relazione fornisce informazioni sul *condizionamento assoluto* del problema; è possibile, inoltre, ottenere un'espressione analoga, per il *condizionamento relativo*. Valutando l'errore relativo nella soluzione si ha:

$$\frac{\|\Delta x\|}{\|x\|} = \frac{\|A^{-1}\Delta b\|}{\|x\|} \leq \frac{\|A^{-1}\|\|\Delta b\|}{\|x\|} \leq \frac{\|A^{-1}\|\|\Delta b\|}{\|b\|/\|A\|}$$

dove l'ultima eguaglianza si ottiene tenendo presente che:

$$\|b\| = \|Ax\| \leq \|A\|\|x\|$$

da cui:

$$\frac{1}{\|x\|} \leq \frac{1}{\|b\|/\|A\|}.$$

In conclusione si è ottenuto che:

$$\frac{\|\Delta x\|}{\|x\|} \leq \|A\|\|A^{-1}\| \frac{\|\Delta b\|}{\|b\|}$$

Quindi la quantità  $\|A\|\|A^{-1}\|$  stima l'amplificazione dell'errore nei dati sulla soluzione.

♣ **Esempio 2.28.** Il sistema di equazioni lineari:

$$\begin{cases} x_1 + .99x_2 = 1.989903 \\ .99x_1 + .98x_2 = 1.970106 \end{cases} \quad (2.25)$$

ha soluzione  $x = (3.0000, -1.0203)^T$ .

Utilizzando un sistema aritmetico floating-point con base  $\beta = 10$  e precisione relativa  $t = 3$  con il metodo di arrotondamento, il sistema (2.25) diventa

$$\begin{cases} x_1 + .99x_2 = 1.99 \\ .99x_1 + .98x_2 = 1.97, \end{cases} \quad (2.26)$$

la cui soluzione è  $x + \Delta x = (1, 1)^T$ .

Pertanto, una perturbazione

$$\Delta b = \begin{pmatrix} -0.000097 \\ +0.000106 \end{pmatrix}$$

sul termine noto, dovuta alla sua rappresentazione nel sistema aritmetico considerato, ha condotto ad una soluzione completamente diversa. In particolare,

$$\Delta x = \begin{pmatrix} 2.0000 \\ 2.0203 \end{pmatrix}$$

Si osservi che:

$$\|\Delta x\|_\infty = 2.0203 \text{ e } \|x\|_\infty = 3.0000$$

da cui:

$$\frac{\|\Delta x\|_\infty}{\|x\|_\infty} = \frac{2.0203}{3.0000} \simeq 0.67 \text{ arrotondato a 2 cifre}$$

Inoltre,

$$\|\Delta b\|_\infty = .106 \times 10^{-3} \text{ e } \|b\|_\infty = 1.989903$$

e quindi

$$\frac{\|\Delta b\|_\infty}{\|b\|_\infty} = \frac{.106 \times 10^{-3}}{1.989903} \simeq 0.53 \times 10^{-4} \text{ arrotondato a 2 cifre.}$$

Da ciò discende che

$$\frac{\|\Delta x\|_\infty}{\|x\|_\infty} \leq 2 \times 10^4 \frac{\|\Delta b\|_\infty}{\|b\|_\infty}$$

ovvero il fattore di amplificazione dell'errore nei dati sulla soluzione è dell'ordine di  $10^4$ .

D'altra parte, poiché

$$A^{-1} = \begin{pmatrix} -9800 & 9900 \\ 9900 & -10000 \end{pmatrix}$$

si ha che:

$$\|A\|_\infty = 1.99 \text{ e } \|A^{-1}\|_\infty = 19900$$

da cui:

$$\|A\| \|A^{-1}\| = 1.99 \times 19900 \simeq 4 \times 10^4$$

si ottiene cioè, a meno di una costante, il fattore di amplificazione dell'errore nei dati sulla soluzione.



Sussiste la seguente:

### Definizione 2.8. (Indice di condizionamento relativo)

*La quantità*

$$\mu(A) = \|A\| \|A^{-1}\|, \quad (2.27)$$

*è detta indice di condizionamento relativo del sistema di equazioni  $Ax = b$ .*

Dall'espressione di  $\mu(A)$ , si deduce che il condizionamento di un sistema lineare è una proprietà intrinseca della matrice dei coefficienti (e non dipende dagli errori di cui è affetto il vettore dei termini noti). Inoltre, è sempre  $\mu(A) \geq 1$ , in quanto:

$$1 = \mu(I) = \|AA^{-1}\| \leq \|A\| \|A^{-1}\| = \mu(A)$$

L'indice di condizionamento misura il massimo fattore di amplificazione dell'errore relativo sulla soluzione, rispetto all'errore relativo sui dati. Un sistema di equazioni è, quindi, *mal condizionato* se  $\mu(A) \gg 1$ , mentre è *ben condizionato* se  $\mu(A) \simeq 1$ . Tale definizione richiede alcune considerazioni. Innanzitutto si deve osservare che, il valore dell'indice di condizionamento è, ovviamente, influenzato dalla norma matriciale adottata e, quindi, dal modo in cui si misura l'errore. Si osserva, a tal proposito, che, posto  $\mu_\alpha(A) = \|A\|_\alpha \|A^{-1}\|_\alpha$  con  $\alpha = 1, 2, \infty$ , sussitono le seguenti relazioni:

$$\frac{1}{n} \mu_2(A) \leq \mu_1(A) \leq n \mu_2(A)$$

$$\frac{1}{n} \mu_\infty(A) \leq \mu_2(A) \leq n \mu_\infty(A)$$

$$\frac{1}{n^2} \mu_1(A) \leq \mu_\infty(A) \leq n^2 \mu_1(A)$$

da cui si ricava che, se  $n$  è grande (ad esempio  $n = 10^6$ ) e  $\mu_\infty(A) = \mathcal{O}(n)$ , il sistema risulterà mal condizionato utilizzando  $\mu_\infty(A)$ , mentre potrebbe risultare ben condizionato utilizzando  $\mu_2(A)$ ; viceversa, però, se  $n$  è piccolo rispetto a  $\mu(A)$ , si verifica che, se  $\|A\| \|A^{-1}\|$  è grande in una norma, lo sarà in tutte le altre. Per tale motivo si è soliti anche definire mal condizionato un sistema di equazioni in cui è almeno  $\mu(A) > \mathcal{O}(n)$ .

L'ipotesi che il solo termine noto sia perturbato, è, in realtà, poco realistica, dato che la stessa matrice  $A$  può essere affetta da errori (in generale è sempre affetta almeno dagli errori di round-off di rappresentazione). È tuttavia possibile ottenere una relazione per il condizionamento anche nel caso più generale, in cui si verifica che l'amplificazione dell'errore presente sui dati continua ad essere influenzata, in modo essenziale, dall'indice di condizionamento  $\mu(A)$ , con  $A$  matrice dei coefficienti del sistema. Supponendo, allora, che i dati costituiti dai coefficienti e dal vettore dei termini noti, siano affetti da errore, per studiare il caso generale, è necessario dimostrare il seguente:

**Lemma 2.1.** *Sia  $\|\cdot\|$  una norma matriciale submoltiplicativa (cioè  $\|AB\| \leq \|A\| \|B\|$ ) e compatibile con una norma vettoriale (cioè  $\|Ax\| \leq \|A\| \|x\| \quad \forall x \neq 0$ ). Se una matrice  $A$  è tale che  $\|A\| < 1$ , si ha:*

1.  $I + A$  è non singolare;
2.  $(1 + \|A\|)^{-1} \leq \|(I + A)^{-1}\| \leq (1 - \|A\|)^{-1}$

**Dimostrazione** Per dimostrare la 1. si consideri un vettore  $x \neq 0$ . Allora si ha:

$$\begin{aligned} \|(I + A)x\| &= \|x + Ax\| \geq \|x\| - \|Ax\| \geq \\ \|x\| - \|x\| \|A\| &= \|x\| (1 - \|A\|) > 0 \end{aligned}$$

Ciò vuol dire che il vettore  $(I + A)x \neq 0 \quad \forall x \neq 0$ , e quindi il sistema di equazioni  $(I + A)x = 0$  ammette solo la soluzione banale e quindi la matrice  $(I + A)$  è non singolare.

Per dimostrare la 2. si osservi che:

$$1 = \|I\| = \|(I + A)(I + A)^{-1}\| \leq \|I + A\| \cdot \|(I + A)^{-1}\| \leq (1 + \|A\|) \cdot \|(I + A)^{-1}\|$$

da cui si ha:

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

e quindi la prima diseguaglianza è dimostrata. Inoltre si ha:

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

da cui:

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

Passando alle norme:

$$\|(I + A)^{-1}\| = \|I - A(I + A)^{-1}\| \leq 1 + \|A\| \cdot \|(I + A)^{-1}\|$$

e quindi:

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

Ma, poiché l'ipotesi  $\|A\| \leq 1 \implies (1 - \|A\|) > 0$ , si ha:

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

e quindi anche la seconda diseguaglianza della 2. è dimostrata. ■

Si è ora in grado di dimostrare il teorema seguente, che fornisce una stima dell'indice di condizionamento relativo, per un sistema lineare.

**Teorema 2.4. [Teorema del condizionamento]**

Sia  $\|\cdot\|$  una norma matriciale submoltiplicativa (cioè  $\|AB\| \leq \|A\|\|B\|$ ) e compatibile con una norma vettoriale (cioè  $\|Ax\| \leq \|A\|\|x\| \quad \forall x \neq 0$ ). Sia inoltre il sistema  $Ax = b$  (con  $A$  non singolare) e si consideri il sistema perturbato:

$$(A + \Delta A)(x + \Delta x) = (b + \Delta b) \quad (2.28)$$

Se

$$\|\Delta A\| < \frac{1}{\|A^{-1}\|} \quad (2.29)$$

posto  $\mu(A) = \|A\|\|A^{-1}\|$  si ha:

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\mu(A)}{1 - \mu(A)\frac{\|\Delta A\|}{\|A\|}} \left( \frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|} \right) \quad (2.30)$$

**Dimostrazione** Posto  $C = A^{-1}\Delta A$  si ha:

$$\|C\| = \|A^{-1}\Delta A\| \leq \|A^{-1}\| \cdot \|\Delta A\| < 1$$

per cui applicando il Lemma alla matrice  $C$  si ha che la matrice  $(I + C) = (I + A^{-1}\Delta A)$  è non singolare. Poiché inoltre  $(A + \Delta A) = A(I + A^{-1}\Delta A)$  si ha che anche  $(A + \Delta A)$  è non singolare. Dal lemma si ha anche che:

$$\|(I + C)^{-1}\| = \|(I + A^{-1}\Delta A)^{-1}\| \leq \frac{1}{1 - \|A^{-1}\Delta A\|} \leq \frac{1}{1 - \|A^{-1}\| \cdot \|\Delta A\|} \quad (2.31)$$

Moltiplicando quindi per  $A^{-1}$  il sistema perturbato (2.28) si ha:

$$A^{-1}(A + \Delta A)(x + \Delta x) = (I + A^{-1}\Delta A)(x + \Delta x) = A^{-1}b + A^{-1}\Delta b$$

da cui:

$$(I + A^{-1}\Delta A)x + (I + A^{-1}\Delta A)\Delta x = x + A^{-1}\Delta b$$

e quindi:

$$(I + A^{-1}\Delta A)\Delta x = x + A^{-1}\Delta b - x - A^{-1}\Delta Ax$$

Isolando a primo membro  $\Delta x$  e passando alle norme si ha:

$$\|\Delta x\| \leq \|(I + A^{-1}\Delta A)^{-1}\| \cdot \|A^{-1}\| (\|\Delta b\| + \|\Delta A\| \cdot \|x\|)$$

da cui, utilizzando anche la (2.31), si ha

$$\|\Delta x\| \leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\| \cdot \|\Delta A\|} (\|\Delta b\| + \|\Delta A\| \cdot \|x\|)$$

e, dunque,

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\| \cdot \|\Delta A\|} \left( \|\Delta A\| + \frac{\|\Delta b\|}{\|x\|} \right)$$

Ma, poiché da  $Ax = b$  si ha  $\|x\| \geq \|b\|/\|A\|$ , dalla precedente espressione, moltiplicando e dividendo per  $\|A\|$ , si ottiene:

$$\frac{\|\Delta x\|}{\|x\|} \leq \frac{\|A^{-1}\| \cdot \|A\|}{1 - \|A^{-1}\| \cdot \|\Delta A\|} \left( \frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|} \right)$$

e quindi, posto  $\mu(A) = \|A\| \cdot \|A^{-1}\|$ , si ha la tesi. ■

Si osservi che da  $\|A\| \|A^{-1}\| \geq 1$ , l'ipotesi (2.29), implica che  $\|\Delta A\| \leq \|A\|$ , la quale è una ipotesi realistica, in quanto afferma che l'ordine di grandezza delle perturbazioni è inferiore all'ordine di grandezza degli elementi di  $A$ . Se cosí non fosse le perturbazioni altererebbero completamente i dati. Inoltre, la stessa ipotesi (2.29) garantisce che:

$$0 < \mu(A) \frac{\|\Delta A\|}{\|A\|} < 1$$

e quindi il denominatore al secondo membro della (2.30) sarà sempre strettamente positivo.

♣ **Esempio 2.29.** Considerata la matrice dei coefficienti del sistema nell'esempio 2.26

$$A = \begin{pmatrix} 1. & 1. \\ 1. & 1.0001 \end{pmatrix}$$

si ha che:

$$A^{-1} = \begin{pmatrix} 10001 & -10000 \\ -10000 & 10000 \end{pmatrix}$$

e quindi:

$$\|A\|_\infty = 2.0001 \text{ e } \|A^{-1}\|_\infty = 20001$$

da cui:

$$\mu(A) \simeq 4 \times 10^4$$

ed il sistema (2.19), come visto, è mal condizionato.

Si osservi inoltre, che nel sistema aritmetico floating-point considerato nell'esempio 2.26, la matrice  $A$  diventa

$$A + \Delta A = \begin{pmatrix} 1. & 1. \\ 1. & 1. \end{pmatrix}$$

che è una matrice singolare. Ciò sembra indicare che una matrice mal condizionata è una matrice “*quasi singolare*”, nel senso che una piccola perturbazione sui suoi elementi conduce ad una matrice singolare. Se si calcola ora la “*distanza*” della matrice  $A$  dalla matrice singolare  $A + \Delta A$ , ovvero l’errore relativo  $\frac{\|\Delta A\|}{\|A\|}$ , si ha che:

$$\frac{\|\Delta A\|_\infty}{\|A\|_\infty} = \frac{0.0001}{2.0001} \simeq 5 \times 10^{-5}$$

e, d’altra parte,

$$\frac{1}{\mu(A)} \simeq 2.5 \times 10^{-5}$$

♣

In generale,  $1/\mu(A)$  è una stima della distanza della matrice  $A$  dalla matrice singolare più vicina, cioè:

$$\frac{1}{\mu(A)} = \min \left\{ \frac{\|A - B\|}{\|A\|} : B \text{ singolare} \right\}$$

Pertanto quanto più è grande l’indice di condizionamento  $\mu(A)$  tanto più la matrice  $A$  è vicina (in norma) ad una matrice singolare, con l’assunzione che una matrice singolare ha indice di condizionamento  $\mu(A) = \infty$ .

♣ **Esempio 2.30.** Si consideri la matrice  $A$  con la sua inversa:

$$A = \begin{pmatrix} 10^{-1} & 0 & \cdots & 0 \\ 0 & 10^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 10^{-1} \end{pmatrix} \implies A^{-1} = \begin{pmatrix} 10 & 0 & \cdots & 0 \\ 0 & 10 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 10 \end{pmatrix}$$

Tale matrice ha determinante  $\det(A) = 10^{-n}$  e l’indice di condizionamento è  $\mu(A) = 1$  e quindi è ben condizionata.

♣

In generale non c’è alcun legame tra indice di condizionamento e determinante di una matrice, nel senso che, avendo posto  $\mu(A) = \infty$  per una matrice singolare (che ha  $\det(A) = 0$ ), si potrebbe essere portati a pensare che un determinante piccolo sia caratteristico delle matrici mal condizionate e che il determinante di una matrice sia legato, in qualche senso, all’ inverso dell’indice di condizionamento.

♣ **Esempio 2.31.** Si considerino i punti  $x_i = i$  per  $i = 1, 2, \dots, 10$  e si consideri la matrice di Vandermonde:

$$V = \begin{pmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ 1 & x_2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^{n-1} \end{pmatrix} = \begin{pmatrix} 1 & x_1 & \cdots & x_1^9 \\ 1 & x_2 & \cdots & x_2^9 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{10} & \cdots & x_{10}^9 \end{pmatrix}$$

È noto che una matrice di Vandermonde è malcondizionata (in questo caso è  $\mu(V) = 3.3064 \times 10^{12}$ ), mentre il determinante è  $\det(V) = +1.8... \times 10^{21}$ .



Stabilito che  $\mu(A)$  permette di sapere di quanto l'errore nei dati si amplifica sulla soluzione, il problema che si pone, assegnata una matrice  $A$ , è calcolarne l'indice di condizionamento. Il calcolo di  $\mu(A)$  coinvolge la valutazione di  $A^{-1}$ , che si è visto essere un problema di elevata complessità computazionale (o comunque di pari complessità rispetto alla risoluzione del sistema). Per tale motivo sono stati sviluppati algoritmi per stimare  $\mu(A)$ , che non richiedono il calcolo dell'inversa di  $A$ . Elementi di software matematico basati su tali algoritmi sono presenti nelle più importanti librerie di software matematico, quali *LAPACK*, *NAg* e *IMSL*.

## 2.8 Accuratezza della soluzione di un sistema di equazioni lineari

In questo paragrafo si vogliono approfondire alcuni aspetti relativi all'accuratezza della soluzione di sistemi di equazioni lineari di ordine  $n$  del tipo<sup>14</sup>:

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \cdots a_{1,n}x_n = b_1 \\ \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots a_{n,n}x_n = b_n \end{cases} \iff Ax = b \quad (2.32)$$

calcolata mediante fattorizzazione *LU* della matrice dei coefficienti,  $A$ , seguita dagli algoritmi di *forward substitution* e di *back substitution*.

Ricordiamo che la fattorizzazione *LU* e la successiva risoluzione del sistema triangolare inferiore  $Ly = b$  mediante la *forward substitution* sono equivalenti all'algoritmo di eliminazione di Gauss, nel senso che, posto  $A^{(0)} = A$  e  $b^{(0)} = b$ , l'algoritmo di eliminazione di Gauss genera una sequenza di sistemi equivalenti:

$$A^{(1)}x = b^{(1)}, \quad A^{(2)}x = b^{(2)}, \quad \dots \quad A^{(n-1)}x = b^{(n-1)}$$

tali che  $U = A^{(n-1)}$  mentre  $L$  è costituita da tutti i moltiplicatori calcolati dall'algoritmo di eliminazione di Gauss ( ponendo inoltre  $l_{i,i} = 1$ ,  $i = 1, \dots, n$  ). Si può osservare, inoltre, che  $b^{(n-1)}$  è la soluzione del sistema triangolare inferiore  $Ly = b$ .

---

<sup>14</sup>Nel seguito, tutti i sistemi di equazioni considerati sono ben condizionati tranne dove specificato diversamente.

Pertanto, al fine della valutazione dell'accuratezza della soluzione del sistema (2.32), analizzeremo la stabilità degli algoritmi di eliminazione di Gauss, di forward substitution e di back substitution, cioè studieremo come si propagano gli errori di round-off in tali algoritmi.

### 2.8.1 Analisi dell'errore di round-off nell'algoritmo di eliminazione di Gauss

♣ **Esempio 2.32.** La soluzione del sistema di equazioni lineari:

$$\begin{cases} 4.4409 \times 10^{-17}x_1 + x_2 = 1 \\ x_1 + x_2 = 2 \end{cases}$$

è:

$$x_1 = 1.000\,000\,000\,000\,000\,044\dots \quad x_2 = 0.999\,999\,999\,999\,999\,955\dots$$

ma la soluzione calcolata utilizzando un calcolatore che sfrutta la doppia precisione dell'aritmetica dello standard IEEE, attraverso

- fattorizzazione  $A = LU$  (mediante algoritmo di Gauss senza pivoting)
- risoluzione di  $Ly = b$  (mediante forward substitution)
- risoluzione di  $Ux = y$  (mediante backward substitution)

è:

$$\hat{x}_1 = 0 \quad \hat{x}_2 = 1$$

Tale soluzione è inaccettabile. È possibile osservare che l'errore relativo commesso è:

$$\frac{\|\Delta x\|_\infty}{\|x\|_\infty} = \frac{\max(|\hat{x}_1 - x_1|, |\hat{x}_2 - x_2|)}{\max(|x_1|, |x_2|)} \simeq 1$$



♣ **Esempio 2.33.** Si risolva il sistema di equazioni  $Ax = b$ , con:

$$A = \begin{pmatrix} 0.001 & 1 \\ 1 & 1 \end{pmatrix} \quad b = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \quad (2.33)$$

nel sistema aritmetico floating-point,  $\mathfrak{I}$ , caratterizzato dai parametri:

$$\mathfrak{I} = \{\beta = 10, \quad t = 3, \quad E_{min} = -6, \quad E_{max} = 6\}$$

il quale ha massima accuratezza relativa  $u = 0.5 \times 10^{-2}$ . La soluzione del sistema è:

$$x_1 = 1.0010\dots \quad x_2 = 0.999\dots$$

**Fattorizzazione LU.** L'algoritmo di eliminazione di Gauss per la fattorizzazione *LU* in questo caso esegue un solo passo. Eseguendo le operazioni nel sistema aritmetico considerato si ottiene:

- $\hat{m}_{2,1} = fl(a_{2,1}^{(0)}/a_{1,1}^{(0)}) = (.100 \times 10^1)/(.100 \times 10^{-2}) = .100 \times 10^4$
- $\hat{a}_{2,2}^{(1)} = fl(a_{2,2}^{(0)} - m_{2,1}a_{1,2}^{(0)}) = .100 \times 10^1 - (.100 \times 10^4) \times (.100 \times 10^1) = .100 \times 10^1 - .100 \times 10^4 = -.100 \times 10^4$

Si osservi che, nella sottrazione presente nel calcolo di  $\hat{a}_{2,2}^{(1)}$ , è stato commesso un errore relativo di round-off:

$$|\delta_s| = \frac{.100 \times 10^1}{.100 \times 10^4} = .100 \times 10^{-2}$$

Si ottiene in definitiva che:

$$\hat{L} = \begin{pmatrix} .100 \times 10^1 & 0 \\ .100 \times 10^4 & .100 \times 10^1 \end{pmatrix} \quad (2.34)$$

$$\hat{A}^{(1)} = \hat{U} = \begin{pmatrix} .100 \times 10^{-2} & .100 \times 10^1 \\ 0 & -.100 \times 10^4 \end{pmatrix}$$

**Forward substitution.** Applicando la forward substitution al sistema  $\hat{L}\hat{y} = b$  si ottiene:

- $\hat{y}_1 = b_1 = .100 \times 10^1$
- $\hat{y}_2 = fl(b_2 - m_{2,1}\hat{y}_1) = .200 \times 10^1 - (.100 \times 10^4) \times (.100 \times 10^1) = .200 \times 10^1 - .100 \times 10^4 = -.100 \times 10^4$

**Back substitution.** Infine applicando la back substitution al sistema  $\hat{U}x = \hat{y}$  si ottiene:

- $\hat{x}_2 = fl(\hat{y}_2/a_{2,2}^{(1)}) = (-.100 \times 10^4)/(-.100 \times 10^4) = 0.100 \times 10^1$
- $\hat{x}_1 = fl(\hat{y}_1 - a_{1,2}^{(1)}\hat{x}_2) = .100 \times 10^1 - (.100 \times 10^1) \times (0.100 \times 10^1) = 0.0$

La soluzione calcolata anche in questo caso è inaccettabile. L'errore relativo commesso è:

$$\frac{\|\Delta x\|_\infty}{\|x\|_\infty} = \frac{\max(|\hat{x}_1 - x_1|, |\hat{x}_2 - x_2|)}{\max(|x_1|, |x_2|)} \simeq 1$$

A questo punto è utile osservare che, relativamente all'esempio 2.33, la fase di fattorizzazione della matrice  $A$  ha generato le matrici  $\hat{L}$  e  $\hat{U}$  della (2.34) tali che:

$$\hat{L}\hat{U} = A' = \begin{pmatrix} 0.001 & 1 \\ 1 & 0 \end{pmatrix}$$

Il prodotto  $\hat{L}\hat{U}$  ricostruisce quindi una matrice  $A'$  che può essere vista come la matrice  $A$  in cui l'elemento  $a_{2,2}$  è stato modificato mediante una perturbazione dell'ordine  $\mathcal{O}(10^0)$ , cioè:

$$A' = \begin{pmatrix} 0.001 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0.001 & 1 \\ 1 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & e_{2,2} \end{pmatrix} = A + E \quad |E| = \mathcal{O}(10^0) \quad (2.35)$$

avendo indicato  $|E| = (|e_{i,j}|)_{i,j=1,\dots,n}$ . In definitiva, quindi, a causa del sistema aritmetico a precisione finita utilizzato, nella fase di fattorizzazione si è, di fatto, fattorizzata la matrice  $A'$  della (2.35) in luogo della matrice  $A$  della (2.33).



Supponiamo di risolvere un generico sistema di equazioni di ordine  $n = 2$ ,

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 = b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 = b_2 \end{cases} \iff Ax = b$$

in un sistema aritmetico a precisione finita,  $\mathfrak{I}$ . Supponiamo che i dati siano esattamente rappresentabili in  $\mathfrak{I}$ . Nella fase di fattorizzazione della matrice, si ha che:

$$\begin{aligned} \hat{m}_{2,1} &= fl(a_{2,1}^{(0)}/a_{1,1}^{(0)}) = a_{2,1}^{(0)}(1 + \delta_d)/a_{1,1}^{(0)} = m_{2,1}(1 + \delta_d) \\ \hat{a}_{2,2}^{(1)} &= (a_{2,2}^{(0)} - m_{2,1}(1 + \delta_d)a_{1,2}^{(0)}(1 + \delta_m))(1 + \delta_s) \end{aligned} \quad (2.36)$$

dove  $|\delta_d| \leq u$ ,  $|\delta_m| \leq u$  e  $|\delta_s| \leq u$  sono gli errori relativi di round-off commessi rispettivamente nella divisione, moltiplicazione e sottrazione. Dalla (2.36) si ha:

$$\begin{aligned} \hat{a}_{2,2}^{(1)} &= a_{2,2}^{(0)} - m_{2,1}a_{1,2}^{(0)} + a_{2,2}^{(0)}\delta_s - m_{2,1}a_{1,2}^{(0)}(\delta_d + \delta_m + \delta_s) + \mathcal{O}(u^2) = \\ &= a_{2,2}^{(1)} + a_{2,2}^{(0)}\delta_s - m_{2,1}a_{1,2}^{(0)}(\delta_d + \delta_m + \delta_s) + \mathcal{O}(u^2) \end{aligned}$$

Siano:

$$\widehat{L} = \begin{pmatrix} 1 & 0 \\ \hat{m}_{2,1} & 1 \end{pmatrix} \quad \widehat{U} = \begin{pmatrix} a_{1,1}^{(0)} & a_{1,2}^{(0)} \\ 0 & \hat{a}_{2,2}^{(1)} \end{pmatrix}$$

Se si effettua in aritmetica esatta il prodotto  $\widehat{L}\widehat{U} = A'$  si ottiene la matrice:

$$A' = \begin{pmatrix} a_{1,1}^{(0)} & a_{1,2}^{(0)} \\ a_{2,1}^{(0)}(1 + \delta_d) & a_{2,2}^{(0)} + e_{2,2} \end{pmatrix}$$

con:

$$e_{2,2} = a_{2,2}^{(0)}\delta_s - m_{2,1}a_{1,2}^{(0)}(\delta_m + \delta_s) + \mathcal{O}(u^2)$$

Si nota che il moltiplicatore  $m_{2,1}$  rappresenta un fattore di amplificazione per gli errori relativi di round-off  $\delta_d$ ,  $\delta_m$ ,  $\delta_s$  commessi nelle operazioni aritmetiche. Di fatto il prodotto  $\widehat{L}\widehat{U}$  ricostruisce una matrice  $A'$  che può essere molto distante dalla matrice  $A$ .

Nell'esempio 2.33 si è avuto  $\hat{m}_{2,1} = 0.1 \times 10^4$ , per cui l'errore relativo di round-off  $|\delta_s| \leq u = 0.5 \times 10^{-2}$  si è amplificato di circa 1000 volte.

Riguardando, dunque, le matrici  $\widehat{L}$  ed  $\widehat{U}$ , risultanti dalla fattorizzazione  $LU$  di  $A$ , come i fattori  $\widehat{L}$  ed  $\widehat{U}$ , calcolati in aritmetica a precisione infinita, della matrice perturbata,  $A + E$ , ovvero effettuando un'analisi della propagazione dell'errore di round-off all'indietro (**backward error analysis**), si ha il seguente risultato:

**Teorema 2.5.** Sia  $A$  una matrice di ordine  $n$ , i cui elementi si suppongono essere numeri macchina. Se ad ogni passo  $k$  della fattorizzazione  $LU$  si ha che  $a_{k,k}^{(k)} \neq 0$  (cioè non si incontrano pivot nulli), allora i fattori calcolati  $\widehat{L}$  e  $\widehat{U}$  soddisfano:

$$\widehat{L}\widehat{U} = A' = A + E, \quad \text{con } |E| \leq 3(n-1)u(|A| + |\widehat{L}||\widehat{U}|) + \mathcal{O}(u^2) \quad (2.37)$$

dove  $|E| = (|e_{i,j}|)_{i,j=1,\dots,n}$ .

**Dimostrazione** La dimostrazione procede per induzione su  $k$ . Poiché il teorema è ovviamente vero per  $k = 1$ , si supponga che la (2.37) sia vera per  $k - 1$ . Sia quindi la matrice:

$$A = \begin{pmatrix} \alpha & w^T \\ v & B \end{pmatrix}$$

dove  $B$  è una matrice di ordine  $k - 1$ ,  $v$  e  $w$  due vettori di ordine  $k - 1$  e  $\alpha$  uno scalare e se ne effettui la fattorizzazione  $LU$ . Allora, nel primo passo di tale fattorizzazione, si ha:

$$\widehat{z} = fl(v/\alpha) = \frac{v}{\alpha} + f \quad |f| \leq u \left| \frac{v}{\alpha} \right|$$

e

$$\widehat{A}_1 = fl(B - \widehat{z}w^T) = B - \widehat{z}w^T + F \quad |F| \leq 2u(|B| + |\widehat{z}||w|^T) + \mathcal{O}(u^2) \quad (2.38)$$

I passi successivi effettuano la fattorizzazione della matrice  $\widehat{A}_1$ , di ordine  $k - 1$ . Per l'ipotesi di induzione, per tale matrice vale la (2.37), cioè:

$$\begin{aligned} \widehat{L}_1\widehat{U}_1 &= \widehat{A}_1 + E_1 \\ |E_1| &\leq 3(k-2)u(|\widehat{A}_1| + |\widehat{L}_1||\widehat{U}_1|) + \mathcal{O}(u^2) \end{aligned}$$

Posto allora:

$$\widehat{L}\widehat{U} = \begin{pmatrix} 1 & 0 \\ \widehat{z} & \widehat{L}_1 \end{pmatrix} \begin{pmatrix} \alpha & w^T \\ 0 & \widehat{U}_1 \end{pmatrix} = A + E$$

dalla (2.38) si ha quindi che

$$|\widehat{A}_1| \leq (1+2u)(|B| + |\widehat{z}||w|^T) + \mathcal{O}(u^2)$$

e quindi usando le ipotesi di induzione sulla matrice  $\widehat{A}_1$  si ha:

$$|E_1 + F| \leq 3(k-1)u(|B| + |\widehat{z}||w|^T + |\widehat{L}_1||\widehat{U}_1|) + \mathcal{O}(u^2)$$

Poiché  $|\alpha f| \leq u|v|$  si ha:

$$|E| \leq 3(k-1)u \left[ \begin{pmatrix} |\alpha| & |w^T| \\ |v| & |B| \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ |\widehat{z}| & |\widehat{L}_1| \end{pmatrix} \begin{pmatrix} |\alpha| & |w^T| \\ 0 & |\widehat{U}_1| \end{pmatrix} \right] + \mathcal{O}(u^2)$$

cioè la tesi. ■

## 2.8.2 Analisi dell'errore di round-off nell'algoritmo di Gauss con pivoting

Analizzando la (2.37), si nota che la perturbazione ricondotta su  $A$  dipende da quanto grandi possono diventare gli elementi<sup>15</sup>  $|l_{i,k}|$  e  $|u_{k,j}|$ . Poichè la matrice  $L$  è costituita da tutti i moltiplicatori  $l_{i,k} = a_{i,k}/a_{k,k}$ , la presenza di elementi  $a_{k,k}$  piccoli rispetto agli elementi  $a_{i,k}$  della stessa colonna, comporta che  $|l_{i,k}| \gg 1$  e quindi la possibilità di grosse perturbazioni  $|e_{i,j}|$  nei dati.

La soluzione a tale problema deve avere quindi come obiettivo quello di rendere gli elementi  $|l_{i,k}| = |a_{i,k}|/|a_{k,k}| \leq 1$  in modo da non amplificare gli errori di round-off. In tal modo, sarà

$$|\hat{L}| \leq 1 \Rightarrow \|\hat{L}\|_\infty \leq n.$$

È noto che una possibile soluzione risiede nella tecnica del **pivoting (parziale o totale)** che consiste, ad ogni passo  $k$ , in una riorganizzazione delle righe e/o delle colonne di  $A$  in modo da avere l'elemento di massimo modulo sempre sulla diagonale. Si ricordi inoltre che l'utilizzo di tale tecnica comporta che ad essere fattorizzata sia la matrice  $A$ , opportunamente moltiplicata per una matrice di permutazione. Ad esempio, relativamente alla tecnica del pivoting parziale si ha:

$$PA = LU \Leftrightarrow A = P^T LU$$

dove  $P$  è una matrice di permutazione,  $P^T$  la sua trasposta ed è  $P^T P = I$ .

A questo punto è necessario effettuare nuovamente l'analisi dell'errore di round-off della fattorizzazione  $A = P^T LU$  ottenuta utilizzando la tecnica del pivoting parziale. A tal fine è utile notare che nella fase di pivoting, cioè nella fase dello scambio delle righe della matrice  $A$ , non vengono effettuate operazioni aritmetiche e quindi non vengono commessi errori di round-off. Il teorema che segue, la cui dimostrazione è analoga a quella del Teorema 2.5, fornisce l'analisi dell'errore di round-off richiesta:

**Teorema 2.6.** *Sia  $A$  una matrice di ordine  $n$ . Se ad ogni passo  $k$  della fattorizzazione  $LU$  con pivoting parziale si ha che  $a_{k,k}^{(k)} \neq 0$  (cioè non si incontrano pivot nulli), allora i fattori calcolati  $\hat{L}$  e  $\hat{U}$  soddisfano:*

$$\hat{L}\hat{U} = A + E, \quad \text{con} \quad |E| \leq 3(n-1)u(|A| + |\hat{P}^T||\hat{L}||\hat{U}|) + \mathcal{O}(u^2) \quad (2.39)$$

<sup>15</sup>Il fattore  $n-1$  non è di importanza significativa e può essere trascurato in questa analisi. Tale fattore è la conseguenza del fatto che nel Teorema 2.5 si sono utilizzati i valori assoluti degli errori di round-off  $|\delta|$  non tenendo conto che essi, essendo con uguale probabilità positivi e negativi, tendono a compensarsi a vicenda. Come sottolineato da Wilkinson lo scopo di tale analisi è di individuare le potenziali instabilità degli algoritmi per migliorarli e non di ottenere una precisa maggiorazione dell'errore di round-off.

dove  $|E| = (|e_{i,j}|)$ ,  $i, j = 1, \dots, n$  e  $P$  è la matrice di permutazione derivante dall'utilizzo del pivoting parziale nell'algoritmo di eliminazione di Gauss.

Dalla (2.39) si ricava che:

$$\frac{|a_{i,j} - a'_{i,j}|}{|a_{i,j}|} = \frac{|e_{i,j}|}{|a_{i,j}|} \leq 3(n-1)u \left( 1 + \frac{\sum_{p_k=1}^n |\hat{l}_{i,p_k}| |\hat{u}_{p_k,j}|}{|a_{i,j}|} \right) + \mathcal{O}(u^2)$$

dove  $p_k = 1, \dots, n$  rappresenta una permutazione degli indici  $k = 1, \dots, n$ . Inoltre, poiché  $\|\hat{P}^T\| = 1$  e  $\|\hat{L}\| \leq n$ , si determina una stima, in norma, della perturbazione relativa:

$$\frac{\|E\|_\infty}{\|A\|_\infty} \leq 3(n-1)u \left( 1 + n \frac{\|\hat{U}\|_\infty}{\|A\|_\infty} \right) + \mathcal{O}(u^2) \quad (2.40)$$

che rappresenta una stima della perturbazione relativa sulla matrice  $A$ , nel caso di pivoting parziale.

A questo punto è opportuno chiedersi quanto grande possa diventare il rapporto  $\|\hat{U}\|_\infty/\|A\|_\infty$  presente nella (2.40), potendo rappresentare, anch'esso, un fattore di amplificazione per gli errori di round-off.

♣ **Esempio 2.34.** Si risolva mediante la fattorizzazione  $A = LU$  con pivoting parziale e gli algoritmi di *forward substitution* e *back substitution* il sistema di equazioni:

$$Ax = b \quad \text{con } A = \begin{cases} -1 & \text{se } i > j \\ 1 & \text{se } i = j \text{ o } j = n \\ 0 & \text{altrimenti} \end{cases} \quad \text{e } b_i = \sum_{j=1}^n a_{i,j}$$

La soluzione di tale sistema è  $x_i = 1 \quad i = 1, \dots, n$ , ma, per un sistema di ordine  $n = 100$ , si ha che la soluzione calcolata con un calcolatore con aritmetica standard IEEE in doppia precisione è:

$$x_i = 1 \quad i = 1, \dots, 53 \quad x_i = 0 \quad i = 54, \dots, 99 \quad x_{100} = 1$$

con un errore relativo  $\|\Delta x\|/\|x\| = 1$ . La soluzione è quindi inaccettabile.

Per capire il perché di tale inaccuratezza, nonostante l'utilizzo della tecnica del pivoting parziale, si osservi che la matrice  $U$  che si ottiene nella fase di fattorizzazione è la seguente:

$$U = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 1 \\ 0 & 1 & 0 & \cdots & 2 \\ 0 & 0 & 1 & \cdots & 2^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 2^{n-1} \end{pmatrix}$$

e quindi  $\|\hat{U}\|_\infty/\|A\|_\infty = \mathcal{O}(2^{n-1}/n)$ .



Dalla 2.40, segue che l'amplificazione dell'errore dipende dai coefficienti

$$|\hat{a}_{kj}^{(k)}| \quad k = 1, \dots, n-1 \quad j = k+1, \dots, n,$$

ovvero dalla crescita degli elementi di  $A$ , al generico passo  $k$ . A tale proposito si fornisce la seguente:

**Definizione 2.9. (Fattore di crescita)** *Relativamente all'algoritmo di eliminazione di Gauss, si definisce il fattore di crescita nel modo seguente:*

$$\rho = \max_{i,j,k} \frac{|\hat{a}_{i,j}^{(k)}|}{\|A\|_\infty}$$

Il fattore di crescita, quindi, misura quanto diventano grandi gli elementi  $|\hat{a}_{i,j}^{(k)}|$  nel corso dell'algoritmo di eliminazione di Gauss, rispetto alla matrice  $A$ . Poiché

$$\|\widehat{U}\|_\infty / \|A\|_\infty \leq n\rho,$$

dalla (2.40), tenendo conto del *fattore di crescita*, si ottiene:

$$\frac{\|E\|_\infty}{\|A\|_\infty} \leq 3(n-1)u(1+n^2\rho) + \mathcal{O}(u^2) \quad (2.41)$$

L'esempio 2.34 mostra che il *fattore di crescita* dà luogo, nell'algoritmo di eliminazione di Gauss, ad un fattore di amplificazione dell'errore di round-off,  $n^3\rho$  nella (2.41), che può ritenersi, in alcuni casi, non trascurabile.

♣ **Esempio 2.35.** Il seguente programma MATLAB calcola il massimo fattore di crescita ottenuto da 100 matrici casuali di ordine  $n = 100$ , i cui elementi sono tali che  $0 \leq a_{i,j} \leq 10$ :

```

n=100;    rho=0;

% ciclo sul numero di matrici
for nmat=1:100
    a=10*rand(n);    norma=norm(a,inf);    rapp=0;

    % fattorizzazione LU
    for k=1:n-1
        % pivoting
        [y,r] = max(abs(a(k:n,k)));  r=r+k-1;
        % scambio delle righe
        t=a(k,k:n);    a(k,k:n)=a(r,k:n);    a(r,k:n)=t;
        % calcolo moltiplicatori
    end
    rho=max(rho,norma);
end

```

```

a(k+1:n,k)=a(k+1:n,k)/a(k,k);
% trasformazione matrice attiva
a(k+1:n,k+1:n)=a(k+1:n,k+1:n)-a(k+1:n,k)*a(k,k+1:n);
%
rt=max(max(abs(a(k:n,k:n)))); 
if rt>rapp      rapp=rt;      end
end

% calcolo di rho
rapp=rapp/norma;
if rapp>rho      rho=rapp;      end

end
disp('massimo rho='); disp(rho)

```

Il risultato fornito è:

```

massimo rho=
0.1879

```



Anche se è possibile definire matrici con fattore di crescita  $\rho = \mathcal{O}(2^n)$ , nella pratica tale valore rimane limitato e non costituisce un effettivo fattore di amplificazione per gli errori di round-off nell'algoritmo di eliminazione di Gauss con pivoting parziale.

Tale risultato, associato alla considerazione che il costo computazionale del pivoting totale è sensibilmente maggiore di quello del pivoting parziale, permette di concludere che l'algoritmo di eliminazione di Gauss con pivoting parziale è un algoritmo praticamente stabile e può essere utilizzato nello sviluppo di software matematico accurato.

### 2.8.3 Analisi dell'errore di round-off nella forward e back substitution

Per completare l'analisi dell'accuratezza della soluzione di un sistema di equazioni lineari  $Ax = b$  è opportuno effettuare la **backward error analysis** anche degli algoritmi di *forward substitution* e di *back substitution*.

In effetti, eseguita la fattorizzazione  $LU$  della matrice dei coefficienti, la soluzione  $\tilde{x}$ , in  $\mathfrak{S}$ , del sistema

$$\widehat{L}\widehat{U}x = (A + E)x = b$$

si ottiene risolvendo i sistemi triangolari,

$$\widehat{L}y = b \quad \text{e} \quad \widehat{U}x = \widehat{y}$$

mediante *forward* e *back substitution*, rispettivamente.

Analizziamo, dunque, dapprima la propagazione degli errori di round-off introdotti dalle operazioni eseguite nella risoluzione di un sistema di equazioni lineari, triangolare inferiore,  $Ly = b$ . Utilizzando la **backward error analysis**, è possibile ricondurre l'errore introdotto nelle operazioni, sulla matrice dei coefficienti. In tal modo  $\hat{y}$  può interpretarsi come la soluzione esatta di un sistema triangolare inferiore, con matrice dei coefficienti perturbata,  $\hat{L} = L + F$ .

Analogamente, nella risoluzione di un sistema triangolare superiore, del tipo  $Ux = y$ , è possibile ricondurre l'errore introdotto nelle operazioni, sulla matrice dei coefficienti. In tal modo  $\tilde{x}$  può interpretarsi come la soluzione esatta di un sistema triangolare superiore, con matrice dei coefficienti perturbata,  $\hat{U} = U + H$ .

Relativamente alla stima della perturbazione ricondotta sugli elementi delle matrici dei coefficienti,  $L$  ed  $U$ , di opportuni sistemi triangolari, inferiori e superiori rispettivamente, valgono i seguenti Teoremi.

**Teorema 2.7.** *Sia  $L$  una matrice triangolare inferiore di ordine  $n$ . Se ad ogni passo,  $i$ , dell'algoritmo di forward substitution si ha  $l_{i,i} \neq 0$ , allora la soluzione calcolata  $\hat{y}$ , di un sistema del tipo  $Ly = b$ , è la soluzione esatta del sistema perturbato:*

$$(L + F)y = b, \quad \text{con} \quad |F| \leq nu|L| + \mathcal{O}(u^2) \quad (2.42)$$

**Dimostrazione** In generale, la componente  $y_i$  viene calcolata mediante:

$$y_i = \frac{b_i - \sum_{j=1}^{i-1} l_{i,j}y_j}{l_{i,i}} \quad i = 1, \dots, n \quad (2.43)$$

Relativamente alla (2.43) si ha che:

$$\hat{y}_1 = \frac{b_1}{l_{1,1}(1 + \delta_d)} \Rightarrow l_{1,1}(1 + \delta_d)\hat{y}_1 = b_1 \quad |\delta_d| \leq u$$

con  $\delta_d$  errore di round-off commesso nella divisione. Il calcolo di  $y_1$  avviene quindi come se l'elemento  $l_{1,1}$  fosse stato perturbato della quantità:

$$|e_{1,1}| = |l_{1,1}\delta_d| \leq |l_{1,1}|u$$

Proseguendo nella *forward substitution* si ha:

$$\hat{y}_2 = \frac{b_2 - l_{2,1}\hat{y}_1(1 + \delta_m)}{l_{2,2}(1 + \delta_d)(1 + \delta_s)} \Rightarrow l_{2,1}(1 + \delta_m)\hat{y}_1 + l_{2,2}(1 + \delta_s)(1 + \delta_d)\hat{y}_2 = b_2$$

con  $\delta_s$  e  $\delta_m$  rispettivamente errori di round-off nella sottrazione e nella moltiplicazione. Il calcolo di  $y_2$  avviene quindi come se  $l_{2,1}$  e  $l_{2,2}$  fossero stati perturbati rispettivamente da:

$$|e_{2,1}| = |l_{2,1}\delta_m| \leq |l_{2,1}|u \quad e \quad |e_{2,2}| = |l_{2,2}(\delta_d + \delta_s + \delta_d\delta_s)| \leq 2|l_{2,2}|u + \mathcal{O}(u^2)$$

Per la componente  $y_i$  si osservi che  $\sum_{j=1}^{i-1} l_{i,j}y_j$  è il prodotto scalare tra le prime  $i-1$  componenti della  $i$ -esima riga di  $L$  e del vettore  $\hat{y}$ . Quindi si ha:

$$fl \left( \sum_{j=1}^{i-1} l_{i,j}y_j \right) = l_{i,1}\hat{y}_1(1 + \theta_{i-1}) + l_{i,2}\hat{y}_2(1 + \theta_{i-2}) + \dots + l_{i,i-1}\hat{y}_{i-1}(1 + \theta_1)$$

con

$$|\theta_i| \leq iu + \mathcal{O}(u^2)$$

da cui si ha:

$$\hat{y}_i = \frac{b_i - (l_{i,1}\hat{y}_1(1 + \theta_{i-1}) + \dots + l_{i,i-1}\hat{y}_{i-1}(1 + \theta_1))}{l_{i,i}(1 + \delta_s)(1 + \delta_d)} \implies$$

$$l_{i,1}\hat{y}_1(1 + \theta_{i-1}) + \dots + l_{i,i}(1 + \delta_d)(1 + \delta_s)\hat{y}_i = b_i$$

cioè  $y_i$  viene calcolato come se gli elementi  $l_{i,j}$  fossero stati perturbati da:

$$\begin{aligned} |e_{i,1}| &= |l_{i,1}\theta_{i-1}| && \leq (i-1)|l_{i,1}|u + \mathcal{O}(u^2) \\ |e_{i,j}| &= |l_{i,j}\theta_{i-j}| && \leq (i-j)|l_{i,j}|u + \mathcal{O}(u^2) \quad j = 2, \dots, i-1 \\ |e_{i,i}| &= |l_{i,i}(\delta_d + \delta_s + \delta_d\delta_s)| && \leq 2|l_{i,i}|u + \mathcal{O}(u^2) \end{aligned}$$

da cui la tesi. ■

**Teorema 2.8.** *Sia  $U$  una matrice triangolare superiore di ordine  $n$ . Se ad ogni passo,  $i$ , dell'algoritmo di back substitution si ha  $u_{i,i} \neq 0$ , allora la soluzione calcolata  $\hat{x}$ , di un sistema del tipo  $Ux = y$ , è la soluzione esatta del sistema perturbato:*

$$(U + H)x = y, \quad \text{con} \quad |H| \leq nu|U| + \mathcal{O}(u^2)$$

(2.44)

**Dimostrazione** Analoga a quella del Teorema 2.7. ■

In entrambi i teoremi, le limitazioni sulle matrici  $F$  ed  $H$  forniscono una stima della perturbazione introdotta sui dati.

Analizziamo, ad esempio, la (2.42). Si osserva che le perturbazioni  $f_{i,j}$  sono piccole

rispetto agli elementi della matrice  $l_{i,j}$ , in quanto:

$$\frac{|f_{i,j}|}{|l_{i,j}|} \leq nu + \mathcal{O}(u^2)$$

In definitiva l'algoritmo di *forward substitution* (così come quello di *back substitution*) può essere considerato stabile.

#### 2.8.4 Analisi dell'errore di round-off nella risoluzione di un sistema lineare mediante algoritmo di Gauss

**Teorema 2.9.** *Sia  $A \in \mathbb{R}^{n \times n}$  e sia  $\tilde{x}$  la soluzione, in  $\mathfrak{S}$ , di un sistema di equazioni lineari  $Ax = b$ , calcolata mediante l'algoritmo di eliminazione di Gauss con pivoting parziale. Allora,  $\tilde{x}$  si può interpretare come la soluzione ottenuta, in aritmetica a precisione infinita, di un sistema con matrice dei coefficienti perturbata,  $A + \Delta A$ :*

$$(A + \Delta A)\tilde{x} = b \quad \text{con} \quad \Delta A = E + G$$

tale che

$$\begin{aligned} \|\Delta A\|_\infty &\leq 3(n-1)u(\|A\|_\infty + \|\widehat{L}\|_\infty \|\widehat{U}\|_\infty) + 2nu\|\widehat{L}\|_\infty \|\widehat{U}\|_\infty + \mathcal{O}(u^2) \leq \\ &\leq 3nu\|A\|_\infty + 3n^2u\|\widehat{U}\|_\infty + 2n^2u\|\widehat{U}\|_\infty + \mathcal{O}(u^2) \leq \\ &\leq 3nu\|A\|_\infty + 5n^2u\|\widehat{U}\|_\infty + \mathcal{O}(u^2) \end{aligned} \tag{2.45}$$

**Dimostrazione** Supponiamo di risolvere un sistema lineare,

$$Ax = b,$$

applicando l'algoritmo di eliminazione di Gauss, ovvero l'algoritmo di fattorizzazione  $LU$ , seguito dagli algoritmi di forward e back substitution.

Osserviamo che, partendo dalla fattorizzazione  $\widehat{L}\widehat{U}$  di  $A + E$ , calcolata in un sistema aritmetico a precisione finita, si ha anche che:

$$(\widehat{L} + F)\widehat{y} = b \quad \text{e} \quad (\widehat{U} + H)\tilde{x} = \widehat{y}$$

per cui:

$$(\widehat{U} + H)\tilde{x} = (\widehat{L} + F)^{-1}b \Rightarrow (\widehat{L} + F)(\widehat{U} + H)\tilde{x} = b$$

e, ancora,

$$(\widehat{L}\widehat{U} + \widehat{L}H + F\widehat{U} + FH)\tilde{x} = b \Rightarrow (A + E + G)\tilde{x} = b,$$

dove

$$\widehat{L}\widehat{U} = A + E \quad \text{e} \quad G = \widehat{L}H + F\widehat{U} + FH$$

Attraverso le stime (2.37), (2.42) e (2.44), si stima, allora, la perturbazione  $\Delta A$ . In particolare:

Dalla fattorizzazione LU:

$$|E| \leq 3(n-1)u(|A| + |\widehat{L}| \cdot |\widehat{U}|) + \mathcal{O}(u^2)$$

Dalla forward substitution:

$$|F| \leq nu|\widehat{L}| + \mathcal{O}(u^2)$$

Dalla back substitution:

$$|H| \leq nu|\widehat{U}| + \mathcal{O}(u^2)$$

da cui

$$\begin{aligned} |\Delta A| &= |E + G| \leq |E| + |G| \leq \\ &\leq 3(n-1)u(|A| + |\widehat{L}| |\widehat{U}|) + 2nu|\widehat{L}||\widehat{U}| + \mathcal{O}(u^2) \end{aligned} \tag{2.46}$$

Applicando la tecnica del **pivoting parziale** e tenendo conto che

$$\|\widehat{L}\|_\infty \leq n,$$

partendo dalla (2.46) e passando alla norma infinito, si ha:

$$\begin{aligned} \|\Delta A\|_\infty &\leq 3(n-1)u(\|A\|_\infty + \|\widehat{L}\|_\infty \|\widehat{U}\|_\infty) + 2nu\|\widehat{L}\|_\infty \|\widehat{U}\|_\infty + \mathcal{O}(u^2) \leq \\ &\leq 3nu\|A\|_\infty + 3n^2u\|\widehat{U}\|_\infty + 2n^2u\|\widehat{U}\|_\infty + \mathcal{O}(u^2) \leq \\ &\leq 3nu\|A\|_\infty + 5n^2u\|\widehat{U}\|_\infty + \mathcal{O}(u^2) \end{aligned} \tag{2.47}$$

che fornisce la stima, in norma, della perturbazione assoluta  $\Delta A$ . ■

Più in generale, indicando con  $\rho_n$ , il fattore di crescita dell'errore di roundoff, introdotto applicando l'algoritmo di eliminazione di Gauss ad una matrice  $A \in \Re^{n \times n}$ , vale il risultato seguente, dovuto a Wilkinson [11]:

**Teorema 2.10.** *Sia  $A \in \Re^{n \times n}$  e sia  $\widehat{x}$  la soluzione di un sistema di equazioni lineari  $Ax = b$ , calcolata mediante l'algoritmo di eliminazione di Gauss con pivoting parziale. Allora:*

$$(A + \Delta A)\widehat{x} = b, \quad \|\Delta A\|_\infty \leq n^2\gamma_{3n}\rho_n\|A\|_\infty,$$

$$\text{con } \gamma_{3n} = \frac{3nu}{1-3nu}.$$

Analogamente, per l'errore relativo si determina la stima della perturbazione relativa sugli elementi di  $A$ :

$$\frac{\|\Delta A\|_\infty}{\|A\|_\infty} \leq 3nu + 5n^2u \frac{\|\widehat{U}\|_\infty}{\|A\|_\infty} + \mathcal{O}(u^2)$$

cioè, tenendo conto del fattore di crescita  $\rho$ , dunque, che

$$\frac{\|\widehat{U}\|_\infty}{\|A\|_\infty} \leq n\rho$$

si ha

$$\boxed{\frac{\|\Delta A\|_\infty}{\|A\|_\infty} \leq 3nu + 5n^3\rho_n u + \mathcal{O}(u^2)} \quad (2.48)$$

$n^3\rho_n$  rappresenta, dunque, il fattore di amplificazione dell'errore di roundoff introdotto nella risoluzione di un sistema di equazioni lineari, mediante algoritmo di eliminazione di Gauss con pivoting parziale. L'errore introdotto nella soluzione calcolata è, dunque, ricondotto ad una perturbazione sulla matrice dei coefficienti, e dal suo ordine di grandezza dipende la stabilità dell'algoritmo di eliminazione di Gauss, con pivoting (parziale o totale). Sostanzialmente si può affermare che **la stabilità dell'algoritmo di eliminazione di Gauss dipende dal fattore di crescita,  $\rho_n$ .**

### Stime del *fattore di crescita*

Per l'algoritmo di eliminazione di Gauss con **pivoting parziale** si dimostra che

$$\rho_n^p \leq 2^{n-1}, \quad (2.49)$$

avendo indicato con  $\rho_n^p$  il fattore di crescita dell'errore di roundoff, introdotto applicando l'algoritmo di eliminazione di Gauss con pivoting parziale, ad una matrice  $A \in \Re^{n \times n}$ ; si riporta, a tal proposito, il risultato seguente, dovuto ad Higham [11]:

**Teorema 2.11.** *Ogni matrice reale,  $A \in \Re^{n \times n}$ , per cui*

$$\rho_n^p(A) = 2^{n-1}$$

*è del tipo*

$$A = DM \begin{bmatrix} T & : & \alpha d \\ 0 & : & \end{bmatrix},$$

*dove  $D = diag(\pm 1)$ ,  $M$  è triangolare inferiore unitaria, con  $m_{ij} = -1$ , per  $i > j$ ,  $T$  è una matrice triangolare superiore, non singolare, arbitraria, di ordine  $n-1$ ,  $d = (1, 2, 4, \dots, 2^{n-1})^T$  ed  $\alpha$  è uno scalare tale che  $\alpha := |a_{1n}| = \max_{i,j} |a_{ij}|$ .*

Si osserva che, variando gli elementi  $m_{ij}$  ( $i > j$ ) ed il vettore  $d$ , nel Teorema 2.11, si possono costruire matrici per le quali  $\rho_n^p$  raggiunge un qualsiasi valore compreso tra 1 e  $2^{n-1}$ . Sebbene esistano matrici per cui  $\rho_n^p$  raggiunge valori elevati, il fattore di crescita si mantiene piccolo, nelle applicazioni, in maniera quasi costante. Wilkinson affermò, a tal proposito, che

[...] È nella nostra esperienza che una sostanziale crescita dell'ordine di grandezza degli elementi di una successione di matrici  $A_r$ , sia un evento estremamente non comune, persino con pivoting parziale... Nessun esempio, che sia sorto in maniera naturale, ha, nella mia esperienza, fornito un incremento per mezzo di un fattore grande quanto 16. ([16], pp. 213-214)

Per riassumere, sebbene ci siano matrici che ricorrono, nelle applicazioni, per le quali il pivoting parziale conduce ad un fattore di crescita moderatamente grande, o persino esponenzialmente grande, nelle applicazioni si individuano fattori di crescita quasi costantemente piccoli. *La spiegazione a questo fenomeno resta uno dei maggiori problemi irrisolti dell'Analisi Numerica.* Il miglior tentativo è dovuto a Trefethen e Schreiber [15], che svilupparono un modello statistico del *fattore di crescita medio*, sia per il pivoting parziale che totale. Tale modello è in accordo con i loro risultati empirici, secondo i quali, per diverse distribuzioni di matrici *random*, il fattore di crescita medio (normalizzato mediante la deviazione standard degli elementi della matrice iniziale) è vicino al valore  $n^{2/3}$  per il pivoting parziale e  $n^{1/2}$  per il pivoting totale (in corrispondenza di  $n \leq 1024$ ). Edelman realizzò esperimenti incisivi, dai quali si deduce che, per matrici random, generate dalla distribuzione normale  $\mathcal{N}(0, 1)$ , il fattore di crescita non normalizzato, per il pivoting parziale, cresce come  $n^{1/2}$  [6].

Riguardo al pivoting totale, Wilkinson [17] dimostrò che

$$\rho_n^t \leq n^{1/2} (2 \cdot 3^{1/2} \cdots n^{1/(n-1)})^{1/2} \sim cn^{1/2} n^{\frac{1}{4} \log n}, \quad (2.50)$$

avendo indicato con  $\rho_n^t$  il fattore di crescita relativo al pivoting totale, e che questo limite non è raggiungibile. Tale limite è una funzione che cresce molto più lentamente di  $2^{n-1}$ , sebbene possa raggiungere valori abbastanza elevati (ad esempio è 3570 per  $n = 100$ ). Analogamente al pivoting parziale, nelle applicazioni il fattore di crescita è, generalmente, piccolo. Wilkinson affermò che

[...] non è stata incontrata, nelle applicazioni, alcuna matrice per la quale  $\rho_1/\rho_n$  fosse grande quanto 8 [17]

e che

[...] non è stata ancora scoperta una matrice per la quale  $\rho_r^t > r$ . [16]

Cryer definì la funzione

$$g(n) = \sup_{A \in \Re^{n \times n}} \rho_n^t(A).$$

Per essa sono noti i seguenti risultati:

- $g(2) = 2$  (si prova banalmente);

- $g(3) = 2\frac{1}{4}$  (dovuto a Tornheim e Cohen);
- $g(4) = 4$  (dovuto a Cryer e Cohen);
- $g(5) < 5.005$  (dovuto a Cohen).

Inoltre, Cryer congettò che, per matrici reali,

$$\rho_n^t(A) \leq n.$$

Questa congettura ha dato luogo ad uno dei più famosi problemi ancora irrisolti in Analisi Numerica ed è stata affrontata, negli anni, da numerosi matematici. Infine, nel 1991 tale congettura fu dimostrata essere falsa. Nello stesso anno Gould [10] determinò una matrice di dimensione  $13 \times 13$ , per la quale il fattore di crescita è 13.0205, nella doppia precisione del sistema aritmetico floating point dello standard IEEE. Un controsenso più incisivo per negare la congettura, è costituito da una matrice di ordine 25, per la quale risulta  $\rho_{25}^t = 32.986341$ . Restano aperte, comunque, problematiche interessanti, come, ad esempio, la determinazione del limite

$$\lim_{n \rightarrow \infty} \frac{g(n)}{n}$$

e la valutazione di  $\rho_n^t$  per matrici di Hadamard.

Concludendo, le stime (2.49) e (2.50) forniscono una limitazione superiore del fattore di crescita dell'errore relativo su  $A$ , per il pivoting parziale e totale, rispettivamente, ma, sperimentalmente, si verifica che, in media, il fattore di crescita si mantiene notevolmente inferiore a  $\rho_n^p$ , per il pivoting parziale, così come a  $\rho_n^t$ , per il pivoting totale.

Inoltre, per il **pivoting totale**:

- non esistono matrici per cui il fattore di crescita è uguale a  $\rho_n^t$ ;
- solitamente il fattore di crescita è  $\rho_n^t < n$ , anche se esistono matrici per cui  $\rho_n^t > n$ .

Per il **pivoting parziale**:

- per ogni  $\rho \in [1, 2^{n-1}]$  esiste una matrice per cui il fattore di crescita è uguale a  $\rho$ ;
- nonostante il fattore di crescita possa essere  $\rho = 2^{n-1}$ , solitamente esso risulta di dimensioni *piccole*.

Infine, si osserva che, utilizzando il *pivoting parziale*, il fattore di crescita è al più il doppio rispetto al *pivoting totale*; considerando, poi, che il pivoting parziale richiede un numero di confronti minore, rispetto al pivoting totale, in generale conviene utilizzare la strategia del *pivoting parziale*.

## 2.9 Sistemi lineari con matrici strutturate

L'algoritmo di fattorizzazione *LU* è stato applicato a matrici generiche, intendendo con ciò che esso non tiene conto del fatto che la matrice possa avere particolari caratteristiche, ad esempio “*pochi*” elementi non nulli disposti secondo un certo schema, oppure elementi simmetrici, rispetto alla diagonale principale, uguali. I paragrafi che seguono sono dedicati alla risoluzione di sistemi lineari con matrice dei coefficienti “*speciale*”; si vedrà infatti come sia possibile sfruttare eventuali proprietà della matrice dei coefficienti per ridurre la complessità di tempo e di spazio richiesta da tale risoluzione. Si noti, a tal proposito, che risolvere un problema cercando di trarre vantaggio dalle particolari caratteristiche del problema stesso costituisce un principio generale dell’Analisi Numerica.

Nei due esempi che seguono sono presentati alcuni problemi descritti da sistemi lineari la cui matrice dei coefficienti ha particolari caratteristiche.

♣ **Esempio 2.36.** Si consideri una rete di resistori uguali tra loro, disposti come in Fig. 2.21. Si indichi con  $R$  la resistenza di ciascuno di essi e con  $v_k$  il potenziale nel  $k$ -mo nodo della rete e si supponga che  $v_0$  e  $v_{n+1}$  siano noti.

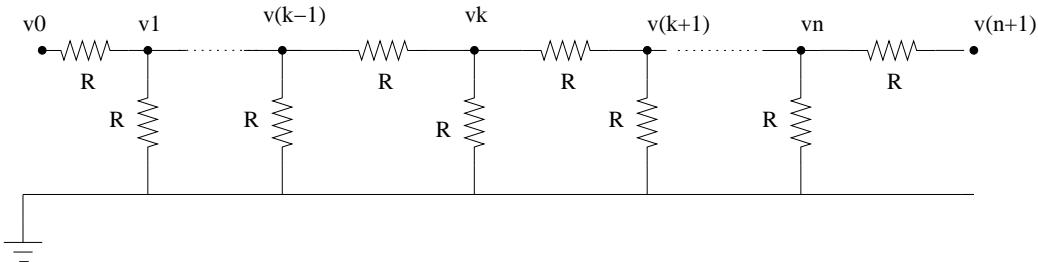


Figura 2.21: Rete di resistori.

Applicando la prima legge di Kirchoff e la legge di Ohm<sup>16</sup>, si ha, per ogni nodo  $k$ , la seguente equazione:

$$\frac{(v_{k-1} - v_k)}{R} - \frac{(v_k - v_{k+1})}{R} = \frac{v_k}{R},$$

ovvero

$$-v_{k-1} + 3v_k - v_{k+1} = 0. \quad (2.51)$$

L’equazione (2.51), scritta per  $k = 1, \dots, n$ , dà luogo al seguente sistema lineare nelle incognite  $v_k$ :

<sup>16</sup>La legge di Ohm afferma che l’intensità della corrente elettrica che percorre un filo metallico a temperatura costante è direttamente proporzionale alla differenza di potenziale esistente tra gli estremi del filo. Indicata con  $v_A - v_B$  la differenza di potenziale tra gli estremi  $A$  e  $B$  del filo e con  $I$  l’intensità della corrente che percorre il filo dal punto  $A$  al punto  $B$ , la legge di Ohm si può esprimere come  $I = (v_A - v_B)/R$ , dove la costante  $R$  è la resistenza del filo.

$$\begin{pmatrix} 3 & -1 & & & \\ -1 & 3 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 3 & -1 \\ & & & -1 & 3 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-1} \\ v_n \end{pmatrix} = \begin{pmatrix} v_0 \\ 0 \\ \vdots \\ 0 \\ v_{n+1} \end{pmatrix}. \quad (2.52)$$

La matrice del sistema (2.52) ha molti elementi nulli e gli elementi non nulli si trovano sulla diagonale principale e sulle due diagonali immediatamente al di sotto ed al di sopra di questa. Tale matrice è inoltre simmetrica, cioè, detto  $a_{i,j}$  il suo generico elemento di posto  $(i,j)$ , risulta  $a_{i,j} = a_{j,i}$ .  $\clubsuit$

**♣ Esempio 2.37.** Si consideri l'equazione differenziale ordinaria (equazione di convezione-diffusione stazionaria in una dimensione):

$$u \frac{dc}{dx}(x) - \alpha \frac{d^2c}{dx^2}(x) = 0, \quad x \in [0, 1], \quad (2.53)$$

con le condizioni al contorno:

$$c(0) = 0, \quad c(1) = 1. \quad (2.54)$$

Per risolvere numericamente il problema (2.53)-(2.54) si può procedere nel modo seguente. Posto  $\Delta x = 1/n$ , con  $n$  intero positivo, si considerino i punti

$$x_0, x_1, \dots, x_n \in [0, 1]$$

tali che

$$x_k = k\Delta x, \quad k = 0, \dots, n,$$

e si approssimino le derivate di  $c(x)$  in tali punti con le espressioni seguenti:<sup>17</sup>

$$\begin{aligned} \frac{dc}{dx}(x_k) &\simeq \frac{1}{2\Delta x} (c_{k+1} - c_{k-1}) + \frac{1}{6\Delta x} (c_{k-2} - 3c_{k-1} + 3c_k - c_{k+1}) \\ &= \frac{1}{\Delta x} \left( \frac{1}{6}c_{k-2} - c_{k-1} + \frac{1}{2}c_k + \frac{1}{3}c_{k+1} \right), \\ \frac{d^2c}{dx^2}(x_k) &\simeq \frac{1}{\Delta x^2} (c_{k-1} - 2c_k + c_{k+1}), \end{aligned}$$

dove  $c_k$  è un'approssimazione di  $c(x_k)$ . Sostituendo le espressioni precedenti nell'equazione (2.53), si ha:

$$\frac{u}{6\Delta x} c_{j-2} - \left( \frac{u}{\Delta x} + \frac{\alpha}{(\Delta x)^2} \right) c_{j-1} + \left( \frac{u}{2\Delta x} + \frac{2\alpha}{(\Delta x)^2} \right) c_j + \left( \frac{u}{3\Delta x} - \frac{\alpha}{(\Delta x)^2} \right) c_{j+1} = 0,$$

ovvero, posto  $R = u/\Delta x$ :

$$\frac{R}{6} c_{j-2} - (R+1) c_{j-1} + \left( \frac{R}{2} + 2 \right) c_j + \left( \frac{R}{3} - 1 \right) c_{j+1} = 0.$$

<sup>17</sup>Non ci si sofferma, in questa sede, su tali approssimazioni delle derivate. Ci si limita solo ad osservare che esse possono essere ottenute utilizzando opportunamente lo sviluppo in serie di Taylor della funzione  $c(x)$  e che sono effettivamente utilizzate nelle applicazioni (Si veda, ad esempio, [8]).

Posto  $c_{-1} = 0$  e tenuto conto delle condizioni (2.54), si perviene al seguente sistema lineare nelle incognite  $c_k$ ,  $k = 1, \dots, n-1$ :

$$\begin{pmatrix} \frac{R}{2} + 2 & \frac{R}{3} - 1 & & \\ R + 1 & \frac{R}{2} + 2 & \frac{R}{3} - 1 & \\ \frac{R}{6} & R + 1 & \frac{R}{2} + 2 & \frac{R}{3} - 1 \\ \ddots & \ddots & \ddots & \ddots \\ & \frac{R}{6} & R + 1 & \frac{R}{2} + 2 & \frac{R}{3} - 1 \\ & & \frac{R}{6} & R + 1 & \frac{R}{2} + 2 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}. \quad (2.55)$$

La matrice dei coefficienti del sistema (2.55) ha gli elementi non nulli disposti solo su quattro diagonali, la diagonale principale, le prime due inferiori e la prima superiore. ♦

Le matrici considerate negli esempi precedenti hanno gli elementi disposti secondo uno schema preciso e si dicono quindi *strutturate*. In particolare, nella matrice dell'esempio 2.36, gli elementi non nulli sono localizzati sulla diagonale principale e sulle due diagonali ad essa adiacenti; nella matrice dell'esempio 2.37, gli elementi non nulli appartengono alla diagonale principale e ad altre diagonali ad essa vicine.

### Definizione 2.10. (Matrice tridiagonale)

Una matrice  $A = (a_{i,j})$  si dice tridiagonale se

$$a_{i,j} = 0 \text{ per } |i - j| > 1.$$

### Definizione 2.11. (Matrice a banda)

Una matrice  $A = (a_{i,j})$  si dice a banda, con ampiezza di banda inferiore  $p$  ed ampiezza di banda superiore  $q$ , se

$$a_{i,j} = 0 \text{ per } i > j + p \text{ e per } j > i + q.$$

Una rappresentazione grafica di una matrice a banda è fornita in Fig. 2.22. Si osservi che una matrice tridiagonale è una particolare matrice a banda, con ampiezze di banda  $p = q = 1$ . Si osservi inoltre che la definizione di matrice a banda non implica che gli elementi della diagonale principale, delle  $p$  diagonali inferiori e delle  $q$  diagonali superiori siano tutti non nulli, ma solo che siano nulli gli elementi che non appartengono a queste diagonali.

Le matrici degli esempi 2.36 e 2.37 sono matrici a banda. La prima ha entrambe le ampiezze di banda uguali ad 1, ovvero è tridiagonale; la seconda ha ampiezza di banda inferiore uguale a 2 ed ampiezza di banda superiore uguale ad 1.

In maniera analoga alla matrice tridiagonale si definisce la matrice *pentadiagonale* ( $p = q = 2$ ), *eptadiagonale* ( $p = q = 3$ ), etc. Anche una matrice *diagonale* si può vedere come una particolare matrice a banda, in cui  $p = q = 0$ . Nel seguito si considerano matrici a banda quadrate, anche se le precedenti definizioni si applicano anche a matrici con un numero di righe diverso dal numero di colonne.

Sono matrici strutturate anche le matrici *triangolari*, i cui elementi non nulli sono appunto disposti secondo una struttura a triangolo. Tali matrici possono essere anche

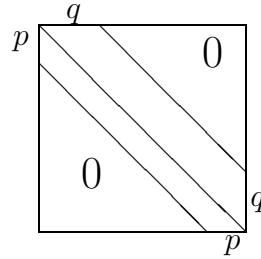


Figura 2.22: Matrice a banda.

a banda, con una delle due ampiezze di banda necessariamente nulla. In tal caso, si parla semplicemente di ampiezza di banda della matrice, facendo riferimento alla banda di ampiezza non nulla. Un esempio di matrice triangolare inferiore a banda ed uno di matrice triangolare superiore a banda è fornito di seguito.

♣ **Esempio 2.38.**

Si considerino le seguenti matrici:

$$L = \begin{pmatrix} 1 & & & \\ -2 & 4 & & \\ & 5 & 1 & \\ & & -7 & 2 \end{pmatrix}, \quad U = \begin{pmatrix} 5 & -3 & & \\ & 1 & -2 & \\ & & -4 & 1 \\ & & & 6 \end{pmatrix}.$$

Tali matrici, oltre ad essere triangolari, sono matrici a banda; la prima con ampiezza di banda (inferiore)  $p = 1$  e la seconda con ampiezza di banda (superiore)  $q = 1$ . ♣

In generale, si dà la seguente definizione:

**Definizione 2.12. (Matrice bidiagonale)**

Una matrice  $A = (a_{i,j})$  si dice bidiagonale inferiore se

$$a_{i,j} = 0 \text{ per } j > i \text{ e per } i > j + 1,$$

e si dice bidiagonale superiore se

$$a_{i,j} = 0 \text{ per } i > j \text{ e per } j > i + 1.$$

## 2.10 Memorizzazione di matrici strutturate

Memorizzare una matrice strutturata in un array bidimensionale non è in generale conveniente dal punto di vista della complessità di spazio, in quanto si memorizzano anche

elementi nulli la cui posizione è nota. Una memorizzazione efficiente deve tener conto della struttura della matrice, evitando di conservare informazioni inutili. D'altra parte, la struttura stessa della matrice suggerisce schemi di memorizzazione più appropriati.

Si inizi col considerare una matrice diagonale, di dimensione  $n \times n$ :

$$A = \begin{pmatrix} a_{1,1} & & & \\ & a_{2,2} & & \\ & & \ddots & \\ & & & a_{n,n} \end{pmatrix}.$$

Se si memorizza tale matrice in un array di dimensione  $n \times n$  si ha una complessità di spazio pari a  $n^2$ , anche se gli elementi che bisogna effettivamente conservare, cioè quelli della diagonale, sono solo  $n$ . Tale osservazione conduce in maniera naturale ad utilizzare un array monodimensionale,  $AD$ , di lunghezza  $n$ , per memorizzare solo tali elementi:

$$AD = (a_{1,1}, a_{2,2}, \dots, a_{n,n}),$$

ovvero

$$AD(i) = a_{i,i}, \quad i = 1, \dots, n.$$

La complessità di spazio si riduce, in tal modo, di un ordine di grandezza.

Nel caso di una matrice tridiagonale di dimensione  $n \times n$ , gli elementi che bisogna effettivamente conservare sono quelli delle tre diagonali, cioè sono  $3n - 2$ .

**Esempio 2.39.** Si consideri la matrice tridiagonale del sistema (2.52) (esempio 2.36). Si vuole determinare uno schema di memorizzazione efficiente, che non conservi cioè gli elementi al di fuori delle tre diagonali. Una scelta naturale è utilizzare tre array monodimensionali, uno per ciascuna diagonale. Detti  $AD$ ,  $AU$  ed  $AL$  tali array, si ha quindi:

$$\begin{aligned} AD &= (\underbrace{3, 3, \dots, 3}_{n \text{ volte}}), \\ AL &= (\underbrace{-1, -1, \dots, -1}_{n-1 \text{ volte}}), \\ AU &= (\underbrace{-1, -1, \dots, -1}_{n-1 \text{ volte}}). \end{aligned}$$

Si noti che la matrice considerata è anche simmetrica e si può quindi evitare di memorizzare la diagonale inferiore o quella superiore, utilizzando così solo due array monodimensionali, con un ulteriore risparmio dello spazio di memoria. ♣

In generale, data una matrice tridiagonale  $A$  di dimensione  $n \times n$ ,

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & & & & \\ a_{2,1} & a_{2,2} & a_{2,3} & & & \\ & \ddots & \ddots & \ddots & & \\ & & & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ & & & & a_{n,n-1} & a_{n,n} \end{pmatrix},$$

si utilizza un array monodimensionale, di lunghezza  $n$ , per memorizzare la diagonale principale e due array monodimensionali, di lunghezza  $n - 1$ , per memorizzare la diagonale inferiore e quella superiore:

$$\begin{aligned} AD(i) &= a_{i,i}, \quad i = 1, \dots, n, \\ AL(i+1) &= a_{i+1,i}, \quad i = 1, \dots, n-1, \\ AU(i) &= a_{i,i+1}, \quad i = 1, \dots, n-1. \end{aligned}$$

Si noti che al primo elemento dell'array  $AL$  è stato assegnato l'indice 2; in tal modo l'indice del generico elemento dell'array risulta essere uguale all'indice di riga del corrispondente elemento della diagonale inferiore. La scelta di assegnare l'indice 1 al primo elemento di  $AL$  è chiaramente equivalente. Tale schema di memorizzazione ha una complessità di spazio pari a  $3n - 2$ , cioè  $\mathcal{O}(n)$ , di un ordine di grandezza inferiore rispetto alla usuale memorizzazione mediante un array bidimensionale.

Se la matrice tridiagonale è simmetrica si può evitare di memorizzare la diagonale inferiore o quella superiore, riducendo la complessità di spazio a  $2n - 1$ . Si osservi che la memorizzazione con due array monodimensionali è adatta anche a matrici bidiagonali.

Per una matrice a banda  $A$ , di dimensione  $n \times n$ , con ampiezza di banda inferiore  $p$  ed ampiezza di banda superiore  $q$ , si potrebbero utilizzare  $p + q + 1$  array monodimensionali, contenenti le  $p$  diagonali inferiori, la diagonale principale e le  $q$  diagonali superiori. In tal modo, però, il numero di array da dichiarare in un programma che opera su matrici a banda sarebbe variabile con la dimensione della matrice.

#### ♣ Esempio 2.40.

Si consideri la matrice a banda del sistema lineare (2.55) (esempio 2.37), che ha ampiezza di banda inferiore  $p = 2$  ed ampiezza di banda superiore  $q = 1$ . Dato che la matrice ha solo quattro diagonali “significative”, si può considerare un array bidimensionale  $AB$  di dimensione  $4 \times n$ , e si possono memorizzare le diagonali della matrice nelle righe di tale array, nel modo seguente:

$$AB = \left( \begin{array}{cccccc} * & \frac{R}{3} - 1 & \cdots & \frac{R}{3} - 1 & \frac{R}{3} - 1 & \frac{R}{3} - 1 \\ \frac{R}{2} + 2 & \frac{R}{2} + 2 & \cdots & \frac{R}{2} + 2 & \frac{R}{2} + 2 & \frac{R}{2} + 2 \\ R + 1 & R + 1 & \cdots & R + 1 & R + 1 & * \\ \frac{R}{6} & \frac{R}{6} & \cdots & \frac{R}{6} & * & * \end{array} \right),$$

dove il simbolo  $*$  indica che al corrispondente elemento dell'array  $AB$  non è stato associato alcun valore. ♣

Nell'esempio 2.40 le diagonali della matrice sono memorizzate, una dopo l'altra, a partire dalla diagonale superiore, nelle righe dell'array. Inoltre, per ogni  $j$ , la  $j$ -ma colonna della matrice si trova nella  $j$ -ma colonna dell'array.

♣ **Esempio 2.41.**

Si consideri una matrice  $A$  di dimensione  $6 \times 6$ , con ampiezze di banda  $p = 1$  e  $q = 2$ :

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & & & \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & & \\ & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & \\ & & a_{4,3} & a_{4,4} & a_{4,5} & a_{4,6} \\ & & & a_{5,4} & a_{5,5} & a_{5,6} \\ & & & & a_{6,5} & a_{6,6} \end{pmatrix}.$$

Utilizzando per la matrice  $A$  lo schema di memorizzazione introdotto nell'esempio 2.40, si ha:

$$AB = \begin{pmatrix} * & * & a_{1,3} & a_{2,4} & a_{3,5} & a_{4,6} \\ * & a_{1,2} & a_{2,3} & a_{3,4} & a_{4,5} & a_{5,6} \\ a_{1,1} & a_{2,2} & a_{3,3} & a_{4,4} & a_{5,5} & a_{6,6} \\ a_{2,1} & a_{3,2} & a_{4,3} & a_{5,4} & a_{6,5} & * \end{pmatrix}.$$

L'elemento  $a_{i,j}$  è dunque memorizzato nella colonna di posto  $j$  e nella riga di posto  $3+i-j = q+1+i-j$ .



In generale, per memorizzare una generica matrice a banda  $A$ , di dimensione  $n \times n$ , con ampiezze di banda  $p$  e  $q$ , si utilizza un array bidimensionale di dimensione  $(p+q+1) \times n$ , nel modo seguente:

$$AB(q+1+i-j, j) = a_{i,j}, \quad \max(1, j-q) \leq i \leq \min(n, j+p).$$

Tale schema di memorizzazione è detto *a banda* (in inglese si parla di *band storage*) ed ha una complessità di spazio pari a  $(p+q+1)n$ . E' chiaro che la memorizzazione a banda è "significativamente" vantaggiosa, rispetto alla usuale memorizzazione mediante un array bidimensionale di dimensione  $n \times n$ , se  $p, q \ll n/2$ .

Analogamente alle matrici a banda, si definiscono schemi di memorizzazione opportuni, per le matrici triangolari.

♣ **Esempio 2.42.**

Si consideri la matrice triangolare inferiore

$$L = \begin{pmatrix} 1 & & & & & \\ -3 & 10 & & & & \\ 2 & 8 & -1 & & & \\ 4 & -5 & 1 & 9 & & \\ 2 & 6 & 4 & -3 & 2 & \\ 14 & -8 & 6 & -4 & 7 & 11 \end{pmatrix},$$

di dimensione  $6 \times 6$ . Il triangolo superiore della matrice, senza la diagonale principale, è costituito da 15 elementi nulli; se per memorizzare tale matrice si utilizza un array bidimensionale con 6 righe e 6 colonne, si "sprecano" 15 su 36 componenti dell'array. Una scelta più efficiente, suggerita in maniera

naturale dalla struttura della matrice, è utilizzare un array monodimensionale,  $LP$ , contenente, una dopo l'altra, le righe, oppure le colonne, di  $L$  (si considera, ovviamente, solo la parte “significativa” di  $L$ ):

$$LP = (1, -3, 10, 2, 8, -1, 4, -5, 1, 9, 2, 6, 4, -3, 2, 14, -8, 6, -4, 7, 11),$$

oppure

$$LP = (1, -3, 2, 4, 2, 14, 10, 8, -5, 6, -8, -1, 1, 4, 6, 9, -3, -4, 2, 7, 11).$$



Per una matrice triangolare inferiore, di dimensione  $n \times n$ ,

$$L = \begin{pmatrix} l_{1,1} & & & \\ l_{2,1} & l_{2,2} & & \\ \vdots & \vdots & \ddots & \\ l_{n,1} & l_{n,2} & \cdots & l_{n,n} \end{pmatrix},$$

si può utilizzare un array monodimensionale di lunghezza  $n(n + 1)/2$ , contenente, una dopo l'altra, le righe oppure le colonne del triangolo inferiore:

$$LP = (l_{1,1}, l_{2,1}, l_{2,2}, \dots, l_{n,1}, l_{n,2}, \dots, l_{n,n}),$$

oppure

$$LP = (l_{1,1}, l_{2,1}, \dots, l_{n,1}, l_{2,2}, \dots, l_{n,2}, \dots, l_{n,n}).$$

Si consideri ad esempio la memorizzazione per righe. La posizione di  $l_{i,j}$ ,  $i \geq j$ , in  $LP$  si può determinare col seguente ragionamento. L'elemento  $l_{i,j}$ , appartenendo alla  $i$ -ma riga di  $L$ , è preceduto in  $LP$  dagli elementi delle  $i - 1$  righe precedenti, che sono

$$1 + 2 + \dots + (i - 1) = \sum_{k=1}^{i-1} k = \frac{i(i - 1)}{2};$$

essendo inoltre il  $j$ -mo elemento della  $i$ -ma riga, è preceduto in  $LP$  da altri  $j - 1$  elementi. L'elemento  $l_{i,j}$  è quindi preceduto in  $LP$  da

$$\frac{i(i - 1)}{2} + j - 1$$

elementi, ovvero si ha

$$LP(i(i - 1)/2 + j) = l_{i,j}, \quad i \geq j.$$

Analogamente si verifica che, se si usa la memorizzazione per colonne, risulta

$$LP(i + (2n - j)(j - 1)/2) = l_{i,j}, \quad i \geq j.$$

Seguendo gli stessi criteri, si ottengono schemi di memorizzazione adatti ad una matrice triangolare superiore

$$U = \begin{pmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,n} \\ & u_{2,2} & \cdots & u_{2,n} \\ & & \ddots & \vdots \\ & & & u_{n,n} \end{pmatrix}.$$

Anche in questo caso si utilizza un array monodimensionale  $UP$ , di lunghezza  $n(n+1)/2$ , per memorizzare le righe di  $U$ :

$$UP = (u_{1,1}, u_{1,2}, \dots, u_{1,n}, u_{2,2}, \dots, u_{2,n}, \dots, u_{n,n}),$$

oppure le colonne:

$$UP = (u_{1,1}, u_{1,2}, u_{2,2}, \dots, u_{1,n}, u_{2,n}, \dots, u_{n,n}).$$

Nel primo caso risulta

$$UP((2n-i)(i-1)/2 + j) = u_{i,j}, \quad i \leq j;$$

nel secondo

$$UP(i + j(j-1)/2) = u_{i,j}, \quad i \leq j.$$

Gli schemi di memorizzazione per matrici triangolari sopra descritti sono detti *packed* (si parla, in inglese, di *packed storage*) e sono utilizzabili anche per la memorizzazione di matrici simmetriche. La scelta di una memorizzazione per righe o per colonne dipende dall'algoritmo che agisce sulla matrice; se l'algoritmo accede agli elementi della matrice per righe è opportuno utilizzare una memorizzazione per righe in modo da avere accessi a locazioni di memoria consecutive e quindi un utilizzo più efficiente della cache memory.

L'uso di schemi di memorizzazione particolari comporta una "riorganizzazione" degli algoritmi che operano sulle corrispondenti matrici. Alcuni esempi al riguardo sono forniti dalle Procedure 2.12 e 2.13. La prima risolve un sistema lineare con matrice triangolare inferiore memorizzata secondo uno schema *packed*, utilizzando l'algoritmo di back substitution particolarizzato per lo schema di memorizzazione in uso. La complessità di tempo è quindi invariata, mentre la complessità di spazio è quasi dimezzata, risultando  $\mathcal{O}(n^2/2)$  anziché  $\mathcal{O}(n^2)$ . La Procedura 2.13 calcola il prodotto di una matrice tridiagonale per un vettore, utilizzando per la matrice una memorizzazione con tre array monodimensionali. Essa differisce da una procedura per il prodotto di una matrice generica per un vettore non solo per il modo in cui si fa riferimento agli elementi della matrice, ma anche per il fatto che non si eseguono le operazioni che coinvolgono gli elementi (nulli) al di fuori delle tre diagonali. Ciò è una conseguenza immediata dello schema di memorizzazione utilizzato per la matrice tridiagonale. La complessità di tempo della Procedura 2.13 risulta quindi  $\mathcal{O}(n)$  operazioni aritmetiche floating-point, mentre nel caso generale è  $\mathcal{O}(n^2)$ . Analogamente, la complessità di spazio di tale procedura è  $\mathcal{O}(n)$ , invece di  $\mathcal{O}(n^2)$ .

```

procedure trisol (in:  $n, lp, b$ ; out:  $b, ierr$ )
  /# SCOPO: risolve un sistema lineare con matrice
  triangolare inferiore in formato packed

  /# SPECIFICHE DEI PARAMETRI:
  /# PARAMETRI DI INPUT:
  var:  $n$  : intero { dimensione del sistema }
  var:  $lp(n(n + 1)/2)$  : reale { matrice del sistema }
  var:  $b(n)$  : reale { vettore dei termini noti }

  /# PARAMETRI DI OUTPUT:
  var:  $b(n)$  : reale { vettore soluzione }
  var:  $ierr$  : intero { indice del primo elemento }
    { diagonale nullo }

```

Procedura 2.12: Risoluzione di un sistema lineare  
con matrice triangolare inferiore in formato packed - continua

```

/# VARIABILI LOCALI:
var:  $i, j, iddiag, irow$  : interi

/# INIZIO ISTRUZIONI:
ierr := 0
 $i := 1$ 
repeat { controllo della singolarità }
     $iddiag := i * (i - 1)/2 + i - 1$ 
    if ( $lp(iddiag) = 0$ ) then
         $ierr := i$ 
    endif
     $i := i + 1$ 
until ( $ierr \neq 0$  or  $i = n + 1$ )
if ( $ierr \neq 0$ ) then { risoluzione del sistema }
     $b(1) := b(1)/lp(1)$ 
    for  $i = 2$  to  $n$ 
         $irow := i(i - 1)/2$ 
        for  $j = 1$  to  $i - 1$ 
             $b(i) := b(i) - lp(irow + j) * b(j)$  { calcolo della soluzione }
            { memorizzata in b}
        endfor
         $b(i) := b(i)/lp(irow + i)$ 
    endfor
endif
end trisol

```

Procedura 2.12: Risoluzione di un sistema lineare  
con matrice triangolare inferiore in formato packed - fine

```

procedure tridvet (in:  $n, ad, al, au, x$ ; out:  $y$ )
  /# SCOPO: calcola il prodotto di una matrice A,
  tridiagonale quadrata, per un vettore x.
  La matrice è memorizzata in tre vettori.

  /# SPECIFICHE DEI PARAMETRI:
  /# PARAMETRI DI INPUT:
  var:  $n$  : intero { dimensione del vettore }
  var:  $ad(n)$  : reale { diagonale principale }
    { della matrice A}
  var:  $al(n - 1)$  : reale { diagonale inferiore }
    { della matrice A}
  var:  $au(n - 1)$  : reale { diagonale superiore }
    { della matrice A}
  var:  $x(n)$  : reale { secondo fattore del prodotto}
  /# PARAMETRI DI OUTPUT:
  var:  $y(n)$  : reale { vettore soluzione }
  /# VARIABILI LOCALI:
  var:  $i$  : interi

  /# INIZIO ISTRUZIONI:
  for  $i = 2$  to  $n - 1$  { componenti  $i = 2, \dots, n - 1$  }
     $y(i) := al(i) * x(i - 1) + ad(i) * x(i) + au(i) * x(i + 1)$ 
  endfor
     $y(1) := ad(1) * x(1) + au(1) * x(2)$  { componente  $i=1$  }
     $y(n) := al(n) * x(n - 1) + ad(n) * x(n)$  { componente  $i=n$  }
  end tridvet

```

Procedura 2.13: Calcolo del prodotto di una matrice tridiagonale per un vettore. La matrice è memorizzata usando tre array monodimensionali

## 2.11 Risoluzione di sistemi lineari con matrice a banda

Si consideri un sistema lineare  $Ax = b$ , dove  $A$  è una matrice a banda. Per risolvere tale sistema si può eseguire la fattorizzazione  $LU$  di  $A$  ed applicare la back e la forward substitution ai sistemi lineari triangolari associati. Si vuole vedere se le matrici  $L$  ed  $U$  derivanti dalla fattorizzazione sono dotate di qualche struttura, dalla quale si possa trarre vantaggio in termini di complessità computazionale.

Nei prossimi paragrafi si analizza dapprima la fattorizzazione  $LU$  senza pivoting e poi quella con pivoting parziale.

### 2.11.1 Fattorizzazione LU senza pivoting di una matrice tridiagonale e risoluzione dei sistemi lineari associati

L'esempio che segue mostra in che modo la fattorizzazione  $LU$  senza pivoting “agisce” su una matrice tridiagonale.

#### ♣ Esempio 2.43.

Si vuole eseguire la fattorizzazione  $LU$  senza pivoting della matrice tridiagonale

$$A = \begin{pmatrix} 5 & -3 & & \\ 1 & 4 & -2 & \\ & -1 & 3 & 1 \\ & & 2 & 5 \end{pmatrix}.$$

Tale fattorizzazione si può ottenere con l'algoritmo di eliminazione di Gauss, cioè con i passi seguenti:

#### passo 1:

- calcolo dei moltiplicatori relativi alla prima colonna:  $m_{2,1} = 1/5$  ( $m_{3,1} = m_{4,1} = 0$ );
- trasformazione della sottomatrice attiva:

$$A^{(1)} = \begin{pmatrix} 5 & -3 & & \\ 0 & 23/5 & -2 & \\ & -1 & 3 & 1 \\ & & 2 & 5 \end{pmatrix};$$

#### passo 2:

- calcolo dei moltiplicatori relativi alla seconda colonna:  $m_{3,2} = -5/23$  ( $m_{4,2} = 0$ );
- trasformazione della sottomatrice attiva:

$$A^{(2)} = \begin{pmatrix} 5 & -3 & & \\ 0 & 23/5 & -2 & \\ 0 & 59/23 & 1 & \\ & 2 & 5 \end{pmatrix};$$

**passo 3:**

- calcolo del moltiplicatore relativo alla quarta colonna:  $m_{4,3} = 46/59$ ;
- trasformazione della sottomatrice attiva:

$$A^{(3)} = \begin{pmatrix} 5 & -3 & & \\ 0 & 23/5 & -2 & \\ & 0 & 59/23 & 1 \\ & & 0 & 249/59 \end{pmatrix}.$$

Si ha quindi:

$$A = LU$$

con

$$L = \begin{pmatrix} 1 & & & \\ 1/5 & 1 & & \\ & -5/23 & 1 & \\ & & 46/59 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 5 & -3 & & \\ 23/5 & -2 & & \\ 59/23 & 1 & & \\ 249/59 & & & \end{pmatrix},$$

cioè  $L$  e  $U$  sono matrici *bidiagonali*. ♣

In generale, i fattori  $L$  ed  $U$  di una matrice tridiagonale (che ammette la fattorizzazione  $LU$ ) sono bidiagonali. Tale risultato si può dimostrare in maniera indipendente dall'algoritmo utilizzato per calcolare  $L$  ed  $U$ .

**♣ Esempio 2.44.**

Sia  $A$  una matrice tridiagonale non singolare di dimensione  $4 \times 4$ :

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & & \\ a_{2,1} & a_{2,2} & a_{2,3} & \\ & a_{3,2} & a_{3,3} & a_{3,4} \\ & & a_{4,3} & a_{4,4} \end{pmatrix},$$

e siano

$$L = \begin{pmatrix} 1 & & & \\ l_{2,1} & 1 & & \\ l_{3,1} & l_{3,2} & 1 & \\ l_{4,1} & l_{4,2} & l_{4,3} & 1 \end{pmatrix}, \quad U = \begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} \\ & u_{2,2} & u_{2,3} & u_{2,4} \\ & & u_{3,3} & u_{3,4} \\ & & & u_{4,4} \end{pmatrix}$$

le matrici ottenute applicando la fattorizzazione  $LU$  senza pivoting ad  $A$ . Calcolando il prodotto  $LU$  si ottiene la matrice

$$\begin{aligned} LU &= \\ &= \begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} & u_{1,4} \\ l_{2,1}u_{1,1} & l_{2,1}u_{1,2} + u_{2,2} & l_{2,1}u_{1,3} + u_{2,3} & l_{2,1}u_{1,4} + u_{2,4} \\ l_{3,1}u_{1,1} & l_{3,1}u_{1,2} + l_{3,2}u_{2,2} & l_{3,1}u_{1,3} + l_{3,2}u_{2,3} + u_{3,3} & l_{3,1}u_{1,4} + l_{3,2}u_{2,4} + u_{3,4} \\ l_{4,1}u_{1,1} & l_{4,1}u_{1,2} + l_{4,2}u_{2,2} & l_{4,1}u_{1,3} + l_{4,2}u_{2,3} + l_{4,3}u_{3,3} & l_{4,1}u_{1,4} + l_{4,2}u_{2,4} + l_{4,3}u_{3,4} + u_{4,4} \end{pmatrix} \end{aligned}$$

e quindi, uguagliando gli elementi di  $LU$  con i corrispondenti elementi di  $A$ , si ricava:<sup>18</sup>

$$\begin{aligned} u_{1,3} &= 0; \\ u_{1,4} &= 0; \\ l_{2,1}u_{1,4} + u_{2,4} &= 0 \implies u_{2,4} = 0; \\ l_{3,1}u_{1,1} &= 0 \implies l_{3,1} = 0; \\ l_{4,1}u_{1,1} &= 0 \implies l_{4,1} = 0; \\ l_{4,1}u_{1,2} + l_{4,2}u_{2,2} &= 0 \implies l_{4,2} = 0. \end{aligned}$$

Risulta dunque:

$$L = \begin{pmatrix} 1 & & & \\ l_{2,1} & 1 & & \\ 0 & l_{3,2} & 1 & \\ 0 & 0 & l_{4,3} & 1 \end{pmatrix}, \quad U = \begin{pmatrix} u_{1,1} & u_{1,2} & 0 & 0 \\ & u_{2,2} & u_{2,3} & 0 \\ & & u_{3,3} & u_{3,4} \\ & & & u_{4,4} \end{pmatrix}.$$

ovvero  $L$  e  $U$  sono bidiagonali. Inoltre, uguagliando gli elementi delle tre diagonali di  $LU$  con i corrispondenti elementi delle tre diagonali di  $A$  e ricordando che sono nulli gli elementi  $u_{i,j}$  con  $i > j$  e gli elementi  $l_{i,j}$  con  $i < j$ , si ricavano le espressioni dei rimanenti elementi di  $L$  e di  $U$ :

$$\begin{aligned} u_{1,1} &= a_{1,1}; \\ u_{1,2} &= a_{1,2}; \\ l_{2,1}u_{1,1} = a_{2,1} &\implies l_{2,1} = a_{2,1}/u_{1,1}; \\ l_{2,1}u_{1,2} + u_{2,2} = a_{2,2} &\implies u_{2,2} = a_{2,2} - l_{2,1}u_{1,2} = a_{2,2} - l_{2,1}a_{1,2}; \\ l_{2,1} \underbrace{u_{1,3}}_{=0} + u_{2,3} = a_{2,3} &\implies u_{2,3} = a_{2,3}; \\ l_{3,1} \underbrace{u_{1,2}}_{=0} + l_{3,2}u_{2,2} = a_{3,2} &\implies l_{3,2} = a_{3,2}/u_{2,2}; \\ l_{3,1} \underbrace{u_{1,3}}_{=0} + l_{3,2}u_{2,3} + u_{3,3} = a_{3,3} &\implies u_{3,3} = a_{3,3} - l_{3,2}u_{2,3} = a_{3,3} - l_{3,2}a_{2,3}; \\ l_{3,1} \underbrace{u_{1,4}}_{=0} + l_{3,2} \underbrace{u_{2,4}}_{=0} + u_{3,4} = a_{3,4} &\implies u_{3,4} = a_{3,4}; \\ l_{4,1} \underbrace{u_{1,3}}_{=0} + l_{4,2} \underbrace{u_{2,3}}_{=0} + l_{4,3}u_{3,3} = a_{4,3} &\implies l_{4,3} = a_{4,3}/u_{3,3} = 0; \\ l_{4,1} \underbrace{u_{1,4}}_{=0} + l_{4,2} \underbrace{u_{2,4}}_{=0} + l_{4,3}u_{3,4} + u_{4,4} = a_{4,4} &\implies u_{4,4} = a_{4,4} - l_{4,3}u_{3,4} = a_{4,4} - l_{4,3}a_{3,4}. \end{aligned}$$

In particolare, gli elementi della diagonale superiore di  $U$  sono uguali a quelli della diagonale superiore di  $A$ . ♣

Il ragionamento dell'esempio precedente si può applicare ad una generica matrice tridiagonale  $A$  di dimensione,  $n \times n$ :

$$A = \begin{pmatrix} d_1 & f_1 & & & \\ c_2 & d_2 & f_2 & & \\ & \ddots & \ddots & \ddots & \\ & & c_{n-1} & d_{n-1} & f_{n-1} \\ & & & c_n & d_n \end{pmatrix}. \quad (2.56)$$

<sup>18</sup>Si noti che  $u_{1,1} \neq 0$  e  $u_{2,2} \neq 0$ , altrimenti  $A$  risulterebbe singolare.

Siano

$$L = \begin{pmatrix} 1 & 0 & \dots & \dots & \dots & 0 \\ l_{2,1} & 1 & 0 & \dots & \dots & 0 \\ l_{3,1} & l_{3,2} & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ l_{n,1} & l_{n,2} & l_{n,3} & \dots & \dots & 1 \end{pmatrix},$$

$$U = \begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \dots & \dots & u_{1,n} \\ 0 & u_{2,2} & u_{2,3} & \dots & \dots & u_{2,n} \\ 0 & 0 & u_{3,3} & \dots & \dots & u_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & 0 & u_{n,n} \end{pmatrix} \quad (2.57)$$

le matrici ottenute applicando la fattorizzazione  $LU$  senza pivoting ad  $A$ . Calcolando il prodotto  $LU$  si ottiene la matrice

$$LU =$$

$$= \begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \dots & u_{1,n} \\ l_{2,1}u_{1,1} & l_{2,1}u_{1,2} + u_{2,2} & l_{2,1}u_{1,3} + u_{2,3} & \dots & l_{2,1}u_{1,n} + u_{2,n} \\ l_{3,1}u_{1,1} & l_{3,1}u_{1,2} + l_{3,2}u_{2,2} & l_{3,1}u_{1,3} + l_{3,2}u_{2,3} + u_{3,3} & \dots & l_{3,1}u_{1,n} + l_{3,2}u_{2,n} + u_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n,1}u_{1,1} & l_{n,1}u_{1,2} + l_{n,2}u_{2,2} & l_{n,1}u_{1,3} + l_{n,2}u_{2,3} + l_{n,3}u_{3,3} & \dots & l_{n,1}u_{1,n} + \dots + u_{n,n} \end{pmatrix}$$

e quindi, uguagliando gli elementi di  $LU$  con i corrispondenti elementi di  $A$ , si

ricava:<sup>19</sup>

$$\begin{aligned}
 & u_{1,3} = 0; \\
 & u_{1,4} = 0; \\
 & \vdots \\
 & u_{1,n} = 0; \\
 l_{2,1}u_{1,4} + u_{2,4} &= 0 \implies u_{2,4} = 0; \\
 & \vdots \\
 l_{2,1}u_{1,n} + u_{2,n} &= 0 \implies u_{2,n} = 0; \\
 l_{3,1}u_{1,1} &= 0 \implies l_{3,1} = 0; \\
 & \vdots \\
 l_{3,1}u_{1,5} + l_{3,2}u_{2,5} + u_{3,5} &= 0 \implies u_{3,5} = 0; \\
 & \vdots \\
 l_{n,1}u_{1,1} &= 0 \implies l_{n,1} = 0; \\
 & \vdots \\
 l_{n,1}u_{1,n-2} + l_{n,2}u_{2,n-2} + \dots + l_{n,n-2}u_{n-2,n-2} &= 0 \implies l_{n,n-2} = 0.
 \end{aligned}$$

Risulta dunque:

$$L = \begin{pmatrix} 1 & 0 & \dots & \dots & \dots & 0 \\ l_{2,1} & 1 & 0 & \dots & \dots & 0 \\ 0 & l_{3,2} & 1 & \dots & \dots & 0 \\ 0 & 0 & l_{4,3} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & l_{n,n-1} & 1 \end{pmatrix}, \\
 U = \begin{pmatrix} u_{1,1} & u_{1,2} & 0 & \dots & \dots & 0 \\ 0 & u_{2,2} & u_{2,3} & 0 & \dots & 0 \\ 0 & 0 & u_{3,3} & u_{3,4} & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & u_{n-1,n-1} & u_{n-1,n} \\ 0 & \dots & \dots & \dots & \dots & u_{n,n} \end{pmatrix}.$$

ovvero  $L$  e  $U$  sono bidiagonali.

Inoltre, uguagliando gli elementi delle tre diagonali di  $LU$  con i corrispondenti elementi delle tre diagonali di  $A$  e ricordando che sono nulli gli elementi  $u_{i,j}$  con  $i > j$  e gli elementi  $l_{i,j}$  con  $i < j$ , si ricavano le espressioni dei rimanenti elementi di  $L$  e di  $U$ .

---

<sup>19</sup>Si noti che  $u_{1,1} \neq 0$  e  $u_{2,2} \neq 0$ , altrimenti  $A$  risulterebbe singolare.

Riscrivendo le due matrici bidiagonali come:

$$L = \begin{pmatrix} 1 & & & \\ l_2 & 1 & & \\ & \ddots & \ddots & \\ & & l_{n-1} & 1 \\ & & & l_n & 1 \end{pmatrix}, \quad U = \begin{pmatrix} u_1 & f_1 & & \\ & u_2 & f_2 & \\ & & \ddots & \ddots \\ & & & u_{n-1} & f_{n-1} \\ & & & & u_n \end{pmatrix}; \quad (2.58)$$

si avrà:

$$\begin{aligned} d_1 &= u_1, & c_2 &= l_2 u_1, \\ d_2 &= l_2 f_1 + u_2, & c_3 &= l_3 u_2, \\ &\vdots & &\vdots \\ d_n &= l_n f_{n-1} + u_n, & c_n &= l_n u_{n-1}, \end{aligned}$$

da cui si ricavano le seguenti espressioni per  $u_i$  e  $l_i$ :

$$\begin{aligned} u_1 &= d_1, & u_i &= d_i - l_i f_{i-1} \quad (i = 2, \dots, n), \\ l_i &= c_i / u_{i-1} \quad (i = 2, \dots, n). \end{aligned}$$

In particolare si osservi che gli elementi della diagonale superiore di  $U$  coincidono con quelli della diagonale superiore di  $A$ .

Le espressioni degli elementi di  $L$  e di  $U$  si possono ricavare anche applicando ad  $A$  l'algoritmo di eliminazione di Gauss formulato per una generica matrice e tenendo conto del fatto che gli elementi di  $L$  al di sotto della prima diagonale inferiore e gli elementi di  $U$  al di sopra della prima diagonale superiore sono necessariamente nulli. L'algoritmo per la fattorizzazione  $LU$  di una matrice tridiagonale consiste quindi nel calcolare, al generico passo  $i$ , la  $i$ -ma componente della diagonale inferiore di  $L$  e la  $i$ -ma componente della diagonale principale di  $U$ , come di seguito mostrato nella Procedura 2.14.

```

procedure lutrid (in:  $n, d, c, f$ ; out:  $l, u$ )
  /* SCOPO: effettua la fattorizzazione LU senza pivoting di una matrice A tridiagonale.
  /* SPECIFICHE DEI PARAMETRI:
  /* PARAMETRI DI INPUT:
var:  $n$  : intero { prima dimensione }
           { della matrice }
var:  $d(n)$  : reale { diagonale principale }
           { della matrice A}
var:  $c(n)$  : reale { diagonale inferiore }
           { della matrice A}
var:  $f(n)$  : reale { diagonale superiore }
           { della matrice A}
  /* PARAMETRI DI OUTPUT:
var:  $l(n)$  : reale { matrice triangolare}
           { inferiore }
var:  $u(n)$  : reale { matrice triangolare }
           { superiore }
  /* VARIABILI LOCALI:
var:  $i$  : interi
  /* INIZIO ISTRUZIONI:
 $u(1) := d(1)$  { calcolo primo elemento}
for  $i = 2$  to  $n$  { fattorizzazione LU }
   $l(i) := c(i)/u(i-1)$ 
   $u(i) := d(i) - l(i) * f(i-1)$ 
endfor
end lutrid

```

Procedura 2.14: Fattorizzazione LU senza pivoting  
di una matrice tridiagonale

Si osservi che il precedente algoritmo non altera la struttura della matrice, nel senso che non trasforma elementi nulli in elementi non nulli.

Nella Procedura 2.14 si eseguono  $2n - 2$  moltiplicazioni o divisioni floating-point e  $n - 1$  addizioni floating-point; la complessità di tempo è quindi

$$T_{LUtrid}(n) = 3n - 3 = \mathcal{O}(n) \text{ flop},$$

dove *flop* indica la generica operazione aritmetica floating-point. La struttura della matrice consente dunque di ottenere un notevole vantaggio computazionale rispetto alla fattorizzazione di una matrice generica, la cui complessità di tempo è  $\mathcal{O}(n^3)$  flop.

Per quanto riguarda la complessità di spazio, si osservi che è possibile memorizzare gli elementi  $l_i$  ed  $u_i$  nelle stesse variabili utilizzate rispettivamente per  $c_i$  e  $d_i$ , in quanto, per ogni  $i$ ,  $c_i$  interviene solo nel calcolo di  $l_i$ , e  $d_i$  solo in quello di  $u_i$ . Con tale strategia, la complessità di spazio dell'algoritmo di fattorizzazione *LU* risulta essere:

$$S_{LUtrid}(n) = 3n - 2 = \mathcal{O}(n).$$

Si consideri ora il sistema lineare  $Ax = b$ , dove  $A$  è una matrice tridiagonale a cui è stata applicata la fattorizzazione *LU*. Per calcolare la soluzione di tale sistema, bisogna risolvere i sistemi triangolari bidiagonali  $Ly = b$  ed  $Ux = y$ . Da  $Ly = b$  si ricava:

$$\begin{aligned} y_1 &= b_1; \\ y_2 &= b_2 - l_2 y_1; \\ &\vdots \\ y_n &= b_n - l_n y_{n-1}; \end{aligned}$$

mentre da  $Ux = y$  si ricava:

$$\begin{aligned} x_n &= y_n / u_n; \\ x_{n-1} &= (y_{n-1} - f_{n-1} x_n) / u_{n-1}; \\ &\vdots \\ x_1 &= (y_1 - f_1 x_2) / u_1. \end{aligned}$$

Gli algoritmi di back e forward substitution applicati alle matrici bidiagonali (2.58) si particolarizzano quindi nel modo illustrato nelle Procedure 2.15 e 2.16.

```

procedure fstrid (in:  $n, l, b$ ; out:  $y$ )
  /# SCOPO: effettua la forward substitution di un sistema
  la cui matrice è triangolare inferiore bidiagonale.

  /# SPECIFICHE DEI PARAMETRI:
  /# PARAMETRI DI INPUT:
  var:  $n$  : intero { dimensione del sistema }
  var:  $l(n)$  : reale { contiene gli elementi non nulli }
    { della matrice del sistema }
  var:  $b(n)$  : reale { vettore dei termini noti }

  /# PARAMETRI DI OUTPUT:
  var:  $y(n)$  : reale { vettore risultante }

  /# VARIABILI LOCALI:
  var:  $i$  : interi

  /# INIZIO ISTRUZIONI:
   $y(1) := b(1)$ 
  for  $i = 2$  to  $n$ 
     $y(i) := b(i) - l(i) * y(i - 1)$ 
  endfor
  end fstrid

```

Procedura 2.15: Forward substitution per un sistema con matrice triangolare inferiore bidiagonale, derivante dalla fattorizzazione  $LU$   
senza pivoting di una matrice tridiagonale

```

procedure bstrid (in:  $n, u, b$ ; out:  $y$ )
  /* SCOPO: effettua la backward substitution di un sistema
   la cui matrice è triangolare superiore bidiagonale.

  /* SPECIFICHE DEI PARAMETRI:
  /* PARAMETRI DI INPUT:
  var:  $n$  : intero { dimensione del sistema }
  var:  $u(n)$  : reale { contiene gli elementi non nulli }
    { della matrice del sistema }
  var:  $b(n)$  : reale { vettore dei termini noti }

  /* PARAMETRI DI OUTPUT:
  var:  $y(n)$  : reale { vettore risultante }

  /* VARIABILI LOCALI:
  var:  $i$  : interi

  /* INIZIO ISTRUZIONI:
 $x(n) := y(n)/u(n)$ 
for  $i = n - 1$  to 1 by  $-1$ 
   $x(i) := (y(i) - f(i) * x(i + 1))/u(i)$ 
endfor
end bstrid

```

Procedura 2.16: Backward substitution per un sistema con matrice triangolare superiore bidiagonale, derivante dalla fattorizzazione  $LU$  senza pivoting di una matrice tridiagonale

Nella Procedura 2.15 si eseguono  $n - 1$  moltiplicazioni ed  $n - 1$  addizioni floating-point, mentre nella Procedura 2.16 si eseguono  $2n - 1$  moltiplicazioni o divisioni floating-point e  $n - 1$  addizioni. La complessità di tempo dei due algoritmi è quindi:

$$T_{Ffstrid}(n) = T_{Bbstrid}(n) = \mathcal{O}(n) \text{ flop.}$$

Nei due algoritmi è inoltre possibile memorizzare la soluzione sovrascrivendo il vettore dei termini noti, senza dover ricorrere all'uso di un'area di memoria aggiuntiva. La complessità di spazio è dunque

$$S_{fstrid}(n) = S_{bstrid}(n) = \mathcal{O}(n) \text{ flop.}$$

In conclusione, le complessità di spazio e di tempo di un algoritmo per la risoluzione (mediante fattorizzazione  $LU$  e back e forward substitution) di un sistema lineare con matrice tridiagonale di dimensione  $n \times n$  sono:

$$\begin{aligned} T_{RISTrid}(n) &= \mathcal{O}(n), \\ S_{RISTrid}(n) &= \mathcal{O}(n). \end{aligned}$$

### 2.11.2 Fattorizzazione LU senza pivoting di una matrice a banda e risoluzione dei sistemi lineari associati

I risultati ottenuti per una matrice tridiagonale si estendono al caso di una generica matrice a banda.

♣ **Esempio 2.45.** Si esegua la fattorizzazione LU senza pivoting della matrice a banda:

$$A = \begin{pmatrix} 1 & 2 & & & \\ 2 & -1 & 1 & & \\ 3 & 1 & 3 & 1 & \\ & 1 & 2 & -2 & 1 \\ & & -1 & 2 & 1 \end{pmatrix},$$

che ha ampiezza di banda superiore  $q = 1$  e ampiezza di banda inferiore  $p = 2$ . Si ha:

**passo 1:**

- calcolo dei moltiplicatori:  $m_{2,1} = 2$ ,  $m_{3,1} = 3$  ( $m_{4,1} = m_{5,1} = 0$ );
- trasformazione della sottomatrice attiva:

$$A^{(1)} = \begin{pmatrix} 1 & 2 & & & \\ & -5 & 1 & & \\ -5 & 3 & 1 & & \\ 1 & 2 & -2 & 1 & \\ & -1 & 2 & 1 \end{pmatrix};$$

**passo 2:**

- calcolo dei moltiplicatori:  $m_{3,2} = 1$ ,  $m_{4,2} = -1/5$  ( $m_{5,2} = 0$ );
- trasformazione della sottomatrice attiva:

$$A^{(2)} = \begin{pmatrix} 1 & 2 & & & \\ & -5 & 1 & & \\ 2 & 1 & 1 & & \\ 11/5 & -2 & 1 & & \\ -1 & 2 & 1 \end{pmatrix};$$

**passo 3:**

- calcolo dei moltiplicatori:  $m_{4,3} = 11/10$ ,  $m_{5,3} = -1/2$ ;
- trasformazione della sottomatrice attiva:

$$A^{(3)} = \begin{pmatrix} 1 & 2 & & \\ & -5 & 1 & \\ & & 2 & 1 \\ & & & -31/10 & 1 \\ & & & 5/2 & 1 \end{pmatrix};$$

**passo 4:**

- calcolo del moltiplicatore:  $m_{5,4} = -25/31$ ;
- trasformazione della sottomatrice attiva:

$$A^{(4)} = \begin{pmatrix} 1 & 2 & & \\ & -5 & 1 & \\ & & 2 & 1 \\ & & & -31/10 & 1 \\ & & & & 56/31 \end{pmatrix}.$$

Si ha quindi:

$$A = LU$$

con

$$L = \begin{pmatrix} 1 & & & & \\ 2 & 1 & & & \\ 3 & 1 & 1 & & \\ & -1/5 & 11/10 & 1 & \\ & & -1/2 & -25/31 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 1 & 2 & & \\ & -5 & 1 & \\ & & 2 & 1 \\ & & & -31/10 & 1 \\ & & & & 56/31 \end{pmatrix};$$

ovvero  $L$  ha ampiezza di banda  $p = 2$  ed  $U$  ha ampiezza di banda  $q = 1$ . ♣

In generale sussiste il risultato seguente che include il caso delle matrici tridiagonali:

**Teorema 2.12.** *Sia  $A$  una matrice a banda di dimensione  $n \times n$ , con ampiezza di banda inferiore  $p$  ed ampiezza di banda superiore  $q$ . Se  $A$  ammette la fattorizzazione  $LU$ , allora  $L$  è una matrice triangolare inferiore a banda, con ampiezza di banda inferiore  $p$ , ed  $U$  è una matrice triangolare superiore a banda, con ampiezza di banda superiore  $q$ .*

**Dimostrazione**

La dimostrazione procede per induzione su  $n$ . Per  $n = 1$  la tesi è banalmente vera. Si suppaga quindi  $n > 1$ . Scrivendo la fattorizzazione nella forma:

$$A = \begin{pmatrix} \alpha & w^T \\ v & B \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ v/\alpha & I_{n-1} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & B - vw^T/\alpha \end{pmatrix} \begin{pmatrix} \alpha & w^T \\ 0 & I_{n-1} \end{pmatrix}$$

È facile osservare che  $B - vw^T/\alpha$  ha ampiezza di banda superiore  $q$  e ampiezza di banda inferiore  $p$ . Sia quindi  $L_1U_1$  la fattorizzazione  $LU$  della matrice  $B - vw^T/\alpha$ . Utilizzando le ipotesi di induzione e osservando che le prime  $q$  componenti di  $w$  e le prime  $p$  di  $v$  sono nulle, segue che:

$$L = \begin{pmatrix} 1 & 0 \\ v/\alpha & L_1 \end{pmatrix} \quad U = \begin{pmatrix} \alpha & w^T \\ 0 & U_1 \end{pmatrix}$$

hanno ampiezza di banda rispettivamente  $p$  e  $q$  e sono tali che  $A = LU$ . ■

Come già osservato per le matrici tridiagonali, al medesimo risultato si giunge se si applica l'algoritmo di eliminazione di Gauss formulato per una generica matrice e si osserva che la struttura a banda della matrice da fattorizzare implica che siano nulli gli elementi di  $L$  al di sotto della  $p$ -ma diagonale inferiore e quelli di  $U$  al di sopra della  $q$ -ma diagonale superiore (cfr. esempio 2.44).

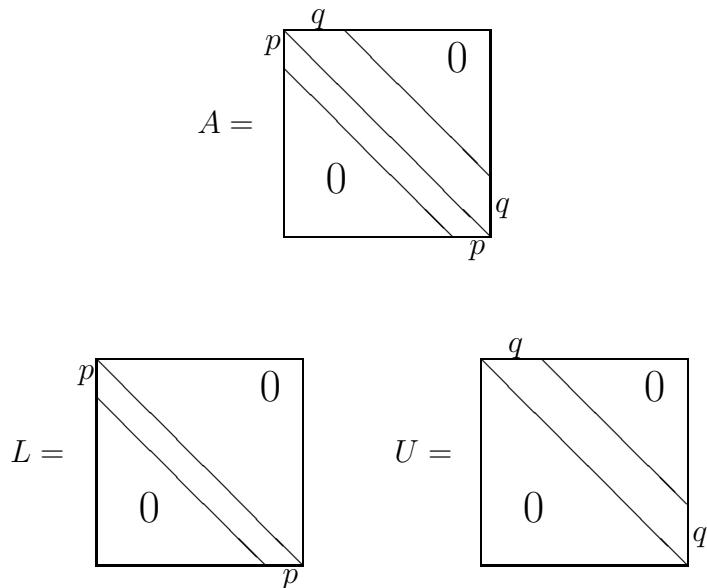


Figura 2.23: Fattorizzazione  $LU$  senza pivoting di una matrice a banda.

La fattorizzazione  $LU$  senza pivoting di una matrice a banda produce quindi fattori che conservano le ampiezze di banda della matrice di partenza (una rappresentazione grafica di ciò è fornita in Fig. 2.23). Al generico passo  $k$  dell'eliminazione di Gauss si calcolano al più  $p$  moltiplicatori, uno per ciascun elemento della banda inferiore nella  $k$ -ma colonna, e si trasformano gli elementi della sottomatrice attiva che appartengono alle righe  $k+1, \dots, \min(k+p, n)$  ed alle colonne  $k+1, \dots, \min(k+q, n)$ . Tale algoritmo può essere eseguito utilizzando per la matrice  $A$  lo schema di memorizzazione a banda di cui è parlato nel §2.10 e scrivendo gli elementi di  $L$  e di  $U$  sui corrispondenti elementi di  $A$  (algoritmo *in place*), come mostrato<sup>20</sup> nella Procedura 2.17. Gli elementi della

<sup>20</sup>Si noti che gli indici delle componenti dell'array  $AB$  possono essere scritti in maniera più “compatta”, riducendo il numero di operazioni aritmetiche intere e quindi migliorando l’efficienza dell’algoritmo. Ad esempio, facendo variare l’indice del ciclo su  $i$  nel modo seguente:

```
for i := q + 2 to min(q + p + 1, q + n - k + 1) ,
```

diagonale principale di  $L$  non sono, ovviamente, memorizzati.

```

procedure bandlu (in:  $n, AB, p, q$ ; out:  $AB$ )
  // SCOPO: effettua la fattorizzazione LU di una matrice A a banda
  memorizzata secondo lo schema band storage.

  // SPECIFICHE DEI PARAMETRI:
  // PARAMETRI DI INPUT:
  var:  $n$  : intero { prima dimensione }
    { della matrice A }
  var:  $AB(p + q + 1, n)$  : reale { matrice da fattorizzare }
  var:  $p$  : intero { ampiezza della banda }
    { inferiore di A }
  var:  $q$  : intero { ampiezza della banda }
    { superiore di A }

  // PARAMETRI DI OUTPUT:
  var:  $AB(p + q + 1, n)$  : reale { matrice fattorizzata }

  // VARIABILI LOCALI:
  var:  $i, j, k$  : interi
```

Procedura 2.7: Fattorizzazione  $LU$  senza pivoting di una matrice a banda memorizzata secondo lo schema *band storage* - continua

---

i moltiplicatori si possono calcolare con l'istruzione

$$AB(i, k) := AB(i, k)/AB(q + 1, k);$$

si può inoltre precalcolare il valore di  $q + 1$ , con un'istruzione del tipo  $r := q + 1$ , e quindi scrivere:

$$AB(i, k) := AB(i, k)/AB(r, k).$$

```


/# INIZIO ISTRUZIONI:
for k = 1 to n - 1
    for i = k + 1 to min(k + p, n)           { calcolo multipl.}
        AB(q + 1 + i - k, k) := AB(q + 1 + i - k, k)/AB(q + 1, k)
    endfor

    for j = k + 1 to min(k + q, n)
        for i = k + 1 to min(k + p, n)
            AB(q + 1 + i - j, j) := AB(q + 1 + i - j, j) +
                -AB(q + 1 + i - k, k) * AB(q + 1 + k - j, j)
        endfor
    endfor
endfor
end bandlu


```

Procedura 2.17: Fattorizzazione  $LU$  senza pivoting di una matrice a banda memorizzata secondo lo schema *band storage* - fine

Si osservi che nell'algoritmo precedente la trasformazione della sottomatrice attiva è eseguita accedendo agli elementi per colonne, cioè si trasformano una dopo l'altra le colonne di tale matrice (il ciclo sull'indice di riga  $i$  è all'interno del ciclo sull'indice di colonna  $j$ ). Scambiando i cicli su  $i$  e  $j$  si ottiene un algoritmo che accede alla sottomatrice attiva per righe, analogo a quello per la fattorizzazione  $LU$  di una matrice generica. I due algoritmi sono equivalenti, cioè producono gli stessi risultati, ed hanno la stessa complessità computazionale; la scelta dell'uno o dell'altro dipende essenzialmente dal linguaggio in cui tale algoritmo è implementato. Se, ad esempio, si usa il Fortran 77, in cui gli array sono memorizzati per colonne, l'algoritmo che accede agli elementi dell'array  $AB$  per colonne consente un utilizzo più efficiente della memoria.

#### ♣ Esempio 2.46.

Sia  $A$  una matrice a banda di dimensione  $10 \times 10$ , con ampiezza di banda inferiore  $p = 2$  ed ampiezza di banda superiore  $q = 4$ :

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} & a_{1,5} & & & & & \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} & a_{2,5} & a_{2,6} & & & & \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} & a_{3,5} & a_{3,6} & a_{3,7} & & & \\ a_{4,2} & a_{4,3} & a_{4,4} & a_{4,5} & a_{4,6} & a_{4,7} & a_{4,8} & & & \\ & a_{5,3} & a_{5,4} & a_{5,5} & a_{5,6} & a_{5,7} & a_{5,8} & a_{5,9} & & \\ & & a_{6,4} & a_{6,5} & a_{6,6} & a_{6,7} & a_{6,8} & a_{6,9} & a_{6,10} & \\ & & & a_{7,5} & a_{7,6} & a_{7,7} & a_{7,8} & a_{7,9} & a_{7,10} & \\ & & & & a_{8,6} & a_{8,7} & a_{8,8} & a_{8,9} & a_{8,10} & \\ & & & & & a_{9,7} & a_{9,8} & a_{9,9} & a_{10,10} & \\ & & & & & & a_{10,8} & a_{10,9} & a_{10,10} & \end{pmatrix}.$$

Contiamo il numero di operazioni aritmetiche floating-point che si eseguono applicando la Procedura 2.17 a tale matrice.

L'algoritmo considerato consta di  $n - 1$  passi, individuati dal contatore  $k$ ; il numero di operazioni varia, al variare di  $k$ , nel modo seguente:

- per  $k = 1, \dots, 6$  ( $6 = n - q$ ) si ha:

$$\begin{aligned}\min(k + p, n) &= \min(k + 2, 10) = k + 2, \\ \min(k + q, n) &= \min(k + 3, 10) = k + 3\end{aligned}$$

e quindi al passo  $k$  si eseguono

- $p = 2$  divisioni per calcolare i moltiplicatori,
- $pq = 8$  addizioni ed altrettante moltiplicazioni per trasformare la sottomatrice attiva,

per un totale di

$$6(2 + 2 \cdot 8) = 108 \text{ flop};$$

- per  $k = 7, 8$  ( $7 = n - q + 1$ ,  $8 = n - p$ ) si ha:

$$\begin{aligned}\min(k + p, n) &= \min(k + 2, 10) = k + 2, \\ \min(k + q, n) &= \min(k + 3, 10) = 10\end{aligned}$$

e quindi al generico passo  $k$  si eseguono

- $p = 2$  divisioni per calcolare i moltiplicatori,
- $p(n - k) = 2(10 - k)$  addizioni ed altrettante moltiplicazioni per trasformare la sottomatrice attiva,

per un totale di

$$\sum_{k=7}^8 (2 + 2 \cdot (10 - k)) = 14 + 10 = 24 \text{ flop};$$

- per  $k = 9$  ( $9 = n - p + 1 = n - 1$ ) si ha:

$$\begin{aligned}\min(k + p, n) &= \min(k + 2, 10) = 10, \\ \min(k + q, n) &= \min(k + 3, 10) = 10\end{aligned}$$

e quindi si eseguono

- $n - k = 1$  divisione per calcolare i moltiplicatori,
- $(n - k)(n - k) = 1$  addizione ed 1 moltiplicazione per trasformare la sottomatrice attiva,

per un totale di

$$3 \text{ flop}.$$

Nell'algoritmo si eseguono dunque

$$108 + 24 + 3 = 135 \text{ flop}.$$

Si osservi che tale numero è significativamente inferiore rispetto al numero di operazioni che si eseguono applicando l'algoritmo di eliminazione di Gauss formulato per una generica matrice, che è  $(4n^3 - 3n^2 - 7n)/6 = 615$ . ♣

Consideriamo una matrice a banda  $A$ , di dimensione  $n \times n$ , con ampiezza di banda inferiore  $p$  ed ampiezza di banda superiore  $q$ :

$$A = \begin{pmatrix} a_{1,1} & \dots & \dots & \dots & a_{1,q+1} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\ a_{p+1,1} & \dots & \dots & \dots & \dots & \dots & a_{p+1,q+1+p} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & a_{n-q,n} \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & a_{n-q+1,n} \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ a_{n,n-p} & \dots & \dots & \dots & \dots & \dots & a_{n,n} \end{pmatrix}.$$

Si suppone che  $p < q$ , e quindi che  $n - q < n - p$ ; l'algoritmo consta di  $n - 1$  passi, individuati dal contatore  $k$ ; il numero di operazioni varia, al variare di  $k$ , nel modo seguente:

- per  $k = 1, \dots, n - q$  risulta

$$\begin{aligned} \min(k + p, n) &= k + p, \\ \min(k + q, n) &= k + q \end{aligned}$$

e quindi al passo  $k$  si eseguono

- $p$  divisioni per calcolare i moltiplicatori,
- $pq$  addizioni ed altrettante moltiplicazioni per trasformare la sottomatrice attiva,

per un totale di

$$(n - q)(p + 2pq) = np + 2npq - pq - 2pq^2 \text{ flop};$$

- per  $k = n - q + 1, \dots, n - p$  risulta

$$\begin{aligned} \min(k + p, n) &= k + p, \\ \min(k + q, n) &= n \end{aligned}$$

e quindi al passo  $k$  si eseguono

- $p$  divisioni per calcolare i moltiplicatori,

–  $p(n - k)$  addizioni ed altrettante moltiplicazioni per trasformare la sottomatrice attiva,

per un totale di

$$\begin{aligned} \sum_{k=n-q+1}^{n-p} (p + 2p(n - k)) &= (q - p)p + 2p \sum_{k=n-q+1}^{n-p} (n - k) = \\ &= (q - p)p + 2p \sum_{i=p}^{q-1} i = (q - p)p + 2p \left( \sum_{i=1}^{q-1} i - \sum_{i=1}^{p-1} i \right) = \\ &= (q - p)p + p(q(q - 1) - p(p - 1)) = pq^2 - p^3 \text{ flop}; \end{aligned}$$

- per  $k = n - p + 1, \dots, n - 1$  risulta

$$\begin{aligned} \min(k + p, n) &= n, \\ \min(k + q, n) &= n \end{aligned}$$

e quindi al passo  $k$  si eseguono

- $n - k$  divisioni per calcolare i moltiplicatori,
- $(n - k)(n - k)$  addizioni ed altrettante moltiplicazioni per trasformare la sottomatrice attiva,

per un totale di

$$\begin{aligned} \sum_{k=n-p+1}^{n-1} ((n - k) + 2(n - k)^2) &= \sum_{i=1}^{p-1} i + 2 \sum_{i=1}^{p-1} i^2 = \\ &= \frac{p(p - 1)}{2} + \frac{p(p - 1)(2p - 1)}{3} = \frac{4p^3 - 3p^2 - p}{6} \text{ flop}. \end{aligned}$$

Sommendo i valori precedentemente ottenuti, si ha:

$$\begin{aligned} np + 2npq - pq - 2pq^2 + pq^2 - p^3 + \frac{4p^3 - 3p^2 - q}{6} &= \\ = 2npq + np - pq^2 - \frac{2p^3 + 3p^2 + p}{6} - pq &\text{ flop}. \end{aligned}$$

e quindi la complessità di tempo dell'algoritmo descritto nella Procedura 2.17 è :

$$T_{LUband}(n, p, q) = \mathcal{O}(npq) \text{ flop}.$$

Si noti che al crescere di  $p$  e  $q$  la complessità di tempo si avvicina a  $\mathcal{O}(n^3)$  flop, cioè a quella dell'algoritmo per una matrice di dimensione  $n \times n$ , e che per  $p, q \simeq n$ , essa è proprio  $\mathcal{O}(n^3)$  flop. L'algoritmo è dunque “significativamente vantaggioso” se  $p, q \ll n$ .

La complessità di spazio è quella richiesta dallo schema di memorizzazione a banda (cfr. § 2.10), ovvero

$$S_{LUband}(n, p, q) = (p + q + 1)n = \mathcal{O}((p + q + 1)n).$$

Si consideri ora la risoluzione dei sistemi lineari aventi come matrici dei coefficienti i fattori  $L$  ed  $U$  calcolati con la Procedura 2.17. Con un ragionamento analogo a quello fatto per le matrici  $L$  ed  $U$  derivanti da una matrice tridiagonale, si hanno gli algoritmi di back e forward substitution riportati nella Procedura 2.18 e 2.19. Si osservi che gli algoritmi sono in place, cioè il vettore soluzione è riscritto sul vettore dei termini noti. Inoltre, in analogia con l'algoritmo descritto nella Procedura 2.17, l'accesso agli elementi dell'array  $AB$  è per colonne; al generico passo  $j$  si ottiene il valore della  $j$ -ma incognita e si aggiornano i valori delle incognite che devono ancora essere calcolate.

```

procedure bandfs (in:  $n, AB, p, q, b$ ; out:  $b$ )
  /# SCOPO : risolve un sistema con matrice a banda A,
  derivante da fattorizzazione LU, utilizzando
  la forward substitution.

  /# SPECIFICHE DEI PARAMETRI:
  /# PARAMETRI DI INPUT:
  var:  $n$  : intero { dimensione del sistema }
  var:  $AB(p + q + 1, n)$  : reale { matrice da fattorizzare }
  var:  $p$  : intero { ampiezza della banda }
    { inferiore di  $A$  }
  var:  $q$  : intero { ampiezza della banda }
    { superiore di  $A$  }
  var:  $b(n)$  : reale { vettore dei termini noti }

  /# PARAMETRI DI OUTPUT:
  var:  $b(n)$  : reale { vettore risultante }
  /# VARIABILI LOCALI:
  var:  $i, j$  : interi

  /# INIZIO ISTRUZIONI:
  for  $j = 1$  to  $n - 1$ 
    for  $i = j + 1$  to  $\min(j + p, n)$ 
       $b(i) = b(i) - AB(q + 1 + i - j, j) * b(j)$ 
    endfor
  endfor
end bandfs

```

Procedura 2.18 : Forward substitution per un sistema con matrice triangolare inferiore derivante dalla fattorizzazione  $LU$   
senza pivoting di una matrice a banda

```

procedure bandbs (in:  $n, AB, p, q, b$ ; out:  $b$ )
  /# SCOPO: risolve un sistema con matrice a banda A,
  derivante da fattorizzazione LU, utilizzando
  la back substitution.

  /# SPECIFICHE DEI PARAMETRI:
  /# PARAMETRI DI INPUT:
  var:  $n$  : intero { dimensione del sistema }
  var:  $AB(p + q + 1, n)$  : reale { matrice da fattorizzare }
  var:  $p$  : intero { ampiezza della banda }
    { inferiore di  $A$  }
  var:  $q$  : intero { ampiezza della banda }
    { superiore di  $A$  }
  var:  $b(n)$  : reale { vettore dei termini noti }

  /# PARAMETRI DI OUTPUT:
  var:  $b(n)$  : reale { vettore risultante }
  /# VARIABILI LOCALI:
  var:  $i, j$  : interi

  /# INIZIO ISTRUZIONI:
  for  $j = n$  to 2 by  $-1$ 
     $b(j) := b(j)/AB(q + 1, j)$ 
    for  $i = \max(j - q, 1)$  to  $j - 1$ 
       $b(i) := b(i) - AB(q + 1 + i - j, j) * b(j)$ 
    endfor
  endfor
   $b(1) := b(1)/AB(q + 1, 1)$ 
end bandbs

```

Procedura 2.19: Back substitution per un sistema con matrice triangolare superiore derivante dalla fattorizzazione  $LU$   
senza pivoting di una matrice a banda

Si verifica che la complessità di tempo della Procedura 2.18 è:

$$T_{FORWband}(n, p) = \mathcal{O}(np) \text{ flop},$$

mentre quella della Procedura 2.19 è :

$$T_{BACKband}(n, q) = \mathcal{O}(nq) \text{ flop}.$$

Dato che gli elementi delle matrici  $L$  ed  $U$  sono memorizzati sui corrispondenti elementi della matrice di partenza  $A$ , la complessità di spazio è la stessa della fattorizzazione  $LU$ , cioè

$$S_{FORWband}(n, p) = S_{BACKband}(n, q) = \mathcal{O}((p + q)n).$$

### 2.11.3 Fattorizzazione LU con pivoting parziale di una matrice a banda e risoluzione dei sistemi lineari associati

Gli algoritmi di fattorizzazione  $LU$  considerati nei paragrafi precedenti non utilizzano alcuna strategia di pivoting; in generale, essi risultano quindi instabili. Si vuole allora vedere se la fattorizzazione  $LU$  con pivoting parziale genera delle matrici  $L$  ed  $U$  dotate di qualche struttura. Dato che il pivoting comporta scambi di righe, ci si aspetta che la struttura a banda della matrice di partenza non venga conservata. Gli esempi che seguono mostrano in che modo il pivoting modifica tale struttura.

#### ♣ Esempio 2.47.

Si esegua la fattorizzazione LU con pivoting parziale della matrice tridiagonale:

$$A = \begin{pmatrix} 1 & 1 & & \\ 2 & 1 & 1 & \\ & 1 & 1 & 2 \\ & & 2 & 1 \end{pmatrix}.$$

Si ha:

**passo 1:**

- pivoting in prima colonna:  $\max_{1 \leq i \leq 4} |a_{i,1}| = 2 = a_{2,1}$  e quindi si scambiano la prima e la seconda riga di  $A$ , ottenendo

$$\tilde{A} = \begin{pmatrix} 2 & 1 & 1 & \\ 1 & 1 & & \\ & 1 & 1 & 2 \\ & & 2 & 1 \end{pmatrix};$$

- calcolo dei moltiplicatori:  $m_{2,1} = 1/2$  ( $m_{3,1} = m_{4,1} = 0$ ), ovvero

$$L^{(1)} = \begin{pmatrix} 1 & & & \\ 1/2 & 1 & & \\ & & 1 & \\ & & & 1 \end{pmatrix},$$

dove  $L^{(i)}$  indica la matrice triangolare unitaria inferiore che contiene, nella prime  $i$  colonne, i moltiplicatori calcolati nei primi  $i$  passi, ed ha tutti gli elementi subdiagonali delle colonne successive alla  $i$ -ma uguali a 0;<sup>21</sup>

- trasformazione della sottomatrice attiva:

$$A^{(1)} = \begin{pmatrix} 2 & 1 & 1 & \\ & 1/2 & -1/2 & \\ 1 & & 1 & 2 \\ & & 2 & 1 \end{pmatrix};$$

**passo 2:**

- pivoting in seconda colonna:  $\max_{2 \leq i \leq 4} |a_{i,2}^{(1)}| = 1 = a_{3,2}^{(1)}$  e quindi si scambiano la seconda e la terza riga di  $A^{(1)}$  ed  $L^{(1)}$ , ottenendo

$$\tilde{A}^{(1)} = \begin{pmatrix} 2 & 1 & 1 & \\ 1 & 1 & -1/2 & 2 \\ 1/2 & & 2 & 1 \end{pmatrix}, \quad \tilde{L}^{(1)} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ 1/2 & & 1 & \\ & & & 1 \end{pmatrix};$$

- calcolo dei moltiplicatori:  $m_{3,2} = 1/2$  ( $m_{4,2} = 0$ ), ovvero

$$\tilde{L}^{(2)} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ 1/2 & 1/2 & 1 & \\ & & & 1 \end{pmatrix};$$

- trasformazione della sottomatrice attiva:

$$A^{(2)} = \begin{pmatrix} 2 & 1 & 1 & \\ & 1 & 1 & 2 \\ & & -1 & -1 \\ & & 2 & 1 \end{pmatrix};$$

**passo 3:**

- pivoting in terza colonna:  $\max_{3 \leq i \leq 4} |a_{i,3}^{(2)}| = 1 = a_{4,3}^{(2)}$  e quindi si scambiano la terza e la quarta riga di  $A^{(2)}$  ed  $L^{(2)}$ , ottenendo

$$\tilde{A}^{(2)} = \begin{pmatrix} 2 & 1 & 1 & \\ 1 & 1 & 2 & \\ 2 & & 1 \\ -1 & -1 & \end{pmatrix}, \quad \tilde{L}^{(2)} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ 1/2 & 1/2 & 1 & \\ & & & 1 \end{pmatrix};$$

---

<sup>21</sup>Tale matrice è stata introdotta solo per mostrare in che modo il pivoting agisce sulla matrice  $L$  che si ottiene con la fattorizzazione  $LU$ .

- calcolo del moltiplicatore:  $m_{4,2} = -1/2$ , ovvero

$$L^{(3)} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ 1/2 & 1/2 & -1/2 & 1 \end{pmatrix};$$

- trasformazione della sottomatrice attiva:

$$A^{(3)} = \begin{pmatrix} 2 & 1 & 1 & 2 \\ & 1 & 1 & \\ & & 2 & 1 \\ & & & -1/2 \end{pmatrix};$$

Eseguendo la fattorizzazione  $LU$  con pivoting della matrice  $A$  si ottiene quindi

$$PA = LU,$$

con

$$L = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ 1/2 & 1/2 & -1/2 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 2 & 1 & 1 & 2 \\ & 1 & 1 & \\ & & 2 & 1 \\ & & & -1/2 \end{pmatrix},$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

Le matrici  $L$  ed  $U$  non sono matrici bidiagonali.  $U$  è una matrice a banda con ampiezza di banda uguale a 2, cioè uguale alla somma delle ampiezze di banda di  $A$ , mentre  $L$  è una generica matrice triagonolare. Si noti però che ciascuna colonna di  $L$  ha al più un elemento non nullo al di fuori della diagonale principale. ♣

♣ **Esempio 2.48.** Si esegua la fattorizzazione  $LU$  con pivoting parziale della matrice a banda, che ha ampiezza di banda inferiore  $p = 2$  ed ampiezza di banda superiore  $q = 1$ :

$$A = \begin{pmatrix} 1 & 1 & & & \\ 2 & -1 & 1 & & \\ 3 & 1 & 3 & 1 & \\ & 1 & 3 & -2 & 1 \\ & & -2 & 1 & 1 \end{pmatrix}.$$

Si ha:

**passo 1:**

- pivoting in prima colonna:  $\max_{1 \leq i \leq 5} |a_{i,1}| = 3 = a_{3,1}$  e quindi si scambiano la prima e la terza riga di  $A$ , ottenendo

$$\tilde{A} = \begin{pmatrix} 3 & 1 & 3 & 1 & \\ 2 & -1 & 1 & & \\ 1 & 1 & & & \\ & 1 & 3 & -2 & 1 \\ & & -2 & 1 & 1 \end{pmatrix};$$

- calcolo dei moltiplicatori:  $m_{2,1} = 2/3$ ,  $m_{3,1} = 1/3$  ( $m_{4,1} = m_{5,1} = 0$ ), ovvero

$$L^{(1)} = \begin{pmatrix} 1 & & & & \\ 2/3 & 1 & & & \\ 1/3 & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix};$$

- trasformazione della sottomatrice attiva:

$$A^{(1)} = \begin{pmatrix} 3 & 1 & 3 & 1 & \\ & -5/3 & -1 & -2/3 & \\ & 2/3 & -1 & -1/3 & \\ 1 & 3 & -2 & 1 & \\ & -2 & 1 & 1 & \end{pmatrix};$$

**passo 2:**

- pivoting in seconda colonna:  $\max_{2 \leq i \leq 5} |a_{i,2}| = 5/3 = |a_{2,2}|$  e quindi non si eseguono scambi;
- calcolo dei moltiplicatori:  $m_{3,2} = -2/5$ ,  $m_{4,2} = -3/5$  ( $m_{5,2} = 0$ ), ovvero

$$L^{(2)} = \begin{pmatrix} 1 & & & & \\ 2/3 & 1 & & & \\ 1/3 & -2/5 & 1 & & \\ & -3/5 & & 1 & \\ & & & & 1 \end{pmatrix};$$

- trasformazione della sottomatrice attiva:

$$A^{(2)} = \begin{pmatrix} 3 & 1 & 3 & 1 & \\ & -5/3 & -1 & -2/3 & \\ & -7/5 & -3/5 & & \\ 12/5 & -12/5 & 1 & & \\ & -2 & 1 & 1 & \end{pmatrix};$$

**passo 3:**

- pivoting in terza colonna:  $\max_{3 \leq i \leq 5} |a_{i,3}| = 12/5 = |a_{4,3}|$  e quindi si scambiano la terza e la quarta riga di  $A^{(2)}$  ed  $L^{(2)}$ , ottenendo

$$\tilde{A}^{(2)} = \begin{pmatrix} 3 & 1 & 3 & 1 & \\ & -5/3 & -1 & -2/3 & \\ & 12/5 & -12/5 & 1 & \\ -7/5 & -3/5 & & & \\ & -2 & 1 & 1 & \end{pmatrix}, \quad \tilde{L}^{(2)} = \begin{pmatrix} 1 & & & & \\ 2/3 & 1 & & & \\ 1/3 & -3/5 & 1 & & \\ & -2/5 & & 1 & \\ & & & & 1 \end{pmatrix};$$

- calcolo dei moltiplicatori:  $m_{4,3} = -7/12$ ,  $m_{5,3} = -5/6$ , ovvero

$$L^{(3)} = \begin{pmatrix} 1 & & & & \\ 2/3 & 1 & & & \\ 1/3 & -3/5 & 1 & & \\ & -2/5 & -7/12 & 1 & \\ & & -5/6 & & 1 \end{pmatrix};$$

- trasformazione della sottomatrice attiva:

$$A^{(3)} = \begin{pmatrix} 3 & 1 & 3 & 1 \\ & -5/3 & -1 & -2/3 \\ & & 12/5 & -12/5 \\ & & & 1 \\ & & & -2 \\ & & & -1 \\ & & & 7/12 \\ & & & 11/6 \end{pmatrix};$$

**passo 4:**

- pivoting in quarta colonna:  $\max_{4 \leq i \leq 5} |a_{i,4}| = 2 = |a_{4,4}|$  e quindi non si eseguono scambi;
- calcolo del moltiplicatore:  $m_{5,4} = 1/2$ , ovvero

$$L^{(4)} = \begin{pmatrix} 1 & & & & \\ 2/3 & 1 & & & \\ & -3/5 & 1 & & \\ 1/3 & -2/5 & -7/12 & 1 & \\ & & -5/6 & 1/2 & 1 \end{pmatrix};$$

- trasformazione della sottomatrice attiva:

$$A^{(4)} = \begin{pmatrix} 3 & 1 & 3 & 1 \\ & -5/3 & -1 & -2/3 \\ & & 12/5 & -12/15 \\ & & & 1 \\ & & & -2 \\ & & & 7/12 \\ & & & 37/24 \end{pmatrix}.$$

In conclusione, si ha:

$$PA = LU,$$

con

$$L = \begin{pmatrix} 1 & & & & \\ 2/3 & 1 & & & \\ & -3/5 & 1 & & \\ 1/3 & -2/5 & -7/12 & 1 & \\ & & -5/6 & 1/2 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 3 & 1 & 3 & 1 \\ & -5/3 & -1 & -2/3 \\ & & 12/5 & -12/15 \\ & & & 1 \\ & & & -2 \\ & & & 7/12 \\ & & & 37/24 \end{pmatrix},$$

$$P = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

La matrice  $U$  ha ampiezza di banda  $3 = p + q$ ;  $L$  ha ampiezza di banda 3, ma ha in ciascuna colonna al più 2 elementi non nulli oltre a quello diagonale. ♣

**Teorema 2.13.** *Sia  $A$  una matrice a banda di dimensione  $n \times n$ , con ampiezza di banda inferiore  $p$  ed ampiezza di banda superiore  $q$ . Eseguendo la fattorizzazione  $LU$  con pivoting parziale di  $A$ , si ottiene:*

$$PA = LU,$$

dove  $U$  è una matrice triangolare superiore a banda con ampiezza di banda uguale a  $p+q$ ,  $L$  è una matrice triangolare inferiore, con al più  $p$  elementi non nulli in ciascuna colonna oltre all'elemento diagonale, e  $P$  è una matrice di permutazione.

**Dimostrazione** Sia  $PA = LU$  la fattorizzazione calcolata dall'algoritmo di eliminazione di Gauss con pivoting parziale e si ricordi che  $P = P_{n-1} \cdots P_1$ . Si ponga  $P^T = (e_{s_1}, \dots, e_{s_n})$  dove  $(s_1, \dots, s_n)$  è una permutazione degli indici  $(1, \dots, n)$ . Se  $s_i > i + p$  allora si ha che il minore principale di ordine  $i$  della matrice  $PA$  è singolare, poiché  $(PA)_{i,j} = a_{s_i,j} \quad j = 1, \dots, s_i - p - 1$  con  $s_i - p - 1 \geq i$ . Ciò comporta l'assurdo che  $A$  e  $U$  sono singolari. Quindi  $s_i \leq i + p$  e di conseguenza  $PA$  ha ampiezza di banda  $p + q$ . Dal teorema 2.12 si ha allora che  $U$  ha ampiezza di banda superiore  $p + q$ . ■

Il pivoting distrugge la struttura a banda della matrice  $A$ , nel senso che l'ampiezza di banda superiore di  $U$  diventa maggiore di quella di  $A$ , mentre nulla si può dire sulla banda di  $L$  (Fig. 2.24). Se si vuole eseguire una fattorizzazione in place non si può quindi memorizzare  $A$  in un array di dimensione  $(p + q + 1) \times n$ , perché non c'è spazio sufficiente per conservare tutta la banda di  $U$ . Dato che  $L$  ha al più  $p$  elementi non nulli in ciascuna colonna, oltre a quello diagonale, lo spazio utilizzato per memorizzare la banda inferiore di  $A$  è adatto a contenere tali elementi; bisogna però “ricordare” quali righe sono state scambiate durante la fattorizzazione, in modo da poter ricostruire la posizione effettiva degli elementi di  $L$ . Per eseguire la fattorizzazione  $LU$  in place, si può dunque utilizzare un array di dimensione  $(2p + q + 1) \times n$ , in cui la matrice  $A$  è memorizzata secondo lo schema a banda, ma supponendo che l'ampiezza di banda superiore sia  $p + q$ , in modo da riservare spazio per gli elementi della banda di  $U$ .

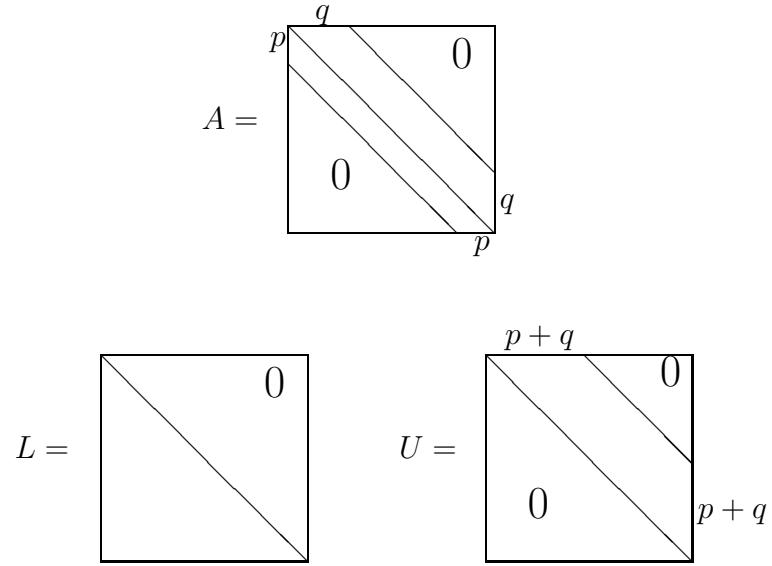
#### ♣ Esempio 2.49.

Si considerino nuovamente la matrice  $A$  dell'esempio 2.48, di dimensione  $n \times n = 5 \times 5$  e con ampiezze di banda  $p = 2$  e  $q = 1$ :

$$A = \begin{pmatrix} 1 & 1 & & & \\ 2 & -1 & 1 & & \\ 3 & 1 & 3 & 1 & \\ & 1 & 3 & -2 & 1 \\ & & -2 & 1 & 1 \end{pmatrix},$$

e le matrici  $L$ ,  $U$  e  $P$  ottenute mediante la fattorizzazione  $LU$  di  $A$  con pivoting:

$$L = \begin{pmatrix} 1 & & & & \\ 2/3 & 1 & & & \\ & -3/5 & 1 & & \\ 1/3 & -2/5 & -7/12 & 1 & \\ & & -5/6 & 1/2 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 3 & 1 & 3 & 1 & \\ & -5/3 & -1 & -2/3 & \\ & & 12/5 & -12/5 & 1 \\ & & & -2 & 7/12 \\ & & & & 37/24 \end{pmatrix},$$

Figura 2.24: Fattorizzazione  $LU$  con pivoting parziale di una matrice a banda.

$$P = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Se si considera un array  $AB$ , di dimensione  $(2p + q + 1) \times n = 6 \times 5$ , e si memorizza  $A$  in  $AB$  nel modo seguente:<sup>22</sup>

$$AB = \begin{pmatrix} * & * & * & + & + \\ * & * & + & + & + \\ * & 1 & 1 & 1 & 1 \\ 1 & -1 & 3 & -2 & 1 \\ 2 & 1 & 3 & 1 & * \\ 3 & 1 & -2 & * & * \end{pmatrix},$$

si può utilizzare tale array per memorizzare anche  $L$  ed  $U$ :

$$AB = \begin{pmatrix} * & * & * & 1 & 0 \\ * & * & 3 & -2/3 & 1 \\ * & 1 & -1 & -12/5 & 7/12 \\ 3 & -5/3 & 12/5 & -2 & 37/24 \\ 2/3 & -3/5 & -7/12 & 1/2 & * \\ 1/3 & -2/5 & -5/6 & * & * \end{pmatrix}$$

(i simboli \* e + indicano rispettivamente le componenti di  $AB$  che non sono proprio usate e quelle che sono usate da  $U$ , ma non da  $A$ ).

<sup>22</sup>In questo caso lo schema di memorizzazione a banda non è conveniente, in quanto il numero di componenti dell'array  $AB$  risulta maggiore del numero di elementi della matrice  $A$ ; ciò è dovuto al fatto che  $p$  e  $q$  sono "prossimi" a  $n/2$ . Lo schema a banda, infatti, fornisce un effettivo vantaggio in termini di complessità computazionale, rispetto all'usuale schema di memorizzazione di una matrice, se  $p, q << n/2$ .

La matrice  $P$  indica la permutazione che bisogna applicare alle righe di  $A$  per ottenere una matrice uguale al prodotto  $LU$ , e quindi indica la posizione di ciascuna riga al termine dell'algoritmo di fattorizzazione. Come già visto nel caso di matrici generiche, tale matrice non viene effettivamente costruita; si costruisce piuttosto un vettore di puntatori che contiene in maniera più compatta le stesse informazioni di  $P$ . ◆

Tenendo conto delle considerazioni precedenti si ottiene l'algoritmo descritto nella Procedura 2.20, che esegue in place la fattorizzazione  $LU$  con pivoting di una matrice a banda, memorizzando  $A$  ed i fattori  $L$  ed  $U$  secondo lo schema a banda, in un array di dimensione  $(2p + q + 1) \times n$ .

```

procedure bandlupiv (in:  $n, p, q, AB$ ; out:  $AB$ )
  /# SCOPO: effettua la fattorizzazione LU
  con pivoting di una matrice a banda A.

  /# SPECIFICHE DEI PARAMETRI:
  /# PARAMETRI DI INPUT:

  var:  $n$  : intero { prima dimensione della }
  { matrice da fattorizzare }

  var:  $AB(p + q + 1, n)$  : reale { matrice da fattorizzare }

  var:  $p$  : intero { ampiezza della banda }
  { inferiore di A }

  var:  $q$  : intero { ampiezza della banda }
  { superiore di A }

  var:  $b(n)$  : reale { vettore dei termini noti }

  /# PARAMETRI DI OUTPUT:

  var:  $AB(p + q + 1, n)$  : reale { matrice fattorizzata }

  /# VARIABILI LOCALI:

  var:  $i, j, k, s$  : interi
  var:  $t$  : reale

  /# INIZIO ISTRUZIONI:
   $s := p + q$ 
  for  $k = 1$  to  $n - 1$ 
     $ip := \min \{r : |AB(s + r, k)| = \max_{1 \leq i \leq p+1} |AB(s + i, k)|\}$ 
     $ipiv(k) := ip + k - 1$ 
    if ( $ip \neq 1$ )
      for  $j = k$  to  $\min(k + q + ip - 1, n)$ 
         $t := AB(s + 1 + k - j, j)$ 
         $AB(s + 1 + k - j, j) := AB(s + ip + k - j, j)$ 
         $AB(s + ip + k - j, j) := t$ 

```

Procedura 2.20: Fattorizzazione  $LU$  con pivoting parziale  
di una matrice a banda - continua

```

    endfor
  endif
  for  $i = k + 1$  to  $\min(k + p, n)$ 
     $AB(s + 1 + i - k, k) := AB(s + 1 + i - k, k) / AB(s + 1, k)$ 
  endfor
  for  $j = k + 1$  to  $\min(k + s, n)$ 
    for  $i = k + 1$  to  $\min(k + p, n)$ 
       $AB(s + 1 + i - j, j) := AB(s + 1 + i - j, j) +$ 
       $- AB(s + 1 + i - k, k) * AB(s + 1 + k - j, j)$ 
    endfor
  endfor
endfor
end bandlupiv

```

**Procedura 2.20: Fattorizzazione  $LU$  con pivoting parziale  
di una matrice a banda - fine**

Si osservi che  $ipiv(k) = l$  indica che al passo  $k$  la riga  $k$ -ma è scambiata con la riga  $l$ -ma; l'array  $ipiv$  è dunque usato diversamente rispetto al caso generale, in cui  $ipiv(k) = l$  indica che al termine dell'algoritmo di fattorizzazione la riga  $k$ -ma si trova nella  $l$ -ma posizione. Per comprendere meglio la differenza tra i due modi di usare  $ipiv$ , consideriamo la fattorizzazione  $LU$  nell'esempio 2.48. Se si usa  $ipiv$  per ricordare lo scambio di righe eseguito al generico passo  $k$ , si ha:

$$ipiv = (3, 2, 4, 4);$$

mentre, in generale  $ipiv$  è:

$$ipiv = (3, 2, 4, 1, 5).$$

Nel primo caso, ad esempio,  $ipiv(4) = 4$  indica che al quarto passo dell'algoritmo la quarta riga non è stata scambiata con alcuna riga, mentre, nel secondo caso  $ipiv(4) = 1$  indica che la prima riga di  $A$  è diventata la quarta al termine della fattorizzazione. Inoltre, dato che l'algoritmo di fattorizzazione consta di 4 passi, il primo array  $ipiv$  ha una componente in meno.

Si osservi inoltre che, al passo  $k$ , lo scambio di righe è eseguito solo sulla sottomatrice attiva, lasciando quindi inalterata la posizione degli elementi di  $L$  appartenenti alle colonne precedenti la  $k$ -ma, in accordo col fatto che  $ipiv(k)$  contiene una informazione relativa al solo passo  $k$ . Tale scelta è coerente anche con la memorizzazione degli elementi subdiagonali di  $L$ , che sono posti nelle ultime  $p$  righe dell'array  $AB$ , senza tener conto della riga alla quale essi appartengono.

Si noti, infine, che nella Procedura 2.20 si suppone che siano state inizialmente azzerate le componenti delle prime  $p$  righe di  $AB$  riservate alle  $p$  diagonali superiori

$U$  generate dalla fattorizzazione; se così non fosse, in seguito agli scambi di righe, si potrebbero avere istruzioni di assegnazione il cui secondo membro contiene variabili non definite.

La Procedura 2.20 si applica, ovviamente, anche ad una matrice  $A$  tridiagonale. Se però si assume che  $A$  sia memorizzata mediante tre array monodimensionali, si può considerare un ulteriore array monodimensionale per gli elementi della seconda diagonale superiore di  $U$  e si possono usare i tre array di  $A$  per gli elementi extradiagonali di  $L$  e per la diagonale principale e la prima diagonale superiore di  $U$ . In questo caso, l'algoritmo va modificato sostituendo opportunamente ad  $AB$  i quattro vettori suddetti.

Con un procedimento analogo a quello della fattorizzazione  $LU$  senza pivoting, si verifica che la Procedura 2.20 ha la complessità di spazio:

$$T_{LUband+piv}(n, p, q) = \mathcal{O}(np(p + q + 1)) \text{ flop};$$

dato che la fattorizzazione è eseguita in place, lo spazio di memoria è essenzialmente quello richiesto dall'array  $AB$ , e quindi la complessità di spazio è:

$$S_{LUband+piv}(n, p, q) = \mathcal{O}((2p + q + 1)n).$$

Si consideri ora la risoluzione dei sistemi lineari  $Ly = Pb$  ed  $Ux = y$ , derivanti dalla fattorizzazione  $LU$  con pivoting della matrice a banda  $A$ , ottenuta con la Procedura 2.20.

L'algoritmo di forward substitution per la risoluzione di  $Ly = Pb$  deve tener conto della effettiva posizione degli elementi di  $L$  e deve associare ad ogni riga di  $L$  il termine noto giusto, utilizzando le informazioni contenute in  $ipiv$ .

#### ♣ Esempio 2.50.

Si vuole risolvere il sistema  $Ly = Pb$ , dove  $L$  e  $P$  sono le matrici derivanti dalla fattorizzazione  $LU$  della matrice  $A$  dell'esempio 2.48 e  $b$  è il seguente vettore:

$$b = (1, 2, -1, -2, 1)^T.$$

Se si suppone che  $L$  sia stata calcolata eseguendo la Procedura 2.20, si ha che gli elementi di  $L$  che non appartengono alla diagonale principale sono memorizzati nelle ultime due righe di un array  $AB$ , di dimensione  $6 \times 5$ , nel modo seguente:

$$AB = \begin{pmatrix} * & * & * & + & + \\ * & * & + & + & + \\ * & + & + & + & + \\ + & + & + & + & + \\ 2/3 & -2/5 & -7/12 & +1/2 & * \\ 1/3 & -3/5 & -5/6 & * & * \end{pmatrix},$$

dove il simbolo  $+$  indica un elemento del fattore  $U$ , e la matrice  $P$  è rappresentata in maniera compatta dal vettore

$$ipiv = (3, 2, 4, 1).$$

la soluzione del sistema si può calcolare con i passi seguenti:

**passo 1:**

- scambio delle componenti di  $b$  e determinazione di  $y_1$ :  $ipiv(1) = 3$  indica che al passo 1 dell'algoritmo di fattorizzazione  $LU$  la prima riga di  $a$  è stata scambiata con la terza; bisogna quindi eseguire lo stesso scambio su  $b$ :

$$b = y^{(0)} = \begin{pmatrix} 1 \\ 2 \\ -1 \\ -2 \\ 1 \end{pmatrix} \longrightarrow \bar{y}^{(0)} = \begin{pmatrix} -1 \\ 2 \\ 1 \\ -2 \\ 1 \end{pmatrix}$$

e risulta

$$y_1 = \bar{y}_1^{(0)} = -1;$$

- aggiornamento del vettore soluzione, utilizzando il valore calcolato di  $y_1$ :

$$y^{(1)} = \begin{pmatrix} y_1^{(1)} \\ y_2^{(1)} \\ y_3^{(1)} \\ y_4^{(1)} \\ y_5^{(1)} \end{pmatrix},$$

con

$$\begin{aligned} y_1^{(1)} &= \bar{y}_1^{(0)} = -1, \\ y_2^{(1)} &= \bar{y}_2^{(0)} - AB(5,1)y_1^{(1)} = 2 + \frac{2}{3} = \frac{8}{3}, \\ y_3^{(1)} &= \bar{y}_3^{(0)} - AB(6,1)y_1^{(1)} = 1 + \frac{1}{3} = \frac{4}{3}, \\ y_4^{(1)} &= \bar{y}_4^{(0)} = -2, \\ y_5^{(1)} &= \bar{y}_5^{(0)} = 1; \end{aligned}$$

**passo 2:**

- scambio delle componenti di  $y^{(1)}$  e determinazione di  $y_2$ :  $ipiv(2) = 2$  e quindi non si esegue alcuno scambio, ovvero

$$\bar{y}^{(1)} = y^{(1)} = \begin{pmatrix} -1 \\ 8/3 \\ 4/3 \\ -2 \\ 1 \end{pmatrix},$$

e risulta

$$y_2 = \bar{y}_2^{(1)} = \frac{8}{3};$$

- aggiornamento del vettore soluzione, utilizzando il valore calcolato di  $y_2$ :

$$y^{(2)} = \begin{pmatrix} y_1^{(2)} \\ y_2^{(2)} \\ y_3^{(2)} \\ y_4^{(2)} \\ y_5^{(2)} \end{pmatrix},$$

con

$$\begin{aligned} y_1^{(2)} &= y_1 = -1, \\ y_2^{(2)} &= y_2 = \frac{8}{3}, \\ y_3^{(2)} &= \bar{y}_3^{(1)} - AB(5,2)y_2^{(2)} = \frac{4}{3} + \frac{2}{5} \cdot \frac{8}{3} = \frac{12}{5}, \\ y_4^{(2)} &= \bar{y}_4^{(1)} - AB(6,2)y_2^{(2)} = -2 + \frac{3}{5} \cdot \frac{8}{3} = -\frac{2}{5}, \\ y_5^{(2)} &= \bar{y}_5^{(1)} = 1; \end{aligned}$$

**passo 3:**

- scambio delle componenti di  $y^{(2)}$  e determinazione di  $y_3$ :  $ipiv(3) = 4$  e quindi si scambiano la terza e la quarta riga di  $y^{(2)}$ , ovvero

$$y^{(2)} = \begin{pmatrix} -1 \\ 8/3 \\ 12/5 \\ -2/5 \\ 1 \end{pmatrix} \longrightarrow \bar{y}^{(2)} = \begin{pmatrix} -1 \\ 8/3 \\ -2/5 \\ 12/5 \\ 1 \end{pmatrix}$$

e risulta

$$y_3 = \bar{y}_3^{(2)} = -\frac{2}{5};$$

- aggiornamento del vettore soluzione, utilizzando il valore calcolato di  $y_3$ :

$$y^{(3)} = \begin{pmatrix} y_1^{(3)} \\ y_2^{(3)} \\ y_3^{(3)} \\ y_4^{(3)} \\ y_5^{(3)} \end{pmatrix},$$

con

$$\begin{aligned} y_1^{(3)} &= y_1 = -1, \\ y_2^{(3)} &= y_2 = \frac{8}{3}, \\ y_3^{(3)} &= y_3 = -\frac{2}{5}, \\ y_4^{(3)} &= \bar{y}_4^{(2)} - AB(5,3)y_3^{(3)} = \frac{12}{5} - \frac{7}{12} \cdot \frac{2}{5} = \frac{13}{6}, \\ y_5^{(3)} &= \bar{y}_5^{(2)} - AB(6,3)y_3^{(3)} = 1 + \frac{5}{6} \cdot \left(-\frac{2}{5}\right) = -\frac{2}{3}; \end{aligned}$$

**passo 4:**

- scambio delle componenti di  $y^{(3)}$  e determinazione di  $y_4$ :  $ipiv(4) = 4$  e quindi non si esegue alcuno scambio, ovvero

$$\bar{y}^{(3)} = y^{(3)} = \begin{pmatrix} -1 \\ 8/3 \\ -2/5 \\ 13/6 \\ 2/3 \end{pmatrix},$$

e risulta

$$y_4 = \bar{y}_4^{(3)} = \frac{13}{6};$$

- aggiornamento del vettore soluzione, utilizzando il valore calcolato di  $y_4$ :

$$y^{(4)} = \begin{pmatrix} y_1^{(4)} \\ y_2^{(4)} \\ y_3^{(4)} \\ y_4^{(4)} \\ y_5^{(4)} \end{pmatrix},$$

con

$$\begin{aligned} y_1^{(4)} &= y_1 = -1, \\ y_2^{(4)} &= y_2 = \frac{8}{3}, \\ y_3^{(4)} &= y_3 = -\frac{2}{5}, \\ y_4^{(4)} &= y_4 = \frac{13}{6}, \\ y_5^{(4)} &= \bar{y}_5^{(3)} - AB(5,4)y_4^{(4)} = \frac{2}{3} - \frac{1}{2} \cdot \frac{13}{6} = -\frac{5}{12}. \end{aligned}$$

Si osservi che risulta

$$y_5 = y_5^{(4)} = -\frac{5}{12}.$$



In generale, la soluzione del sistema  $Ly = Pb$  si può calcolare con una back substitution per colonne, in cui al generico passo  $j$  si “aggiornano” le componenti del vettore soluzione a partire dalla  $j$ -ma e si scambia la componente di posto  $j$  del vettore aggior- nato con quella di posto  $ipiv(j)$ , ottenendo la  $j$ -ma componente del vettore soluzione<sup>23</sup>, come mostrato nella Procedura 2.21.

---

<sup>23</sup>Gli elementi diagonali della matrice  $L$  sono uguali ad 1.

```

procedure forsub3d (in:  $n, b, AB, p, q, ipiv;$  out:  $b)$ 

/# SCOPO : risolve un sistema con matrice a banda A,
derivante da fattorizzazione LU con pivotin
parziale, utilizzando la forward substitution.

/# SPECIFICHE DEI PARAMETRI:
/# PARAMETRI DI INPUT:

var:  $n$  : intero { dimensione del sistema }
var:  $AB(p + q + 1, n)$  : reale { matrice da fattorizzare }
var:  $p$  : intero { ampiezza della banda }
{ inferiore di  $A$  }
var:  $q$  : intero { ampiezza della banda }
{ superiore di  $A$  }
var:  $b(n)$  : reale { vettore dei termini noti }
var:  $ipiv(n)$  : reale { vettore di permutazione }

/# PARAMETRI DI OUTPUT:

var:  $b(n)$  : reale { vettore risultante }

/# VARIABILI LOCALI:

var:  $i, j, s$  : interi
var:  $t$  : reale

/# INIZIO ISTRUZIONI:
 $s := p + q$ 
for  $j = 1$  to  $n - 1$ 

```

Procedura 2.21: Forward substitution per un sistema  
con matrice triangolare inferiore derivante dalla fattorizzazione *LU*  
con pivoting parziale di una matrice a banda - continua

```

if ( $j \neq ipiv(j)$ ) then
     $t := b(j)$ 
     $b(j) := b(ipiv(j))$ 
     $b(ipiv(j)) := t$ 
endif
for  $i = j + 1$  to  $\min(j + p, n)$  { aggiornamento componenti }
     $b(i) := b(i) - AB(q + 1 + i - j, j) * b(j)$ 
endfor
endfor
end forsub3d

```

Procedura 2.21: Forward substitution per un sistema  
con matrice triangolare inferiore derivante dalla fattorizzazione  $LU$   
con pivoting parziale di una matrice a banda - fine

Tale algoritmo ha la stessa complessità di tempo della Procedura 2.19 che risolve il sistema  $Ly = b$  derivante dalla fattorizzazione senza pivoting:

$$T_{FORWband+piv}(n, p) = \mathcal{O}(np) \text{ flop}.$$

Per quanto riguarda il sistema  $Ux = y$ , dato che gli scambi sulla matrice  $U$  sono stati effettivamente eseguiti durante la fattorizzazione, la soluzione si può calcolare con la Procedura 2.20, considerando  $q + p$  al posto di  $p$ . La complessità di tempo in questo caso è

$$T_{BACKband+piv}(n, p + q) = \mathcal{O}(n(p + q)) \text{ flop}.$$

La complessità di spazio dei due algoritmi è

$$S_{FORWband+piv}(n, p) = S_{BACKband+piv}(n, p + q) = \mathcal{O}(n(p + q)),$$

in quanto entrambi considerano l'intero array  $AB$ , anche se operano solo su parte di esso.

## 2.12 Fattorizzazione di matrici simmetriche

### 2.12.1 Fattorizzazione $LDL^T$

Le matrici simmetriche, come quelle a banda, sono dotate di una struttura, ed è naturale chiedersi se si possa trarre qualche vantaggio da essa nell'esecuzione della fattorizzazione  $LU$ .

#### ♣ Esempio 2.51.

Si calcoli la fattorizzazione  $LU$  della matrice simmetrica

$$A = \begin{pmatrix} 2 & 1 & 3 & 1 \\ 1 & 1 & 2 & 1 \\ 3 & 2 & -2 & 3 \\ 1 & 1 & 3 & 4 \end{pmatrix}.$$

Applicando l'algoritmo di eliminazione di Gauss senza pivoting, si ha:

**passo 1:**

- calcolo dei moltiplicatori relativi alla prima colonna:  $m_{2,1} = 1/2$ ,  $m_{3,1} = 3/2$ ,  $m_{4,1} = 1/2$ ;
- trasformazione della sottomatrice attiva:

$$A^{(1)} = \begin{pmatrix} 2 & 1 & 3 & 1 \\ 1/2 & 1/2 & 1/2 & 1/2 \\ 1/2 & -13/2 & 3/2 \\ 1/2 & 3/2 & 7/2 \end{pmatrix}$$

**passo 2:**

- calcolo dei moltiplicatori relativi alla seconda colonna:  $m_{3,2} = 1$ ,  $m_{4,2} = 1$ ;
- trasformazione della sottomatrice attiva:

$$A^{(2)} = \begin{pmatrix} 2 & 1 & 3 & 1 \\ 1/2 & 1/2 & 1/2 & 1/2 \\ -7 & 1 & & \\ 1 & 3 & & \end{pmatrix};$$

**passo 3:**

- calcolo del moltiplicatore relativo alla terza colonna:  $m_{4,3} = -1/7$ ;
- trasformazione della sottomatrice attiva:

$$A^{(3)} = U \begin{pmatrix} 2 & 1 & 3 & 1 \\ 1/2 & 1/2 & 1/2 & 1/2 \\ -7 & 1 & & \\ & & 22/7 & \end{pmatrix}.$$

Si noti che, ad ogni passo, la sottomatrice trasformata, sulla quale si deve operare al passo successivo, è simmetrica; è quindi possibile calcolarne solo gli elementi diagonali e quelli che si trovano al di sopra, o al di sotto, della diagonale principale. Ciò consente di dimezzare approssimativamente la complessità di tempo dell'algoritmo di eliminazione di Gauss.

L'algoritmo precedente calcola i fattori  $L$  ed  $U$ :

$$L = \begin{pmatrix} 1 & & & \\ 1/2 & 1 & & \\ 3/2 & 1 & 1 & \\ 1/2 & 1 & -1/7 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 2 & 1 & 3 & 1 \\ & 1/2 & 1/2 & 1/2 \\ & & -7 & 1 \\ & & & 22/7 \end{pmatrix}.$$

Posto

$$D = \text{diag}(2, 1/2, -7, 22/7),$$

si ha:

$$D^{-1}U = \begin{pmatrix} 1 & 1/2 & 3/2 & 1/2 \\ & 1 & 1 & 1 \\ & & 1 & -1/7 \\ & & & 1 \end{pmatrix} = L^T;$$

la matrice  $A$  risulta quindi fattorizzata nel modo seguente:

$$A = LDL^T,$$

dove  $L$  è la matrice unitaria triangolare inferiore ottenuta con la fattorizzazione  $LU$  senza pivoting e  $D$  è una matrice diagonale, i cui elementi sulla diagonale principale coincidono con i corrispondenti elementi della matrice  $U$ , risultante dalla fattorizzazione suddetta.

♦

Applicando ad una matrice simmetrica l'algoritmo di eliminazione di Gauss senza pivoting, si ottengono sottomatrici attive simmetriche. Inoltre, a partire dalla fattorizzazione  $LU$  di una matrice simmetrica  $A$ , si è pervenuti ad un'altra fattorizzazione,  $A = LDL^T$ , che tiene conto della proprietà di simmetria. I risultati ottenuti valgono in generale per le matrici simmetriche che ammettono la fattorizzazione  $LU$ .

**Teorema 2.14.** *Sia  $A \in \mathbb{R}^{n \times n}$  una matrice simmetrica. Allora, se  $A$  ammette la fattorizzazione  $LU$ , si ha:*

$$a_{i,j}^{(k)} = a_{j,i}^{(k)}, \quad k = 0, \dots, n-1, \quad i, j = k+1, \dots, n,$$

dove  $a_{i,j}^{(k)}$  è il generico elemento della matrice  $A^{(k)}$  ottenuta al  $k$ -mo passo dell'algoritmo di eliminazione di Gauss, e  $A^{(0)} = A$ .

#### Dimostrazione

La dimostrazione è basata sul principio di induzione.

Per  $k = 0$  la tesi è vera in quanto  $A^{(0)} = A$ . Si supponga ora che la tesi sia vera per  $k = s-1$ . Sfruttando tale ipotesi, per  $i, j = s+1, \dots, n$  si ha:

$$a_{i,j}^{(s)} = a_{i,j}^{(s-1)} - \frac{a_{i,s}^{(s-1)}}{a_{s,s}^{(s-1)}} a_{s,j}^{(s-1)} = a_{j,i}^{(s-1)} - \frac{a_{s,j}^{(s-1)}}{a_{s,s}^{(s-1)}} a_{i,s}^{(s-1)} = a_{j,i}^{(s-1)} - \frac{a_{j,s}^{(s-1)}}{a_{s,s}^{(s-1)}} a_{s,i}^{(s-1)} = a_{j,i}^{(s)},$$

e quindi la tesi è vera anche per  $k = s$ . La tesi è dunque vera per  $k = 0, \dots, n-1$ . ■

### Teorema 2.15. [Fattorizzazione $LDL^T$ ]

Sia  $A \in \Re^{n \times n}$  una matrice simmetrica tale che tutte le sue sottomatrici principali sono non singolari. Esistono allora una ed una sola matrice  $L \in \Re^{n \times n}$  triangolare unitaria inferiore ed una ed una sola matrice  $D \in \Re^{n \times n}$  diagonale tali che

$$A = LDL^T.$$

Inoltre,  $L$  coincide con la matrice triangolare inferiore della fattorizzazione  $LU$  di  $A$  e gli elementi diagonali di  $D$  coincidono con i corrispondenti elementi diagonali della matrice triangolare superiore della fattorizzazione  $LU$  di  $A$ , ovvero  $L^T = D^{-1}U$ .

#### Dimostrazione

Per ipotesi,  $A$  ammette una ed una sola fattorizzazione  $LU$ , con  $L \in \Re^{n \times n}$  triangolare unitaria inferiore ed  $U \in \Re^{n \times n}$  triangolare superiore non singolare. Posto

$$D = \text{diag}(u_{1,1}, \dots, u_{n,n}),$$

dove  $u_{i,i}$  è l' $i$ -mo elemento diagonale di  $U$ , e

$$M^T = D^{-1}U,$$

si ha:

$$A = LDM^T, \quad (2.59)$$

con  $M$  triangolare unitaria inferiore.

Si vuole ora dimostrare che risulta  $L = M$ . Moltiplicando ambo i membri di (2.59) a sinistra per  $M^{-1}$  ed a destra per  $(M^T)^{-1}$ , si ha:

$$M^{-1}A(M^T)^{-1} = M^{-1}LD.$$

Dato che  $M^{-1}A(M^T)^{-1}$  è una matrice simmetrica e  $M^{-1}LD$  è una matrice triangolare inferiore, dalla relazione precedente discende che  $M^{-1}L$  è una matrice diagonale; inoltre,  $M^{-1}L$  è una matrice unitaria, in quanto prodotto di matrici triangolari inferiori unitarie, e quindi  $M^{-1}L = I$ , con  $I \in \Re^{n \times n}$  matrice identica, ovvero

$$M = L.$$

L'unicità delle matici  $L$  e  $D$  discende dall'unicità della fattorizzazione  $LU$ . Si supponga infatti che esistano  $L_1, L_2 \in \Re^{n \times n}$  triangolari unitarie inferiori e  $D_1, D_2 \in \Re^{n \times n}$  diagonali tali che

$$A = L_1 D_1 L_1^T = L_2 D_2 L_2^T.$$

Posto  $U_1 = D_1 L_1^T$  e  $U_2 = D_2 L_2^T$ , si ha:

$$A = L_1 U_1 = L_2 U_2,$$

con  $U_1$  e  $U_2$  triangolari superiori, e quindi, per l'unicità della fattorizzazione  $LU$  risulta

$$L_1 = L_2, \quad U_1 = U_2$$

e quindi

$$L_1 D_1 = L_2 D_2.$$

Poiché  $L_1 = L_2$ , risulta anche  $D_1 = D_2$ . ■

I risultati precedenti suggeriscono che il calcolo della fattorizzazione  $LDL^T$  si può eseguire modificando opportunamente l'algoritmo di eliminazione di Gauss, utilizzato per ottenere la fattorizzazione  $LU$ . Inoltre, grazie alla simmetria della matrice  $A$  e delle sottomatrici attive, ad ogni passo, dopo avere calcolato i moltiplicatori, si può operare solo sul triangolo superiore della sottomatrice attiva corrente. L'algoritmo si può eseguire *in place*, memorizzando gli elementi del triangolo superiore di  $A$  al posto dei corrispondenti elementi di  $L^T$  e gli elementi diagonali di  $D$  al posto di quelli di  $A$ ; gli elementi diagonali di  $L$ , essendo tutti uguali ad 1, non sono memorizzati. Una descrizione di tale algoritmo è fornita di seguito.

```

procedure ldlt (in:  $n, a$ ; out:  $a$ )
  /* SCOPO: effettua la fattorizzazione  $LDL^T$ 
  /* SPECIFICHE DEI PARAMETRI:
  /* PARAMETRI DI INPUT:
  var:  $n$  : intero { dimensione del sistema }
  var:  $a(n, n)$  : reale { matrice del sistema }
    { da fattorizzare }

  /* PARAMETRI DI OUTPUT:
  var:  $a(n, n)$  : reale { matrice del sistema }
    { fattorizzata }

  /* VARIABILI LOCALI:
  var:  $i, j, k$  : interi
  var:  $m$  : reale

  /* INIZIO ISTRUZIONI:
  for  $k = 1$  to  $n - 1$ 
    for  $i = k + 1$  to  $n$ 
       $m := a(k, i) / a(k, k)$ 
      for  $j = i$  to  $n$ 
         $a(i, j) := a(i, j) - m * a(k, j)$ 
      endfor
       $a(k, i) := m$ 
    endfor
  endfor
end ldlt

```

Procedura 2.22: Fattorizzazione  $LDL^T$  di una matrice tridiagonale - fine

L'algoritmo precedente consta di  $n - 1$  passi, individuati dal contatore  $k$ . Al  $k$ -passo si eseguono:

- $n - k$  divisioni floating-point per calcolare i moltiplicatori,  $m$ , ovvero gli elementi subdiagonali della  $k$ -ma colonna di  $L$ ;

- $\sum_{i=k+1}^n (n-i+1) = \sum_{i=1}^{n-k} i = (n-k)(n-k+1)/2$  addizioni ed altrettante moltiplicazioni floating-point per trasformare il triangolo superiore della sottomatrice attiva.

L'algoritmo richiede in totale

$$\begin{aligned} \sum_{k=1}^{n-1} \left( n - k + 2 \frac{(n-k)(n-k+1)}{2} \right) &= \sum_{k=1}^{n-1} k + \sum_{k=1}^{n-1} k(k+1) = \\ &= 2 \frac{n(n-1)}{2} + \frac{n(n-1)(2n-1)}{6} = n^2 - n + \frac{2n^3 - 3n^2 + n}{6} = \\ &= \frac{2n^3 + 3n^2 - 5n}{6} \end{aligned}$$

operazioni aritmetiche floating-point. La complessità di tempo è dunque circa la metà di quella dell'algoritmo utilizzato per costruire i fattori  $L$  ed  $U$  di una matrice quadrata:

$$T_{LDL^T}(n) = \mathcal{O}\left(\frac{n^3}{6}\right) \text{ flop.}$$

La complessità di spazio dell'algoritmo è quella dell'algoritmo di eliminazione di Gauss per matrici generiche, cioè  $\mathcal{O}(n^2)$ , a meno che non si utilizzi uno schema di memorizzazione *packed* (cfr. §2.10); in quest'ultimo caso la complessità di spazio è

$$S_{LDL^T}(n) = \mathcal{O}\left(\frac{n^2}{2}\right).$$

Si osservi che la Procedura 2.22 costruisce esplicitamente il triangolo superiore della matrice  $L^T$  e lo memorizza sul triangolo superiore della matrice di partenza. Una variante di tale algoritmo, che costruisce direttamente la matrice  $L$ , memorizzandola sul triangolo inferiore di  $A$ , si può ricavare imponendo l'uguaglianza tra gli elementi della matrice  $LDL^T$  ed i corrispondenti elementi di  $A$ , con un ragionamento analogo a quello utilizzato, ad esempio, per l'algoritmo di fattorizzazione  $LU$  di una matrice tridiagonale.

La risoluzione di un sistema lineare  $Ax = b$ , con matrice dei coefficienti simmetrica fattorizzata come  $A = LDL^T$ , si ottiene risolvendo due sistemi triangolari ed uno diagonale:

$$Ly = b, \quad Dw = y, \quad L^T x = w;$$

si noti che la risoluzione del sistema  $Ly = b$  può essere eseguita senza trasporre effettivamente la matrice  $L^T$  costruita con l'algoritmo precedente.

L'algoritmo di eliminazione di Gauss è stato utilizzato per costruire la fattorizzazione  $LDL^T$  senza applicare alcuna strategia di pivoting e risulta, in generale, instabile. Dunque, anche se una matrice simmetrica  $A$  è dotata di fattorizzazione  $LDL^T$ , l'esecuzione della Procedura 2.22 in un sistema aritmetico a precisione finita, può condurre a risultati

inaccettabili. Si vuole dunque studiare come si comporta l'algoritmo di eliminazione di Gauss con pivoting parziale quando viene eseguita su una matrice simmetrica.

♣ **Esempio 2.52.**

Si applichi l'algoritmo di eliminazione di Gauss con pivoting parziale alla matrice:

$$A = \begin{pmatrix} 1/2 & 1 \\ 1 & 2 \end{pmatrix}.$$

L'esecuzione del pivoting parziale al primo passo dell'algoritmo genera la matrice

$$A' = \begin{pmatrix} 1 & 2 \\ 1/2 & 1 \end{pmatrix},$$

che non è più simmetrica.



In generale, applicando ad una matrice simmetrica l'algoritmo di eliminazione di Gauss con pivoting parziale, si distrugge la simmetria della matrice, perdendo in tal modo i vantaggi, in termini di complessità computazionale, derivanti proprio dalla simmetria. Esistono però alcune classi di matrici che non richiedono l'esecuzione del pivoting.

### 2.12.2 Alcune matrici che non richiedono l'esecuzione del pivoting

♣ **Esempio 2.53.**

Si applichi l'algoritmo di eliminazione di Gauss con pivoting parziale alla matrice:

$$A = \begin{pmatrix} 4 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 4 \end{pmatrix}.$$

Si ha:

**passo 1:**

- pivoting in prima colonna:  $\max_{1 \leq i \leq 3} |a_{i,1}| = 4 = a_{1,1}$  e quindi non si eseguono scambi;
- calcolo dei moltiplicatori:  $m_{2,1} = 1/4$ ,  $m_{3,1} = 1/4$ ;
- trasformazione della sottomatrice attiva:

$$A^{(1)} = \begin{pmatrix} 4 & 1 & 1 \\ 0 & 11/4 & 3/4 \\ 0 & 3/4 & 15/4 \end{pmatrix}.$$

**passo 2:**

- pivoting in seconda colonna:  $\max_{2 \leq i \leq 3} |a_{i,2}^{(1)}| = 11/4 = a_{2,2}^{(1)}$  e quindi non si eseguono scambi;
- calcolo del moltiplicatore:  $m_{3,2} = 3/11$ ;
- trasformazione della sottomatrice attiva:

$$A^{(1)} = \begin{pmatrix} 4 & 1 & 1 \\ 0 & 11/4 & 3/4 \\ 0 & 0 & 39/11 \end{pmatrix}.$$

Si noti che, in entrambi i passi, l'elemento di massimo modulo della colonna da annullare nella sottomatrice attiva si trova sulla diagonale e quindi non bisogna eseguire scambi di righe. 

La matrice dell'esempio precedente non richiede l'applicazione del pivoting nell'esecuzione dell'algoritmo di eliminazione di Gauss. Ciò è dovuto al fatto che tale matrice, oltre ad essere simmetrica, è a diagonale strettamente dominante.

### Definizione 2.13. (Matrice a diagonale dominante)

Una matrice  $A \in \mathbb{R}^{n \times n}$  si dice a diagonale dominante se

$$|a_{i,i}| \geq \sum_{\substack{i, j=1 \\ j \neq i}}^n |a_{i,j}|, \quad i = 1, 2, \dots, n. \quad (2.60)$$

In particolare, se in (2.60) vale il segno  $>$ , la matrice si dice a diagonale strettamente dominante.

Per le matrici simmetriche a diagonale strettamente dominante sussiste il

**Teorema 2.16.** Sia  $A \in R^{n \times n}$  una matrice a diagonale strettamente dominante. Allora la matrice  $A$  è non singolare e, applicando l'algoritmo di eliminazione di Gauss senza pivoting, si ottiene, ad ogni passo, una sottomatrice attiva a diagonale strettamente dominante.

**Dimostrazione** La sottomatrice attiva al  $k$ -mo passo dell'algoritmo di eliminazione di Gauss,  $\bar{A}^{(k-1)}$ , è formata dalle ultime  $n - k - 1$  righe e  $n - k - 1$  colonne della matrice trasformata,  $A^{(k-1)}$ , ottenuta al passo  $k - 1$ :

$$\bar{A}^{(k-1)} = \begin{pmatrix} a_{k,k}^{(k-1)} & \dots & a_{k,n}^{(k-1)} \\ \vdots & \ddots & \vdots \\ a_{n,k}^{(k-1)} & \dots & a_{n,n}^{(k-1)} \end{pmatrix}.$$

Per dimostrare che  $\bar{A}^{(k-1)}$  è a diagonale dominante per ogni  $k = 1, \dots, n$ , si può applicare il principio di induzione.

Per  $k = 1$  risulta  $\bar{A}^{(k-1)} = A$  e quindi la tesi è vera. Si osservi che, essendo  $A$  a diagonale dominante, si ha:

$$|a_{1,1}| > \sum_{\substack{i=1 \\ i \neq j}}^n |a_{i,j}| \geq 0.$$

e quindi  $a_{1,1} \neq 0$ .

Si supponga ora che  $\bar{A}^{(k-2)}$ ,  $k \geq 2$ , sia a diagonale strettamente dominante. Per ogni  $i = k, \dots, n$  si ha:

$$\begin{aligned} \sum_{\substack{j=k \\ j \neq i}}^n |a_{i,j}^{(k-1)}| &= \sum_{\substack{j=k \\ j \neq i}}^n \left| a_{i,j}^{(k-2)} - \frac{a_{i,k-1}^{(k-2)}}{a_{k-1,k-1}^{(k-2)}} a_{k-1,j}^{(k-2)} \right| \leq \\ &\leq \sum_{\substack{j=k \\ j \neq i}}^n |a_{i,j}^{(k-2)}| + \frac{|a_{i,k-1}^{(k-2)}|}{|a_{k-1,k-1}^{(k-2)}|} \sum_{\substack{j=k \\ j \neq i}}^n |a_{k-1,j}^{(k-2)}|, \end{aligned} \quad (2.61)$$

dove  $a_{k-1,k-1}^{(k-2)} \neq 0$  in quanto  $\bar{A}^{(k-2)}$  è a diagonale strettamente dominante. In virtù di questa proprietà risulta inoltre, per  $i = k, \dots, n$ ,

$$\begin{aligned} \sum_{\substack{j=k \\ j \neq i}}^n |a_{i,j}^{(k-2)}| + |a_{i,k-1}^{(k-2)}| &< |a_{i,i}^{(k-2)}|, \\ \sum_{\substack{j=k \\ j \neq i}}^n |a_{k-1,j}^{(k-2)}| + |a_{k-1,i}^{(k-2)}| &< |a_{k-1,k-1}^{(k-2)}|, \end{aligned}$$

e quindi, da (2.61), si ottiene:

$$\begin{aligned} \sum_{\substack{j=k \\ j \neq i}}^n |a_{i,j}^{(k-1)}| &< \left( |a_{i,i}^{(k-2)}| - |a_{i,k-1}^{(k-2)}| \right) + \frac{|a_{i,k-1}^{(k-2)}|}{|a_{k-1,k-1}^{(k-2)}|} \left( |a_{k-1,k-1}^{(k-2)}| - |a_{k-1,i}^{(k-2)}| \right) = \\ &= |a_{i,i}^{(k-2)}| - \frac{|a_{i,k-1}^{(k-2)}|}{|a_{k-1,k-1}^{(k-2)}|} |a_{k-1,i}^{(k-2)}| \leq |a_{i,i}^{(k-2)}| - \frac{a_{i,k-1}^{(k-2)}}{a_{k-1,k-1}^{(k-2)}} a_{k-1,i}^{(k-2)} \leq |a_{i,i}^{(k-1)}|. \end{aligned}$$

La matrice  $\bar{A}^{(k-1)}$  è dunque a diagonale strettamente dominante ed è così dimostrato che tutte le sottomatrici attive sono a diagonale strettamente dominante.

Per dimostrare che  $A$  è non singolare, basta osservare che, al generico passo  $k$  dell'algoritmo di eliminazione di Gauss, risulta

$$a_{k,k}^{(k-1)} \neq 0.$$

■

Una conseguenza dei teoremi 2.16 e 2.14 è che, se una matrice è simmetrica a diagonale dominante, tutte le sottomatrici attive hanno come elemento di massimo modulo

della colonna pivot l'elemento diagonale di tale colonna, e quindi la strategia di pivoting è inutile. La fattorizzazione  $LDL^T$  si può dunque calcolare con la Procedura 2.22.

### 2.12.3 Fattorizzazione di Cholesky

**Definizione 2.14. (Matrice definita positiva)**

Una matrice  $A \in \mathbb{R}^{n \times n}$ , simmetrica, si dice definita positiva se, per ogni  $x \in \mathbb{R}^n$ ,  $x \neq 0$ , risulta

$$x^T Ax > 0.$$

Si dimostra che, se una matrice è simmetrica definita positiva, l'errore di roundoff nell'esecuzione dell'algoritmo di eliminazione di Gauss senza pivoting, in un sistema aritmetico floating-point a precisione finita, si mantiene limitato in maniera paragonabile a quanto avviene con l'applicazione del pivoting [9]. Non ci si sofferma però su tale argomento, in quanto alle matrici simmetriche definite positive è possibile applicare un algoritmo di fattorizzazione *ad hoc*, come spiegato nel prossimo paragrafo.

Si è visto, nei paragrafi precedenti, che una matrice  $A$  simmetrica, in opportune ipotesi, ha una fattorizzazione  $LDL^T$ . Si vuole ora vedere se è possibile ottenere una fattorizzazione del tipo  $A = LL^T$ , dove  $L$  è una matrice triangolare inferiore (con elementi diagonali in generale diversi da 1).<sup>24</sup>

♣ **Esempio 2.54.** Si provi a fattorizzare la matrice simmetrica

$$A = \begin{pmatrix} 16 & 4 & 20 \\ 4 & 10 & 2 \\ 20 & 2 & 30 \end{pmatrix},$$

nel prodotto  $LL^T$ , dove  $L$  è una matrice reale triangolare inferiore:

$$L = \begin{pmatrix} l_{1,1} & & \\ l_{2,1} & l_{2,2} & \\ l_{3,1} & l_{3,2} & l_{3,3} \end{pmatrix}.$$

Moltiplicando  $L$  per la sua trasposta, si ha:

$$LL^T = \begin{pmatrix} l_{1,1}^2 & l_{1,1}l_{2,1} & l_{1,1}l_{3,1} \\ l_{1,1}l_{2,1} & l_{2,1}^2 + l_{2,2}^2 & l_{2,1}l_{3,1} + l_{2,2}l_{3,2} \\ l_{1,1}l_{3,1} & l_{3,1}l_{2,1} + l_{2,2}l_{3,2} & l_{3,1}^2 + l_{3,2}^2 + l_{3,3}^2 \end{pmatrix},$$

---

<sup>24</sup>Anche se la matrice  $L$  considerata in  $A = LL^T$  differisce dalla matrice  $L$  in  $LDL^T$ , si continuerà ad indicare entrambe con la lettera  $L$ , a meno che ciò non generi confusione, in quanto questa è la notazione solitamente utilizzata.

che è una matrice simmetrica. Uguagliando gli elementi del triangolo inferiore (superiore) di  $A$  con i corrispondenti elementi di  $L$ , si ricava

$$\begin{aligned} l_{1,1} &= \sqrt{a_{1,1}} = \sqrt{16} = 4; \\ l_{2,1} &= a_{2,1}/l_{1,1} = 4/4 = 1; \\ l_{3,1} &= a_{3,1}/l_{1,1} = 20/4 = 5; \\ l_{2,2} &= \sqrt{a_{2,2} - l_{2,1}^2} = \sqrt{10 - 1} = 3; \\ l_{3,2} &= (a_{3,2} - l_{3,1}l_{2,1})/l_{2,2} = (2 - 5)/3 = -1; \\ l_{3,3} &= \sqrt{a_{3,3} - l_{3,1}^2 - l_{3,2}^2} = \sqrt{30 - 25 - 1} = 2; \end{aligned}$$

ovvero

$$L = \begin{pmatrix} 4 & & \\ 1 & 3 & \\ 5 & -1 & 2 \end{pmatrix}.$$

L'esistenza di  $L$  dipende dal fatto che gli argomenti delle radici quadrate sono sempre non negativi; ciò garantisce non solo che gli elementi diagonali  $l_{i,i}$  esistano in  $\mathbb{R}$ , ma anche che non vengano effettuate divisioni per 0 nel calcolo degli altri elementi di  $L$ .

La matrice  $A$  è definita positiva. Ciò si può verificare applicando il criterio di Sylvester (Teorema B.22, Appendice B.9), ovvero controllando che il determinante di ciascuna sottomatrice principale di  $A$  sia maggiore di 0. Si ha infatti:

$$\begin{aligned} \det(A_1) &= \det(16) = 16, \\ \det(A_2) &= \det \begin{pmatrix} 16 & 4 \\ 4 & 10 \end{pmatrix} = 144, \\ \det(A_3) &= \det \begin{pmatrix} 16 & 4 & 20 \\ 4 & 10 & 2 \\ 20 & 2 & 30 \end{pmatrix} = 576 \end{aligned}$$

e quindi la matrice è definita positiva. ♣

### Teorema 2.17. [Fattorizzazione di Cholesky]

*Sia  $A \in \mathbb{R}^{n \times n}$  simmetrica definita positiva. Allora esiste una ed una sola matrice triangolare inferiore  $L \in \mathbb{R}^{n \times n}$ , con  $l_{k,k} > 0$  per  $k = 1, \dots, n$ , tale che*

$$A = LL^T.$$

**Dimostrazione** Se  $A \in \mathbb{R}^{n \times n}$  è definita positiva, per il criterio di Sylvester tutte le sottomatrici principali di  $A$  sono non singolari e quindi  $A$  ammette una ed una sola fattorizzazione del tipo

$$A = GDG^T,$$

con  $G \in \mathbb{R}^{n \times n}$  triangolare unitaria inferiore e  $D = \text{diag}(d_1, \dots, d_n) \in \mathbb{R}^{n \times n}$  non singolare. Dato che  $\det(A_k) = d_1 d_2 \dots d_k$ , per il criterio di Sylvester risulta anche

$$d_k > 0, \quad k = 1, 2, \dots, n.$$

La matrice

$$D^{\frac{1}{2}} = \text{diag} \left( \sqrt{d_1}, \dots, \sqrt{d_n} \right)$$

è quindi reale e non singolare e, posto

$$L = GD^{\frac{1}{2}},$$

risulta

$$A = LL^T,$$

con  $L$  triangolare inferiore tale che  $l_{k,k} > 0$  per  $k = 1, \dots, n$ . L'unicità della fattorizzazione  $LL^T$  è una conseguenza dell'unicità della fattorizzazione  $GDG^T$ . Si supponga, infatti, che esistano  $L, \bar{L} \in \Re^{n \times n}$ , tali che

$$l_{k,k}, \bar{l}_{k,k} > 0, \quad k = 1, \dots, n,$$

e

$$A = LL^T = \bar{L}\bar{L}^T.$$

Posto

$$\begin{aligned} B &= \text{diag}(l_{1,1}, \dots, l_{n,n}), & G &= LB^{-1}, & D &= (B^2)^{-1}, \\ \bar{B} &= \text{diag}(\bar{l}_{1,1}, \dots, \bar{l}_{n,n}), & \bar{G} &= \bar{L}\bar{B}^{-1}, & \bar{D} &= (\bar{B}^2)^{-1}, \end{aligned}$$

si ha:

$$A = GDG^T = \bar{G}\bar{D}\bar{G}^T,$$

da cui discende

$$D = (B^2)^{-1} = \bar{D} = (\bar{B}^2)^{-1}, \quad G = LB^{-1} = \bar{G} = \bar{L}\bar{B}^{-1},$$

e quindi

$$L = \bar{L}.$$

■

La dimostrazione precedente è basata sul Teorema 2.15, riguardante l'esistenza e l'unicità della fattorizzazione  $LDL^T$  di una matrice. Una dimostrazione indipendente da tale teorema è riportata in nota<sup>25</sup>.

La fattorizzazione  $LL^T$  di una matrice  $A \in \Re^{n \times n}$  simmetrica definita positiva è detta *fattorizzazione di Cholesky* di  $A$  e la matrice  $L$  è detta *fattore di Cholesky* di  $A$ . La dimostrazione del Teorema 2.17 fornisce un procedimento per il calcolo della matrice  $L$

<sup>25</sup> **Dimostrazione 2** Si procede per induzione. Per  $n = 1$  basta porre  $l_{1,1} = \sqrt{a_{1,1}}$  e la tesi è vera. Si supponga ora che la tesi sia vera per  $n = k - 1 > 1$ , cioè tale che, data  $A_{k-1} \in \Re^{(k-1) \times (k-1)}$  simmetrica definita positiva, esista  $L_{k-1} \in \Re^{(k-1) \times (k-1)}$  triangolare inferiore, con tutti gli elementi diagonali positivi, tale che

$$A_{k-1} = L_{k-1}L_{k-1}^T.$$

Si considerino le seguenti matrici  $A_k, L_k \in \Re^{k \times k}$ , la prima simmetrica definita positiva e la seconda triangolare inferiore:

$$A_k = \begin{pmatrix} A_{k-1} & y \\ y^T & a_{k,k} \end{pmatrix}, \quad L_k = \begin{pmatrix} L_{k-1} & 0 \\ w^T & l_{k,k} \end{pmatrix},$$

con  $y, w \in \Re^{(k-1)}$  vettori colonna e  $l_{k,k} > 0$ . Imponendo

$$A_k = L_k L_k^T,$$

a partire dalla fattorizzazione  $GDG^T$ . Un algoritmo differente per il calcolo di  $L$  si ottiene ragionando come nell'esempio 2.54: si considera una matrice  $L = (l_{i,j})$  triangolare inferiore, si calcola il prodotto  $LL^T$  e si impone l'uguaglianza tra gli elementi di  $A$  ed i corrispondenti elementi di  $LL^T$ , ovvero

$$\begin{aligned} a_{j,j} &= \sum_{k=1}^j l_{j,k} l_{k,j}^T = \sum_{k=1}^j l_{j,k}^2, \quad j = 1, \dots, n, \\ a_{i,j} &= \sum_{k=1}^j l_{i,k} l_{k,j}^T = \sum_{k=1}^j l_{i,k} l_{j,k}, \quad j = 1, \dots, n-1, \quad i = j+1, \dots, n. \end{aligned}$$

Da ciò si ricava:

$$\begin{aligned} l_{j,j} &= \sqrt{a_{j,j} - \sum_{k=1}^{j-1} l_{j,k}^2}, \quad j = 1, \dots, n, \\ l_{i,j} &= \frac{a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} l_{j,k}}{l_{j,j}}, \quad j = 1, \dots, n-1, \quad i = j+1, \dots, n. \end{aligned} \tag{2.64}$$

Gli elementi di  $L$  si possono quindi determinare in  $n$  passi, calcolando, al passo  $j$ , prima l'elemento diagonale della colonna  $j$ -ma e poi gli altri elementi di tale colonna che si trovano al di sotto di quello diagonale. Si noti che i calcoli possono essere eseguiti in

si ha:

$$\begin{aligned} A_{k-1} &= L_{k-1} L_{k-1}^T, \\ y &= L_{k-1} w, \\ y^T &= w^T L_{k-1}^T, \\ a_{k,k} &= w^T w + l_{kk}^2. \end{aligned} \tag{2.62}$$

La prima relazione in (2.62) è vera per ipotesi di induzione; dalla seconda (ovvero dalla terza) si ricava univocamente  $w$ , in quanto  $L_{k-1}$ , avendo tutti gli elementi diagonali positivi, è non singolare; dalla quarta relazione, infine, si ricava:

$$l_{k,k} = \sqrt{a_{k,k} - w^T w}. \tag{2.63}$$

Per concludere la dimostrazione dell'esistenza di  $L_k$  occorre quindi dimostrare che l'argomento della radice quadrata in (2.63) è positivo. Considerati i seguenti vettori:

$$z = A_{k-1}^{-1} y, \quad x = \begin{pmatrix} z \\ 1 \end{pmatrix},$$

si ha:

$$\begin{aligned} x^T A_k x &= z^T A_{k-1} z - 2z^T y + a_{k,k} = z^T y - 2z^T y + a_{k,k} = \\ &= -z^T y + a_{k,k} = a_{k,k} - y^T A_{k-1}^{-1} y = a_{k,k} - y^T (L_{k-1} L_{k-1}^T)^{-1} y = \\ &= a_{k,k} - (L_{k-1}^{-1} y)^T (L_{k-1}^{-1} y) = a_{k,k} - w^T w; \end{aligned}$$

dato che  $x^T A_k x > 0$  ( $A_k$  è definita positiva), risulta  $a_{k,k} - w^T w > 0$ , e quindi l'esistenza di  $L_k$  è dimostrata. L'unicità di  $L_k$  discende dall'unicità di  $L_{k-1}$ , di  $w$  e di  $l_{k,k}$ . Il teorema è così dimostrato.

place, come mostrato nella Proceura 2.23<sup>26</sup>. Per costruire  $l_{i,j}$  si utilizzano, oltre ad  $a_{i,j}$ , gli elementi della  $i$ -ma e della  $j$ -ma riga di  $L$  che appartengono alle prime  $j - 1$  colonne, come schematizzato in Fig. 2.25.

---

<sup>26</sup>Le istruzioni `for i = j + 1 to n` e `for k = 1 to j - 1` si possono scambiare tra loro, lasciando inalterato il valore finale di  $a_{i,j}$ . In questo caso l'algoritmo, anziché calcolare definitivamente, uno dopo l'altro, gli  $n - j$  elementi extradiagonali della colonna  $j$ -ma, esegue  $j - 1$  aggiornamenti parziali di tali elementi, ottenendo la colonna  $j$ -ma di  $L$  al termine del  $(j - 1)$ -mo aggiornamento. Le due versioni dell'algoritmo differiscono per il modo in cui si accede agli elementi della parte di  $L$  già calcolata, per ottenere una nuova colonna.

```

procedure cholesky (in:  $n, a$ ; out:  $a$ )
  /# SCOPO: effettua la fattorizzazione
  di Cholesky di una matrice  $A$ .

  /# SPECIFICHE DEI PARAMETRI:
  /# PARAMETRI DI INPUT:
  var:  $n$  : intero { dimensione della matrice  $A$  }
  var:  $a(n, n)$  : reale { matrice da fattorizzare }

  /# PARAMETRI DI OUTPUT:
  var:  $a(n, n)$  : reale { matrice fattorizzata }
  /# VARIABILI LOCALI:
  var:  $i, j, k$  : interi

  /# INIZIO ISTRUZIONI:
  for  $j = 1$  to  $n$  { Calcolo degli elementi diagonali}
    for  $k = 1$  to  $j - 1$ 
       $a_{j,j} := a_{j,j} - a_{j,k}^2$ 
    endfor
     $a_{j,j} := \sqrt{a_{j,j}}$ 
    for  $i = j + 1$  to  $n$  { Calcolo degli elementi del triangolo}
      { inferiore non diagonali}
      for  $k = 1$  to  $j - 1$ 
         $a_{i,j} := a_{i,j} - a_{i,k}a_{j,k}$ 
      endfor
       $a_{i,j} := a_{i,j}/a_{j,j}$ 
    endfor
  endfor
end cholesky

```

Procedura 2.23: Algoritmo di Cholesky

Calcoliamo la complessità di tempo della Procedura 2.23, al generico passo  $j$ ,

- il calcolo dell'elemento diagonale  $a_{j,j}$  richiede  $j - 1$  moltiplicazioni, altrettante addizioni ed una radice quadrata, ovvero

$$2(j - 1) \text{ flop} + 1 \text{ radice quadrata};$$

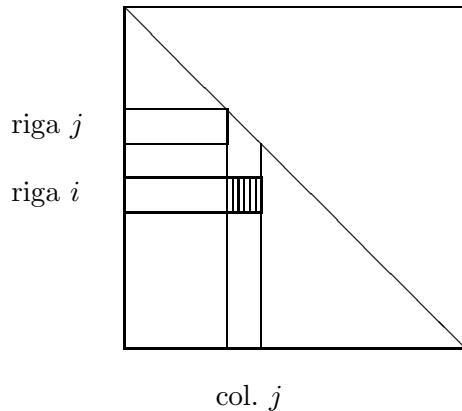


Figura 2.25: Rappresentazione grafica dell'algoritmo di Cholesky.

- il calcolo di un elemento extradiagonale  $l_{i,j}$ ,  $i > j$ , richiede  $j - 1$  moltiplicazioni, altrettante addizioni ed una divisione, e quindi il calcolo degli  $n - j$  elementi extradiagonali della colonna  $j$ -ma richiede

$$(n - j)(2j - 1) \text{ flop}.$$

In totale, l'algoritmo richiede

$$\begin{aligned} & \sum_{j=1}^n (2(j-1) + (n-j)(2j-1)) = \\ &= 2 \sum_{j=1}^{n-1} j + 2n \sum_{j=1}^{n-1} j - 2 \sum_{j=1}^{n-1} j^2 - n^2 + \sum_{j=1}^{n-1} j + n \\ &= (2n+3) \frac{n(n-1)}{2} - 2 \frac{n(n-1)(2n-1)}{6} - n^2 - n(n-1) \\ &= \frac{2n^3 + 3n^2 - 5n}{6} \\ & e \\ & n \text{ radici quadrate}. \end{aligned}$$

La complessità di tempo dell'algoritmo è dunque

$$T_{Chol}(n) = \mathcal{O}\left(\frac{n^3}{6}\right) \text{ flop}.$$

Si noti che, come era prevedibile, il numero di operazioni aritmetiche floating-point dell'algoritmo di Cholesky è circa la metà di quello dell'algoritmo di eliminazione di Gauss.

Se per memorizzare la matrice  $A$ , e quindi la matrice  $L$ , si utilizza un array bidimensionale di dimensione  $n \times n$ , la complessità di spazio dell'algoritmo di Cholesky è:

$$S_{Chol}(n) = n^2;$$

se invece si utilizza uno schema di memorizzazione packed, la complessità di spazio diventa

$$S_{Chol}^{packed}(n) = \frac{n(n+1)}{2} = \mathcal{O}\left(\frac{n^2}{2}\right)$$

e quindi risulta dimezzata. L'algoritmo relativo a quest'ultimo si può ottenere dalla Procedura 2.23, sostituendo ad  $a_{i,j}$  l'elemento corrispondente dell'array  $LP$  utilizzato per la memorizzazione packed.

Analogamente all'algoritmo di eliminazione di Gauss, l'algoritmo di Cholesky, richiede l'esecuzione di istruzioni del tipo<sup>27</sup>

$$\begin{aligned} a_{i,j} &:= a_{i,j} - a_{i,k}a_{j,k}, \\ a_{i,j} &:= a_{i,j}/a_{j,j}; \end{aligned} \quad (2.65)$$

è quindi naturale chiedersi se sia necessaria qualche strategia di pivoting per evitare che, al generico passo  $j$ , l'elemento  $a_{j,j}$  sia “tropppo grande” e quindi gli  $a_{i,j}$ , che risultano essere gli  $a_{i,k}$  dei passi successivi, crescano in modo da amplificare significativamente l'errore.

#### ♣ Esempio 2.55.

Applichiamo l'algoritmo di Cholesky alla matrice simmetrica definita positiva

$$A = \begin{pmatrix} .0001 & .01 \\ .01 & 100 \end{pmatrix},$$

utilizzando un sistema aritmetico floating-point con base  $\beta = 10$ , precisione  $t = 2$  e massima accuratezza dinamica. In tale sistema,  $A$  ha la seguente rappresentazione:

$$\tilde{A} = A = \begin{pmatrix} .10 \times 10^{-3} & .10 \times 10^{-1} \\ .10 \times 10^{-1} & .10 \times 10^3 \end{pmatrix}$$

si ha inoltre:

$$\begin{aligned} \tilde{l}_{1,1} &= \sqrt{a_{1,1}} = .10 \times 10^{-1}, \\ \tilde{l}_{2,1} &= a_{2,1}/\tilde{l}_{1,1} = (.10 \times 10^{-1})/(.10 \times 10^{-1}) = .10 \times 10^1, \\ \tilde{l}_{2,2} &= \sqrt{a_{2,2} - \tilde{l}_{2,1}^2} = \sqrt{.10 \times 10^3 - (.10 \times 10)^2} = \\ &= \sqrt{.10 \times 10^3 - .1 \times 10} = \sqrt{.10 \times 10^3} = .10 \times 10^2, \end{aligned}$$

---

<sup>27</sup>Si noti che, per  $i = j$ , la prima delle (2.65) corrisponde alla seguente istruzione per il calcolo dell'elemento diagonale  $j$ -mo:

$$a_{j,j} := a_{j,j} - a_{j,k}^2.$$

e quindi

$$\tilde{L} = \begin{pmatrix} .10 \times 10^{-1} & \\ & .10 \times 10^1 \end{pmatrix}.$$

Eseguendo il prodotto  $\tilde{L}\tilde{L}^T$  nel sistema aritmetico considerato, si ha:

$$\tilde{L}\tilde{L}^T = \begin{pmatrix} .10 \times 10^{-3} & .1 \times 10^{-1} \\ .1 \times 10^{-1} & .1 \times 10^3 \end{pmatrix} = A.$$

Anche se  $a_{1,1}$  è due ordini di grandezza più piccolo di  $a_{2,1}$ , non si ha una crescita “eccessiva” degli elementi di  $\tilde{L}$ , come invece ci si potrebbe aspettare, e quindi una crescita dell’errore di roundoff tale da invalidare il calcolo. ♣

Dalla prima relazione in (2.64) si ricava che

$$l_{j,1}^2 + l_{j,2}^2 + \dots + l_{j,j}^2 = a_{j,j}, \quad j = 1, \dots, n,$$

dove tutti i termini a primo membro sono non negativi, da cui si ottiene la limitazione a priori per gli elementi di  $\tilde{L}$ :

$$|\tilde{l}_{j,k}| \leq \sqrt{a_{j,j}}, \quad k = 1, \dots, j, j = 1, \dots, n.$$

Ogni elemento di  $\tilde{L}$  è dunque limitato dalla radice quadrata dell’elemento diagonale di  $A$  che si trova sulla medesima riga, indipendentemente dalla grandezza degli elementi “pivot” che compaiono al denominatore nei calcoli e dei conseguenti “moltiplicatori”. In altri termini, gli elementi della matrice  $L$  non possono crescere in maniera tale da invalidare il calcolo. L’algoritmo di Cholesky risulta quindi stabile.

### Teorema 2.18.

*Dato un sistema di equazioni  $Ax = b$  con  $A \in R^{n \times n}$  simmetrica definita positiva. Se  $\hat{L}$  è la matrice triangolare inferiore calcolata dall’algoritmo di Cholesky e  $\hat{x}$  la soluzione calcolata del sistema, allora si ha:*

$$(A + \Delta A)\hat{x} = b \quad \text{con} \quad \|\Delta A\|_2 \leq c_n u \|A\|_2 + o(u^2)$$

con  $c_n = o(n^2)$  costante dipendente solo da  $n$ .

**Dimostrazione** Si cominci col dimostrare che, detta  $u$  la massima accuratezza relativa e posto  $\gamma_i = iu/(1 - iu)$ , per ogni fissato  $j$  si ha:

$$|a_{i,j} - \sum_{k=1}^i \hat{l}_{i,k} \hat{l}_{j,k}| \leq \gamma_i \sum_{k=1}^i |\hat{l}_{i,k}| |\hat{l}_{j,k}| \quad i = j+1, \dots, n \quad (2.66)$$

Ciò può essere facilmente provato che detto:

$$\hat{s} = fl \left( a_{i,j} - \sum_{k=1}^i \hat{l}_{i,k} \hat{l}_{j,k} \right)$$

e ricordando che  $fl(a \text{ op } b) = (a \text{ op } b)(1 + \delta)$  con  $|\delta| \leq u$ , si ha:

$$\hat{s} = a_{i,j}(1 + \delta_1) \cdots (1 + \delta_i) - \sum_{k=1}^i \hat{l}_{i,k} \hat{l}_{j,k} (1 + \epsilon_k)(1 + \delta_k) \cdots (1 + \delta_i)$$

con  $|\epsilon_i| |\delta_i| \leq u$ . Dividendo per  $(1 + \delta_1) \cdots (1 + \delta_i)$  si ottiene:

$$\frac{\hat{s}}{(1 + \delta_1) \cdots (1 + \delta_i)} = a_{i,j} - \sum_{k=1}^i \hat{l}_{i,k} \hat{l}_{j,k} \frac{(1 + \epsilon_k)}{(1 + \delta_1) \cdots (1 + \delta_{k-1})}$$

Ricordando infine che  $\prod_{k=1}^i (1 + \delta_k)^{\rho_k} = 1 + \theta_i$  con  $\rho_k = \pm 1$  e  $|\theta_i| \leq \gamma_i$  si ha  $\hat{s}(1 + \theta_i) = a_{i,j} - \sum_{k=1}^i \hat{l}_{i,k} \hat{l}_{j,k} (1 + \theta_k)$ , da cui la (2.66). Analogamente si può provare che

$$|a_{j,j} - \sum_{k=1}^j \hat{l}_{j,k}^2| \leq \gamma_j \sum_{k=1}^j |\hat{l}_{i,k}^2| \quad (2.67)$$

Dalle (2.66) e (2.67) si ha quindi che, se si suppone  $nu < 1$ , si ottiene:

$$\hat{L}\hat{L}^T = A + \Delta A \quad \text{con} \quad \gamma_{n+1}|\hat{L}| |\hat{L}^T| \leq nu|\hat{L}| |\hat{L}^T| + o(u^2) \quad (2.68)$$

Poiché per la risoluzione di sistemi triangolari valgono<sup>28</sup>:

$$\begin{aligned} (\hat{L} + \Delta L)\hat{y} &= b & |\Delta L| &\leq nu|\hat{L}| + o(u^2) \\ (\hat{U} + \Delta U)\hat{x} &= \hat{y} & |\Delta U| &\leq nu|\hat{U}| + o(u^2/2) \end{aligned}$$

allora si ha:

$$b = (\hat{L} + \Delta L)(\hat{L}^T + \Delta L^T)\hat{x} = (A + \Delta A + \hat{L}\Delta L^T + \Delta L\hat{L}^T + \Delta L\Delta L^T)\hat{x} = (A + \Delta A') \quad (2.69)$$

dove  $|\Delta A'| \leq c_n nu|\hat{L}| |\hat{L}^T| + o(u^2)$ .

Si osservi che

$$\| |L| |L^T| \|_2 = \| |L| \|_2^2 \leq nn \| |L| \|_2^2 = n \| A \|$$

e, analogamente, dalla (2.68) si ha:

$$\| |\hat{L}| |\hat{L}^T| \|_2 \leq n(1 - n\gamma_n)^{-1} \| A \|_2$$

da cui, utilizzando la (2.69) si ottiene

$$\| \Delta A' \|_2 \leq \| \Delta A \|_2 \leq c_n u \| A \|_2 + o(u^2)$$

cioè la tesi. ■

Il teorema precedente asserisce quindi che risolvere un sistema di equazioni lineari con matrice simmetrica definita positiva, equivale alla risoluzione di un sistema perturbato, dove le perturbazioni sui coefficienti rimangono limitate.

Si osservi, però, che alcune “difficoltà numeriche” si possono comunque presentare.

---

<sup>28</sup>E ricordando che  $L^T$  è triangolare superiore

♦ **Esempio 2.56.**

Si consideri la matrice simmetrica definita positiva:

$$A = \begin{pmatrix} 100 & 9.71 \times 10^{-2} \\ 9.71 \times 10^{-2} & 9.43 \times 10^{-5} \end{pmatrix},$$

il cui fattore di Cholesky, arrotondato a quattro cifre decimali significative, è

$$L = \begin{pmatrix} 10 & \\ 9.710 \times 10^{-3} & 9.662 \times 10^{-3} \end{pmatrix}$$

Se si calcola  $L$ , mediante l'algoritmo di Cholesky, in un sistema aritmetico floating-point con base  $\beta = 10$  e precisione  $t = 3$ , dotato di massima accuratezza dinamica, si ha:

$$\begin{aligned} \tilde{l}_{1,1} &= \sqrt{a_{1,1}} = .100 \times 10^2, \\ \tilde{l}_{2,1} &= a_{2,1}/\tilde{l}_{1,1} = (.971 \times 10^{-1})/(.100 \times 10^2) = .971 \times 10^{-2}, \\ \tilde{l}_{2,2} &= \sqrt{a_{2,2} - \tilde{l}_{2,1}^2} = \sqrt{.943 \times 10^{-3} - .943 \times 10^{-4}} = 0, \end{aligned}$$

ovvero:

$$\tilde{L} = \begin{pmatrix} 10 & \\ 9.71 \times 10^{-4} & 0 \end{pmatrix},$$

che ha il secondo elemento diagonale uguale a 0. Eseguendo il prodotto  $\tilde{L}\tilde{L}^T$ , nel sistema aritmetico floating-point considerato, si ha:

$$\tilde{L}\tilde{L}^T = \begin{pmatrix} 100 & 9.71 \times 10^{-2} \\ 9.71 \times 10^{-2} & 0 \end{pmatrix},$$

che non è una matrice definita positiva. Ciò dipende dal fatto che la matrice  $A$  è “lontanamente” definita positiva; i suoi autovalori, arrotondati a tre cifre significative, sono infatti  $\lambda_1 = 100$  e  $\lambda_2 = 9.34 \times 10^{-5}$ , e quindi uno di essi è prossimo a zero. ♣

In generale, nell'esecuzione dell'algoritmo di Cholesky si posso avere problemi di instabilità numerica nel caso in cui la matrice  $A$  non è “sufficientemente” definita positiva, ovvero se uno o più autovalori sono prossimi a zero.

## 2.13 Software matematico per la risoluzione di sistemi lineari

Come già osservato nei paragrafi precedenti, uno dei package piú utilizzati per la risoluzione di problemi di algebra lineare con matrice piena o a banda è *LAPACK* [1]. Tale package è il risultato di una intensa attività rivolta allo sviluppo di software matematico

efficiente, accurato e portabile per la risoluzione dei problemi suddetti<sup>29</sup> ed è disponibile, come software di dominio pubblico, al sito web <http://www.netlib.org/lapack>.

LAPACK è scritto in Fortran 77 e contiene routine per la risoluzione di sistemi di equazioni lineari e problemi di autovalori e valori singolari, basate essenzialmente su metodi di fattorizzazione della matrice dei coefficienti del problema. Le routine sono disponibili in singola ed in doppia precisione, per problemi reali e complessi. Esistono inoltre interfacce a LAPACK o conversioni/traduzioni di esso in Fortran 95, C, C++ e Java.

LAPACK contiene, tra l'altro, routine per la risoluzione di sistemi lineari con matrice speciale (tridiagonale, a banda, simmetrica definita positiva, simmetrica definita positiva a banda, simmetrica definita positiva tridiagonale, simmetrica non definita, triangolare a banda). Il package include inoltre routine che eseguono operazioni connesse con la risoluzione di un sistema, quali, ad esempio, la stima dell'indice di condizionamento della matrice dei coefficienti o dell'errore nella soluzione calcolata.

Per le matrici simmetriche, hermitiane o triangolari è usata sia la memorizzazione convenzionale, nel triangolo superiore o inferiore di un array bidimensionale, sia la memorizzazione di tipo *packed*. Per matrici a banda è utilizzato lo schema di memorizzazione a banda, mentre per quelle tridiagonali è utilizzata la memorizzazione mediante tre vettori.

Le routine di LAPACK sono suddivise in tre classi:

- routine *driver*, ciascuna delle quali risolve un problema completo, quale, ad esempio, il calcolo della soluzione di un sistema lineare;
- routine *computazionali*, ciascuna delle quali risolve un singolo problema computazionale, quale, ad esempio, il calcolo della fattorizzazione *LU* di una matrice o la stima dell'indice di condizionamento;
- routine *ausiliarie*, ciascuna delle quali risolve un singolo sottoproblema che si presenta nei problemi computazionali suddetti.

Le routine driver chiamano, in generale, routines computazionali, le quali chiamano a loro volta routine ausiliarie.

Le routine driver e computazionali hanno un nome a cinque o a sei caratteri<sup>30</sup>, rispettivamente del tipo XYZZZ o XYZZZZ, in cui

---

<sup>29</sup>LAPACK costituisce l'evoluzione e l'estensione, per i calcolatori moderni a memoria gerarchica, dei package *LINPACK* [3] ed *EISPACK* [13], sviluppati negli anni '70 per la risoluzione di sistemi lineari e di problemi lineari di minimi quadrati e per il calcolo di autovalori ed autovettori, utilizzando essenzialmente metodi di fattorizzazione. Gli algoritmi implementati sono stati riorganizzati in termini di operazioni tra matrici, quali, ad esempio, prodotti matrice-matrice, eseguite mediante *Basic Linear Algebra Subprograms (BLAS)* [12, 5, 4], per tener conto della gerarchia di memoria della macchina in uso e quindi minimizzare il traffico di dati da e verso tale memoria, con conseguente aumento dell'efficienza. Si noti che il raggiungimento di elevate prestazioni richiede una implementazione ottimizzata di BLAS, fornita da numerose case costruttrici di computer o generabile in maniera automatica utilizzando il software ATLAS

<sup>30</sup>In LAPACK sono presenti due tipi di routine driver: i *driver semplici*, che calcolano la soluzione di un problema completo, ed i *driver esperti* che, oltre al calcolo della soluzione, sono in grado di

- X indica il tipo di dato su cui la routine agisce ( S = REAL, D = DOUBLE PRECISION, C = COMPLEX, Z = COMPLEX\*16 o DOUBLE COMPLEX );
- YY indica il tipo di matrice (GB = generale a banda, GT = generale tridiagonale, PO = simmetrica o Hermitiana definita positiva - memorizzazione convenzionale, PP = simmetrica o Hermitiana definita positiva - memorizzazione packed, SB = simmetrica a banda, etc.);
- ZZ o ZZZ indica il tipo di operazione effettuata (SV = risoluzione di un sistema - driver semplice, TRF = fattorizzazione, TRS = risoluzione del sistema fattorizzato, CON = stima dell'indice di condizionamento, etc.).

Ad esempio, le routine SPOSV e SPPSV risolvono entrambe un sistema lineare reale con matrice dei coefficienti simmetrica definita positiva, operando su dati in singola precisione; SPOSV richiede che la matrice sia memorizzata in un array bidimensionale, mentre SPPSV assume una memorizzazione packed. Analogamente, DGBTRF esegue la fattorizzazione *LU* di una matrice reale a banda e DGBTRS risolve i sistemi lineari triangolari a banda associati a tale fattorizzazione; entrambe le routine eseguono le operazioni aritmetiche in doppia precisione. Maggiori dettagli sono forniti in [1].

Numerose routine per la risoluzione di sistemi lineari con matrice speciale sono presenti nella Libreria Fortran 77 del NAg versione *Mark 19*. In particolare, per la risoluzione di sistemi lineari con matrice generale o speciale, oltre alle routine dei capitoli F01 e F04, la libreria contiene, nel capitolo F07, le routine di LAPACK<sup>31</sup> (si veda, a tal proposito, la pagina web <http://www.nag.co.uk/numeric/f1/manual/html/FLlibrarymanual.asp>).

Routine di LAPACK, infine, sono utilizzate da funzioni di MATLAB per risolvere problemi del tipo suddetto. Ad esempio, la funzione *chol*, di cui si parla nel prossimo paragrafo, è basata sulle routine DPOTRF e ZPOTRF, che eseguono la fattorizzazione di Cholesky rispettivamente di una matrice reale in doppia precisione e di una complessa in doppia precisione.

## 2.14 Risoluzione di sistemi lineari in ambiente MATLAB

### 2.14.1 Elementi di base di MATLAB

In ambiente MATLAB la memorizzazione di un vettore viene effettuata racchiudendo tra parentesi quadre una sequenza di numeri, separati da uno spazio bianco o da una virgola; ad esempio

---

eseguire altre operazioni, quali, ad esempio, la stima dell'indice di condizionamento o il raffinamento (miglioramento dell'accuratezza) della soluzione calcolata e la stima dell'errore corrispondente. Il nome di un driver semplice ha cinque caratteri; quello di un driver esperto ne ha sei ed è ottenuto aggiungendo una X alla fine del nome dell'analogico driver semplice.

<sup>31</sup>La libreria Fortran 77 del NAg include anche le routine di LAPACK per i problemi di minimi quadrati e di autovalori, nel capitolo F08.

```
>> x=[1 2 3]
x =
1     2     3
```

oppure

```
>> x=[1, 2, 3];
```

Il punto e virgola alla fine dell'istruzione consente di memorizzare, ma non visualizzare sullo schermo, il risultato di un'operazione.

Analogamente, per definire una matrice si separano gli elementi delle righe con spazi o virgole, mentre le diverse righe con punti e virgola (oppure andando a capo ad ogni nuova riga).

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

```
A =
1     2     3
4     5     6
7     8     9
```

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

L'operatore di trasposizione in MATLAB è il carattere *apice*', quindi sono analoghe le istruzioni

```
>> y=[1; 4; 7]
```

```
y =
```

```
1
4
7
```

```
>> y=[1 4 7]'
```

```
y =
```

```
1
4
7
```

L'accesso agli elementi di una matrice si ottiene tramite indici; l'elemento  $a_{mn}$  è indirizzato come  $A(m, n)$ ; ad esempio

```
>> A(2,3)
```

```
ans =
```

```
6
```

cioè il comando  $A(2, 3)$  fa riferimento all'elemento di posto (2,3) della matrice  $A$ . Più in generale, per prelevare l'elemento sulla riga  $i$  e sulla colonna  $j$  della matrice  $A$  e memorizzarlo nella variabile  $a$ , si utilizza l'istruzione

```
>> a=A(i,j)
```

La riga  $m$ -esima è indirizzata come  $A(m,:)$ , dove tutte le colonne sono indicate con due punti:

```
>> A(2,:)
```

```
ans =
```

```
4      5      6
```

mentre la colonna  $n$ -esima è indirizzata come  $A(:,n)$ , dove tutte le righe sono indicate con due punti:

```
>> A(:,3)
```

```
ans =
```

```
3
6
9
```

## 2.14.2 Operazioni matriciali

Le operazioni algebriche tra matrici si eseguono con gli usuali operatori  $+, -, *, /$ ; in particolare, date due matrici  $A$  e  $B$ , di dimensioni opportune, si possono definire le seguenti operazioni:

```
>> S=A+B; % somma di due matrici
>> P=A*B; % prodotto righe per colonne
>> At=A'; % trasposta di una matrice
```

Altre funzioni operanti su matrici (e, quindi, su vettori riga o colonna) sono:

```
max, min,
sort,
sum, prod.
```

Esistono, poi, particolari operatori ( $.*$ ,  $. /$ ,  $.^$ ) che permettono di effettuare operazioni su matrici, elemento per elemento, senza ricorrere a cicli. Ad esempio, se A e B sono due matrici, per moltiplicarle elemento per elemento basta eseguire l'istruzione

```
>> C=A.*B;
```

Tra le funzioni che operano essenzialmente su matrici si ricordano:

- **det**, che restituisce il determinante di una matrice

```
>> det(A)
```

```
ans =
```

```
0
```

- **size**, che restituisce le dimensioni di una matrice

```
>> size(A)
```

```
ans =
```

```
3      3
```

- **rank**, che restituisce il rango di una matrice

```
>> rank(A)
```

```
ans =
```

```
2
```

Si riporta un elenco di alcune funzioni predefinite, per la creazione di matrici particolarmente usate:

---

<code>eye(n)</code>	: matrice identica di ordine $n$
<code>zeros(m,n)</code>	: matrice di 0, con $m$ righe e $n$ colonne
<code>ones(m,n)</code>	: matrice di 1, con $m$ righe e $n$ colonne
<code>rand(m,n)</code>	: matrice generata in maniera casuale ( <i>random</i> ), costituita da valori compresi tra 0 e 1
<code>diag(X)</code>	: se $X$ è un vettore con $n$ elementi, costruisce una matrice quadrata, diagonale, di dimensione $n \times n$ , con gli elementi di $X$ sulla diagonale. Se $X$ è una matrice quadrata di dimensione $n \times n$ , produce un vettore di $n$ elementi, uguali a quelli sulla diagonale di $X$ .

---

MATLAB presenta, inoltre, un `help` in linea, con informazioni sulla sintassi di tutte le funzioni disponibili; ad esempio, per avere informazioni relative all'utilizzo di una particolare funzione *fun*, è sufficiente digitare

```
>> help fun
```

Il solo comando

```
>> help
```

elenca tutte le categorie di funzioni disponibili (i **toolbox**), mentre, ad esempio,

```
>> help signal
```

fornisce informazioni su tutte le funzioni presenti nella categoria **signal**.

♣ **Esempio 2.57.** Calcolare il prodotto scalare

$$s = u * v^T$$

tra i vettori

$$u = (5, 3, -2, -4, -1) \quad v = (2, -1, 0, -7, 2)$$

Memorizzando i due vettori mediante i comandi

```
>> u=[5 3 -2 -4 -1]
```

```
u =
```

$$\begin{matrix} 5 & 3 & -2 & -4 & -1 \end{matrix}$$

---

```
>> v=[2 -1 0 -7 2]
v =
2     -1      0     -7      2
```

si determina il risultato, come output generato dal comando

```
>> s=u*v'
s =
33
```



♣ **Esempio 2.58.** Assegnata la matrice

$$\begin{pmatrix} 5 & 3 & -6 \\ 7 & 2 & 0 \\ -4 & 8 & 1 \end{pmatrix}$$

si vogliono calcolare

1. A1=A\*A;
2. A2=A'\*A;
3. A3=.\*A;
4. d1=diag(A);
5. d2=diag(A,1);
6. e=exp(A);
7. sq=sqrt(A);
8. el=exp(log(A+7));
9. m=max(A);
10. sn=sign(A);

A tal fine si memorizza, dapprima, la matrice A come

```
>> A=[5 3 -6; 7 2 0; -4 8 1]
A =
```

$$\begin{matrix} 5 & 3 & -6 \\ 7 & 2 & 0 \\ -4 & 8 & 1 \end{matrix}$$

Si ottengono, dunque, i risultati mediante le istruzioni

```
>> A1=A*A

A1 =
    70   -27   -36
    49    25   -42
    32    12    25

>> A2=A'*A

A2 =
    90    -3   -34
    -3    77   -10
   -34   -10    37

>> A3=A.*A

A3 =
    25     9    36
    49     4     0
    16    64     1

>> d1=diag(A)

d1 =
    5
    2
    1

>> d2=diag(A,1)

d2 =
    3
    0

>> e=exp(A)

e =
    1.0e+03 *
    0.1484    0.0201    0.0000
    1.0966    0.0074    0.0010
    0.0000    2.9810    0.0027

>> sq=sqrt(A)
```

```

sq =
2.2361      1.7321      0 + 2.4495i
2.6458      1.4142      0
0 + 2.0000i  2.8284      1.0000

>> el=exp(log(A+7))

el =
12.0000  10.0000  1.0000
14.0000  9.0000   7.0000
3.0000   15.0000  8.0000

>> m=max(A)

m =
7     8     1

>> sn=sign(A)

sn =
1     1    -1
1     1     0
-1    1     1

```

Con l'istruzione `save` è possibile salvare, in un file con estensione `.mat`, ad esempio `ristest.mat`, i risultati ottenuti:

```
>> save ristest A1 A2 A3 d1 d2 e sq el m sn
```

Con l'istruzione `clear`

```
>> clear
```

si può liberare la memoria dai risultati di tutte le operazioni eseguite; con il comando `load` è, poi, possibile caricare il file di risultati creato

```
>> load ristest
```

In tal modo, in memoria, saranno nuovamente presenti le variabili salvate nel file `ristest`.



### 2.14.3 Metodo di eliminazione di Gauss e fattorizzazione LU

Si supponga di voler risolvere in ambiente MATLAB un sistema lineare del tipo

$$Ax = B$$

ovvero di voler determinare il vettore  $x$  come

$$x = A^{-1}B.$$

MATLAB consente di risolvere il problema eseguendo le istruzioni

```
>> x=A\B;
```

o, analogamente,

```
>> x=inv(A)*B;
```

con

```
>> inv(A); % inversa della matrice A
```

Più in generale la divisione tra matrici in MATLAB può essere eseguita con due diversi simboli:

- $x = A \setminus B$  esegue  $x = A^{-1} \cdot B$ , risolvendo il sistema  $Ax = B$ ; in tal caso si parla anche di *divisione a sinistra*.
- $x = D/C$  esegue  $D \cdot C^{-1}$ , risolvendo  $x \cdot C = D$ ; in tal caso si parla anche di *divisione a destra*.

Nel secondo caso si possono utilizzare i due comandi equivalenti:

```
>> x=D/C;
```

oppure

```
>> x=D*inv(C);
```

Riassumendo

- **slash** / esegue la divisione con la matrice posta a destra;
- **backslash** \ esegue la divisione con la matrice posta a sinistra.

MATLAB determina la fattorizzazione LU di una matrice A mediante l'istruzione

```
>> [L,U]=lu(A);
```

che restituisce, in output, le matrici L, triangolare inferiore e U, triangolare superiore. In realtà, nel caso in cui il metodo di eliminazione di Gauss preveda l'applicazione della tecnica di pivoting e, dunque, uno scambio di righe, la matrice L, restituita in output, è la matrice risultante dal prodotto del fattore triangolare inferiore della decomposizione LU, con la matrice di permutazione,  $P$ , tale che

$$PA = LU;$$

l'espressione esplicita delle tre matrici, L,U e P, si può, invece, ottenere, mediante l'istruzione

```
>> [L,U,P]=lu(A);
```

♣ **Esempio 2.59.** Risolvere il seguente sistema lineare

$$\left\{ \begin{array}{l} 2x_1 - 4x_2 + 7x_3 + 4x_4 = 5 \\ 9x_1 + 3x_2 + 2x_3 - 7x_4 = -1 \\ 5x_1 + 2x_2 - 3x_3 + x_4 = -3 \\ 6x_1 - 5x_2 + 4x_3 - 3x_4 = 2 \end{array} \right.$$

e determinare i fattori della decomposizione LU della matrice dei coefficienti.

Memorizzando la matrice A mediante l'istruzione

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

A =

$$\begin{matrix} 2 & -4 & 7 & 4 \\ 9 & 3 & 2 & -7 \\ 5 & 2 & -3 & 1 \\ 6 & -5 & 4 & -3 \end{matrix}$$

ed il vettore dei termini noti

```
>> b=[5 -1 -3 2]'
```

b =

$$\begin{matrix} 5 \\ -1 \\ -3 \\ 2 \end{matrix}$$

la soluzione del sistema si ottiene come output generato dall'istruzione

```
>> x=A\b
```

x =

```
-0.1704
-0.1137
0.6614
0.0640
```

I fattori della decomposizione LU di A si deducono mediante l'istruzione

```
>> [L,U]=lu(A);
```

che genera il seguente output:

```
>> [L,U]=lu(A)
```

L =

```
0.2222 0.6667 1.0000 0
1.0000 0 0 0
0.5556 -0.0476 -0.8339 1.0000
0.6667 1.0000 0 0
```

U =

```
9.0000 3.0000 2.0000 -7.0000
0 -7.0000 2.6667 1.6667
0 0 4.7778 4.4444
0 0 0 8.6744
```

In particolare, l'istruzione

```
>> [L,U,P]=lu(A);
```

restituisce

L =

```
1.0000 0 0 0
0.6667 1.0000 0 0
0.2222 0.6667 1.0000 0
0.5556 -0.0476 -0.8339 1.0000
```

U =

```
9.0000 3.0000 2.0000 -7.0000
0 -7.0000 2.6667 1.6667
0 0 4.7778 4.4444
0 0 0 8.6744
```

P =

$$\begin{matrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{matrix}$$



## 2.14.4 MATLAB e le matrici speciali

In ambiente MATLAB non sono presenti funzioni predefinite specifiche per la risoluzione di sistemi lineari con matrice a banda o simmetrica non definita. Per i sistemi lineari con matrice simmetrica definita positiva è, invece, disponibile la funzione `chol`, che esegue la fattorizzazione di Cholesky di una matrice  $A$ ,

$$A = R^T R,$$

dove  $R$  è una matrice triangolare superiore, con elementi diagonali positivi, calcolando il fattore di Cholesky,  $R$ , mediante l'istruzione

```
>> R=chol(A)
```

### ♣ Esempio 2.60.

Si vuole risolvere il sistema lineare  $Ax = b$ , con

$$A = \begin{pmatrix} 22 & 2 & 12 & 15 \\ 2 & 47 & 18 & -10 \\ 12 & 18 & 25 & 21 \\ 15 & -10 & 21 & 47 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \\ 3 \\ -2 \end{pmatrix}.$$

La matrice  $A$  è simmetrica definita positiva. Per verificare che essa gode di quest'ultima proprietà si può applicare il criterio di Sylvester (Teorema B.22, Appendice B.9), utilizzando la funzione `det`.

Le istruzioni

```
>> A = [ 22    2    12    15
         2    47    18   -10
        12    18    25    21
        15   -10    21    47];
>> d2 = det(A(1:2,1:2))
>> d3 = det(A(1:3,1:3))
>> d4 = det(A)
```

generano il seguente output:

```
d2 =
          1030
d3 =
          12718
d4 =
          137641
```

da cui si ricava che la matrice è definita positiva (il determinante della sottomatrice principale di  $A$  di ordine 1 è, banalmente, 22). Poiché una matrice simmetrica è definita positiva se e solo se i suoi autovalori sono positivi (cfr. § B.9), la precedente verifica si può eseguire anche utilizzando la funzione `eig`, che calcola gli autovalori (ed anche gli autovettori) di una matrice quadrata. L'istruzione

```
>> lambda = eig(A)
```

fornisce, in output, un vettore, memorizzato nella variabile `lambda`, di componenti gli autovalori di  $A$ :

```
lambda =
14.1028
 2.5464
56.4276
67.9232
```

che sono, appunto, reali positivi. Utilizzando la routine `chol` si ottiene:

```
>> R=chol(A)
```

```
R =
 4.6904    0.4264    2.5584    3.1980
      0    6.8424    2.4712   -1.6608
      0        0    3.5139    4.8158
      0        0        0    3.2898
```

Naturalmente, eseguendo il prodotto

```
>> C = R'*R
```

si ottiene

```
C =
22.0000    2.0000   12.0000   15.0000
 2.0000   47.0000   18.0000  -10.0000
12.0000   18.0000   25.0000   21.0000
15.0000  -10.0000   21.0000   47.0000
```

che coincide con la matrice  $A$ .

La soluzione del sistema  $Ax = b$  si può, dunque, eseguire con le istruzioni

```
>> w = R'\b
>> x = R \w
```

che corrispondono ad una forward e ad una back substitution e producono i seguenti risultati:

```
>> w = R'\b
```

```
w =
```

```
0.2132
0.1329
0.6051
-1.6339
```

```
>> x = R \w
```

```
x =
```

```
-0.0439
-0.4092
0.8529
-0.4967
```

Si osservi che il calcolo della soluzione del sistema  $Ax = b$  si può eseguire con l'istruzione

```
>> x = A \ b
```

che fornisce, in output, direttamente il vettore  $x$ :

```
x =
```

```
-0.0439
-0.4092
0.8529
-0.4967
```

In questo caso, prima di procedere alla risoluzione del sistema, MATLAB controlla se la matrice è simmetrica e se ha tutti gli elementi diagonali positivi; in tal caso prova ad eseguire la fattorizzazione di Cholesky. Se la matrice è definita positiva, come in questo caso, la fattorizzazione di Cholesky termina senza errore ed i sistemi triangolari ad essa associati sono risolti con i relativi algoritmi di sostituzione.<sup>32</sup>




---

<sup>32</sup>Se l'algoritmo di Cholesky fallisce, il sistema è risolto mediante la fattorizzazione  $LU$ .

## 2.15 Esercizi di Calcolo matriciale

### 2.15.1 Alcuni esercizi sulle operazioni tra vettori e matrici

**Esercizio 1** Calcolare  $\alpha \underline{x}$ ,  $\underline{x}^T \underline{y}$ ,  $\underline{x} \underline{y}^T$  e  $\underline{x} + \underline{y}$  per:

- a)  $\alpha = 2.5$ ,  $\underline{x} = [6 \ 2]^T$ ,  $\underline{y} = [4 \ -2]^T$ ;
- b)  $\alpha = -2$ ,  $\underline{x} = \underline{y} = [4 \ -2 \ 2]^T$ .

**Esercizio 2** Posto:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad B = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}, \quad E = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \end{pmatrix},$$

$$\underline{c} = \begin{pmatrix} 4 \\ 1 \end{pmatrix}, \quad \underline{d} = \begin{pmatrix} 2 \\ 3 \end{pmatrix},$$

calcolare  $AB$  e  $BA$ . Usare tali risultati per determinare, se definiti:

- a)  $2A - 3B$ ,  $ABA$ ,  $A^2B$ ,  $BA^2$ ,  $EBA$ ,  $ABE$ ,  $BAE$ ;
- b)  $\underline{c}^T B$ ,  $E\underline{d}$ ,  $\underline{d}^T A$ .

**Esercizio 3** Verificare, utilizzando  $A$ ,  $B$ ,  $\underline{c}$  e  $\underline{d}$  dell'esercizio precedente, che:

- a)  $(\underline{c}^T A)B = \underline{c}^T(AB)$ ;
- b)  $(A + B)\underline{d} = A\underline{d} + B\underline{d}$ ;
- c)  $A(BA) = (AB)A$ .

**Esercizio 4** Verificare che per moltiplicare una matrice  $r \times s$  per una matrice  $s \times t$  sono richieste al piú  $rstM$  e  $rt(s-1)A$ .

**Esercizio 5** Scrivere un sottoprogramma FORTRAN  $dot(X, Y, N, C)$  che calcoli  $c = \underline{x}^T \cdot \underline{y}$ , con  $\underline{x}, \underline{y} \in R^n$ ,  $c \in R$ .

**Esercizio 6** Scrivere un sottoprogramma FORTRAN  $saxpy(S, X, Y, N, Z)$  che calcoli  $\underline{z} = s\underline{x} + \underline{y}$ , con  $\underline{x}, \underline{y}, \underline{z} \in R^n$ ,  $s \in R$ .

**Esercizio 7** Scrivere un sottoprogramma FORTRAN  $matvet(A, X, M, N, Y)$  che calcoli  $\underline{y} = A\underline{x}$ , con  $\underline{x}, \underline{y} \in R^n$ ,  $A \in R^{m \times n}$ .

**Esercizio 8** Scrivere un sottoprogramma FORTRAN  $add(S, A, B, M, N, C)$  che calcoli  $C = A + sB$ , con  $A \in R^{m \times n}$ ,  $B \in R^{m \times n}$  e  $s \in R$ .

**Esercizio 9** Scrivere un sottoprogramma FORTRAN  $mult(A, B, M, N, P, C)$  che calcoli  $C = AB$ , con  $A \in R^{m \times n}$ ,  $B \in R^{n \times p}$ .

### 2.15.2 Alcuni esercizi sul calcolo dell'inversa di una matrice

**Esercizio 1** Per le seguenti matrici:

$$A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix},$$

calcolare  $AB$ ,  $A^{-1}$ ,  $B^{-1}$  e  $(AB)^{-1}$ .

**Esercizio 2** Risolvere, mediante il calcolo dell'inversa della matrice dei coefficienti, i seguenti sistemi:

$$\begin{cases} x_1 + 2x_2 = -3 \\ 4x_1 + 3x_2 = 18 \end{cases} \quad \begin{cases} -2x_1 + x_2 = -1 \\ 3x_1 - x_2 = 3 \end{cases} \quad \begin{cases} 2x_1 - x_2 = 0 \\ 2x_1 + x_2 = 1 \end{cases}$$

**Esercizio 3** Determinare quante operazioni aritmetiche sono necessarie per risolvere  $A\underline{x} = \underline{b}$  calcolando  $A^{-1}\underline{b}$ , con  $A \in R^{3 \times 3}$ .

### 2.15.3 Alcuni esercizi su norme di vettori e matrici

**Esercizio 1** Per

$$\underline{x} = \begin{pmatrix} 2 \\ -5 \\ 3 \end{pmatrix}, \quad A = \begin{pmatrix} -1 & 1 & -4 \\ 2 & 2 & 0 \\ 3 & 3 & 2 \end{pmatrix}, \quad A^{-1} = \begin{pmatrix} -2 & 7 & -4 \\ 2 & -5 & 4 \\ 0 & -3 & 2 \end{pmatrix},$$

calcolare  $\|\underline{x}\|$ ,  $\|A\|$  e  $\|A^{-1}\|$ , con  $\|\cdot\| = \|\cdot\|_1$ ,  $\|\cdot\|_2$ ,  $\|\cdot\|_\infty$ .

**Esercizio 2** Per ciascuno dei seguenti vettori  $\underline{x}$ , calcolare  $\|\underline{x}\|_1$ ,  $\|\underline{x}\|_2$ ,  $\|\underline{x}\|_\infty$ :

$$[2 \ -5]^T, \ [2 \ -5 \ 5]^T, \ [0 \ 0 \ 0]^T.$$

**Esercizio 3** Per ciascuna delle seguenti coppie di vettori, calcolare  $\underline{x}^T \underline{y}$ ,  $\|\underline{x}\|_2$ ,  $\|\underline{y}\|_2$  e verificare che  $|\underline{x}^T \underline{y}| \leq \|\underline{x}\|_2 \|\underline{y}\|_2$  (*diseguaglianza di Schwarz*):

- a)  $\underline{x} = [1 \ -3]^T$ ,  $\underline{y} = [2 \ 1]^T$ ;
- b)  $\underline{x} = [-3 \ -4]^T$ ,  $\underline{y} = [1 \ -1]^T$ ;
- c)  $\underline{x} = [2 \ -3]^T$ ,  $\underline{y} = [-6 \ 9]^T$ .

**Esercizio 4** Verificare che  $\|\underline{x}\|_\infty \leq \|\underline{x}\|_2 \leq \|\underline{x}\|_1$ ,  $\forall \underline{x} \in R^n$ , e che l'uguaglianza può presentarsi anche per vettori non nulli.

### 2.15.4 Alcuni esercizi sulle proprietà delle matrici

**Esercizio 1** Per la seguente matrice:

$$A = \begin{pmatrix} -3 & 9 & 6 \\ 9 & 3 & -3 \\ -6 & 6 & -3 \end{pmatrix},$$

- a) valutare i minori  $M_{1,1}$ ,  $M_{1,3}$ ,  $M_{2,2}$ ,  $M_{3,2}$ ,  $M_{3,3}$ ;
- b) valutare i minori complementari  $A_{1,1}$ ,  $A_{2,2}$ ,  $A_{3,2}$ ,  $A_{3,3}$ ;
- c) valutare  $\det A$ .

**Esercizio 2** Calcolare i determinanti delle seguenti matrici:

$$a) \begin{pmatrix} 2 & -2 & 4 \\ 4 & 2 & 6 \\ 0 & 4 & -2 \end{pmatrix}, \quad b) \begin{pmatrix} 2 & -2 \\ 0 & 4 \end{pmatrix}, \quad c) \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad d) (-7).$$

**Esercizio 3** Verificare che  $\det L$  con  $L$  triangolare inferiore è uguale al prodotto degli elementi diagonali.

**Esercizio 4** Verificare che  $\det I_p = 1$ ,  $\forall p$ .

**Esercizio 5** Verificare che per  $P$  matrice di permutazione si ha che  $\det P = \pm 1$ .

**Esercizio 6** Verificare che  $\det(A^n) = (\det A)^n$ ,  $\forall n \in N$ .

## 2.16 Esercizi di Algebra Lineare

**Esercizio 1** La Pennsyltucky Pharmaceutical prepara una linea specialistica di tre integratori dietetici usando proporzioni diverse di tre sostanze naturali: estratto di broccoli, gingko biloba e olio di ginseng. Il prodotto 1 è realizzato combinando quantità uguali di tutte e tre le sostanze; il prodotto 2 è costituito prevalentemente da gingko biloba, a quattro dosi di esso corrisponde una dose di estratto di broccoli e olio di ginseng; il prodotto 3 impiega sei dosi di olio di ginseng, due di gingko biloba e una di estratto di broccoli. Queste combinazioni possono essere modellizzate da un insieme di equazioni lineari: se indichiamo con  $x_1$  la quantità di estratto di broccoli, con  $x_2$  la quantità di gingko biloba e con  $x_3$  la quantità di olio di ginseng, e denotiamo con  $p_1$ ,  $p_2$  e  $p_3$  la quantità di ciascun prodotto, allora otteniamo il sistema di equazioni

$$\begin{aligned} x_1 + x_2 + x_3 &= p_1 \\ x_1 + 4x_2 + x_3 &= p_2 \\ x_1 + 2x_2 + 6x_3 &= p_3. \end{aligned}$$

Data la quantità di produzione desiderata per ciascun prodotto  $p_i$ ,  $i = 1, 2, 3$ , per ciascuna sostanza possiamo determinare la quantità da comprare risolvendo questo sistema di equazioni. Questo particolare modello è molto semplice, soprattutto per la dimensione piccola del sistema, e non richiede alcuna tecnica specializzata per la sua risoluzione. Tuttavia, il successo di questa linea di integratori ha convinto la Pennsyltucky Pharmaceutical ad espandere sostanzialmente questo settore dei suoi affari, che sta pensando di produrre 50 prodotti fatti da altrettante sostanze naturali. La risoluzione di un sistema lineare  $50 \times 50$  non è da prendere alla leggera in assenza di un buon software [7].

### 2.16.1 Esercizi §2.4

**Esercizio 1** Risolvere i seguenti sistemi triangolari utilizzando gli algoritmi di back e forward substitution:

$$a) \begin{cases} x &= 6 \\ 4x + 6y &= 0 \\ 8x + 10y + 12z &= 8 \end{cases} \quad b) \begin{cases} 18x + 15y + 12z &= 12 \\ &+ 9y + 6z = 0 \\ &9z = 9 \end{cases}$$

$$c) \begin{cases} 2x &= -2 \\ 14x + 2y &= -10 \\ 12x + 10y + 2z &= 8 \\ 8x + 6y + 4z + 8u &= -10 \end{cases}$$

$$d) \begin{cases} 6x + 3y + 3z + 3u &= -3 \\ 6y + 3z - 6u &= 6 \\ 3z + 3u &= -3 \\ &6u = -6 \end{cases}$$

$$e) \begin{cases} 2x + 4y + 2z + 2u &= 12 \\ 2y + 10z - 6u &= -20 \\ 2z - 4u &= -10 \\ &2u = 4 \end{cases}$$

**Esercizio 2** Dimostrare che:

- a) se  $U$  è una matrice triangolare superiore invertibile, allora  $U^{-1}$  è triangolare superiore;
- b) se  $L$  è una matrice triangolare inferiore con elementi diagonali tutti uguali ad 1 (*unitaria*), allora  $L^{-1}$  è triangolare inferiore unitaria;
- c) il prodotto di due matrici triangolari superiori (inferiori) è una matrice triangolare superiore (inferiore).

**Esercizio 3** Scrivere un sottoprogramma FORTRAN  $bs(U, B, N, X, INDERR)$  per la procedura di Back-substitution.

**Esercizio 4** Scrivere un sottoprogramma FORTRAN  $fs(L, B, N, X, INDERR)$  per la procedura di Forward-substitution.

## 2.16.2 Esercizi §2.5

### Esercizi §2.5.2

**Esercizio 1** Risolvere, mediante l'algoritmo di eliminazione di Gauss e l'algoritmo di back substitution, il seguente sistema:

$$\begin{cases} 4x - 6y = 16 \\ 8x - 10y + 2z = 30 \\ 4x + 8z = 2 \end{cases}$$

**Esercizio 2** Risolvere, mediante l'algoritmo di eliminazione di Gauss e l'algoritmo di back substitution, i seguenti sistemi:

$$a) \begin{cases} 6x_1 - 2x_2 + 2x_3 = -2 \\ 18x_1 - 4x_2 + 2x_3 = -18 \\ 6x_1 + 2x_2 - 4x_3 = -18 \end{cases}$$

$$b) \begin{cases} -3x_1 - 3x_2 + 6x_3 = -15 \\ -9x_1 - 3x_2 + 21x_3 = -66 \\ 3x_1 - 9x_2 - 3x_3 = 30 \end{cases}$$

$$c) \begin{cases} -6x_1 - 12x_2 + 18x_3 = 0 \\ 2x_1 + 8x_2 + 2x_3 = 12 \\ 4x_1 + 16x_2 + 6x_3 = 26 \end{cases}$$

$$d) \begin{cases} 2x_1 + 4x_2 - 10x_3 + 2x_4 = 6 \\ 4x_1 - 6x_2 + 8x_3 - 4x_4 = 4 \\ 8x_1 + 2x_2 - 12x_3 + 6x_4 = 22 \\ 10x_1 + 18x_2 - 40x_3 + 2x_4 = 20 \end{cases}$$

**Esercizio 3** Calcolare, mediante l'algoritmo di eliminazione di Gauss, l'inversa di:

$$A = \begin{pmatrix} 4 & 6 \\ 6 & 8 \end{pmatrix}$$

**Esercizio 4** Applicare l'algoritmo di eliminazione di Gauss per determinare l'inversa destra delle seguenti matrici. Verificare se le matrici ottenute sono anche inverse sinistre (e quindi inverse).

$$a) \begin{pmatrix} 8 & -24 \\ 32 & 48 \end{pmatrix} \quad b) \begin{pmatrix} -2 & 4 & 2 \\ 0 & 2 & -4 \\ 2 & 8 & -2 \end{pmatrix} \quad c) \begin{pmatrix} 6 & 12 & 6 \\ 12 & 6 & -42 \\ 6 & 18 & 33 \end{pmatrix}$$

**Esercizio 5** Applicare l'algoritmo di eliminazione di Gauss ai seguenti sistemi. Se, ad un certo passo  $k$ ,  $a_{k,k}^{(k-1)} = 0$ , determinare una strategia che consenta comunque di ottenere la soluzione del sistema.

$$a) \begin{cases} 4x_1 + 12x_2 + 4x_3 = 12 \\ 8x_1 + 64x_2 + 8x_3 = -8 \\ \quad 8x_2 + 8x_3 = -24 \end{cases}$$

$$b) \begin{cases} 10x_1 + 10x_2 - 10x_3 = -20 \\ 10x_1 + 10x_2 - 20x_3 = 40 \\ -5x_1 + 5x_2 + 15x_3 = 20 \end{cases}$$

$$c) \begin{cases} 4x_2 + 2x_3 = 6 \\ -2x_1 + 2x_2 = 4 \\ 4x_1 - 2x_2 + 6x_3 = -10 \end{cases}$$

$$d) \begin{cases} 12x_2 + 6x_3 + 24x_4 = -18 \\ 6x_1 - 6x_2 - 12x_3 = 0 \\ 6x_1 + 36x_4 = -12 \\ -6x_1 + 18x_2 + 12x_3 = 0 \end{cases}$$

$$e) \begin{cases} 3x_1 - 3x_2 + 3x_4 = -9 \\ 6x_1 - 9x_2 - 3x_3 + 9x_4 = -18 \\ 9x_1 - 9x_2 + 6x_3 + 15x_4 = 27 \\ 12x_1 - 6x_2 + 6x_3 = -42 \end{cases}$$

**Esercizio 6** Risolvere il seguente sistema, mediante l'algoritmo di eliminazione di Gauss e l'algoritmo di back substitution, utilizzando un sistema aritmetico floating-point normalizzato con  $t = 2$  e la tecnica dell'arrotondamento:

$$\begin{cases} 0.98x_1 + 0.43x_2 = 0.91 \\ -0.61x_1 + 0.23x_2 = 0.48 \end{cases}$$

Confrontare la soluzione ottenuta con la soluzione vera  $x_1^* = 0.005946\dots$ ,  $x_2^* = 2.102727\dots$ , e con la migliore approssimazione a 2 cifre  $x_1' = 0.0059$ ,  $x_2' = 2.1$ .

**Esercizio 7** Scrivere una procedura per l'algoritmo di eliminazione di Gauss utilizzando l'operazione di tipo *saxpy*.

### Esercizi §2.5.3

**Esercizio 1** Determinare il numero di moltiplicazioni/divisioni necessario per risolvere, mediante l'algoritmo di eliminazione di Gauss e di back substitution, un sistema di equazioni  $p \times p$ , per  $p = 10; p = 50; p = 70; p = 90$ .

**Esercizi §2.5.4**

**Esercizio 1** Stabilire, applicando l'algoritmo di eliminazione di Gauss con pivoting parziale, se il seguente sistema ammette soluzioni:

$$\begin{cases} 2x + 2y + 2z = 0 \\ 2x + 4y + 6z = 0 \\ 6x + 10z + 14z = 2 \end{cases}$$

**Esercizio 2** Applicare l'algoritmo di eliminazione di Gauss con pivoting parziale ai seguenti sistemi. Discutere l'esistenza di soluzioni.

$$a) \begin{cases} 3x_1 + 6x_2 - 10x_3 = 6 \\ 6x_1 - 9x_2 + 12x_3 = 12 \\ 12x_1 + 3x_2 - 18x_3 = 24 \end{cases}$$

$$b) \begin{cases} 2x_1 + 4x_2 - 2x_3 + 4x_4 = 8 \\ 4x_1 + 14x_2 + 2x_3 + 2x_4 = 28 \\ 6x_1 + 16x_2 - 2x_3 + 8x_4 = 34 \\ 10x_1 + 18x_2 - 40x_3 + 2x_4 = 20 \end{cases}$$

**Esercizio 3** Verificare, applicando l'algoritmo di eliminazione di Gauss con pivoting parziale, che le seguenti matrici sono singolari:

$$a) \begin{pmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix} \quad b) \begin{pmatrix} -2 & 4 & 6 \\ 0 & 2 & 8 \\ 4 & -8 & -12 \end{pmatrix} \quad c) \begin{pmatrix} 3 & -3 & -3 & -6 \\ 6 & -9 & 6 & -3 \\ 9 & -15 & 15 & 0 \\ 3 & -6 & 9 & 3 \end{pmatrix}$$

**Esercizio 4** Verificare, applicando l'algoritmo di eliminazione di Gauss con pivoting parziale, che il seguente sistema:

$$\begin{cases} 4x_1 - 8x_2 + 12x_3 = 4 \\ 8x_1 + 4kx_2 + 24x_3 = 24 \\ -4x_1 + 12x_2 + 4(k-3)x_3 = 0 \end{cases}$$

- a) per  $k = 0$  ha infinite soluzioni;
- b) per uno specifico valore di  $k$ , da determinare, non ha soluzioni;
- c) per tutti gli altri valori di  $k$  ammette una sola soluzione.

**Esercizio 5** Risolvere, mediante l'algoritmo di eliminazione di Gauss con pivoting parziale, il seguente sistema:

$$\begin{cases} 3x + 3y - 3z = 6 \\ 9x + 9y + 3z = 6 \\ 3x + 3z = 0 \end{cases}$$

**Esercizio 6** Risolvere, mediante l'algoritmo di eliminazione di Gauss con pivoting parziale, i seguenti sistemi (esercizio 5 di Esercizi §2.5.2):

$$a) \begin{cases} 4x_1 + 12x_2 + 4x_3 = 12 \\ 8x_1 + 64x_2 + 8x_3 = -8 \\ 8x_2 + 8x_3 = -24 \end{cases}$$

$$b) \begin{cases} 10x_1 + 10x_2 - 10x_3 = -20 \\ 10x_1 + 10x_2 - 20x_3 = 40 \\ -5x_1 + 5x_2 + 15x_3 = 20 \end{cases}$$

$$c) \begin{cases} 4x_2 + 2x_3 = 6 \\ -2x_1 + 2x_2 = 4 \\ 4x_1 - 2x_2 + 6x_3 = -10 \end{cases}$$

$$d) \begin{cases} 12x_2 + 6x_3 + 24x_4 = -18 \\ 6x_1 - 6x_2 - 12x_3 = 0 \\ 6x_1 + 36x_4 = -12 \\ -6x_1 + 18x_2 + 12x_3 = 0 \end{cases}$$

$$e) \begin{cases} 3x_1 - 3x_2 + 3x_4 = -9 \\ 6x_1 - 9x_2 - 3x_3 + 9x_4 = -18 \\ 9x_1 - 9x_2 + 6x_3 + 15x_4 = 27 \\ 12x_1 - 6x_2 + 6x_3 = -42 \end{cases}$$

**Esercizio 7** Risolvere, applicando l'algoritmo di eliminazione di Gauss *i)* senza pivoting parziale e *ii)* con pivoting parziale, i seguenti sistemi utilizzando un sistema aritmetico floating-point con  $t = 3$ .

$$a) \begin{cases} 0.005x_1 + x_2 + x_3 = 2 \\ x_1 - 2x_2 + x_3 = 4 \\ -3x_1 - x_2 + 6x_3 = 2 \end{cases} \quad b) \begin{cases} x_1 - x_2 + 2x_3 = 5 \\ 2x_1 - 2x_2 + x_3 = 1 \\ 3x_1 - 2x_2 + 7x_3 = 20 \end{cases}$$

$$c) \begin{cases} 1.19x_1 + 2.37x_2 - 7.31x_3 + 1.75x_4 = 2.78 \\ 2.15x_1 - 9.76x_2 + 1.54x_3 - 2.08x_4 = 6.27 \\ 10.7x_1 - 1.11x_2 + 3.78x_3 + 4.49x_4 = 9.03 \\ 2.17x_1 + 3.58x_2 + 1.70x_3 + 9.33x_4 = 5.00 \end{cases}$$

Confrontare i risultati.

**Esercizio 8** Il pivoting parziale non sempre produce buoni risultati. Per verificare ciò, applicare l'algoritmo di Gauss con pivoting parziale, utilizzando un sistema aritmetico floating-point normalizzato con  $t$  cifre, al seguente sistema:

$$\begin{cases} 2x_1 + x_2 + x_3 = 1 \\ x_1 + \epsilon x_2 + \epsilon x_3 = 2\epsilon \\ x_1 + \epsilon x_2 - \epsilon x_3 = \epsilon \end{cases}$$

con  $\epsilon$  molto piccolo e noto alla massima accuratezza del sistema aritmetico. Mostrare, inoltre, che ponendo  $z_1 = x_1/\epsilon$ ,  $z_2 = x_2$ ,  $z_3 = x_3$ , e dividendo la II e la III equazione per  $\epsilon$  così da ottenere il sistema:

$$\begin{cases} 2\epsilon z_1 + z_2 + z_3 = 1 \\ z_1 + z_2 + z_3 = 2 \\ z_1 + z_2 - z_3 = 1 \end{cases}$$

l'algoritmo di Gauss con pivoting parziale fornisce una soluzione accurata.

**Esercizio 9** Scrivere un sottoprogramma FORTRAN *gausspiv(A, B, N, INDERR)* per il metodo di eliminazione di Gauss con pivoting parziale (Procedura 2.9), utilizzando l'operazione di tipo *saxpy*.

### Esercizi §2.6

**Esercizio 1** Applicare l'algoritmo di fattorizzazione LU con pivoting parziale, per calcolare il determinante e l'inversa della seguente matrice:

$$A = \begin{pmatrix} 1.723 & -1.421 & 3.784 \\ 0.113 & 4.071 & 1.213 \\ 5.131 & -4.010 & -2.176 \end{pmatrix}$$

**Esercizio 2** Calcolare la fattorizzazione *LU* (con *L* unitaria) della seguente matrice:

$$\begin{pmatrix} 2 & 8 & 0 \\ 0 & 6 & 2 \\ 0 & 2 & 0 \end{pmatrix}$$

**Esercizio 3** Risolvere, calcolando la fattorizzazione *LU* (con *L* unitaria), il seguente sistema:

$$\begin{cases} 12x_1 - 15x_2 + 3x_3 = 45 \\ 6x_1 + 12x_3 = 3 \\ 6x_1 - 9x_2 = 24 \end{cases}$$

**Esercizio 4** Applicare l'algoritmo di eliminazione di Gauss per costruire *L* (*L* unitaria) ed *U* tale che la seguente matrice  $A = LU$ :

$$A = \begin{pmatrix} 8 & -4 & 4 \\ 4 & 6 & -2 \\ 8 & -6 & -4 \end{pmatrix}.$$

**Esercizio 5** Risolvere, utilizzando l'algoritmo di fattorizzazione *LU*, i seguenti sistemi:

$$a) \begin{cases} 8x_1 + 64x_2 + 8x_3 = -8 \\ 4x_1 + 12x_2 + 4x_3 = 12 \\ 8x_2 + 8x_3 = -24 \end{cases}$$

$$b) \begin{cases} 10x_1 + 10x_2 - 10x_3 = -20 \\ -5x_1 + 5x_2 + 15x_3 = 20 \\ 10x_1 + 10x_2 - 20x_3 = 40 \end{cases}$$

$$c) \begin{cases} 4x_1 - 2x_2 + 6x_3 = -10 \\ 4x_2 + 2x_3 = 6 \\ -2x_1 + 2x_2 = 4 \end{cases}$$

$$d) \begin{cases} 6x_1 - 6x_2 - 12x_3 = 0 \\ 12x_2 + 6x_3 + 24x_4 = -18 \\ 6x_1 + 36x_4 = -12 \\ -6x_1 + 18x_2 + 12x_3 = 0 \end{cases}$$

$$e) \begin{cases} 12x_1 - 6x_2 + 6x_3 = -42 \\ 6x_1 - 9x_2 - 3x_3 + 9x_4 = -18 \\ 9x_1 - 9x_2 + 6x_3 + 15x_4 = 27 \\ 3x_1 - 3x_2 + 3x_4 = -9 \end{cases}$$

**Esercizio 6** Per ciascuna matrice  $A$  seguente calcolare la sua fattorizzazione  $LU$  (con  $L$  unitaria) e verificare che  $A = LU$ :

$$a) \begin{pmatrix} 4 & 8 \\ 4 & 2 \end{pmatrix} \quad b) \begin{pmatrix} -6 & 8 & 14 \\ 4 & 8 & 6 \\ 2 & 4 & -12 \end{pmatrix} \quad c) \begin{pmatrix} -4 & -8 & 4 & -4 \\ 2 & 10 & 10 & -16 \\ -2 & 0 & 14 & -22 \\ 4 & 14 & 6 & -6 \end{pmatrix}$$

**Esercizio 7** Determinare la fattorizzazione  $LU$  della seguente matrice  $A$ , con  $L$  ed  $U$  definite come segue, calcolando gli elementi di  $L$  e di  $U$  in base alla posizione  $A = LU$ :

$$A = \begin{pmatrix} 6 & 6 \\ 12 & -3 \end{pmatrix}; \quad L = \begin{pmatrix} l_{1,1} & 0 \\ l_{2,1} & l_{2,2} \end{pmatrix}; \quad U = \begin{pmatrix} 1 & u_{1,2} \\ 0 & 1 \end{pmatrix}$$

**Esercizio 8** Esprimere la seguente uguaglianza di matrici con 4 equazioni scalari:

$$\begin{pmatrix} a & 0 \\ b & c \end{pmatrix} \begin{pmatrix} d & e \\ 0 & f \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \quad (LU = A)$$

Dedurre che  $A$  può avere infinite fattorizzazioni  $LU$ . Determinare quella per cui  $L$  è unitaria.

**Esercizio 9** Dimostrare che se una matrice  $A$  non singolare ha una fattorizzazione  $LU$  in cui  $L$  è triangolare inferiore unitaria e  $U$  è triangolare superiore, allora  $L$  ed  $U$  sono uniche (Sugg: Supponendo che  $A = L_1U_1 = L_2U_2$  da cui si ha  $L_2^{-1}L_1 = U_2U_1^{-1}$ , dimostrare che l'uguaglianza sussiste solo se I e II membro sono uguali alla matrice identica).

**Esercizio 10** Verificare che la matrice:

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

non ha una fattorizzazione  $LU$  in cui  $L$  è unitaria.

**Esercizio 11** Verificare che ogni matrice della forma:

$$\begin{pmatrix} 0 & a \\ 0 & b \end{pmatrix}$$

ha una fattorizzazione  $LU$ . Verificare che, anche se  $L$  è triangolare inferiore unitaria, la fattorizzazione non è unica.

**Esercizio 12** Verificare che ogni matrice della forma:

$$\begin{pmatrix} 0 & 0 \\ a & b \end{pmatrix}$$

ha una fattorizzazione  $LU$ . Verificare se ha una fattorizzazione  $LU$  in cui  $L$  è unitaria.

**Esercizio 13** Determinare tutte le fattorizzazioni  $LU$ , in cui  $L$  è unitaria, della seguente matrice:

$$A = \begin{pmatrix} 1 & 3 \\ 5 & 15 \end{pmatrix}$$

**Esercizio 14** Calcolare la fattorizzazione  $LU$ , in cui  $L$  è triangolare inferiore ed  $U$  è triangolare superiore unitaria, della seguente matrice:

$$A = \begin{pmatrix} 3 & 0 & 1 \\ 0 & -1 & 3 \\ 1 & 3 & 0 \end{pmatrix}$$

**Esercizio 15** Calcolare la fattorizzazione  $LU$ , in cui sia  $L$  sia  $U$  sono unitarie, della seguente matrice:

$$A = \begin{pmatrix} 1 & 5 \\ 3 & 16 \end{pmatrix}$$

**Esercizio 16** Utilizzare la fattorizzazione  $LU$  (con  $L$  unitaria) per calcolare le soluzioni  $\underline{x}_1$ ,  $\underline{x}_2$  e  $\underline{x}_3$  di  $A\underline{x}_i = \underline{b}_i$ ,  $i = 1, 2, 3$  con:

$$A = \begin{pmatrix} 12 & 8 & -2 \\ -8 & 32 & 20 \\ 4 & -16 & 8 \end{pmatrix} ;$$

$$\underline{b}_1 = \begin{pmatrix} 58 \\ 92 \\ -64 \end{pmatrix}; \quad \underline{b}_2 = \begin{pmatrix} 34 \\ -4 \\ 20 \end{pmatrix}; \quad \underline{b}_3 = \begin{pmatrix} 20 \\ 24 \\ -12 \end{pmatrix}$$

**Esercizio 17** Verificare che la complessità computazionale richiesta per risolvere  $k$  sistemi  $A\underline{x}_i = \underline{b}_i$ ,  $1 \leq i \leq k$ , con  $A$  matrice qualsiasi,  $A \in R^{n \times n}$ , calcolando la fattorizzazione  $LU$  di  $A$ , è sempre minore di quella richiesta per calcolare  $A^{-1}$  con l'algoritmo di eliminazione di Gauss e quindi calcolare  $\underline{x}_i = A^{-1}\underline{b}_i$ ,  $1 \leq i \leq k$ .

**Esercizio 18** Scrivere una versione dell'algoritmo di fattorizzazione  $LU$  (con  $L$  unitaria) che calcoli, al  $k$ -mo passo, la  $k$ -ma riga di  $L$  e la  $k$ -ma riga di  $U$  (conseguentemente al  $k$ -mo passo l'ordine degli elementi da calcolare è  $l_{k,1}, l_{k,2}, \dots, l_{k,k-1}, u_{k,k}, u_{k,k+1}, \dots, u_{k,n}$ ) (*algoritmo di Doolittle per righe*). Scrivere poi una versione che calcoli, al  $k$ -mo passo, la  $k$ -ma colonna di  $U$  e la  $k$ -ma colonna di  $L$  (conseguentemente al  $k$ -mo passo l'ordine degli elementi da calcolare è  $u_{1,k}, u_{2,k}, \dots, u_{k,k}, l_{k+1,k}, u_{k+2,k}, \dots, l_{n,k}$ ) (*algoritmo di Doolittle per colonne o di Crout*).

**Esercizio 19** Determinare quale matrice di permutazione  $P$ , di dimensione  $3 \times 3$ , ha l'effetto di scambiare la I con la III riga, di una matrice  $A \in R^{3 \times 3}$ .

**Esercizio 20** Risolvere, usando la fattorizzazione  $LU$  (con  $L$  unitaria), *i*) senza e *ii*) con pivoting parziale, i seguenti sistemi:

$$a) \begin{cases} 3x_1 + 21x_2 + 18x_3 = 6 \\ 3x_1 - 6x_2 - 6x_3 = 0 \\ 6x_1 + 6x_2 - 12x_3 = 24 \end{cases} \quad b) \begin{cases} 2x_1 - 2x_2 + 4x_3 = 10 \\ 4x_1 - 4x_2 + 2x_3 = 2 \\ 6x_1 - 4x_2 + 14x_3 = 40 \end{cases}$$

$$c) \begin{cases} 3x_1 + 6x_2 + 3x_3 - 3x_4 = 3 \\ 3x_1 + 3x_2 + 3x_3 + 3x_4 = 6 \\ -6x_1 + 3x_2 - 6x_3 + 9x_4 = 3 \\ 9x_1 - 3x_2 + 12x_3 - 3x_4 = 0 \end{cases}$$

**Esercizio 21** Per ciascuna matrice  $A$  seguente, determinare la sua fattorizzazione  $P^T LU$ , applicando l'algoritmo di fattorizzazione  $LU$  con pivoting parziale.

$$a) \begin{pmatrix} 0 & 4 \\ 4 & 8 \end{pmatrix} \quad b) \begin{pmatrix} 2 & 6 & -4 \\ -4 & -12 & 11 \\ 3 & 14 & -16 \end{pmatrix} \quad c) \begin{pmatrix} 4 & 7.56 & 0.36 & -4.38 \\ -8 & -15.12 & 0.72 & 6.76 \\ 6 & 21 & 4 & 8 \\ 10 & 12 & 2 & -16 \end{pmatrix}$$

**Esercizio 22** Utilizzare l'algoritmo di fattorizzazione  $LU$  con pivoting parziale, per calcolare il determinante e l'inversa (se esiste) delle seguenti matrici:

$$a) A = \begin{pmatrix} 1 & 7 & 6 \\ 1 & -2 & -1 \\ 2 & 2 & -4 \end{pmatrix} \quad b) A = \begin{pmatrix} 1 & 2 & 3 \\ 5 & 7 & -1 \\ 7 & 8 & -11 \end{pmatrix}$$

### Esercizi §2.6.1

**Esercizio 1** Risolvere, applicando l'algoritmo di fattorizzazione  $LU$  con pivoting parziale con scambio virtuale delle righe, i seguenti sistemi: (esercizio 5 di Esercizi §2.5.2)

$$a) \begin{cases} 4x_1 + 12x_2 + 4x_3 = 12 \\ 8x_1 + 64x_2 + 8x_3 = -8 \\ 8x_2 + 8x_3 = -24 \end{cases}$$

$$b) \begin{cases} 10x_1 + 10x_2 - 10x_3 = -20 \\ 10x_1 + 10x_2 - 20x_3 = 40 \\ -5x_1 + 5x_2 + 15x_3 = 20 \end{cases}$$

$$c) \begin{cases} 4x_2 + 2x_3 = 6 \\ -2x_1 + 2x_2 = 4 \\ 4x_1 - 2x_2 + 6x_3 = -10 \end{cases}$$

$$d) \begin{cases} 12x_2 + 6x_3 + 24x_4 = -18 \\ 6x_1 - 6x_2 - 12x_3 = 0 \\ 6x_1 + 36x_4 = -12 \\ -6x_1 + 18x_2 + 12x_3 = 0 \end{cases}$$

$$e) \begin{cases} 3x_1 - 3x_2 + 3x_4 = -9 \\ 6x_1 - 9x_2 - 3x_3 + 9x_4 = -18 \\ 9x_1 - 9x_2 + 6x_3 + 15x_4 = 27 \\ 12x_1 - 6x_2 + 6x_3 = -42 \end{cases}$$

**Esercizio 2** Determinare, per ciascun sistema dell'esercizio precedente, il vettore finale  $IPIV$ .

**Esercizio 3** Scrivere un sottoprogramma FORTRAN  $sistlin(A, B, N, X, INDERR)$  per la risoluzione di un generico sistema lineare.

### 2.16.3 Esercizi §2.7

**Esercizio 1** Dimostrare che, qualunque sia la norma utilizzata, si ha che

$$\mu(A) = \|A\| \|A^{-1}\| \geq 1$$

**Esercizio 2** Per la seguente matrice

$$A = \begin{pmatrix} 1 & 0 \\ 0 & k \end{pmatrix}, \quad k > 0$$

determinare:

- a)  $\mu(A)$  in norma infinito;
- b) la più “vicina” matrice singolare in un sistema aritmetico floating-point normalizzato caratterizzato dai seguenti parametri:  $\beta = 10; t = 3; E_{min} = -3; E_{max} = 3$ , utilizzando la tecnica di troncamento.

**Esercizio 3** Calcolare  $\mu(A)$  in norma infinito della seguente matrice:

$$A = \begin{pmatrix} 1 & 1+\epsilon \\ 1-\epsilon & 1 \end{pmatrix}, \quad \epsilon > 0.$$

**Esercizio 4** Calcolare  $\mu(A)$  in norma infinito della seguente matrice:

$$A = \begin{pmatrix} 14 & 16 \\ 18 & 20 \end{pmatrix}$$

**Esercizio 5** Data la seguente matrice:

$$A = \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}$$

la cui matrice inversa è:

$$A^{-1} = \begin{pmatrix} 1 & -k \\ 0 & 1 \end{pmatrix},$$

si ha che, sia in norma uno sia in norma infinito:

$$\|A\| = \|A^{-1}\| = 1 + k, \quad \forall k \geq 0$$

cosicché  $\mu(A) = (1+k)^2$ , che è grande per  $k$  grande. Se si considera il sistema di equazioni  $A\underline{x} = \underline{b}$  con:

$$\underline{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

si ha che la soluzione è

$$\underline{x} = \begin{pmatrix} 1-k \\ 1 \end{pmatrix}.$$

Perturbando il termine noto mediante  $\delta \neq 0$  a

$$\underline{b} + \delta \underline{b} = \begin{pmatrix} 1 + \delta \\ 1 + \delta \end{pmatrix},$$

si ha che la variazione  $\delta \underline{x}$  nella soluzione è:

$$\delta \underline{x} = \begin{pmatrix} \delta(1 - k) \\ \delta \end{pmatrix}.$$

Determinare una maggiorazione di  $\|\delta \underline{x}\|/\|\underline{x}\|$  usando la norma infinito, per mostrare che il sistema è ben condizionato, anche avendo  $A$  un indice di condizionamento grande.

**Esercizio 6** Ripetere l'esercizio precedente ponendo:

$$\underline{b} = \begin{pmatrix} 1 \\ -1/k \end{pmatrix}.$$

Verificare che in questo caso il sistema è mal condizionato.

**Esercizio 7** Verificare, introducendo una perturbazione  $\delta \underline{b} = (0., 0.00001)^T$  nel termine noto  $\underline{b}$ , che il seguente sistema, la cui soluzione esatta è  $\underline{x} = (1., 1.)^T$ , è mal condizionato:

$$\begin{cases} x_1 + x_2 = 2.00000 \\ 1.00001x_1 + x_2 = 2.00001 \end{cases}$$

Calcolare  $\mu(A)$  usando la norma infinito.

**Esercizio 8** Considerato il seguente sistema:

$$\begin{cases} 2.67x_1 + 1.59x_2 = 1.08 \\ 1.41x_1 + 0.84x_2 = 0.57 \end{cases}$$

che ha soluzione esatta  $x_1 = 1.41$ ,  $x_2 = 0.84$ , determinare:

- a)  $\delta \underline{b}$  cosicché, sostituendo  $\underline{b}$  con  $\underline{b} + \delta \underline{b}$ , la soluzione calcolata sia  $x_1 = 0.47$ ,  $x_2 = -0.11$ ;
- b) se il sistema è ben o mal condizionato;
- c) l'indice di condizionamento usando la norma infinito.

**Esercizio 9** Utilizzare MATLAB per ottenere una soluzione approssimata del sistema dell'esercizio precedente. Determinare l'errore assoluto tra la soluzione calcolata e la soluzione esatta, in norma infinito.

**Esercizio 10** Risolvere il seguente sistema:

$$\begin{cases} 1.34x_1 + 7.21x_2 + 1.04x_3 = 9.60 \\ 3.18x_1 + 4.01x_2 + 0.98x_3 = 8.17 \\ 2.84x_1 - 2.40x_2 - 2.24x_3 = -23.4 \end{cases}$$

applicando l'algoritmo di eliminazione di Gauss con pivoting parziale, utilizzando un sistema aritmetico floating-point con  $t = 3$ . Perturbare, poi, il termine noto della 1 equazione a 9.59, e risolvere alla stessa maniera il sistema così ottenuto. Cosa si può dedurre per questo sistema?

## 2.17 Esercizi con il MATLAB

### 2.17.1 Esercizi §2.14.4

**Esercizio 1** Definire i seguenti vettori:

- a)  $a = (1.01 \ 2.3 \ \sqrt{2} \ \cos(1))$
- b)  $b = (1 \ 2 \ 3 \ 4)$  mediante l'operatore *colon*
- c)  $c = (4 \ 3 \ 2 \ 1)$  mediante l'operatore *colon*
- d)  $d = (1 \ 1.5 \ 2 \ 2.5 \ 3)$  mediante l'operatore *colon*.

Calcolare poi la somma e la differenza dei vettori  $b$  e  $c$ , infine calcolare il loro prodotto scalare, verificando la relazione:

$$bc^T = cb^T$$

**Esercizio 2** Estrarre la diagonale principale, la prima colonna, la seconda riga e il blocco  $2 \times 2$ ,  $A_{1,2;1,2}$  della seguente matrice:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

**Esercizio 3** Costruire la matrice C aggiungendo alla matrice A dell'esercizio precedente la riga

$$a = \begin{pmatrix} 7 & 8 & 9 \end{pmatrix}$$

calcolare poi la trasposta di C, e la matrice che si ottiene aggiungendo a C la colonna  $a^T$ .

**Esercizio 4** Data la matrice A dell'esercizio 2 e la matrice:

$$B = \begin{pmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{pmatrix}$$

provare tutte le possibili combinazioni di A e B e le loro trasposte per costruire una nuova matrice, e riflettere sui risultati ottenuti.

**Esercizio 5** Verificare con un esempio le seguenti relazioni:

$$\begin{aligned}(C^T)^T &= C \\ (C^T + D^T) &= (C + D)^T \\ (\lambda C)^T &= \lambda C^T\end{aligned}$$

(Si utilizzi la matrice C dell'esercizio 3,  $\lambda = 0.5$ ,  $D = \lambda C$ ).

**Esercizio 6** Si calcoli la trasposta della seguente matrice:

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

Si verifichi che  $E^T - E = 0$ . Perché?

**Esercizio 7** Costruire le seguenti matrici:

- a) la matrice diagonale  $4 \times 4$  D con  $D_{ii} = i$  per  $i=1,2,3,4$
- b) la matrice identica di ordine 4
- c) la matrice  $4 \times 4$  O, tale che  $O_{ii} = 1$
- d) la matrice nulla  $4 \times 4$
- e) i vettori riga  $(1, 1, 1, 1)$  e  $(0, 0, 0, 0)$ .

**Esercizio 8** Date le matrici

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \quad B = \begin{pmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{pmatrix}$$

calcolarne il prodotto. Verificare che  $(AB)^T = B^T A^T$ , e che le matrici  $A^T A$ ,  $B^T B$  sono quadrate e simmetriche.

**Esercizio 9** Date le seguenti matrici:

$$\begin{aligned}A_{11} &= \begin{pmatrix} 1 & 2 \\ 5 & 6 \end{pmatrix} & A_{12} &= \begin{pmatrix} 3 & 4 \\ 7 & 8 \end{pmatrix} & A_{21} &= \begin{pmatrix} 9 & 10 \\ 13 & 14 \end{pmatrix} & A_{22} &= \begin{pmatrix} 11 & 12 \\ 15 & 16 \end{pmatrix} \\ B_{11} &= \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} & B_{12} &= B_{11} & B_{21} &= \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix} & B_{22} &= B_{21}\end{aligned}$$

si consideri

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

Si verifichi che

$$A + B = \begin{pmatrix} A_{11} + B_{11} & A_{12} + B_{12} \\ A_{21} + B_{21} & A_{22} + B_{22} \end{pmatrix}$$

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

**Esercizio 10** Costruire una m-function che conta gli elementi nulli in una matrice.

**Esercizio 11** Dato il seguente sistema lineare:

$$\begin{cases} x_1 + 2x_3 + 3x_4 = 1 \\ -x_1 + 2x_2 + 2x_3 - 3x_4 = -1 \\ x_2 + x_3 + 4x_4 = 2 \\ 6x_1 + 2x_2 + 2x_3 + 4x_4 = 1 \end{cases}$$

si chiede di:

- a) Verificare che il sistema è compatibile.
- b) Risolvere il sistema mediante l'operatore \.
- c) Risolvere il sistema mediante la fattorizzazione LU della matrice dei coefficienti.
- d) Risolvere il sistema calcolando l'inversa della matrice dei coefficienti.
- e) Calcolare mediante la funzione *flops* la complessità di tempo dei tre procedimenti.

**Esercizio 12** Calcolare la norma uno, la norma euclidea, la norma infinito di:

$$A = \begin{pmatrix} 2 & -3 & 1 \\ 6 & 2 & 0 \\ -1 & 3 & 2 \end{pmatrix} \quad b = \begin{pmatrix} 4 \\ -1 \\ 6 \end{pmatrix}.$$

Verificare poi le seguenti relazioni valide in generale:

$$\|A\|_1 = \max_j \sum_i |a_{ij}| \quad \|A\|_\infty = \max_i \sum_j |a_{ij}|$$

$$\|b\|_1 = \sum_j |b_j| \quad \|b\|_1 = \max_j |b_j|$$

**Esercizio 13** Data la matrice dell'esercizio precedente calcolare  $\mu(A)$  e verificare le seguenti relazioni:

- a)  $\mu(A) = \|A\| \|A^{-1}\|$
- b)  $\mu(A) = \mu(\alpha A)$  per  $\forall \alpha$
- c)  $\mu(AA^T) = \mu(A)^2$
- d)  $\mu(A) = \mu(A^T) = \mu(A^{-1})$

**Esercizio 14** Risolvere con MATLAB i sistemi lineari del secondo esercizio del §2.16.2, e verificare i risultati ottenuti, inoltre valutare il condizionamento delle matrici dei coefficienti sia tramite *cond* che tramite *rcond*.

**Esercizio 15** Risolvere con MATLAB i sistemi lineari del quinto esercizio del §2.16.2, e verificare i risultati ottenuti, inoltre valutare il condizionamento delle matrici dei coefficienti sia tramite *cond* che tramite *rcond*.

**Esercizio 16** Costruire due m-function una per la backward substitution e una per la forward substitution. Testarle poi con i sistemi del primo esercizio del §2.16.1.



# Bibliografia

- [1] Anderson E., Bai Z., Bischof C., Blackford S., Demmel J., Dongarra J., Du Croz J., Greenbaum A., Hammarling S., McKenney A., Sorensen D. - *LAPACK Users' Guide* - terza edizione, SIAM, 1999.
- [2] Demmel J. - *Applied numerical linear algebra* - SIAM, 1997.
- [3] Dongarra J.J., Bunch J.R., Moler C.B., Stewart G.W. - *LINPACK Users' Guide* - SIAM, 1979.
- [4] Dongarra J.J., Du Croz J., Duff I.S., Hammarling S. - *A set of Level 3 Basic Linear Algebra Subprograms* - ACM Transactions on Mathematical Software, vol. 16 (1990), pp. 1-17.
- [5] Dongarra J.J., Du Croz J., Hammarling S., Hanson R.J. - *An extended set of FORTRAN basic linear algebra subroutines* - ACM Transactions on Mathematical Software, vol. 14 (1988), pp. 1-17.
- [6] Edelman A. e Mascarenhas W. - On the complete pivoting conjecture for a Hadamard matrix of order 12. *Linear and Multilinear Algebra*, 38(3):181-188, 1995.
- [7] Epperson J.F. - *Introduzione all'analisi numerica* Teoria, metodi, algoritmi - McGraw-Hill.
- [8] Fletcher C.A.J. - *Computational Techniques for Fluid Dynamics* - volume 1, seconda edizione, Springer, 1991.
- [9] Golub G.H., Van Loan C.F. - *Matrix Computations* - third ed., The Johns Hopkins University Press, 1997.
- [10] Gould N.I.M. - *On growth in Gaussian elimination with complete pivoting*. - SIAM J. Matrix Anal. Appl., 12(2): 354-361, 1991.
- [11] Higham N.J. - *Accuracy and Stability of Numerical Algorithms* - II ed. SIAM
- [12] Lawson C.L., Hanson R.J., Kincaid D., Krogh F.T. - *Basic linear algebra subprograms for Fortran usage* - ACM Transactions on Mathematical Software, vol. 5 (1979), pp. 308-323.

- [13] Smith B.T., Boyle J.M., Dongarra J.J., Garbow B.S., Ikebe Y., Klema V.C., Moler C.B. - *Matrix Eigensystem Routines - EISPACK Guide* - Lecture Notes in Computer Science, vol. 6, Springer-Verlag, 1976.
- [14] Trefethen L.N., Bau D. - *Numerical linear algebra* - SIAM, 1997.
- [15] Trefethen Lloyd N. e Schreiber R.S. - *Average-case stability of Gaussian elimination* - SIAM J. Matrix Anal. Appl., 11(3):335-360, 1990.
- [16] Wilkinson J.H. - *The Algebraic Eigenvalue Problem* - Oxford University Press, 1965. xviii+662pp.
- [17] Wilkinson J.H. - *Error analysis of direct methods of matrix inversion* - J. Assoc. Comput. Mach., 8:281-330, 1961.

# Capitolo 3

## La rappresentazione di dati

### 3.1 Introduzione

Esistono molti problemi della *fisica*<sup>1</sup> in cui la relazione tra due (o più) grandezze  $X, Y$  è nota solo attraverso un certo numero di valutazioni o misure  $x_i$  di  $X$  e  $y_i$  di  $Y$  ( $i = 1, 2, 3, \dots$ ), ottenute attraverso esperimenti, indagini statistiche, o semplici rilevamenti diretti.

In tali circostanze, può essere necessario:

- rappresentare questi dati in modo più conciso di quanto consenta la semplice descrizione “elemento per elemento”;
- desumere nuove informazioni sulla base dei dati già noti.

La deduzione di informazioni nuove può consistere semplicemente nella stima di valori di  $Y$  corrispondenti a prefissati valori di  $X$  *senza dover eseguire ulteriori esperimenti o misurazioni*, oppure nella valutazione di grandezze nuove, ottenute da  $Y$  mediante operazioni opportune, quali ad esempio un integrale o il calcolo di una derivata. Questi obiettivi possono essere raggiunti mediante la formulazione di un opportuno **modello matematico**.

In questo capitolo ci occupiamo del problema della costruzione di un modello matematico nel caso in cui si abbia solo un numero finito di informazioni. Più precisamente, illustreremo metodi, algoritmi e software per la **rappresentazione di dati**.

♣ **Esempio 3.1.** In Italia, dal 1921 ad oggi sono stati effettuati censimenti ogni dieci anni, fatta eccezione nel 1941. A partire dai dati registrati in questi censimenti, ci si propone di stimare la variazione subita dalla popolazione italiana tra un censimento e l’altro.

---

<sup>1</sup>Qui e nel seguito per problema *fisico* si intende un qualunque problema che riguarda un fenomeno concreto del mondo reale (fisico, chimico, economico, sociale, etc.).

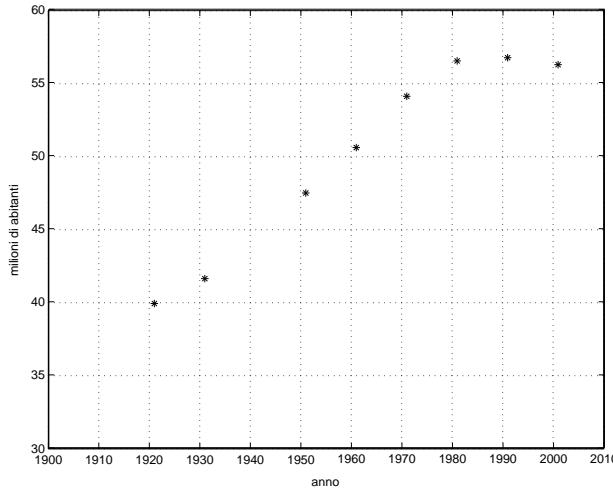


Figura 3.1: Dati registrati nei censimenti effettuati in Italia dal 1921 al 2001 <sup>2</sup>

anno	1921	1931	1951	1961	1971	1981	1991	2001
n. abit. ( $\times 10^3$ )	39944	41652	47516	50624	54137	56557	56778	56305

Rappresentiamo come punti di un piano cartesiano le informazioni relative ad ogni censimento, cioè riportiamo l'anno in cui il censimento è stato effettuato, sull'asse delle ascisse, ed il numero di abitanti rilevati in quell'anno sull'asse delle ordinate (vedi Fig. 3.1).

Osservando la figura, abbiamo una rappresentazione immediata dell'andamento della popolazione italiana dal primo all'ultimo censimento. Tale descrizione però è **parziale**, perché ottenuta da un **numero finito di informazioni** ciascuna relativa **unicamente** al momento in cui è stato effettuato il censimento. Pertanto, basandoci anche su una certa consuetudine, per poter stimare cosa accade tra un censimento e l'altro congiungiamo a due a due tali punti con un segmento di retta, come mostrato nella figura seguente.

Si noti come, in Fig. 3.2, a partire da un numero finito di dati siamo riusciti a dare una descrizione generale del problema che, tra l'altro, consente eventualmente di dedurre nuove informazioni. Ad esempio, se siamo interessati a stimare la velocità di crescita del numero di abitanti in Italia, per ciascun decennio, basta calcolare il coefficiente angolare del relativo segmento di retta.

Il grafico<sup>3</sup> disegnato in Figura 3.2, si può ritenere un **modello matematico attendibile**.<sup>4</sup>



<sup>3</sup>[...] Se per certi valori  $x_1, x_2, \dots, x_n$  assunti da  $x$ , si conoscono o si sanno calcolare i corrispondenti valori  $y_1, y_2, \dots, y_n$  di  $y$ , fissati due assi cartesiani, si posson costruire nel piano i punti di coordinate  $(x_1, y_1)$ ,  $(x_2, y_2)$ , ...,  $(x_n, y_n)$ , i quali son vertici di una linea poligonale, che rappresenta con tanta maggior fedeltà l'andamento della funzione  $y$  di  $x$ , quanto più vicini sono i suoi successivi vertici. Questa linea è il *grafico* della funzione. [...] [5]

<sup>4</sup>Per ora possiamo solo intuire il significato della parola *attendibile* basandoci su considerazioni empiriche o sull'esperienza che ciascuno ha. Come vedremo in seguito, tale parola avrà un significato più preciso relativamente alla definizione che daremo, in questo contesto, di modello matematico.

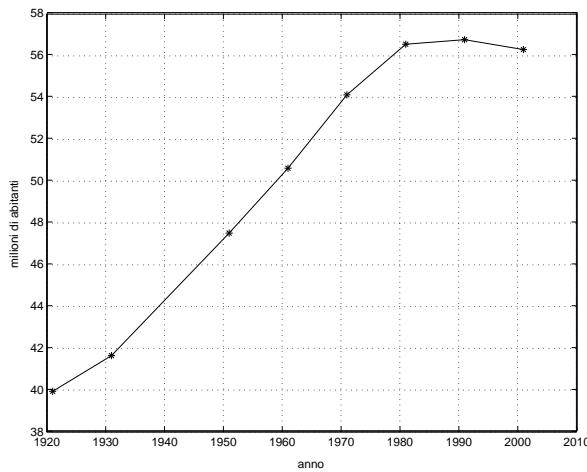


Figura 3.2: Un modello matematico per descrivere l'andamento della popolazione italiana dal 1921 al 2001

Prima dell'avvento dei calcolatori, una delle applicazioni più vaste del calcolo numerico è stata la costruzione di tavole relative alle funzioni elementari trascendenti, che non fossero combinazioni lineari delle quattro operazioni aritmetiche, cioè logaritmi, funzioni trigonometriche, etc. Una volta costruite le tavole, per valori dell'argomento opportunamente fissati<sup>5</sup>, si poneva il problema di come valutare la funzione nei valori non *tabulati* cioè non presenti nella tabella<sup>6</sup>. L'idea più naturale era quella di fissare un intervallo delimitato da due valori consecutivi nella tabulazione, congiungere le ordinate corrispondenti ai punti estremi dell'intervallo con un segmento di retta e calcolare, per ogni ascissa interna all'intervallo fissato, l'ordinata del punto sul segmento di retta con quella determinata ascissa. In particolare, per ciascun intervallo di tabulazione, venivano precalcolati anche i coefficienti dell'equazione di ciascun segmento di retta, in modo che fosse ridotto al minimo il numero di operazioni aritmetiche necessarie alla valutazione in un qualsiasi punto.

La rappresentazione delle funzioni matematiche elementari rientrava nell'ambito della costruzione di un modello matematico a partire da un numero finito di informazioni. Le tavole, infatti, per quanto fossero estese, consentivano di disporre delle valutazioni di una funzione solo in un numero finito e prefissato di punti. Solo con l'avvento dei

<sup>5</sup>Nei primi anni del XX secolo E. Whittaker, docente all'Università di Edimburgo, introduceva l'interpolazione ai suoi studenti come uno strumento per ... *leggere tra le righe di una tabella matematica.*".

<sup>6</sup>Ad esempio utilizzando strumenti costruiti appositamente, come le macchine analizzatrici, a partire dai valori assunti negli archi notevoli ( $15^\circ, 30^\circ, 45^\circ, 60^\circ$ ) si valutavano meccanicamente le funzioni trigonometriche in altri valori ottenuti da combinazioni degli archi notevoli.

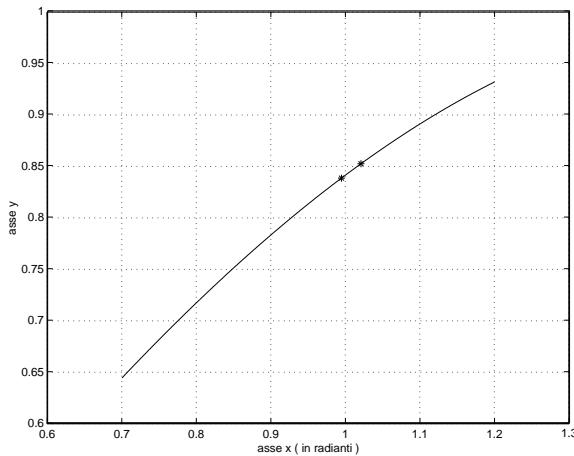


Figura 3.3: Grafico della funzione  $y = \sin(x)$  nell'intervallo  $[57^\circ, 58^\circ 5']$

calcolatori, si è potuto parlare di *generazione* delle funzioni elementari<sup>7</sup> cioè della rappresentazione di tali funzioni in modo da consentire la loro efficiente valutazione per qualunque valore *rappresentabile* dell'argomento e con la *massima accuratezza possibile*.

♣ **Esempio 3.2.** Calcolare il valore della funzione  $\sin(x)$  per  $x = 1 \text{ rad}$ . avendo a disposizione una tavola che fornisce i valori delle funzioni trigonometriche quando gli argomenti sono espressi in gradi. In questo caso, poiché  $1 \text{ rad} = 57^\circ 17' 44.81''$  le informazioni deducibili dalla tavola sono:

$$\begin{aligned}\sin(57^\circ) &= 0.83867 \\ \sin(58^\circ) &= 0.84805\end{aligned}$$

avendo supposto di avere per ciascun punto della tavola un'approssimazione corretta a cinque cifre significative. Supponendo per la funzione  $\sin(x)$  un andamento lineare<sup>8</sup> nell'intervallo  $[57^\circ, 58^\circ 5']$ , uniamo con un segmento di retta i punti assegnati (vedi Fig. 3.4).

Come si può anche osservare dai grafici in Fig. 3.3, 3.4, i due punti sono talmente vicini che solo un "ingrandimento" consente di distinguerli. In questo caso, la curva  $y = \sin(x)$ (linea tratteggiata) e

<sup>7</sup> Attualmente le funzioni elementari dei compilatori dei linguaggi di programmazione ad alto livello sono rappresentate da opportune combinazioni lineari di polinomi.

<sup>8</sup> Questa ipotesi è coerente, tenuto conto che localmente, cioè su intervalli di ampiezza *sufficientemente piccola* ogni curva può essere *assimilata* alla retta secante passante per i due estremi dell'intervallo. In particolare, considerato l'intervallo  $[x_1, x_2]$  sostituendo alla funzione  $f(x)$  rappresentante la curva, il polinomio di Taylor di primo grado (supponendo che la funzione sia derivabile) si ha:

$$y = f(x) \approx f(x_1) + f'(x)(x - x_1) \quad x \in [x_1, x_2]$$

sostituendo la derivata col rapporto incrementale della funzione calcolato tra  $x_1$  e  $x_2$ :

$$y \approx f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1) \quad x \in [x_1, x_2]$$

si ottiene l'equazione della retta secante congiungente i punti  $(x_1, y_1)$  e  $(x_2, y_2)$ .

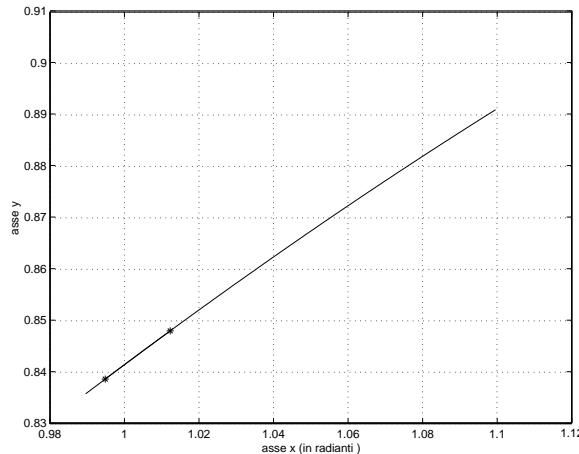


Figura 3.4: Grafico della funzione  $y = \sin(x)$  nell'intervallo  $[57^{\circ}17', 58^{\circ}18']$

la retta secante (linea continua) "praticamente" coincidono, nel senso che non si riesce ad apprezzarne la differenza.

Il valore di  $\sin(x)$  per  $x = 57^{\circ}17'44.81'' = 1\text{rad}$  è quello assunto dalla funzione che rappresenta il segmento di retta nell'intervallo tra i due punti di tabulazione  $[57^{\circ}17', 57^{\circ}18']$ . Il modello matematico costruito in questo esempio si può ritenere *attendibile* a patto che i due punti estremi dell'intervallo siano "sufficientemente" vicini. ♣

Spesso, nel rappresentare un insieme di dati occorre tener conto di ulteriori informazioni che riguardano la natura del problema fisico da cui provengono. Infatti, in assenza di questa informazione, fissato l'insieme di dati, possono esistere differenti modelli i quali, evidentemente, descrivono in maniera diversa l'uno dall'altro, i dati assegnati. In altre parole, un'informazione fondamentale nella costruzione di un modello matematico che rappresenti i dati è la *forma* (retta, parabola, sinusoida, etc.) del modello stesso.

♣ **Esempio 3.3.** Siano assegnati i seguenti punti:

x	-2	$-\pi/2$	-1	$\pi/3$	$\pi/2$	3
y	0.41	1.08	-0.13	-0.39	-1.58	0.97

A partire da questi punti, possiamo costruire un polinomio che passi per questi punti, oppure, se ad esempio sappiamo che i dati provengono dalla misura di una marea, che in genere è periodica, possiamo costruire un polinomio trigonometrico che passi per essi.

Come si osserva dalla Figura 3.6 la descrizione del fenomeno cambia in modo evidente a seconda della scelta della funzione. ♣

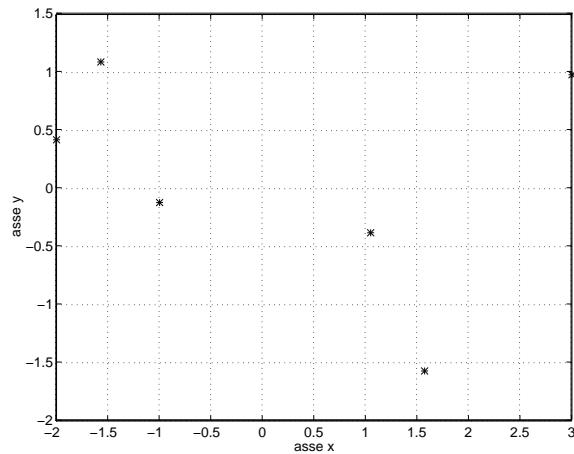


Figura 3.5: Dati ottenuti da misurazioni del livello del mare

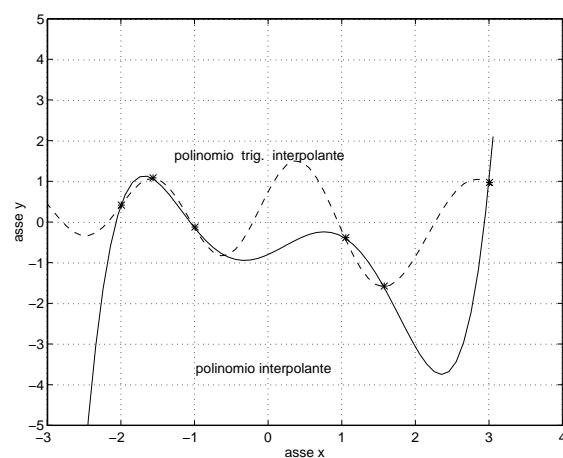


Figura 3.6: Un confronto tra due modelli interpolanti

Tuttavia, se nell'acquisire i dati che descrivono il problema, si introducono perturbazioni dovute, ad esempio, allo strumento di misura utilizzato, può accadere che sebbene la dipendenza tra le grandezze in esame debba soddisfare, ad esempio, ad una legge lineare, i punti che in un sistema di riferimento cartesiano rappresentano tali grandezze non sono allineati. Come fare in questi casi a costruire un modello attendibile, cioè un modello che tenga conto sia delle perturbazioni introdotte sia della natura del fenomeno? In tal caso un modello che, in un *certo senso*, si scosti *poco* dai punti senza che sia costretto a passare per essi (fitting), così da non esaltare l'eventuale errore di misura, può risultare molto più ragionevole di uno che sia invece vincolato a passare per tali punti.

♣ **Esempio 3.4.** Una massa di 5 kg è sospesa verticalmente ad una estremità di una molla. Supponendo che la molla sia fissata all'altra estremità si vuole determinare la costante d'elasticità della molla.

Per piccoli spostamenti dalla posizione di equilibrio, la forza esercitata dalla massa sulla molla è proporzionale allo spostamento subito dalla molla e il fattore di proporzionalità risulta essere proprio la costante di elasticità  $K$ . In altre parole, lo spostamento  $x$  della molla soddisfa alla *Legge di Hooke*:

$$F = -K x$$

dove  $F$  è la forza esercita dalla molla (tensione della molla)<sup>9</sup>. Introduciamo un sistema di riferimento cartesiano  $(x, F)$ , con l'origine coincidente con la posizione di equilibrio della molla cioè  $0 \equiv (x = 0, F = 0)$  e misuriamo per alcuni valori  $x_i$  dello spostamento  $x$ , la tensione  $F_i$  esercitata dalla molla in corrispondenza degli  $x_i$  (vedi Fig. 3.7), e rappresentiamo i punti di coordinate  $(x_i, F_i)$ .

x (cm)	2.5	5.	10.	17.5	22.5	30.	32.5	37.5	42.5
F (Kg)	0.6804	1.7690	2.9937	5.3070	7.0760	8.5275	8.8904	9.344	9.5707

Dalla Fig. 3.7 si può osservare che i punti di coordinate  $(x_i, F_i)$ , non sono allineati. Pertanto se congiungiamo a due a due tali punti abbiamo un andamento lineare *a tratti*. Poiché la *pendenza* di questi *tratti* di retta, cioè il loro coefficiente angolare dovrebbe rappresentare il valore stimato della costante  $K$ , in questo modo avremmo diversi valori per  $K$ , uno per ogni *tratto* di retta. In altre parole, il modello ottenuto congiungendo a due a due i punti risulterebbe *poco attendibile*.

Ciò è dovuto al fatto che, nel misurare la tensione esercitata dalla molla, i valori  $F_i$  sono, in maniera inevitabile, affetti da errori di misurazione. Invece di considerare i tratti di rette passanti per i punti (linea tratteggiata), se consideriamo la retta rappresentata in Fig. 3.8 con la linea continua appare evidente che essa è quella che meglio tiene conto dell'andamento lineare del fenomeno, dal momento che si avvicina ai punti pur non passando necessariamente per essi. In questo modo si è ottenuto un modello matematico attendibile. Il coefficiente angolare di questa retta è il valore stimato per  $K$ .



<sup>9</sup>Il segno negativo sta ad indicare che in seguito ad un allungamento della molla ( $x > 0$ ) la tensione esercitata dalla molla è di tipo attrattivo perché è orientata nel verso opposto all'allungamento. Il contrario avviene in seguito ad una compressione della molla ( $x < 0$ ).

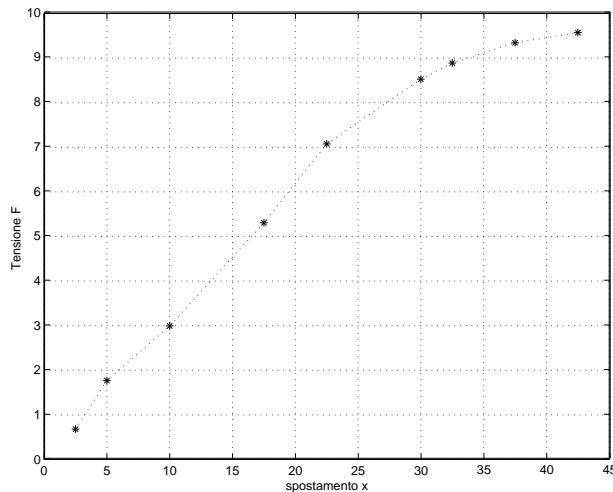
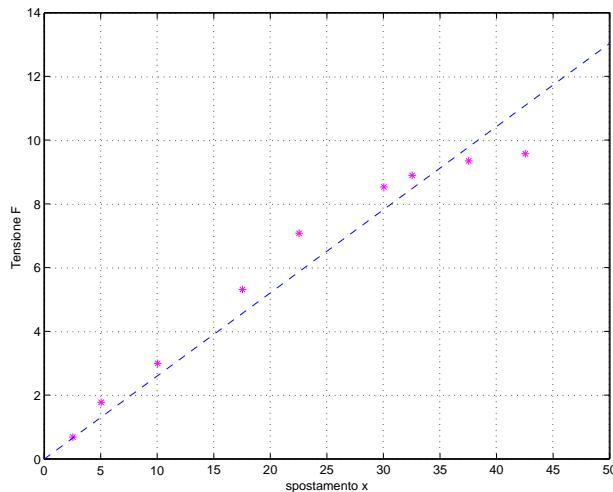
Figura 3.7: Tensione  $F$  esercitata dalla molla in funzione dello spostamento  $x$ 

Figura 3.8: Un modello matematico per stimare la costante elastica di una molla

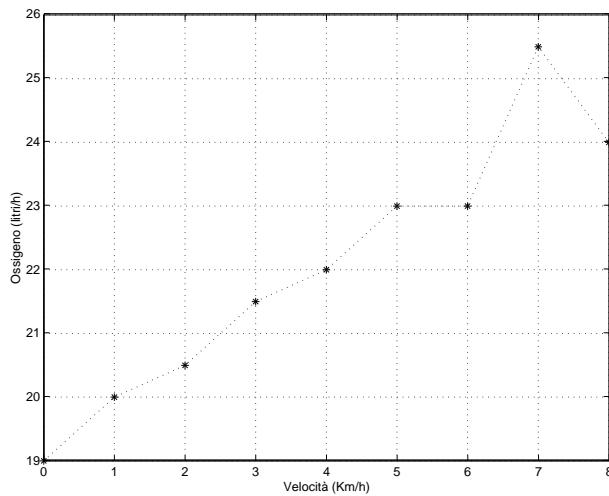


Figura 3.9: Quantità di ossigeno respirata in funzione della velocità dell'individuo.

♣ **Esempio 3.5.** Mediante un'apparecchiatura detta spirometro si può valutare il volume d'aria che entra o esce dai polmoni di un individuo, e quindi il volume d'ossigeno inspirato. In un esperimento vengono effettuate varie misure su un soggetto in movimento. Si vuole dare una descrizione generale di quanto ossigeno viene respirato a varie velocità.

v (Km/h)	0	1	2	3	4	5	6	7	8
oss (litri/h)	19	21	20.5	21.5	22	24	24	25.5	24

Considerazioni sulla natura fisica del fenomeno fanno presumere che la quantità di ossigeno respirata dipenda linearmente dalla velocità dell'individuo. Ma anche in questo caso, i punti non sono allineati. Adottiamo come modello per descrivere il fenomeno, la retta disegnata in Fig. 3.10 è quella che, in un *certo senso*, si scosta il meno possibile dai punti senza però passare per essi, così da non esaltare l'errore di misura da cui sono affetti. Ne deduciamo, osservando anche la Fig. 3.8, che un modello il cui grafico non debba necessariamente passare per i punti assegnati, risulta molto più ragionevole di uno invece vincolato a passare per tali punti.



Come abbiamo visto negli esempi precedenti, il primo passo nella risoluzione di un qualsiasi problema concreto consiste nella **individuazione delle relazioni esistenti tra le grandezze, note ed incognite, che intervengono nel problema**, in altre

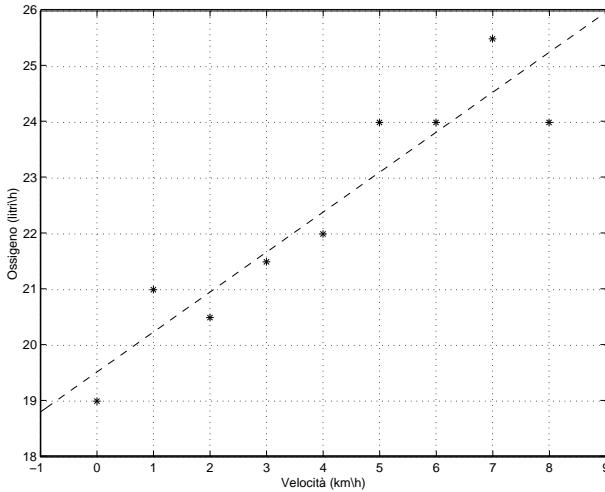


Figura 3.10: Un modello matematico per descrivere i dati registrati dallo spirometro.

parole, nell'individuazione del **modello matematico** che descrive il problema. La formulazione di un modello matematico, negli esempi citati, si traduce nella determinazione di una funzione  $f$ , definita in un intervallo  $I$ , con  $I \supset \{x_i\}$ , che:

- i) rappresenti i dati  $(x_i, y_i)$ ;
- ii) conservi eventuali altre proprietà della correlazione tra le grandezze; e che
- iii) consenta di ottenere nuove informazioni eventualmente richieste.

Allora, per costruire un modello matematico che soddisfi ai requisiti (i)  $\div$  (iii), cioè necessari perché il modello sia *attendibile*, è opportuno tener conto delle leggi che governano il fenomeno. Queste informazioni possono essere di aiuto nel definire la *forma* del modello cioè il tipo di funzione  $f$  (ad es. una retta, un parabola, una funzione trigonometrica, etc.). Inoltre, è importante osservare che in alcuni casi affinchè un modello possa essere considerato attendibile non è necessario che il grafico di  $f$  passi per tutti i punti assegnati. Questo accade nell'esempio 3.4 e nell'esempio 3.5.  
Possiamo dare allora la seguente:

### Definizione 3.1. (Fitting o modello)

Dato un insieme finito di dati  $D = \{(x_i, y_i)_{i=1,\dots,n}\}$  appartenenti ad un intervallo  $I$ , tale cioè che  $I \supset \{x_i\}$ , ogni funzione  $f$ , definita su  $I$  che descrive  $D$ , si dice un **fitting o modello** per  $D$ ; tale funzione è poi detta **interpolante**  $D$  se sono verificate delle condizioni sulla funzione e/o sulle sue derivate nei punti assegnati, cioè se sono soddisfatte le condizioni:

$$f(x_i) = y_i \quad (\text{in generale } f^{(j)}(x_i) = y_i^j, j \in J \subseteq \mathbf{N}_0) \quad \forall i = 1, \dots, n. \quad (3.1)$$

Le condizioni (3.1) sono dette *condizioni di interpolazione* in quanto caratterizzano il modello interpolante.

Se invece, si richiede che la funzione  $f$  sia tale che<sup>10</sup>:

$$f = \arg \min_{i=1, \dots, n} |f(x_i) - y_i|$$

allora la funzione  $f$  fornisce un modello **approssimante**. Per determinare un modello per  $D$  soddisfacente i requisiti (i) ÷ (iii) è dunque necessario in primo luogo stabilire se  $f$  debba:

- solo *approssimare*;
- solo *interpolare*;
- *interpolare ed approssimare*.

Nel caso in cui si scelga un modello approssimante, è poi necessario anche poter stabilire una **misura** di quanto  $f$  si *scosti* dai punti in  $D$ .

È naturale, a questo punto, chiedersi perché introdurre i modelli approssimanti che impongono la scelta ulteriore di un criterio con cui misurare lo scostamento quando i modelli interpolanti consentono di rappresentare fedelmente l'insieme di dati di cui si dispone.

Come già accennato, il ricorso ad un modello interpolante vincolato fortemente ai dati ha senso solo quando questi ultimi siano affetti da errori *trascurabili*. In caso contrario, quando si presume che i dati siano affetti da errori *non trascurabili*, non avrebbe senso vincolare una funzione ad assumere tali valori perché ciò potrebbe comportare un'amplificazione dell'errore. Osserviamo che, negli esempi descritti abbiamo supposto che nel problema del censimento l'errore possa essere considerato trascurabile, mentre nell'esperimento dello spirometro, o della molla, l'errore di misura ha fatto sí che i punti non fossero più allineati, cioè l'errore non può essere considerato trascurabile. Pertanto, nel primo caso si è scelto un modello interpolante, negli altri, si è preferito un modello solo approssimante.

Ciò premesso diamo le seguenti definizioni:

### Definizione 3.2. (Problema di interpolazione)

*Dati*  $n$  *valori distinti*  $(x_i)_{i=1, \dots, n}$  *detti nodi*, ed  $n$  *valori corrispondenti*

---

<sup>10</sup>In questo contesto viene preso in considerazione il problema di approssimazione detto "di migliore approssimazione", in cui si richiede che la distanza tra la funzione approssimante e i punti assegnati sia minima; quando si richiede che la distanza sia minore di  $\epsilon$ , si parla, semplicemente, di approssimazione di ordine  $\epsilon$ .

$(y_i)_{i=1,\dots,n}$ , si vuole determinare una funzione  $f$ , detta **funzione interpolante**, che nei nodi  $(x_i)_{i=1,\dots,n}$  soddisfi a certe condizioni, dette **condizioni di interpolazione**. Tali condizioni, in generale, sono vincoli che la funzione interpolante  $f$  (e/o le sue derivate), deve soddisfare nei punti  $(x_i, y_i)_{i=1,\dots,n}$ .

Mentre, un problema di approssimazione di dati può essere enunciato nel modo seguente<sup>11</sup>:

#### Definizione 3.4. (Problema di approssimazione)

Dati  $n$  valori distinti  $(x_i)_{i=1,\dots,n}$ , detti **nodi** ed  $n$  valori corrispondenti  $(y_i)_{i=1,\dots,n}$ , si vuole determinare una funzione  $f$ , detta **funzione approssimante**, la cui distanza nei nodi  $(x_i)_{i=1,\dots,n}$  dai valori  $(y_i)_{i=1,\dots,n}$  sia minima; la scelta della misura di tale distanza e il tipo di vincolo imposto a tale distanza qualifica il problema di approssimazione considerato.

Come abbiamo anche visto dagli esempi, l'interpolazione e l'approssimazione forniscono due modelli sostanzialmente differenti, anche se **storicamente** si tendeva a confonderli. Uno dei motivi è che tra le funzioni approssimanti, potremmo scegliere quelle che nei nodi assumono i valori  $y_i$ , in altre parole potremmo scegliere di costruire funzioni approssimanti che siano anche interpolanti. In realtà come vedremo in seguito, questo ha senso in alcuni problemi di analisi numerica.

Il capitolo è organizzato come segue: cominciamo a prendere in esame il problema dell'interpolazione, successivamente quello dell'approssimazione. In particolare, in 3.2, introduciamo l'**interpolazione polinomiale di Lagrange**, in cui si richiede che la funzione interpolante sia un polinomio e che le condizioni di interpolazione siano vincoli imposti **unicamente** sulla funzione interpolante.

In 3.5 viene presentato un particolare problema di approssimazione di dati: quello in cui la funzione approssimante è un polinomio costruito richiedendo che sia minima la somma dei quadrati delle distanze dei punti assegnati. In tal caso si ha l'**approssimazione nel senso dei minimi quadrati**.

## 3.2 Il problema dell'interpolazione

Come abbiamo detto, la costruzione di un modello matematico che descriva in modo attendibile dei dati affetti da errori che possono essere considerati trascurabili, conduce

---

<sup>11</sup>In questo contesto verrà discussa una formulazione semplificata del problema dell'approssimazione di dati. Più in generale sussiste la seguente

**Definizione 3.3.** Data una funzione  $f$ , definita su uno spazio di Banach  $\mathcal{B}$ , si vuole determinare una funzione  $\tilde{f}$ , che sia semplice da determinare e da valutare, in termini di complessità computazionale, che si scosti il meno possibile dalla funzione  $f$ . Lo scostamento tra  $f$  e  $\tilde{f}$  è misurato dalla norma sullo spazio  $\mathcal{B}$  della differenza tra  $f$  e  $\tilde{f}$ :  $\|f - \tilde{f}\|_{\mathcal{B}}$ . La funzione  $\tilde{f}$  è detta **funzione approssimante**.

ad una funzione interpolante.

♣ **Esempio 3.6.** Una pallottola è sparata verso l'alto dalla sommità di un edificio alto 100 metri. Viene osservato che dopo 10 secondi dal lancio la pallottola raggiunge la massima altezza di 590 metri. Si vuole disegnare la traiettoria seguita dalla pallottola finché questa raggiunge il suolo.

Il moto della pallottola è un moto uniformemente accelerato con accelerazione uguale all'accelerazione di gravità  $g$ . Se introduciamo un sistema di riferimento  $(x, y)$  in cui sull'asse  $x$  riportiamo il tempo espresso in secondi, e sull'asse  $y$  riportiamo l'altezza della pallottola con l'origine coincidente con la base dell'edificio, la traiettoria seguita dalla pallottola è una parabola di equazione  $y = ax^2 + bx + c$ , il cui vertice rappresenta il punto in cui la pallottola raggiunge la massima altezza; in questo punto, inoltre, la velocità è nulla.

Per disegnare la traiettoria della pallottola, ovvero per tracciare il grafico della parabola  $y = ax^2 + bx + c$ , è necessario conoscere i tre coefficienti della parabola,  $a, b, c$ . A tal fine imponiamo che la curva passi per il punto di coordinate  $(0, 100)$ , (posizione iniziale al tempo iniziale) e per il punto di coordinate  $(10, 590)$  (massima altezza dopo 10 secondi). Richiediamo, infine, che nel punto di coordinate  $(10, 590)$  ci sia il vertice della parabola (vedi Fig. 3.11).

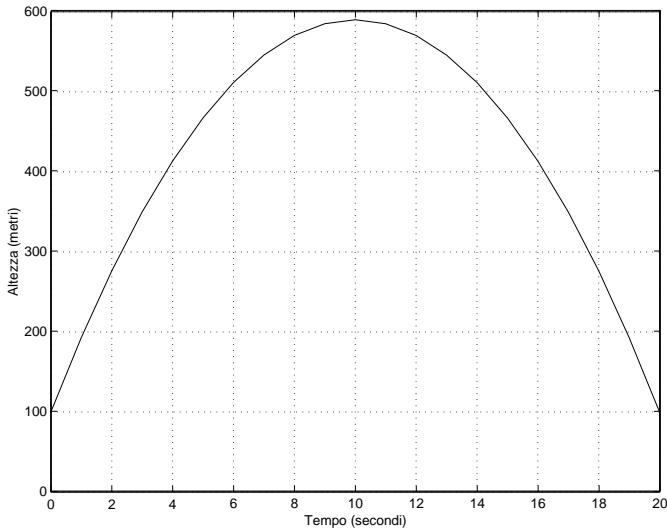


Figura 3.11: Traiettoria seguita dalla pallottola

Osserviamo che tale curva è stata ottenuta conoscendo due punti appartenenti alla curva e la "pendenza" della curva in uno dei due punti<sup>12</sup>. ♣

Le condizioni richieste alla funzione interpolante possono riguardare esclusivamente la funzione stessa oppure coinvolgere anche le sue derivate.

**♣ Esempio 3.7.** Calcolare il valore della funzione  $f(x) = \log(x)$  per  $x = 1.54$  avendo a disposizione una tavola che fornisce i valori della funzione logaritmo per valori di  $x$  distanziati di 0.1.

Considerato l'intervallo di tabulazione  $[x_1 = 1.5, x_2 = 1.6]$  e supponendo che siano tabulati valori corretti a 4 cifre significative, si ha:

$x_i$	1.5	1.6
$\log(x_i)$	0.4055	0.4700

utilizziamo come funzione interpolante la retta passante per  $(x_1, y_1), (x_2, y_2)$ . Tale retta ha equazione:

$$\frac{x - x_2}{x_1 - x_2} = \frac{y - y_2}{y_1 - y_2} \implies y = y_2 + (y_1 - y_2) \frac{x - x_2}{x_1 - x_2} = r(x) ,$$

valutiamo la funzione  $y = r(x)$  in  $\bar{x} = 1.54$ :

$$r(1.54) = 0.47 + (0.4055 - 0.47) \frac{1.54 - 1.6}{1.5 - 1.6} = 0.4313 .$$

---

<sup>12</sup>Imponendo che il vertice della parabola coincida con il punto di massima altezza, in effetti, abbiamo imposto che la "pendenza" della parabola, cioè la tangente alla parabola nel vertice sia la retta parallela all'asse delle ascisse di equazione  $y = 590$ .

Tale quantità rappresenta il valore di  $\log(1.54) = 0.4317$ , corretto a tre cifre significative<sup>13</sup>.



In generale, se le condizioni richieste alla funzione interpolante coinvolgono unicamente i valori che essa assume in certi punti si ha il **problema di interpolazione di Lagrange**.

### Definizione 3.5. (Problema di interpolazione di Lagrange)

*Dati  $n$  nodi distinti  $(x_i)_{i=1,\dots,n}$ , ed  $n$  valori  $(y_i)_{i=1,\dots,n}$ , determinare una funzione  $f$  tale che:*

$$f(x_i) = y_i, \quad i = 1, \dots, n$$

Tali condizioni sono dette **condizioni di interpolazione di Lagrange**.

Il problema di interpolazione di Lagrange richiede quindi che per ciascun nodo sia assegnata una ed una sola condizione; uno schema grafico dell'interpolazione di Lagrange è il seguente:

nodi	$x_1$	$x_2$	$\dots$	$\dots$	$x_n$
condizioni	$y_1$	$y_2$	$\dots$	$\dots$	$y_n$

**Schema riassuntivo dell'Interpolazione di Lagrange**

♣ **Esempio 3.8.** Calcolare  $f(1.54) = \log(1.54)$  supponendo che sia noto il valore che la funzione logaritmo assume per  $x_1 = 1.5$ .

Poiché manca il valore in un altro nodo non possiamo procedere come nell'esempio 3.7, però possiamo tener conto del fatto che, essendo  $f'(x) = \frac{1}{x}$ , è possibile conoscere il valore di  $f'(x_1)$ .

Si ha:

$$f(1.5) = 0.4055, \quad f'(1.5) = \frac{1}{1.5} = \frac{2}{3} = 0.6666$$

Per calcolare  $\log(1.54)$ , determiniamo tra tutte le rette che passano per il punto di ascissa  $x_1 = 1.5$  quella di coefficiente angolare  $f'(1.5)$ :

$$y = r_1(x) = y_1 + f'(1.5) \cdot (x - x_1) = 0.4055 + 0.6666 \cdot (x - 1.5)$$

e sostituendo a  $x$  il valore 1.54, calcoliamo  $r_1(1.54)$ :

$$r_1(1.54) = y_1 + f'(1.5) \cdot (1.54 - 1.5) = 0.4055 + 0.6666 \cdot 0.04 = 0.4322$$

Tale valore è un'approssimazione di  $\log(1.54)$  corretta a 2 cifre significative<sup>14</sup>.



<sup>13</sup>Osserviamo che tale valore è stato ottenuto supponendo che nell'intervallo [1.5, 1.6] il grafico della funzione logaritmo fosse assimilabile al segmento di retta secante passante per i punti della curva di ascissa 1.5 e 1.6

<sup>14</sup>Osserviamo che tale approssimazione è stata ottenuta supponendo che nell'intervallo [1.5, 1.54] il grafico della funzione logaritmo sia assimilabile alla retta tangente passante per il punto della curva di ascissa 1.5.

Se le condizioni di interpolazione coinvolgono anche le derivate della funzione si ha il **problema di interpolazione di Hermite**:

**Definizione 3.6. (Problema di interpolazione di Hermite)**

Assegnati  $n$  nodi distinti  $(x_i)_{i=1,\dots,n}$ ,  $n$  interi positivi  $l_1, l_2, \dots, l_n$ , tali che:

$$\sum_{i=1}^n l_i = m$$

ed  $m$  valori  $(y_i^j)$  con  $i = 1, \dots, n$ ;  $j = 0, \dots, l_i - 1$ , determinare la funzione  $f$  tale che<sup>15</sup>:

$$f^{(j)}(x_i) = y_i^j \quad i = 1, \dots, n; \quad j = 0, \dots, l_i - 1$$

Il problema di interpolazione di Hermite richiede quindi che per ciascun nodo sia assegnata almeno una condizione; inoltre, se in un nodo è assegnata una condizione sulla derivata di un ordine  $q$  allora devono essere necessariamente assegnate in quel nodo tutte le condizioni sulle derivate di ordine inferiore, cioè le condizioni sulle derivate di ordine  $j$  con  $j = 0, \dots, q - 1$ <sup>16</sup>. Una rappresentazione grafica di ciò è la seguente:

nodi	$x_1$	$x_2$	...	...	$x_n$
condizioni	$y_1$	$y_2$	...	...	$y_n$
	$y_1^1$	$y_2^1$	...	...	$y_n^1$
	$y_1^2$		...	...	$y_n^2$
	$y_1^3$		...	...	$y_n^3$

Interpolazione di Hermite

Osserviamo che nella formulazione generale di un problema di interpolazione, sia esso di Lagrange o di Hermite, non abbiamo precisato la *forma* della funzione interpolante. Dovendo costruire un modello per questi dati, uno dei primi quesiti che si pone è il seguente: quale forma scegliere per la funzione interpolante ? In generale, la scelta della *forma* dipende esclusivamente dal problema e dagli obiettivi

<sup>15</sup>Nel seguito si userà la notazione  $f^{(j)}(x_i)$  per indicare la derivata di ordine  $j$  della funzione  $f$  valutata nel punto  $x_i$  ed  $f^{(0)} \equiv f$

<sup>16</sup>Se invece in corrispondenza di un nodo ci sono delle *lacune* cioè mancano una o più condizioni sulla derivata della funzione approssimante si ha il **problema di interpolazione di Hermite - Birckoff**.

per i quali si costruisce il modello interpolante<sup>17</sup>.

In base a tale scelta si caratterizza il **tipo di interpolazione** cioè ad esempio si parla di:

- **interpolazione polinomiale**, se si richiede che la funzione interpolante sia un polinomio di un certo grado;
- **interpolazione trigonometrica**, se si richiede che la funzione interpolante sia un polinomio trigonometrico di un certo grado;
- **interpolazione mediante spline**, se si richiede che la funzione interpolante sia una spline di un certo grado.

Ad esempio, in Figura 3.11, è disegnato il grafico di una funzione interpolante di tipo polinomiale, in particolare una parabola.

Nel seguito focalizziamo l'attenzione sull'**interpolazione polinomiale**.

### 3.2.1 Esistenza e unicità del polinomio interpolante di Lagrange

Il primo problema da affrontare ai fini della costruzione del polinomio interpolante riguarda la sua esistenza e, soprattutto, la sua unicità.

♣ **Esempio 3.9.** Siano  $P_1$  e  $P_2$  due punti distinti; è noto che per  $P_1$  e  $P_2$  passa un'unica retta, ma quante sono le parabole che passano per tali punti? Come si può anche osservare dalla Figura 3.12, fissati  $P_1$  e  $P_2$ , al variare del punto  $P$  esiste una ed una sola parabola, con asse di simmetria parallelo all'asse  $y$ , passante per  $P_1$ ,  $P_2$  e  $P$ . Potendo variare  $P$  in infiniti modi esistono infinite parabole che passano per  $P_1$  e  $P_2$ .

In un sistema di assi cartesiani  $(x, y)$  una parabola si rappresenta con un'equazione di secondo grado del tipo:

$$y = p(x) = a_2x^2 + a_1x + a_0 \quad a_0, a_1, a_2 \in \mathbb{R}, \quad a_2 \neq 0$$

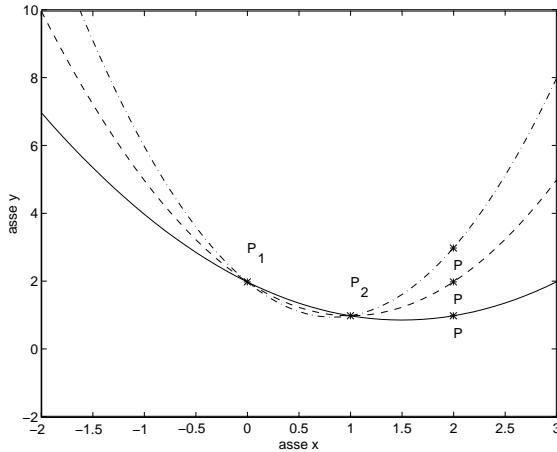
con  $a_0, a_1, a_2$  univocamente determinati<sup>18</sup>, al variare del punto  $P$  esiste una ed una sola terna di coefficienti che individuano univocamente la parabola passante per  $P_1$ ,  $P_2$  e  $P$ . Potendo variare  $P$  in infiniti modi, si ha che esistono infinite terne di coefficienti  $a_0, a_1, a_2$  per cui la corrispondente parabola passa sia per  $P_1$ , sia per  $P_2$ .

In altre parole, dati due punti esiste un'unica retta passante per essi ma i due punti, da soli, non bastano a determinare univocamente la parabola passante per essi. ♣

<sup>17</sup>Osserviamo che specificare la *forma* del modello interpolante equivale a specificare lo spazio  $\mathcal{F}$  nel quale viene formulato il problema di interpolazione e al quale la funzione interpolante  $f$  deve appartenere.

<sup>18</sup>Con ciò si intende dire che una parabola si può caratterizzare come il luogo dei punti di coordinate  $(x, y)$  che soddisfano ad una equazione di secondo grado del tipo

$$y = p(x) = a_2x^2 + a_1x + a_0 \quad a_0, a_1, a_2 \in \mathbb{R}, \quad a_2 \neq 0$$

Figura 3.12: Parabole interpolanti i punti  $P_1$ ,  $P_2$  e  $P$ 

Il risultato al quale si è pervenuto nell'esempio precedente, è in realtà a carattere generale. Consideriamo il problema seguente:

**Problema di interpolazione polinomiale di Lagrange:**

Assegnati due punti distinti del piano,  $P_1 \equiv (x_1, y_1)$  e  $P_2 \equiv (x_2, y_2)$ , con  $x_1 \neq x_2$ , determinare per quali valori di  $m$  esiste ed è unico il polinomio<sup>19</sup>:

$$p(x) = a_m x^m + a_{m-1} x^{m-1} + \dots + a_0$$

di grado al più  $m$ , tale che:

$$\begin{cases} p(x_1) = y_1 \\ p(x_2) = y_2 \end{cases} \quad (3.2)$$

Tale problema, come mostrato in Fig. 3.13:

- se  $m = 0$ , non ammette soluzione (a meno che  $y_1 = y_2 = a_0$ , nel qual caso la soluzione è la retta parallela all'asse delle ascisse e tale che tutti suoi punti hanno ordinata  $a_0$ , cioè  $p(x) \equiv a_0$ ).
- se  $m = 1$ , ammette un'unica soluzione (esiste un'unica retta passante per i due punti assegnati  $y = p(x) = a_1 x + a_0$ , con  $a_0$  e  $a_1$  univocamente determinati);
- se  $m = 2$ , ammette infinite soluzioni, cioè in infiniti modi è possibile scegliere la terna dei coefficienti  $a_0, a_1, a_2$  del polinomio  $p(x)$  soddisfacente le condizioni (3.2)

<sup>19</sup>Cioè esiste un'unica  $(m+1)$ -pla di valori  $a_0, a_1, \dots, a_m$ .

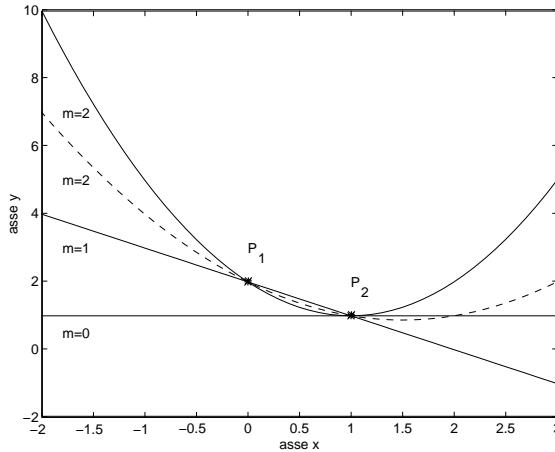


Figura 3.13: Polinomi di grado  $m$  interpolanti i punti  $P_1$  e  $P_2$

Si può dunque concludere che **il polinomio  $p \in \Pi_m$ , interpolante gli  $n = 2$  punti assegnati esiste ed è unico se e solo se  $m = 1 = n - 1$ .**

Che succede se invece di due punti ne consideriamo tre? In altri termini, dati tre punti di che grado deve essere il polinomio interpolante tali punti perchè sia l'unico? E se i punti sono quattro, cinque, etc.?

Ebbene, la relazione tra il numero di punti e il grado del polinomio interpolante, appena ottenuta per due punti, è di carattere generale, sussistendo qualunque siano il numero dei punti assegnati e la scelta dei punti  $(x_i, y_i)$  il seguente:

**Teorema 3.1. [Unicità del polinomio interpolante di Lagrange]**

*Dati  $n$  nodi distinti  $(x_i)_{i=1,\dots,n}$  ed  $n$  valori corrispondenti  $(y_i)_{i=1,\dots,n}$ , il polinomio  $p$  di grado al più  $m$  ( $p \in \Pi_m$ ), tale che:*

$$p(x_i) = y_i, \quad i = 1, \dots, n$$

*è unico se:*

$$m \leq n - 1.$$

**Dimostrazione** Supponiamo che esistano due polinomi distinti  $q$  e  $r$ , con  $q \in \Pi_{n-1}$  e  $r \in \Pi_{n-1}$  tali che:

$$q(x_i) = y_i, \quad r(x_i) = y_i \quad i = 1, \dots, n$$

dimostriamo che  $q = r$ .

Il polinomio differenza  $p(x) = q(x) - r(x) \in \Pi_{n-1}$  è soluzione del problema omogeneo.

$$p(x_i) = 0 \quad i = 1, \dots, n \tag{3.3}$$

Dimostriamo quindi che se  $m = n - 1$  il problema omogeneo (3.3) ammette solo la soluzione identicamente nulla.

Osserviamo che la (3.3) equivale a richiedere che il polinomio  $p$  ammetta  $n$  zeri distinti:  $x_1, \dots, x_n$ . Per il *Teorema fondamentale dell'algebra* un polinomio di grado al più  $n - 1$  può ammettere al più  $n - 1$  zeri distinti, pertanto  $p \in \Pi_{n-1}$  deve essere necessariamente il polinomio identicamente nullo. ■

Esiste una seconda dimostrazione del Teorema 3.1, che tra l'altro fornisce una condizione necessaria e sufficiente affinché il polinomio interpolante sia unico e, inoltre, fornisce anche un metodo per costruire tale polinomio<sup>20</sup>:

**Dimostrazione 2** Posto:

$$p(x) = a_m x^m + a_{m-1} x^{m-1} + \dots + a_0 \in \Pi_m,$$

dimostriamo che il polinomio  $p(x)$  tale che:

$$p(x_i) = y_i, \quad i = 1, \dots, n$$

esiste ed è unico se e solo se  $m = n - 1$ .

Osserviamo che quanto richiesto equivale a dimostrare che il sistema lineare:

$$\left\{ \begin{array}{l} a_0 + a_1 x_1 + \dots + a_{m-1} x_1^{m-1} + a_m x_1^m = y_1 \\ a_0 + a_1 x_2 + \dots + a_{m-1} x_2^{m-1} + a_m x_2^m = y_2 \\ \vdots \quad \dots \quad \dots \quad \dots \quad \dots \quad \vdots \\ a_0 + a_1 x_n + \dots + a_{m-1} x_n^{m-1} + a_m x_n^m = y_n \end{array} \right. \quad (3.4)$$

di dimensione  $n \times (m + 1)$ , nelle incognite  $a_0, a_1, \dots, a_{m-1}, a_m$  ammette un'unica soluzione.

Detta  $A$  la matrice dei coefficienti di questo sistema, indichiamo con  $r$  il rango di  $A$ , e supponiamo che la matrice  $A$  abbia rango massimo, cioè  $r = \min\{n, m + 1\}$ . Indicato con:

$$N(A) = \{x \in \Re^{m+1} : Ax = 0\}$$

il sottospazio nullo di  $A$ , e con

$$R(A) = \{y \in \Re^n : \exists x \in \Re^{m+1} : Ax = y\}$$

il codominio di  $A$  (o range di  $A$ ), ricordiamo che:

$$\dim(N(A)) + \dim(R(A)) = \dim(N(A)) + r = m + 1$$

pertanto la soluzione, se esiste, è unica se e solo se:

$$N(A) = \{\underline{0}\}$$

e quindi se e solo se:

$$\dim(N(A)) = 0$$

cioè se e solo se  $r = m + 1$ . Da ciò deriva:

$$r = m + 1 \leq n \quad (3.5)$$

---

<sup>20</sup>Come vedremo, però, questo metodo non è computazionalmente praticabile perché generalmente dà luogo ad un problema mal condizionato

D'altra parte, comunque si scelga il vettore dei termini noti  $b \in \Re^n$  perché sia sempre garantita l'esistenza della soluzione del sistema lineare (3.4), è necessario che  $b \in R(A)$ , con:

$$R(A) = \{y \in \Re^n : \exists x : Ax = y\}$$

Poiché:

$$\dim(R(A)) = r = m + 1 \quad (3.6)$$

si ha:

$$m + 1 \geq n \quad (3.7)$$

Dalla (3.5) e dalla (3.7) deriva:

$$m + 1 = n$$

e cioè la tesi.

Dunque, il sistema (3.4) è un sistema quadrato di dimensione  $n$ , che ha il rango uguale al numero di colonne (e quindi anche al numero di righe). Si può verificare che il determinante di questa matrice ha l'espressione seguente:

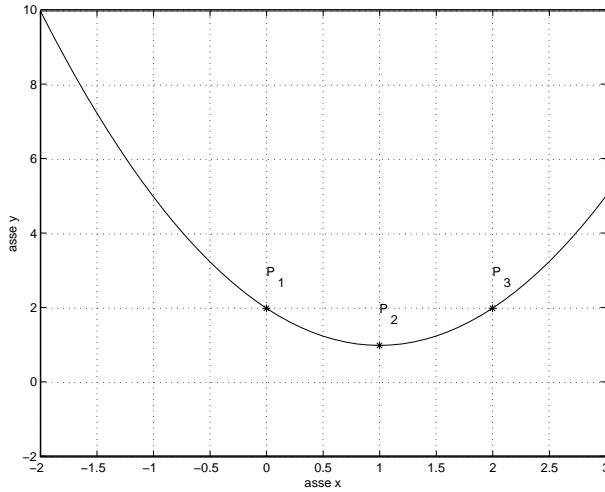
$$\prod_{1 \leq j < i \leq n} (x_i - x_j) = (x_n - x_{n-1}) \cdots (x_n - x_1)(x_{n-1} - x_{n-2}) \cdots (x_{n-1} - x_1) \cdots (x_3 - x_2)(x_3 - x_1)(x_2 - x_1)$$

il cui valore è sicuramente non nullo se i nodi  $x_1, x_2, \dots, x_n$  sono a due a due distinti tra loro. Pertanto il sistema (3.4) ammette un'unica soluzione.  $\square$

Si osservi che il Teorema 3.1 garantisce l'esistenza e l'unicità del polinomio interpolante  $p \in \Pi_m$ , con  $m = n - 1$ , cioè nello spazio dei polinomi di grado al più  $n - 1$ , se il numero di coefficienti da calcolare concide col numero di condizioni di interpolazione. In particolare è possibile che il polinomio abbia grado inferiore al numero di punti meno uno. Ad esempio se si considerano tre punti allineati, il polinomio interpolante non ha grado 2 bensì  $1 < 2 = n - 1$ . In generale questo caso si presenta quando c'è una *ridondanza* tra i dati assegnati.

### 3.2.2 Esistenza ed unicità del polinomio interpolante di Hermite

Come per l'interpolazione di Lagrange, anche nel caso di interpolazione di Hermite, al fine di costruire il polinomio interpolante bisogna innanzitutto verificare sotto quali condizioni questo esiste ed è unico.

Figura 3.14: Parabola interpolante i punti  $P_1, P_2$  e  $P_3$ 

♣ **Esempio 3.10.** Dati tre punti del piano  $P_1 \equiv (x_1, y_1)$ ,  $P_2 \equiv (x_2, y_2)$  e  $P_3 \equiv (x_3, y_3)$ , si vuole costruire la parabola  $y = y(x) = ax^2 + bx + c$ , passante per tali punti, cioè tale che:

$$\begin{cases} y(x_1) = ax_1^2 + bx_1 + c = y_1 \\ y(x_2) = ax_2^2 + bx_2 + c = y_2 \\ y(x_3) = ax_3^2 + bx_3 + c = y_3 \end{cases} \quad (3.8)$$

Assegnati tre punti del piano (crf. Teorema 3.1), esiste un'unica parabola (del tipo  $y = y(x)$ , ovvero con asse di simmetria parallelo all'asse delle ordinate) passante per tali punti. Se, ad esempio, i punti sono:

$$P_1 \equiv (0, 2), \quad P_2 \equiv (1, 1), \quad P_3 \equiv (2, 2),$$

il sistema diventa:

$$\begin{cases} a \cdot 0 + b \cdot 0 + c = 2 \\ a \cdot 1 + b \cdot 1 + c = 1 \\ a \cdot 4 + b \cdot 2 + c = 2 \end{cases}$$

ed ammette la soluzione:

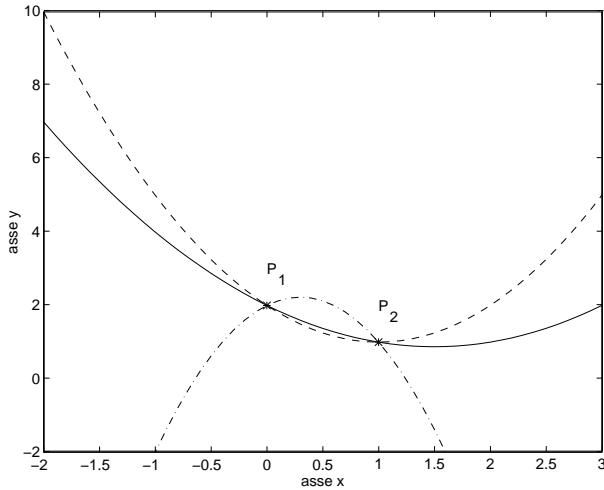
$$a = 1, \quad b = -2, \quad c = 2.$$

Quindi, l'unica parabola interpolante  $P_1, P_2$  e  $P_3$  è (Fig. 3.14):

$$y = y(x) = x^2 - 2x + 2.$$

Le condizioni (3.8) sono vincoli richiesti unicamente alla funzione  $y = y(x)$  e quindi il problema in esame è un problema di interpolazione di Lagrange. Supponendo invece che il punto  $P_3$  non sia assegnato, è possibile individuare comunque la stessa parabola  $y = x^2 - 2x + 2$ ? Da quanto detto è chiaro che l'appartenza dei due punti  $P_1$  e  $P_2$  alla parabola non è sufficiente, ma occorre aggiungere altre condizioni.

Osserviamo che conoscere la "pendenza",  $m$ , della curva in un punto significa conoscere la tangente alla parabola in quel punto avendo quest'ultima coefficiente angolare proprio  $m$ . In altre parole, abbiamo un'ulteriore informazione utile alla individuazione della parabola.

Figura 3.15: Parabole interpolanti i punti  $P_1$  e  $P_2$ 

Se poi consideriamo come punto sulla parabola, il vertice, la pendenza nel vertice è zero, cioè la tangente alla parabola nel vertice ha coefficiente angolare nullo, cioè è parallela all'asse delle ascisse.

Come si osserva dalla Fig. 3.15, delle tre parabole per  $P_1$  e  $P_2$  solo quella indicata dalla linea tratteggiata (- - -) ha il vertice in  $P_2$ . Se siamo interessati a costruire la parabola che oltre a passare per i due punti,  $P_1 = (0, 2)$  e  $P_2 = (1, 1)$ , ha il vertice in  $P_2$ , imponiamo anche questa condizione.

Come abbiamo notato, richiedere che la parabola abbia come vertice il punto  $P_2 \equiv (x_2, y_2)$  equivale a imporre che la pendenza della curva sia zero, cioè che derivata prima della funzione  $y = y(x)$  in  $x_2$  assuma valore nullo; le condizioni da imporre sono quindi:

$$\begin{cases} y(x_i) = y_i, \quad i = 1, 2 \\ y'(x_2) = 0 \end{cases} \iff \begin{cases} ax_1^2 + bx_1 + c = y_1 \\ ax_2^2 + bx_2 + c = y_2 \\ 2ax_2 + b = 0 \end{cases} \iff \begin{cases} c = 2 \\ a + b + c = 1 \\ 2a + b = 0 \end{cases}$$

da cui deriva:

$$a = 1 \quad b = -2 \quad c = 2.$$

Le condizioni richieste coinvolgono non solo la funzione  $y = y(x)$ , ma anche la sua derivata prima. La parabola determinata nel primo e nel secondo caso in questo esempio è la stessa; ciò che è cambiato è il tipo di condizioni richieste per costruirla univocamente, e cioè l'appartenenza alla parabola dei tre punti (nel caso di interpolazione lagrangiana) o l'appartenenza alla parabola dei due punti unita alla condizione che uno dei due sia il vertice (nel caso di interpolazione di Hermite). ♣

In generale, se non imponiamo alcun vincolo sul grado del polinomio, possiamo costruire infiniti polinomi che soddisfano alle condizioni di interpolazione di Hermite. Sia nel problema di interpolazione polinomiale di Lagrange sia in quello di Hermite ciò che assicura l'unicità del polinomio interpolante è la richiesta che il **numero di condizioni di interpolazione coincida con il numero di incognite da calcolare**

cioè con il numero di coefficienti del polinomio interpolante. Più precisamente, sussiste il seguente:

**Teorema 3.2. [Unicità del polinomio interpolante di Hermite]**

Assegnati  $n$  nodi distinti  $(x_i)_{i=1,\dots,n}$ ,  $n$  interi positivi  $l_1, l_2, \dots, l_n$ , tali che:

$$\sum_{i=1}^n l_i = m$$

ed  $m$  valori  $(y_i^j)$  con  $i = 1, \dots, n$ ;  $j = 0, \dots, l_i - 1$ , il polinomio  $p \in \Pi_q$  tale che:

$$p^{(j)}(x_i) = y_i^j \quad i = 1, \dots, n; j = 0, \dots, l_i - 1$$

è unico se

$$q \leq m - 1$$

**Dimostrazione** Supponiamo che esistano due polinomi  $q$  e  $r$ , con  $q \in \Pi_{m-1}$  e  $r \in \Pi_{m-1}$  tali che:

$$q^{(j)}(x_i) = y_i^{(j)}, \quad r^{(j)}(x_i) = y_i^{(j)} \quad i = 1, \dots, n \quad j = 0, \dots, l_i - 1$$

Dimostriamo che essi sono identicamente uguali. Osserviamo che il polinomio differenza  $p = q - r$ , di grado al più  $m - 1$ , è soluzione del problema omogeneo:

$$p^{(j)}(x_i) = 0 \quad i = 1, \dots, n, \quad j = 0, \dots, l_i - 1 \tag{3.9}$$

Dimostriamo quindi che il problema omogeneo (3.9) ammette solo la soluzione identicamente nulla. Osserviamo che la (3.9) equivale a richiedere che il polinomio  $p(x)$  ammette  $n$  zeri distinti:  $x_1, \dots, x_n$  ciascuno con molteplicità  $l_i$ . Poiché:

$$\sum_{i=1}^n l_i = m$$

il polinomio  $p$  ammette  $m$  zeri. Per il *Teorema fondamentale dell'algebra* un polinomio non nullo di grado al più  $m - 1$  può ammettere al più  $m - 1$  zeri, pertanto  $p \in \Pi_{m-1}$  deve essere necessariamente il polinomio identicamente nullo. ■

### 3.2.3 Algoritmi per la costruzione e valutazione del polinomio interpolante di Lagrange

Risolto il problema dell'esistenza e dell'unicità del polinomio interpolante di Lagrange, occorre affrontare quello di una sua efficiente costruzione e valutazione.

Come abbiamo visto negli esempi precedenti, in generale l'obiettivo per cui si risolve un problema di interpolazione è essenzialmente quello di:

1. **costruire** il polinomio interpolante, ovvero determinare i suoi coefficienti in una fissata base;

e/o

2. tracciare il grafico del polinomio interpolante, ossia **valutarlo** in più punti.

Nel seguito prenderemo in esame i seguenti metodi:

1. il metodo dei coefficienti indeterminati,
2. la Formula di Lagrange,
3. la Formula di Newton.

### 3.2.4 Metodo dei coefficienti indeterminati

La dimostrazione del teorema di unicità del polinomio interpolante di Lagrange fornisce un metodo costruttivo per la sua determinazione; come abbiamo potuto osservare infatti, le condizioni di interpolazione equivalgono alla risoluzione di un sistema di equazioni lineari.

♣ **Esempio 3.11.** Si vuole costruire il polinomio  $p \in \Pi_3$  interpolante i punti:

$$(x_1, y_1) \equiv (1, 2), \quad (x_2, y_2) \equiv (1.5, 5), \quad (x_3, y_3) \equiv (2, 4), \quad (x_4, y_4) \equiv (2.5, -2),$$

cioè tale che:

$$\begin{cases} p(x_1) = y_1 \\ p(x_2) = y_2 \\ p(x_3) = y_3 \\ p(x_4) = y_4 \end{cases} \quad (3.10)$$

Il polinomio  $p$  è del tipo:

$$p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3.$$

Per costruire  $p$  occorre pertanto determinare i suoi coefficienti  $a_0, a_1, a_2, a_3$ . Dalle condizioni di interpolazione (3.10) segue che:

$$\begin{cases} a_0 + a_1 + a_2 + a_3 = 2 \\ a_0 + a_1 \cdot 1.5 + a_2 \cdot 2.25 + a_3 \cdot 3.375 = 5 \\ a_0 + a_1 \cdot 2 + a_2 \cdot 4 + a_3 \cdot 8 = 4 \\ a_0 + a_1 \cdot 2.5 + a_2 \cdot 6.25 + a_3 \cdot 15.625 = -2 \end{cases} \quad (3.11)$$

e, quindi, per determinare i coefficienti di  $p$  si può risolvere il sistema di equazioni lineari (3.11), che in forma matriciale si esprime come:

$$Va = y,$$

dove:

$$V = \begin{pmatrix} 1. & 1. & 1. & 1. \\ 1. & 1.5 & 2.25 & 3.375 \\ 1. & 2. & 4. & 8. \\ 1. & 2.5 & 6.25 & 15.625 \end{pmatrix}, \quad a = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}, \quad y = \begin{pmatrix} 2 \\ 5 \\ 4 \\ -2 \end{pmatrix}.$$



Più precisamente, un modo per costruire il polinomio  $p \in \Pi_{n-1}$ :

$$p(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1},$$

che interpoli gli  $n$  punti assegnati  $(x_i, y_i)_{i=1,\dots,n}$ , cioè tale che:

$$p(x_i) = y_i \quad i = 1, \dots, n \tag{3.12}$$

è risolvere il sistema di equazioni lineari che si ottiene dalle condizioni (3.12), e cioè il sistema:

$$\left\{ \begin{array}{l} a_0x_1^0 + a_1x_1^1 + \dots + a_{n-2}x_1^{n-2} + a_{n-1}x_1^{n-1} = y_1 \\ a_0x_2^0 + a_1x_2^1 + \dots + a_{n-2}x_2^{n-2} + a_{n-1}x_2^{n-1} = y_2 \\ \vdots \qquad \dots \qquad \dots \qquad \dots \qquad \dots \qquad \vdots \\ a_0x_n^0 + a_1x_n^1 + \dots + a_{n-2}x_n^{n-2} + a_{n-1}x_n^{n-1} = y_n \end{array} \right. \tag{3.13}$$

in cui le incognite sono i coefficienti del polinomio interpolante. Una volta calcolati i coefficienti, il polinomio può essere valutato per qualsiasi valore di  $x$  utilizzando **l'algoritmo di Horner**.

Questo metodo è noto come **metodo dei coefficienti indeterminati**<sup>21</sup>.

---

<sup>21</sup>Questo metodo non è praticabile da un punto di vista computazionale perché conduce ad un problema mal condizionato

```

procedure Horner(in: n, a,  $\tilde{x}$  ; out: p)

/# SCOPO: valutazione di un polinomio.

/# SPECIFICHE PARAMETRI:
/# PARAMETRI DI INPUT:

var: n : intero { numero dei punti }
var:  $\tilde{x}$  : reale { punto in cui si richiede }
{ la valutazione del polinomio }
var: a(n) : array di reali { coefficienti del polinomio }
{ interpolante}

/# PARAMETRI DI OUTPUT:

var: p : reale { valore del polinomio in  $\tilde{x}$  }

/# VARIABILI LOCALI:
var: i : intero

/# INIZIO ISTRUZIONI:
p := a(n);
for i = n - 1, 1 step -1 do
    p := p ·  $\tilde{x}$  + a(i);
endfor

end Horner

```

Procedura 3.1: Algoritmo di Horner per la valutazione di un polinomio

### 3.2.5 Formula di Lagrange

♣ **Esempio 3.12.** Consideriamo il seguente problema di interpolazione. Siano  $(x_1, y_1)$ ,  $(x_2, y_2)$  le coordinate di due punti del piano.

Costruire la retta  $y = p(x)$  interpolante i due punti  $(x_1, y_1)$ ,  $(x_2, y_2)$  cioè tale che:

$$\begin{cases} p(x_1) = y_1 \\ p(x_2) = y_2 \end{cases}$$

Poniamo:

$$p(x) = y_1 \cdot l_1(x) + y_2 \cdot l_2(x)$$

con  $l_1$  e  $l_2$  polinomi di primo grado da determinare.

Per calcolare  $l_1$  e  $l_2$ , osserviamo che:

$$\begin{cases} p(x_1) = y_1 \longrightarrow y_1 l_1(x_1) + y_2 l_2(x_1) = y_1 \\ p(x_2) = y_2 \longrightarrow y_1 l_1(x_2) + y_2 l_2(x_2) = y_2 \end{cases}$$

Queste condizioni sono sicuramente verificate se richiediamo che i due polinomi  $l_1$  e  $l_2$  sono tali che:

$$\begin{cases} l_1(x_1) = 1, \quad l_1(x_2) = 0 \\ l_2(x_1) = 0, \quad l_2(x_2) = 1 \end{cases}$$

Dovendo essere  $l_1$  e  $l_2$  polinomi di primo grado, si ha:

$$\begin{cases} l_1(x_2) = 0 \longrightarrow l_1(x) = a(x - x_2) \\ l_2(x_1) = 0 \longrightarrow l_2(x) = b(x - x_1) \end{cases}$$

e:

$$\begin{cases} l_1(x_1) = 1 = a(x_1 - x_2) \longrightarrow a = \frac{1}{(x_1 - x_2)} \\ l_2(x_2) = 1 = b(x_2 - x_1) \longrightarrow b = \frac{1}{(x_2 - x_1)} \end{cases}$$

Cioè, l'equazione della retta per i due punti assegnati è del tipo:

$$y = p(x) = y_1 \cdot l_1(x) + y_2 \cdot l_2(x) = y_1 \cdot \frac{(x - x_2)}{(x_1 - x_2)} + y_2 \cdot \frac{(x - x_1)}{(x_2 - x_1)} \quad (3.14)$$

La (3.14) è detta **Formula di Lagrange** per il polinomio  $p(x)$  interpolante i punti  $(x_1, y_1)$  e  $(x_2, y_2)$ . I polinomi:

$$l_1(x) = \frac{(x - x_2)}{(x_1 - x_2)}, \quad l_2(x) = \frac{(x - x_1)}{(x_2 - x_1)}$$

sono detti **polinomi fondamentali di Lagrange**.



L'idea di base della Formula di Lagrange è esprimere il polinomio interpolante di Lagrange  $p \in \Pi_{n-1}$  nella forma:

$$p(x) = y_1 \cdot l_1(x) + \cdots + y_n \cdot l_n(x) \quad (3.15)$$

con ciascun  $l_i$  polinomio di grado  $n-1$ <sup>22</sup>. Il problema consiste nel determinare i polinomi  $l_i$ .

Osserviamo che, dovendo essere:

$$p(x_i) = y_i \quad i = 1, n$$

segue che i polinomi  $l_i$  devono essere tali che:

$$l_i(x_k) = \begin{cases} 1 & i = k, \\ 0 & i \neq k \end{cases} \quad . \quad (3.16)$$

Dalla seconda delle (3.16) segue che ciascun polinomio  $l_i$  ammette  $n-1$  zeri, e cioè i punti  $x_k$ , con  $k = 1, n$  e  $k \neq i$ . Pertanto ciascun polinomio  $l_i$  è:

$$l_i(x) = a_i \cdot (x - x_1) \cdot (x - x_2) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)$$

Il coefficiente  $a_i$  viene determinato utilizzando la prima delle (3.16) e cioè:

$$l_i(x_i) = 1$$

da cui:

$$a_i = \frac{1}{(x_i - x_1) \cdot (x_i - x_2) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)}$$

In conclusione, i **polinomi fondamentali di Lagrange**  $l_i$  di grado  $n-1$  sono:

$$l_i(x) = \prod_{j=1, j \neq i}^n \frac{(x - x_j)}{(x_i - x_j)} \quad i = 1, \dots, n \quad (3.17)$$

e dunque la **Formula di Lagrange** per  $p \in \Pi_{n-1}$  diventa:

$$p(x) = \sum_{i=1}^n y_i \prod_{j=1, j \neq i}^n \frac{(x - x_j)}{(x_i - x_j)}$$

---

<sup>22</sup>Più in generale, si può richiedere che la funzione interpolante,  $f \in \mathcal{X}$ , sia esprimibile nella forma:

$$f(x) = \sum_{i=1}^n a_i b_i(x)$$

con  $\{b_i\}$  funzioni opportunamente fissate nello spazio  $\mathcal{X}$ . Il problema consiste quindi nel calcolare i coefficienti  $a_i$  di tale rappresentazione. Ricordiamo che una formulazione di questo tipo per la funzione interpolante è anche detta **modello lineare**.

La complessità di tempo della formula di Lagrange si calcola contando, dapprima, il numero di operazioni richieste dalla costruzione di ciascuno degli  $n$  polinomi fondamentali di Lagrange, ovvero

$$l_i(x) = \underbrace{\prod_{j=1, j \neq i}^n \frac{(x - x_j)}{(x_i - x_j)}}_{\substack{(n-1) \text{ fattori} \\ (n-2)M}} \Rightarrow (n-1)(2A + 1M) + (n-2)M = 4n - 5 \text{ flop}$$

per  $i = 1, \dots, n$ . Si aggiunge, poi, per ciascun polinomio, l'operazione floating point relativa al prodotto di  $l_i(x)$  con  $y_i$  e si somma il tutto  $n$  volte,

$$\begin{aligned} p(x) &= \underbrace{\sum_{i=1}^n y_i \cdot l_i(x)}_{\substack{n \text{ addendi} \\ (n-1)A}} \Rightarrow \\ &\Rightarrow n[2(n-1)A + (2n-2)M] + (n-1)A = n(4n-3) - 1 \text{ flop} \end{aligned}$$

Concludendo, la complessità asintotica di tempo richiesta dall'algoritmo per la costruzione del polinomio interpolante espresso nella formula di Lagrange, è

$$T(n) = \mathcal{O}(n^2) \text{ flop}$$

Per la valutazione del polinomio interpolante, costruito con l'algoritmo descritto, occorre valutare, per un fissato valore di  $x$ , ciascun polinomio fondamentale di Lagrange e quindi al variare di  $x$  occorre effettuare di nuovo tutte le valutazioni<sup>23</sup>, con una complessità di tempo asintotica di

$$T(n) = \mathcal{O}(n^2) \text{ flop.}$$

### 3.2.6 Formula di Newton

♣ **Esempio 3.13.** Sia  $p$  il polinomio di grado zero interpolante il punto  $P_1 = (x_1, y_1) \equiv (1, 2)$ ; dunque:

$$y = p(x) \equiv a_0 = y_1 = 2 \quad (3.18)$$

---

<sup>23</sup>Come vedremo, esistono algoritmi più efficienti per la valutazione in un punto della formula di Lagrange. Tali algoritmi si basano su un opportuno cambio di base nella rappresentazione del polinomio interpolante.

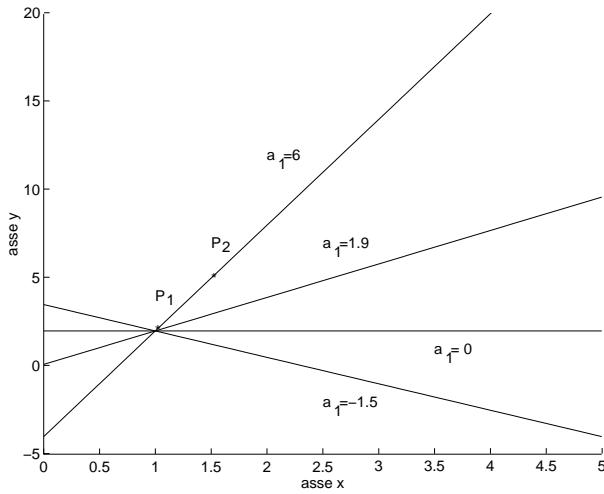


Figura 3.16: Fascio di rette per il punto  $P_1$  ottenuto al variare del coefficiente  $a_1 = -1.5, 0, 1.9, 6$

è l'equazione della retta parallela all'asse delle ascisse passante per il punto  $P_1$ . A partire dall'equazione di questa retta, si vuole determinare quella della retta passante oltre che per  $P_1$  anche per un altro punto  $P_2 \equiv (1.5, 5)$ .

Considerata la generica retta per il punto  $P_1$ , di equazione<sup>24</sup>:

$$y(x) = 2 + a_1(x - 1), \quad (3.20)$$

calcoliamo il coefficiente  $a_1$  imponendo l'appartenenza del punto  $P_2$ . Come si osserva dal grafico in Fig. 3.16, al variare del coefficiente  $a_1$  si ottiene una (ed una sola) delle infinite rette passanti per il punto  $P_1$ , di equazione (3.20). Imponendo l'appartenenza al *fascio di rette* anche del punto  $P_2$ , si determina univocamente quella passante per entrambi i punti  $P_1$  e  $P_2$ , e, quindi, il valore del corrispondente coefficiente  $a_1$ .

Dunque:

$$5 = y(1.5) = 2 + a_1(1.5 - 1) \implies a_1 = \frac{5 - 2}{1.5 - 1} = \frac{3}{0.5} = 6. \quad (3.21)$$

La retta interpolante i punti  $P_1$  e  $P_2$  ha, quindi, equazione:

$$y = q(x) = p(x) + a_1(x - 1) = 2 + 6(x - 1). \quad (3.22)$$

---

<sup>24</sup>Considerata l'equazione della generica retta del piano:

$$y = ax + b, \quad (3.19)$$

imponendo che questa passi per il punto  $(x_1, y_1)$  si ha:

$$y_1 = y(x_1) = ax_1 + b \implies b = y_1 - ax_1,$$

e, quindi, sostituendo nella (3.19):

$$y(x) = ax + y_1 - ax_1 = y_1 + a(x - x_1)$$

e cioè si ha l'equazione del **fascio di rette** di centro il punto  $(x_1, y_1)$ .

Analogamente, considerando un ulteriore punto  $P_3 \equiv (2, 4)$ , per determinare l'equazione della parabola passante per i tre punti  $P_1, P_2$  e  $P_3$  basta considerare l'equazione della generica parabola<sup>25</sup> per i punti  $P_1$  e  $P_2$ :

$$y = q(x) + a_2(x - 1)(x - 1.5)$$

e calcolare il coefficiente  $a_2$  imponendo l'appartenenza a tale parabola del punto  $P_3$ . Quindi:

$$4 = y(2) = q(2) + a_2(2 - 1)(2 - 1.5) = 2 + 6 \cdot 1 + a_2 \cdot (2 - 1)(2 - 1.5)$$

da cui:

$$a_2 = \frac{4 - 2 - 6}{0.5} = -8. \quad (3.25)$$

La parabola interpolante i punti  $P_1, P_2$  e  $P_3$ , allora, ha equazione:

$$y = r(x) = q(x) + a_2(x - 1)(x - 1.5) = 2 + 6(x - 1) - 8(x - 1)(x - 1.5). \quad (3.26)$$

Aggiungendo il punto  $P_4 \equiv (2.5, -2)$ , per costruire il polinomio di terzo grado interpolante i punti  $P_1, P_2, P_3$  e  $P_4$  si consideri l'equazione della curva di terzo grado (cubica) per i punti  $P_1, P_2$  e  $P_3$ , che, analogamente a quanto dimostrato per la retta e la parabola, è del tipo:

$$y = s(x) = r(x) + a_3(x - 1)(x - 1.5)(x - 2). \quad (3.27)$$

Imponendo che  $s$  interpoli anche il punto  $P_4$ , si ha:

$$\begin{aligned} -2 &= s(2.5) = 2 + 6(2.5 - 1) - 8(2.5 - 1)(2.5 - 1.5) + a_3(2.5 - 1)(2.5 - 1.5)(2.5 - 2) = \\ &= 2 + 6 \cdot 1.5 - 8 \cdot 1.5 \cdot 1 + a_3 \cdot 1.5 \cdot 1 \cdot 0.5 = -1 + 0.75 \cdot a_3 \end{aligned}$$

da cui si ricava il valore del coefficiente  $a_3$ :

$$a_3 = \frac{-1}{0.75} = -1.33\hat{3}.$$

---

<sup>25</sup>Considerata l'equazione della parabola nel piano, del tipo:

$$y = y(x) = ax^2 + bx + c, \quad (3.23)$$

imponendo che a questa appartengano i punti di coordinate  $(x_1, y_1), (x_2, y_2)$ , si ha:

$$\begin{cases} y_1 = ax_1^2 + bx_1 + c \\ y_2 = ax_2^2 + bx_2 + c \end{cases} \quad (3.24)$$

Ricavando dalle (3.24) i coefficienti  $b$  e  $c$  in funzione del coefficiente  $a$ , si ha:

$$\begin{cases} b = \frac{y_2 - y_1}{x_2 - x_1} - a(x_2 + x_1) \\ c = \frac{y_1(x_2 - x_1) - x_1(y_2 - y_1)}{x_2 - x_1} + ax_1x_2 \end{cases}$$

Sostituendo le espressioni ottenute di  $b$  e  $c$  nella (3.23), segue che:

$$y(x) = a(x - x_1)(x - x_2) + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1,$$

cioè:

$$y(x) = a(x - x_1)(x - x_2) + q(x),$$

con  $q(x)$  retta interpolante  $(x_1, y_1)$  e  $(x_2, y_2)$ .

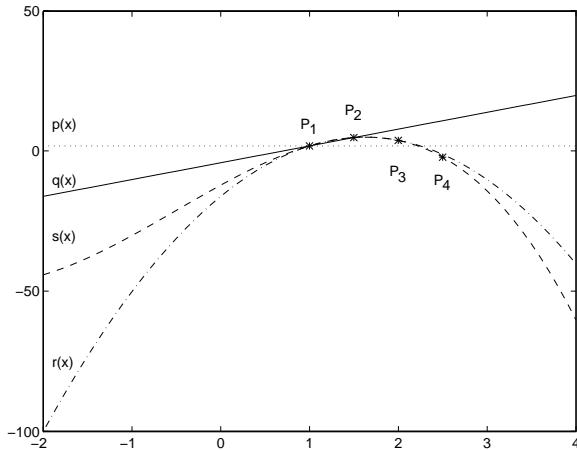


Figura 3.17: Polinomi interpolanti di grado zero, uno, due e tre relativi rispettivamente ai punti  $P_1$ ,  $P_1$  e  $P_2$ ,  $P_1$ ,  $P_2$  e  $P_3$ ,  $P_1$ ,  $P_2$ ,  $P_3$  e  $P_4$

Il polinomio di terzo grado interpolante i punti  $P_1, P_2, P_3$  e  $P_4$  ha, dunque, equazione:

$$\begin{aligned} y = s(x) &= r(x) + a_3(x-1)(x-1.5)(x-2) = \\ &= 2 + 6(x-1) - 8(x-1)(x-1.5) + 1.33\hat{3} \cdot (x-1)(x-1.5)(x-2). \end{aligned} \quad (3.28)$$

La (3.28) prende il nome di **Formula di Newton** per il polinomio interpolante. Tenendo conto delle (3.18), (3.22), (3.26), (3.27) notiamo che il polinomio di terzo grado  $s$  interpolante i quattro punti  $P_1, P_2, P_3, P_4$ , è stato ottenuto a partire dal polinomio di secondo grado  $r$  interpolante i tre punti  $P_1, P_2$  e  $P_3$  che a sua volta è stato ottenuto a partire dal polinomio di primo grado  $q$  interpolante i due punti  $P_1$  e  $P_2$  costruito a partire dal polinomio di grado zero  $p$  interpolante il punto  $P_1$ .

Una rappresentazione grafica dei polinomi così costruiti è mostrata in Figura 3.17.

♣

In generale, dati i punti di coordinate  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ , l'idea della formula di Newton è quella di **costruire il polinomio interpolante gli  $n$  punti a partire dal polinomio interpolante i primi  $n-1$  punti**. Naturalmente, tale procedimento va applicato in modo ricorrente, perché il polinomio interpolante i primi  $n-1$  punti viene a sua volta costruito a partire dal polinomio interpolante i primi  $n-2$  punti, e così proseguendo.

Il procedimento costruttivo della formula di Newton si può schematizzare come segue:

- si costruisce il polinomio  $p$  di grado zero interpolante il punto  $(x_1, y_1)$ :  
 $y = p(x) \equiv y_1 = a_0$ .
- si costruisce il polinomio  $q \in \Pi_1$  interpolante i punti  $(x_1, y_1)$  e  $(x_2, y_2)$ :  $q(x)$  è del

tipo:

$$y = q(x) = p(x) + a_1(x - x_1) = y_1 + a_1(x - x_1)$$

Il coefficiente  $a_1$  viene determinato imponendo la condizione di interpolazione nel punto  $(x_2, y_2)$ :

$$y_2 = q(x_2) = y_1 + a_1(x_2 - x_1) \rightarrow a_1 = \frac{y_2 - y_1}{x_2 - x_1}$$

- si costruisce il polinomio  $r \in \Pi_2$  interpolante i punti  $(x_1, y_1)$ ,  $(x_2, y_2)$  e  $(x_3, y_3)$ : tale polinomio è del tipo:

$$y = r(x) = p(x) + a_1(x - x_1) + a_2(x - x_1)(x - x_2)$$

Il coefficiente  $a_2$  viene determinato imponendo la condizione di interpolazione nel punto  $(x_3, y_3)$ :

$$y_3 = r(x_3) = y_1 + a_1(x_3 - x_1) + a_2(x_3 - x_1)(x_3 - x_2)$$

$$\rightarrow a_2 = \frac{y_3 - y_1 - a_1(x_3 - x_1)}{(x_3 - x_1)(x_3 - x_2)} = \frac{y_3 - y_1 - \frac{y_2 - y_1}{x_2 - x_1}(x_3 - x_2 + x_2 - x_1)}{(x_3 - x_2)(x_3 - x_1)} =$$

$$= \frac{y_3 - y_1 - [(y_2 - y_1) + \frac{y_2 - y_1}{x_2 - x_1}(x_3 - x_2)]}{(x_3 - x_1)(x_3 - x_2)} = \frac{\frac{y_3 - y_1}{x_3 - x_2} - \frac{y_2 - y_1}{x_3 - x_2} - a_1}{x_3 - x_1} =$$

$$= \frac{\frac{y_3 - y_1 - y_2 + y_1}{x_3 - x_2} - a_1}{x_3 - x_1} = \frac{\frac{y_3 - y_2}{x_3 - x_2} - a_1}{x_3 - x_1}$$

- dati  $k + 1$  nodi:  $x_1, \dots, x_{k+1}$ , si costruisce il polinomio interpolante  $p \in \Pi_k$  a partire da  $q \in \Pi_{k-1}$ , polinomio interpolante i primi  $k$  nodi  $x_1, \dots, x_k$  secondo una formula del tipo:

$$p(x) = q(x) + r(x)$$

con  $r \in \Pi_k$ . Il polinomio  $r$  si determina imponendo le condizioni di interpolazione:

$$p(x_i) = y_i \quad i = 1, \dots, k + 1$$

In particolare:

- Da  $p(x_i) = q(x_i) + r(x_i)$ ,  $i = 1, \dots, k$ , segue che  $r(x_i) = 0$ , per  $i = 1, \dots, k$  cioè:

$$r(x) = a_k(x - x_1)(x - x_2) \dots (x - x_k)$$

con  $a_k$  incognita.

- Il coefficiente  $a_k$  viene calcolato imponendo la condizione di interpolazione sul nodo  $k + 1$ -mo:

$$p(x_{k+1}) = q(x_{k+1}) + a_k(x_{k+1} - x_1)(x_{k+1} - x_2) \dots (x_{k+1} - x_k) = y_{k+1}$$

da cui si ricava  $a_k$ .

- Quindi:

$$p(x) = q(x) + a_k(x - x_1)(x - x_2) \dots (x - x_k)$$

- in generale, dati  $n$  nodi, il polinomio interpolante  $p \in \Pi_{n-1}$  è del tipo:

$$p(x) = q(x) + a_{n-1}(x - x_1)(x - x_2) \dots (x - x_{n-1})$$

dove  $q \in \Pi_{n-2}$  è il polinomio interpolante i primi  $n - 1$  nodi.

Si ottiene, quindi, l'espressione seguente:

$$p(x) = a_0 + a_1(x - x_1) + \dots + a_{n-1}(x - x_1)(x - x_2) \dots (x - x_{n-1}), \quad (3.29)$$

nota come **Formula di Newton** per il polinomio interpolante di Lagrange.

### 3.2.7 Formula di Newton e differenze divise

Per costruire i coefficienti del polinomio interpolante espresso nella formula di Newton si utilizza il concetto di **differenza divisa**.

♣ **Esempio 3.14.** Considerati i punti:

$$P_1(x_1 = 1, y_1 = 2), P_2(x_2 = 1.5, y_2 = 5), P_3(x_3 = 2, y_3 = 4)$$

dalla costruzione del polinomio interpolante espresso nella formula di Newton, osserviamo che il coefficiente  $a_0$  di tale polinomio coincide con l'ordinata del primo punto di interpolazione:

$$a_0 = 2 = y_1$$

Per il coefficiente  $a_1$  si ha che:

$$a_1 = \frac{y_2 - y_1}{x_2 - x_1} = \frac{5 - 2}{1.5 - 1} = 6$$

e,

$$a_2 = \frac{\frac{y_3 - y_2}{x_3 - x_2} - a_1}{x_3 - x_1} = \frac{\frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_2 - x_1}}{x_3 - x_1} = \frac{\frac{4 - 5}{2 - 1.5} - 6}{2 - 1} = -2 - 6 = -8$$

Posto:

$$y[x_1] = y_1, \quad y[x_2] = y_2, \quad y[x_3] = y_3,$$

(differenze divise di ordine zero), si ha quindi

$$\begin{aligned} a_0 &= y_1 = y[x_1] \\ a_1 &= \frac{y[x_2] - y[x_1]}{x_2 - x_1} \end{aligned} \quad (3.30)$$

Posto:

$$y_{12} = y[x_1, x_2] = \frac{y[x_2] - y[x_1]}{x_2 - x_1}$$

e

$$y_{23} = y[x_2, x_3] = \frac{y[x_3] - y[x_2]}{x_3 - x_2}$$

(differenze divise di ordine uno), si ha<sup>26</sup>

$$\begin{aligned} a_0 &= y_1 = y[x_1] \\ a_1 &= y_{12} = y[x_1, x_2] \\ a_2 &= \frac{y[x_2, x_3] - y[x_1, x_2]}{x_3 - x_1} \end{aligned} \quad (3.31)$$

In presenza di 4 nodi  $x_1, x_2, x_3, x_4$  si considerano anche:

$$\begin{aligned} y[x_4] &= y_4 \\ y_{34} &= y[x_3, x_4] = \frac{y[x_4] - y[x_3]}{x_4 - x_3} \end{aligned}$$

Se si pone:

$$y_{123} = y[x_1, x_2, x_3] = \frac{y[x_2, x_3] - y[x_1, x_2]}{x_3 - x_1}$$

e

$$y_{234} = y[x_2, x_3, x_4] = \frac{y[x_3, x_4] - y[x_2, x_3]}{x_4 - x_2},$$

(differenza divise di ordine due), si ha:

$$\begin{aligned} a_0 &= y_1 = y[x_1] \\ a_1 &= y_{12} = y[x_1, x_2] \\ a_2 &= y_{123} = y[x_1, x_2, x_3] \\ a_3 &= \frac{y[x_2, x_3, x_4] - y[x_1, x_2, x_3]}{x_4 - x_1} \end{aligned} \quad (3.32)$$

Più in generale, con  $n$  nodi si ha:

---

<sup>26</sup>Nel seguito applicheremo due proprietà delle differenze divise che riguardano l'invarianza del risultato rispetto a qualsiasi permutazione dei nodi. Più precisamente, useremo il fatto che

$$y[x_i, x_j] = y[x_j, x_i]$$

e che, più in generale con tre nodi  $y[x_i, x_j, x_k]$  non cambia al variare della disposizione dei nodi. Questa proprietà si può dimostrare vera per le differenze divise di qualsiasi ordine.

$$\begin{aligned}
 a_0 &= y[x_1], \\
 a_1 &= y[x_1, x_2], \\
 a_2 &= y[x_1, x_2, x_3] \\
 a_3 &= y[x_1, x_2, x_3, x_4] \\
 &\vdots \\
 a_{n-1} &= y[x_1, x_2, \dots, x_n]
 \end{aligned}$$



Considerato il polinomio interpolante di Lagrange, espresso nella formula di Newton (3.29), e indicato con

$$a_k = y[x_1, x_2, \dots, x_{k+1}]$$

il k-mo coefficiente  $a_k$ , sottolineando la dipendenza esclusiva dai punti di interpolazione, vediamo come tali quantità possono essere determinate. A tal fine, riscriviamo il polinomio interpolante nella forma:

$$\begin{aligned}
 p(x) &= y[x_1] + y[x_1, x_2](x - x_1) + \dots + \\
 &+ y[x_1, \dots, x_n](x - x_1)(x - x_2) \dots (x - x_{n-1})
 \end{aligned}$$

e, utilizzando le condizioni di interpolazione introduciamo le **differenze divise**.

Poiché:

$$p(x_1) = y_1 \longrightarrow y[x_1] = y_1$$

si definiscono le **differenze divise di ordine zero** nel modo seguente:

$$y[x_i] = y_i, \quad \forall i = 1, n$$

Da:

$$p(x_2) = y_2 \longrightarrow y[x_1] + y[x_1, x_2](x_2 - x_1) = y_2$$

si ricava:

$$y[x_1, x_2] = \frac{y[x_2] - y[x_1]}{x_2 - x_1}$$

da cui si definiscono le **differenze divise di ordine uno**:

$$y[x_i, x_{i+1}] = \frac{y[x_{i+1}] - y[x_i]}{x_{i+1} - x_i}$$

In generale, la **differenza divisa di ordine n**, si definisce per ricorrenza:

**Definizione 3.7. (Differenza divisa di ordine  $n$ )**

$$\begin{aligned} y[x_i] &= y_i \\ y[x_1, \dots, x_{n+1}] &= \frac{y[x_2, \dots, x_{n+1}] - y[x_1, \dots, x_n]}{x_{n+1} - x_1} \end{aligned} \quad (3.33)$$

Dimostriamo ora che tutti i coefficienti del polinomio interpolante di Lagrange nella (3.29) sono esprimibili in termini di differenze divise relative ai nodi di interpolazione <sup>27</sup>

**Teorema 3.4.** *Sia  $p \in \Pi_{n-1}$  il polinomio interpolante di Lagrange relativo ai punti  $(x_i, y_i)_{i=1, \dots, n}$ . Se  $a_k$  è il  $k$ -mo coefficiente di  $p$  espresso nella formula di Newton, allora  $a_k$  è la differenza divisa di ordine  $k$  relativa ai nodi  $x_1, x_2, \dots, x_{k+1}$ , cioè:*

$$a_k = y[x_1, x_2, \dots, x_{k+1}]$$

---

<sup>27</sup>In generale sussiste la seguente:

**Definizione 3.8. (Operatore alle differenze in avanti)**

Data la successione  $\{y_i\}_{i=1,2,\dots}$ , l'operatore:

$$\Delta y_i = y_{i+1} - y_i$$

è detto **operatore alle differenze in avanti** e, più in generale l'operatore:

$$\Delta^n y_i = \Delta(\Delta^{n-1} y_i) \quad n \geq 1 \quad (\Delta^0 \equiv 1)$$

è detto **operatore alle differenze in avanti di ordine  $n$** .

Si dimostra che:

**Teorema 3.3.** *Nell'ipotesi che i nodi  $(x_i)_{i=1, \dots, n}$  siano equidistanziati, posto*

$$h = x_i - x_{i-1}, \quad \forall x_i,$$

*la differenza divisa  $y[x_1, \dots, x_n]$  relativa agli  $n$  valori  $x_1, \dots, x_n$ , si esprime come:*

$$y[x_1, \dots, x_n] = \frac{1}{(n-1)!h^{n-1}} \Delta^{n-1} y_1$$

**Dimostrazione** La dimostrazione procede per induzione sul numero di nodi  $n$ . Per  $n = 1$ , discende dalla definizione di differenza divisa di ordine zero. Assumiamo che sia vera per  $k$ , dimostriamo che è vera per  $k + 1$ .

Dalla definizione di differenza divisa di ordine  $k + 1$ , segue:

$$y[x_1, \dots, x_{k+1}] = \frac{y[x_2, \dots, x_{k+1}] - y[x_1, \dots, x_k]}{x_{k+1} - x_1}$$

Applicando l'ipotesi di induzione alle quantità al secondo membro si ha:

$$\begin{aligned} y[x_1, \dots, x_{k+1}] &= \frac{1}{k \cdot h} \left[ \frac{1}{(k-1)!h^{k-1}} \Delta^{k-1} y_2 - \frac{1}{(k-1)!h^{k-1}} \Delta^{k-1} y_1 \right] = \\ &= \frac{1}{k!h^k} \cdot \Delta^{k-1} (y_2 - y_1) = \frac{1}{k!h^k} \cdot \Delta^{k-1} (\Delta y_1) = \frac{1}{k!h^k} \Delta^k y_1 \end{aligned} \quad (3.34)$$



cioè:

$$\begin{aligned} p(x) &= y[x_1] + y[x_1, x_2](x - x_1) + \dots + \\ &+ y[x_1, x_2, \dots, x_n](x - x_1)(x - x_2) \dots (x - x_{n-1}) \end{aligned}$$

**Dimostrazione** Si procede per induzione su  $k$ .

Per  $k = 0$ , la tesi è banalmente vera in quanto

$$a_0 = p(x) = y_1 = y[x_1]$$

Supponiamo la tesi vera per  $k = j - 1$  e dimostriamo che è vera per  $k = j$ .

Siano

- $p \in \Pi_{j-1}$  il polinomio interpolante i nodi  $x_1, x_2, \dots, x_j$ ;
- $q \in \Pi_{j-1}$  il polinomio interpolante i nodi  $x_2, \dots, x_{j+1}$ ;

Per ipotesi di induzione:

- $p(x) = r(x) + y[x_1, x_2, \dots, x_j](x - x_1)(x - x_2) \dots (x - x_{j-1})$
- $q(x) = s(x) + y[x_2, \dots, x_{j+1}](x - x_2)(x - x_3) \dots (x - x_j)$ ;

Consideriamo il polinomio

$$P(x) = \frac{(x - x_1)q(x) - (x - x_{j+1})p(x)}{(x_{j+1} - x_1)} \quad (3.35)$$

Poiché  $P(x_i) = y_i$ ,  $i = 1, 2, \dots, j + 1$ ,  $P(x)$  è il polinomio interpolante i nodi  $x_1, x_2, \dots, x_{j+1}$ , e quindi utilizzando la formula di Newton:

$$P(x) = p(x) + a_j(x - x_1)(x - x_2) \dots (x - x_j), \quad \forall x \quad (3.36)$$

Per l'unicità del polinomio interpolante le espressioni (3.35) e (3.36) devono necessariamente individuare lo stesso polinomio e quindi, per il principio di identità dei polinomi, coincidono tutti i coefficienti omologhi. In particolare, coincidono i coefficienti di grado massimo:

$$a_k = \frac{y[x_2, x_3, \dots, x_{j+1}] - y[x_1, x_2, \dots, x_j]}{x_{j+1} - x_1}$$

da cui:

$$a_k = y[x_1, x_2, \dots, x_{j+1}]$$

■

---

Analogamente, si definisce l'**operatore alle differenze all'indietro** nel modo seguente:

**Definizione 3.9. (Operatore alle differenze all'indietro)**

Data la successione  $\{y_i\}_{i=1,2,\dots}$ , l'operatore:

$$\nabla y_i = y_i - y_{i-1}$$

è detto **operatore alle differenze all'indietro** e, più in generale l'operatore:

$$\nabla^n y_i = \nabla(\nabla^{n-1} y_i) \quad n \geq 1 \quad (\nabla^0 \equiv 1)$$

è detto **operatore alle differenze all'indietro di ordine n**

### 3.2.8 Un algoritmo per la formula di Newton: aspetti implementativi

Per costruire la formula di Newton (3.29) occorre calcolare innanzitutto le differenze divise  $y[x_1], y[x_1, x_2], \dots, y[x_1, \dots, x_n]$ .

Consideriamo la tabella seguente, dove con  $k$  abbiamo indicato l'ordine della generica differenza divisa:

$x_i$	$y_i$	$k = 1$	$k = 2$	$k = 3$	$\dots$	$k = n - 1$
$x_1$	$y_1$					
$x_2$	$y_2$	$y[x_1, x_2]$		$y[x_1, x_2, x_3]$		
$x_3$	$y_3$		$y[x_2, x_3]$			
$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$y[x_1, x_2, \dots, x_n]$
$x_{n-2}$	$y_{n-2}$		$y[x_{n-2}, x_{n-1}]$			
$x_{n-1}$	$y_{n-1}$			$y[x_{n-2}, x_{n-1}, x_n]$		
$x_n$	$y_n$		$y[x_{n-1}, x_n]$			

Tavola 1: Le differenze divise

Osservando la tavola, si nota che le differenze divise sono disposte per colonna in modo che ogni elemento di ciascuna colonna può essere calcolato, utilizzando la definizione (3.33), a partire dai due elementi ad esso adiacenti nella colonna precedente; ad esempio:

- la differenza divisa del primo ordine  $y[x_1, x_2] = (y_2 - y_1)/(x_2 - x_1)$  può essere calcolata utilizzando  $y_1$  e  $y_2$  cioè a partire dalle differenze divise di ordine zero relative rispettivamente ai nodi  $x_1$  e  $x_2$ ;
- la differenza divisa del secondo ordine  $y[x_1, x_2, x_3]$ , viene calcolata a partire da  $y[x_1, x_2]$  e  $y[x_2, x_3]$  cioè a partire dalle differenze divise di ordine uno relative ai nodi  $x_1, x_2, x_3$ ;
- così proseguendo fino a  $y[x_1, \dots, x_n]$ .

I coefficienti del polinomio interpolante espresso nella formula di Newton si trovano lungo la diagonale superiore.

L'algoritmo per il calcolo delle differenze divise è dunque costituito da un ciclo di  $n - 1$  passi in ciascuno dei quali si costruiscono le differenze divise di un prefissato ordine e cioè dell'ordine uguale al passo corrente.

In altre parole, calcoliamo le differenze divise, *procedendo per colonne*, da sinistra verso destra, nel modo seguente:

- passo 1: calcolo della I colonna delle differenze divise di ordine 1,

- passo 2: calcolo della II colonna delle differenze divise di ordine 2,
- passo  $n - 1$ : calcolo dell'unico elemento della  $(n - 1)$ -ma colonna delle differenze divise di ordine  $n - 1$ .

Una prima versione dell'algoritmo è quella illustrata nel riquadro seguente.

```

procedure diffdivise(in: n, x, y ; out: y)

/# SCOPO: calcolo dei coefficienti del polinomio interpolante
di Lagrange, espresso nella formula di Newton.

/# SPECIFICHE PARAMETRI:
/# PARAMETRI DI INPUT:

var: n : intero { numero dei punti }
var: x(n) : array di reali { nodi di interpolazione }
var: y(n) : array di reali { ordinate corrispondenti }
{ ai nodi di interpolazione }

/# PARAMETRI DI OUTPUT:
var: y(n) : array di reali { coefficienti del polinomio }
{ interpolante }

/# VARIABILI LOCALI:
var: k : intero

/# INIZIO ISTRUZIONI:
{ ciclo sul numero di passi }
for k = 1, n - 1 do
{ calcolo della colonna k-ma }
endfor
end diffdivise

```

Procedura 3.2: Algoritmo per il calcolo dei coefficienti del polinomio interpolante espresso nella formula di Newton

Ad ogni passo viene anche calcolato uno dei coefficienti del polinomio interpolante, che quindi dobbiamo opportunamente memorizzare.

Volendo utilizzare il vettore  $(y_1, \dots, y_n)$ , che inizialmente contiene le ordinate corrispondenti ai nodi di interpolazione, si pone il problema di stabilire in quale ordine le differenze divise di ciascuna colonna debbano essere calcolate.

Infatti, osserviamo che al primo passo, ad esempio, cioè nel calcolo della prima colonna:

$$y[x_1, x_2], \dots, y[x_{n-1}, x_n]$$

dopo aver calcolato  $y[x_1, x_2]$ , che rappresenta il coefficiente  $a_1$  del polinomio, questo non può essere memorizzato al posto di  $y_1$  perché  $y_1$  è il coefficiente  $a_0$ , e quindi non può

essere perso, né tantomeno al posto di  $y_2$  perché questo valore serve anche per il calcolo dell'elemento successivo  $y[x_2, x_3]$ .

Una strategia migliore può essere quella di calcolare gli elementi di una prefissata colonna procedendo, in ciascuna colonna, dal basso verso l'alto e cioè:

- primo passo: calcolare nell'ordine  $y[x_{n-1}, x_n] \dots y[x_1, x_2]$ ;
- secondo passo: calcolare nell'ordine  $y[x_{n-2}, x_{n-1}, x_n] \dots y[x_1, x_2, x_3]$ ;
- così proseguendo fino a  $y[x_1, \dots, x_n]$ .

In questo modo si può memorizzare l'elemento di volta in volta calcolato al posto dell'elemento che tra i due che hanno contribuito al calcolo, si trova più in basso nella colonna precedente, in quanto quest'ultimo non interverrà più nel calcolo delle successive differenze divise. Ad esempio al primo passo il valore di  $y[x_{n-1}, x_n]$  ottenuto utilizzando i valori di  $y_{n-1}$  e di  $y_n$ , si memorizza al posto di  $y_n$ , quello di  $y[x_{n-2}, x_{n-1}]$  calcolato a partire da  $y_{n-2}$  e di  $y_{n-1}$ , si memorizza al posto di  $y_{n-1}$ , ed infine il valore di  $y[x_1, x_2]$  al posto di  $y_2$ ; analogo procedimento viene ripetuto per ciascun passo.

Riassumendo, abbiamo:

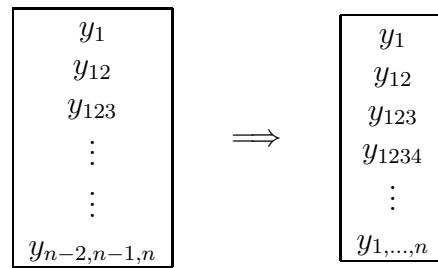
- **passo 1:** si calcolano le  $n - 1$  differenze divise  $y_{12}, y_{23}, \dots, y_{n-1,n}$  e si memorizzano al posto degli ultimi  $n - 1$  elementi dell'array  $(y_1, y_2, \dots, y_n)$ .

$$\begin{array}{|c|} \hline y_1 \\ \hline y_2 \\ \hline \vdots \\ \hline y_n \\ \hline \end{array} \implies \begin{array}{|c|} \hline y_1 \\ \hline y_{12} \\ \hline \vdots \\ \hline y_{n-1,n} \\ \hline \end{array}$$

- **passo 2:** si calcolano  $n - 2$  differenze divise  $y_{123}, y_{234}, \dots, y_{n-2,n-1,n}$  e si memorizzano negli ultimi  $n - 2$  elementi di  $(y_1, y_2, \dots, y_n)$ .

$$\begin{array}{|c|} \hline y_1 \\ \hline y_{12} \\ \hline \vdots \\ \hline y_{n-1,n} \\ \hline \end{array} \implies \begin{array}{|c|} \hline y_1 \\ \hline y_{12} \\ \hline y_{123} \\ \hline \vdots \\ \hline y_{n-2,n-1,n} \\ \hline \end{array}$$

- **passo n-1:** si calcola un solo elemento, e cioè  $y_{1,2,\dots,n}$  l'unica differenza divisa di ordine  $n - 1$ , e si memorizza al posto dell'ultima componente del vettore  $(y_1, \dots, y_n)$ .



Una versione in Pascal - like dell'algoritmo illustrato precedentemente è la seguente:

```

procedure diffdivise(in: n, x, y ; out: y)

/# SCOPO: calcolo dei coefficienti del polinomio interpolante
di Lagrange, espresso nella formula di Newton.

/# SPECIFICHE PARAMETRI:
/# PARAMETRI DI INPUT:
var: n : intero { numero dei punti }
var: x(n) : array di reali { nodi di interpolazione }
var: y(n) : array di reali { ordinate corrispondenti }
{ ai nodi di interpolazione }

/# PARAMETRI DI OUTPUT:
var: y(n) : array di reali { coefficienti del polinomio }
{ interpolante }

/# VARIABILI LOCALI:
var: i, k : interi

/# INIZIO ISTRUZIONI:
{ ciclo sull'ordine delle differenze divisa }
for k = 1, n - 1 do
{ calcolo delle differenze divise di ordine k }
for i = n, k + 1, step -1 do
{ calcolo elemento i-mo }
endfor
endfor
end diffdivise

```

Procedura 3.3: Algoritmo per il calcolo dei coefficienti del polinomio interpolante espresso nella formula di Newton

La versione finale è, dunque, la seguente:

```

procedure diffdivise(in:n, x, y ; out: y)
  /* SCOPO: calcolo dei coefficienti del polinomio interpolante
   di Lagrange, espresso nella formula di Newton.

  /* SPECIFICHE PARAMETRI:
  /* PARAMETRI DI INPUT:
  var: n : intero { numero dei punti }
  var: x(n) : array di reali { nodi di interpolazione }
  var: y(n) : array di reali { ordinate corrispondenti }
                                { ai nodi di interpolazione }

  /* PARAMETRI DI OUTPUT:
  var: y(n) : array di reali { coefficienti del polinomio }
                                { interpolante }

  /* VARIABILI LOCALI:
  var: k, i : interi

  /* INIZIO ISTRUZIONI:
  { ciclo sull'ordine delle differenze divise }
  for k = 1, n - 1 do
    { calcolo delle differenze divise di ordine k }
    for i = n, k + 1, step -1 do
      { calcolo elemento i-mo }
      yi = (yi - yi-1) / (xi - xi-k)
    endfor
  endfor
end diffdivise

```

#### Procedura 3.4: Algoritmo per il calcolo delle differenze divise

L'algoritmo calcola in ciascuno degli  $n - 1$  passi, un numero di differenze divise pari a  $n - i$  se  $i$  indica il generico passo. Quindi calcola complessivamente:

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

differenze divise, e, poiché ciascuna richiede due addizioni e una divisione, la complessità asintotica di tempo dell'algoritmo per il calcolo dei coefficienti del polinomio è:

$$T(n) = \mathcal{O}(n^2) \text{ flop}$$

Analogamente si può sviluppare l'algoritmo che procede "per diagonali" in cui, ad ogni passo, si calcola una diagonale della tabella delle differenze divise. In particolare, riferendoci ancora alla Tavola 1, delle differenze divise, l'algoritmo "per diagonali" calcola, al primo passo, la differenza divisa  $y[x_1, x_2]$ , al secondo passo calcola le due differenze divise  $y[x_2, x_3]$  e  $y[x_1, x_2, x_3]$ , al terzo passo calcola le tre differenze divise  $y[x_3, x_4]$ ,  $y[x_2, x_3, x_4]$  e  $y[x_1, x_2, x_3, x_4]$  e così procedendo fino al passo  $(n - 1)$ -esimo, in cui sono calcolate  $n - 1$  differenze divise e, precisamente,  $y[x_{n-1}, x_n], \dots, y[x_1, x_2, \dots, x_n]$ . La complessità di tempo di questo algoritmo è, ancora,

$$T(n) = \mathcal{O}(n^2) \text{ flop}$$

perché le operazioni sono le stesse dell'algoritmo "per colonna", ciò che è cambiato è semplicemente l'ordine con cui vengono calcolate le differenze divise.

### 3.2.9 Valutazione del polinomio interpolante

Calcolati i coefficienti  $a_i$ , il polinomio interpolante di Lagrange può essere valutato in un fissato valore  $\tilde{x}$  mediante una particolare versione dell'algoritmo di Horner:

```

procedure Horner_Newton(in: n, x, a,  $\tilde{x}$  ; out: p)

    /* SCOPO: valutazione del polinomio interpolante di Lagrange
       espresso nella formula di Newton.

    /* SPECIFICHE PARAMETRI:
    /* PARAMETRI DI INPUT:
    var: n      : intero      { numero dei punti }
    var: x(n)  : array di reali { nodi di interpolazione}
    var:  $\tilde{x}$    : reale       { punto in cui si richiede }
                                { la valutazione del polinomio }
    var: a(n)  : array di reali { coefficienti del polinomio }
                                { interpolante }

    /* PARAMETRI DI OUTPUT:
    var: p      : reale       { valore del polinomio in  $\tilde{x}$  }

    /* VARIABILI LOCALI:
    var: i      : intero

    /* INIZIO ISTRUZIONI:
    p := a(n);
    for i = n - 1, 1 step -1 do
        p := p · ( $\tilde{x}$  - x(i)) + a(i);
    endfor
end Horner_Newton

```

Procedura 3.5: Algoritmo di Horner per la valutazione di un polinomio espresso nella formula di Newton

Ad ogni passo l'algoritmo effettua una moltiplicazione ed una addizione, pertanto la complessità di tempo di questo algoritmo è  $T(n) = \mathcal{O}(n)$  flop.

♣ **Esempio 3.15.** Calcolare i coefficienti del polinomio interpolante  $n + 1$  punti noti i coefficienti del polinomio interpolante i primi  $n$  punti.

Per la formula di Newton il polinomio  $p \in \Pi_n$  interpolante i punti  $(x_1, y_1), \dots, (x_{n+1}, y_{n+1})$  differisce da  $q \in \Pi_{n-1}$  interpolante i punti  $(x_1, y_1), \dots, (x_n, y_n)$  solo nell'ultimo termine, cioè :

$$p(x) = q(x) + a_n(x - x_1) \cdot (x - x_2) \cdots (x - x_n)$$

con:

$$a_n = y[x_1, x_2, \dots, x_{n+1}],$$

pertanto è necessario calcolare solo l'ultimo coefficiente di  $p$ , cioè  $a_{n+1}$ . Per tale motivo aggiungiamo innanzitutto un elemento in più nelle prime due colonne della tavola delle differenze divise, cioè rispettivamente  $x_{n+1}$  e  $y_{n+1}$ . Successivamente si devono calcolare le differenze divise che si ottengono con l'aggiunta di questo punto:

$$y[x_1, x_{n+1}], \dots, y[x_1, \dots, x_{n+1}].$$

Dunque, supponendo di aver calcolato e memorizzato nel vettore  $y$  le prime  $n$  differenze divise:

$$y[x_1], y[x_1, x_2], \dots, y[x_1, \dots, x_n]$$

come precedentemente descritto, un modo per calcolare le nuove differenze divise è quello di combinare al primo passo ( $k=1$ )  $y[x_{n+1}]$  con  $y[x_1]$  per calcolare  $y[x_1, x_{n+1}]$  e memorizzare quest'ultima differenza divisa in  $y_{n+1}$ ; successivamente, al secondo passo ( $k=2$ ), combinando  $y[x_1, x_{n+1}]$  con  $y[x_1, x_2]$ , si calcola  $y[x_1, x_2, x_{n+1}]$  e si memorizza il risultato su  $y[x_1, x_{n+1}]$ ; al terzo passo ( $k = 3$ ) si calcola  $y[x_1, x_2, x_3, x_{n+1}]$  a partire da  $y[x_1, x_2, x_{n+1}]$  e  $y[x_1, x_2, x_3]$  e si memorizza il risultato su  $y[x_1, x_2, x_{n+1}]$ . Così procedendo, dopo  $n$  passi, su  $y_{n+1}$  si ottiene la differenza divisa di ordine  $n$   $y[x_1, x_2, \dots, x_{n+1}]$  corrispondente al coefficiente  $a_{n+1}$ :

$x_i$	$y_i$	$k = 1$	$k = 2$	...	$k = n$
$x_1$	$y[x_1]$				
$x_2$	$y[x_1, x_2]$				
$x_3$	$y[x_1, x_2, x_3]$				
$\vdots$	$\vdots$				
$x_n$	$y[x_1, \dots, x_n]$				
$x_{n+1}$	$y[x_{n+1}]$	$y[x_1, x_{n+1}]$	$y[x_1, x_2, x_{n+1}]$	...	$y[x_1, \dots, x_{n+1}]$

Tavola 2: Aggiunta di un punto nella tavola delle differenze divise



### 3.2.10 Un elemento di software per l'interpolazione polinomiale

Una routine di software matematico per la costruzione del polinomio interpolante di Lagrange espresso nella formula di Newton deve prevedere come dati di input il vettore dei nodi di interpolazione e quello contenente i corrispondenti valori. La routine, dopo aver calcolato i coefficienti del polinomio interpolante mediante le differenze divise, valuta il polinomio utilizzando l'algoritmo di Horner. Si può prevedere la valutazione in uno o più punti. Pertanto un altro dato di input è il vettore contenente i valori di valutazione, eventualmente costituito da un solo elemento. Per quanto riguarda i controlli sui dati di input è opportuno che la routine verifichi che i nodi siano a due a due distinti e che siano ordinati in senso crescente ed eventualmente li ordini. Pertanto, le specifiche di una tale routine potrebbero essere del tipo:

```
SUBROUTINE NEWTON(X,Y,N,M,Z,PZ,IFAIL)
```

dove:

- $X$  è il vettore di lunghezza  $N$  contenente i nodi di interpolazione;
- $Y$  è il vettore di lunghezza  $N$  contenente le ordinate dei punti di interpolazione;
- $N$  è il numero dei punti di interpolazione;
- $M$  è il numero dei punti di valutazione del polinomio interpolante;
- $Z$  è il vettore contenente i punti in cui si vuole valutare il polinomio interpolante;
- $PZ$  è il vettore contenente i valori assunti dal polinomio interpolante nei punti assegnati;
- $IFAIL$  è l'indicatore di errore.

Un esempio di programma chiamante per tale subroutine, scritto nel linguaggio di programmazione ad alto livello FORTRAN, è il seguente:

```

PROGRAM MAINNEWTON

C
C Dichiarazione delle variabili
C
    INTEGER N,M, IFAIL,I
    REAL X(100), Y(100), Z(100), PZ(100)
C
C Lettura dei dati di input
C
    WRITE(*,*)'INSERIRE NUMERO DEI PUNTI'
    'DI INTERPOLAZIONE'
    READ(*,*) 'N= ',N
    WRITE(*,*)'INSERIRE',N,'NODI DI INTERPOLAZIONE:'
    READ(*,*) (X(I),I=1,N)
    WRITE(*,*)'INSERIRE GLI',N,'VALORI CORRISPONDENTI:'
    READ(*,*) (Y(I),I=1,N)
    WRITE(*,*)'INSERIRE NUMERO DI PUNTI'
    'IN CUI VALTARE IL POLINOMIO'
    READ(*,*) M
    WRITE(*,*)'INSERIRE',M,'VALORI'
    READ(*,*) (Z(I),I=1,N)
C
C Chiamata alla subroutine NEWTON
C
    CALL NEWTON(X,Y,N,Z,PZ,IFAIL)
C
C Stampa dei risultati
C
    IF (IFAIL .EQ. 0) THEN
        WRITE(*,*) (PZ(I), I=1,N)
        STOP
    END

```

Procedura 3.6: Programma chiamante per la routine Newton

```

SOUTROUTINE NEWTON(X,Y,N,M,Z,PZ,IFAIL)

C
C Dichiarazione delle variabili
C
REAL X(N), Y(N), Z(M), PZ(M)
INTEGER N,M,IFAIL
C
C Chiamata alla routine DIST che verifica se
C i nodi assegnati sono distinti.
C
CALL DIST(X,N,IFAIL)

IF (IFAIL .EQ. 0) THEN
C
C Se i nodi sono distinti si chiama la routine DIFFDIV
C per il calcolo delle differenze divise
C
CALL DIFFDIV(X,Y,N)
C
C Chiamata alla routine HORNER per la valutazione
C del polinomio interpolante.
C
CALL HORNER (X,Y,N,Z,M,PZ)

ENDIF
RETURN
END

```

Procedura 3.7: Subroutine Newton

### 3.3 Metodi costruttivi per il polinomio interpolante di Hermite

#### 3.3.1 Formula di Lagrange

♣ **Esempio 3.16.** Nell'esempio 3.10 abbiamo costruito la parabola passante per i punti  $P_1 \equiv (0, 2)$ ,  $P_2 \equiv (1, 1)$  e avente vertice in  $P_2$ . La parabola è:

$$p(x) = x^2 - 2x + 2$$

Il polinomio  $p \in \Pi_2$  è quindi il polinomio interpolante di Hermite relativo ai punti  $P_1$  e  $P_2$ :

$$\begin{cases} y(x_1) = y_1^0 = 2 \\ y(x_2) = y_2^0 = 1 \\ y^{(1)}(x_2) = y_2^1 = 0 \end{cases}$$

o, equivalentemente:

nodi	$x_1$	$x_2$
condizioni	$y_1^0$	$y_2^0$
		$y_2^1$

Scriviamo  $p$  nella forma:

$$p(x) = y_1^0 \cdot h_{10}(x) + y_2^0 \cdot h_{20}(x) + y_2^1 \cdot h_{21}(x)$$

con  $h_{ij}$  polinomi di secondo grado. Determiniamo tali polinomi utilizzando le condizioni di interpolazione. Da

$$\begin{cases} p(x_1) = y_1^0 & \rightarrow y_1^0 \cdot h_{10}(x_1) + y_2^0 \cdot h_{20}(x_1) + y_2^1 \cdot h_{21}(x_1) = y_1^0 \\ p(x_2) = y_2^0 & \rightarrow y_1^0 \cdot h_{10}(x_2) + y_2^0 \cdot h_{20}(x_2) + y_2^1 \cdot h_{21}(x_2) = y_2^0 \\ p'(x_2) = y_2^1 & \rightarrow y_1^0 \cdot h'_{10}(x_2) + y_2^0 \cdot h'_{20}(x_2) + y_2^1 \cdot h'_{21}(x_2) = y_2^1 \end{cases}$$

Queste condizioni sono sicuramente verificate se richiediamo che i polinomi  $h_{10}, h_{20}, h_{21}, h'_{10}, h'_{20}, h'_{21}$  siano tali che:

$$\begin{cases} h_{10}(x_1) = 1 & h_{20}(x_1) = 0 & h_{21}(x_1) = 0 \\ h_{10}(x_2) = 0 & h_{20}(x_2) = 1 & h_{21}(x_1) = 0 \\ h'_{10}(x_2) = 0 & h'_{20}(x_2) = 0 & h'_{21}(x_2) = 1 \end{cases}$$

Dovendo essere polinomi di secondo grado,

- per il polinomio  $h_{20}$ , essendo la derivata prima un polinomio di primo grado e dovendo avere, dalla terza condizione, lo zero in  $x_2$  risulta

$$h'_{20}(x) = a(x - x_2)$$

da cui integrando:

$$h_{20}(x) = a/2 \cdot (x - x_2)^2 + c$$

dalla seconda condizione si ha  $c = 1$ , e imponendo la prima segue che  $a = \frac{-2}{(x_1 - x_2)^2}$ ;

- per il polinomio  $h_{10}$ , analogamente, essendo la derivata prima un polinomio di primo grado, e dovendo avere, dalla terza condizione, lo zero in  $x_2$  risulta:

$$h'_{10}(x) = b(x - x_2)$$

da cui integrando:

$$h_{10}(x) = b/2 \cdot (x - x_2)^2 + d$$

dalla seconda condizione si ricava  $d = 0$ , e imponendo la prima segue che  $b = \frac{2}{(x_1 - x_2)^2}$ ;

- per il polinomio  $h_{21}$ , dalle prime due condizioni risulta che esso ha i due zeri  $x_1$  e  $x_2$ , inoltre essi sono zeri semplici perché il polinomio è di secondo grado quindi può avere al più due zeri distinti, per cui:

$$h_{21}(x) = f(x - x_1)(x - x_2)$$

imponendo la terza condizione si ricava:

$$f = 1/(x_2 - x_1)$$

La rappresentazione di  $p$  nella forma

$$p(x) = y_1^0 \cdot h_{10}(x) + y_2^0 \cdot h_{20}(x) + y_2^1 \cdot h_{21}(x)$$

con  $h_{ij}$  polinomi di secondo grado, univocamente determinati dalle condizioni di interpolazione, è detta **Formula di Lagrange** per il polinomio di Hermite. ♣

Analogamente al problema di interpolazione di Lagrange, è possibile esprimere il polinomio interpolante di Hermite relativo ai dati:

- $x_i$ ,  $i = 1, \dots, n$ ;
- $y_i^j$ ,  $i = 1, \dots, n$ ,  $j = 0, \dots, l_i - 1 \forall i$ ,
- $\sum_{i=1}^n l_i = m + 1$

nella forma:

$$q(x) = \sum_{j=0}^{l_1-1} y_1^j h_{1j}(x) + \sum_{j=0}^{l_2-1} y_2^j h_{2j}(x) + \dots + \sum_{j=0}^{l_n-1} y_n^j h_{nj}(x)$$

dove

$$h_{ij} \in \Pi_m, \quad i = 1, \dots, n$$

e

$$h_{ij}^{(r)}(x_s) = \begin{cases} 1 & (i, j) = (s, r) \\ 0 & \text{altrimenti} \end{cases}$$

### 3.3.2 Formula di Newton

Anche per la formula di Newton è possibile ricavare un'espressione analoga valida per costruire il polinomio interpolante di Hermite.

♣ **Esempio 3.17.** Consideriamo il seguente problema di interpolazione di Lagrange:

nodi	$x_1$	$\tilde{x}_1$	$x_2$	$\tilde{x}_2$
condizioni	$y_1$	$\tilde{y}_1$	$y_2$	$\tilde{y}_2$

e costruiamo il polinomio interpolante nella formula di Newton:

$$\begin{aligned} q(x) = & y[x_1] + y[x_1, \tilde{x}_1](x - x_1) + \\ & + y[x_1, \tilde{x}_1, x_2](x - x_1)(x - \tilde{x}_1) + \\ & + y[x_1, \tilde{x}_1, x_2, \tilde{x}_2](x - x_1)(x - \tilde{x}_1)(x - x_2) \end{aligned}$$

Immaginiamo ora di far coincidere i nodi a due a due cioè:

$$\tilde{x}_1 \longrightarrow x_1$$

e

$$\tilde{x}_2 \longrightarrow x_2$$

in maniera tale che la condizione in  $\tilde{x}_1$  diventi ora una "seconda" condizione su  $x_1$ , e analogamente la condizione su  $\tilde{x}_2$  diventi una "seconda" condizione su  $x_2$ . Il problema di interpolazione diventa un problema con due nodi e quattro condizioni, due su ciascun nodo:

nodi	$x_1$	$x_2$
condizioni	$y_1$	$y_2$

Se poniamo  $y_1^1 = \tilde{y}_1$  e  $y_2^1 = \tilde{y}_2$  ci siamo ricondotti ad un problema di interpolazione di Hermite. In tal caso, la formula di Newton diventa:

$$\begin{aligned} q(x) = & y[x_1] + y[x_1, x_1](x - x_1) + \\ & + y[x_1, x_1, x_2](x - x_1)(x - x_1) + \\ & + y[x_1, x_1, x_2, x_2](x - x_1)^2(x - x_2) \end{aligned}$$

In questo caso la formula di Newton è stata ricondotta ad una analoga in cui i coefficienti sono espressi in termini delle differenze divise. Come si nota, i nodi sono ripetuti pertanto è naturale aspettarsi che queste quantità non possano essere definite (e quindi calcolate) in maniera analoga alle usuali differenze divise. Nasce dunque il problema di dare una definizione alle **differenze divise su nodi multipli** ♣

Per derivare la formula di Newton per il polinomio interpolante di Hermite è necessario dimostrare alcuni risultati relativi alle differenze divise su nodi multipli.

**Proposizione 3.1.** *Siano*

$$x_1, x_2, \dots, x_k \in [a, b]$$

*Sia, inoltre*

$$y = f(x), \quad x \in [a, b]$$

*una funzione derivabile in  $[a, b]$ . Se :*

$$\begin{aligned} f(x_i) &= y_i \\ f^{(1)}(x_i) &= y_i^1 \end{aligned} \tag{3.37}$$

*allora:*

$$\lim_{x_j \rightarrow x_i} y[x_i, x_j] = y_i^1, \quad i = 1, \dots, k$$

**Dimostrazione** Si ha:

$$\lim_{x_j \rightarrow x_i} y[x_i, x_j] = \lim_{x_j \rightarrow x_i} \frac{y_j - y_i}{x_j - x_i} = \lim_{x_j \rightarrow x_i} \frac{f(x_j) - f(x_i)}{x_j - x_i} = f'(x_i) = y_i^1$$

■

A fronte di questo risultato, è naturale dare la seguente

**Definizione 3.10. (Differenza divisa di ordine uno su nodi coincidenti)**  
*La differenza divisa di ordine uno, su due nodi coincidenti:*

$$y[x_i, x_i]$$

*è definita nel modo seguente:*

$$y[x_i, x_i] = y_i^1$$

*dove  $y_i^1$  rappresenta la seconda condizione di interpolazione nel nodo  $x_i$ .*

Analogamente, dimostriamo la seguente:

**Proposizione 3.2.** *Siano:*

$$x_1, x_2, \dots, x_k \in [a, b]$$

*Sia, inoltre,:*

$$y = f(x) \in C^2[a, b], \quad x \in [a, b]$$

*Se*

$$\begin{aligned} f(x_i) &= y_i \\ f^{(1)}(x_i) &= y_i^1 \\ f^{(2)}(x_i) &= y_i^2 \end{aligned} \tag{3.38}$$

*allora:*

$$\lim_{x_j \rightarrow x_i} y[x_i, x_i, x_j] = \frac{y_i^2}{2}, \quad i = 1, \dots, k$$

**Dimostrazione** Si ha:

$$\begin{aligned} \lim_{x_j \rightarrow x_i} y[x_i, x_i, x_j] &= \\ &= \lim_{x_j \rightarrow x_i} \frac{-y[x_i, x_i] + y[x_i, x_j]}{x_j - x_i} = \lim_{x_j \rightarrow x_i} \frac{-f'(x_i) + \frac{f(x_j) - f(x_i)}{x_j - x_i}}{x_j - x_i} \end{aligned} \tag{3.39}$$

Sviluppiamo  $f(x)$  in serie di Taylor di punto iniziale  $x_i$ , troncata al secondo ordine e valutiamo lo sviluppo in  $x_j$ . Si ha:

$$f(x_j) = f(x_i) + f'(x_i)(x_j - x_i) + \frac{1}{2}f''(x_i)(x_j - x_i)^2 + o((x_i - x_j)^2)$$

Sostituendo nella (3.39), si ottiene:

$$\frac{-f'(x_i) + \frac{f(x_j) - f(x_i)}{x_j - x_i}}{x_j - x_i} = \frac{-f'(x_i) + f'(x_i) + 1/2 \cdot f''(x_i)(x_j - x_i)}{x_j - x_i} = \frac{f''(x_i)}{2}$$

■

**Definizione 3.11. (Differenza divisa di ordine due su nodi coincidenti)**

*La differenza divisa di ordine due, su tre nodi coincidenti:*

$$y[x_i, x_i, x_i]$$

*è definita nel modo seguente:*

$$y[x_i, x_i, x_i] = \frac{y_i^2}{2}$$

*dove  $y_i^2$  rappresenta la terza condizione di interpolazione nel nodo  $x_i$ .*

In generale, se  $y_i^k$  rappresenta la condizione  $(k+1)$ -ma nel nodo  $x_i$  allora, la differenza divisa di ordine  $k$  sul nodo  $x_i$  si definisce nel modo seguente:

$$y \underbrace{[x_i, x_i, \dots, x_i]}_{k+1} = \frac{y_i^k}{k!}$$

mentre per il calcolo della differenza su nodi multipli, sussiste la seguente formula ricorrente:

$$\begin{aligned} & y \underbrace{[x_i, \dots, x_i]}_k \underbrace{x_j, \dots, x_j}_s \underbrace{x_p, \dots, x_p}_r = \\ & = \frac{y \underbrace{[x_i, \dots, x_i]}_{k-1} \underbrace{x_j, \dots, x_j}_{s-1} \underbrace{x_p, \dots, x_p}_{r-1} - y \underbrace{[x_i, \dots, x_i]}_k \underbrace{x_j, \dots, x_j}_s \underbrace{x_p, \dots, x_p}_{r-1}}{x_p - x_i} \end{aligned} \quad (3.40)$$

Le differenze divise su nodi multipli si possono calcolare secondo uno schema analogo a quello costruito per le differenze divise su nodi distinti. Precisamente, si costruisce una tabella a doppia entrata: lungo le righe si dispongono le differenze divise di ordine  $k$ , lungo le colonne si dispongono i nodi avendo cura di ripetere i nodi  $x_i$  tante volte quante sono le condizioni imposte sul nodo. Ad esempio, considerato il seguente problema di interpolazione di Hermite:

nodi	$x_1$	$x_2$	$x_3$
condizioni	$y_1^0$	$y_2^0$	$y_3^0$
	$y_2^1$	$y_3^1$	
	$y_2^2$	$y_3^2$	

Schema grafico di un problema di interpolazione di Hermite

Si ottiene la tavola seguente:

$x_i$	$y_i$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
$x_1$	$y_1$	$y[x_1, x_2]$				
$x_2$	$y_2$		$y[x_1, x_2, x_2]$			
$x_2$	$y_2$	$y[x_2, x_2]$		$y[x_1, x_2, x_2, x_2]$		
$x_2$	$y_2$	$y[x_2, x_2]$	$y[x_2, x_2, x_2]$	$y[x_2, x_2, x_2, x_3]$	$y[x_1, x_2, x_2, x_2, x_3]$	$y[x_1, x_2, x_2, x_2, x_3, x_3]$
$x_2$	$y_2$	$y[x_2, x_3]$	$y[x_2, x_2, x_3]$		$y[x_2, x_2, x_3, x_3]$	
$x_3$	$y_3$	$y[x_2, x_3]$	$y[x_2, x_3, x_3]$			
$x_3$	$y_3$	$y[x_3, x_3]$				

Tavola 3: Differenze divise su nodi multipli

♣ **Esempio 3.18.** Sia  $f \in C^k[a, b]$ . Costruiamo il polinomio interpolante di Hermite relativo ai seguenti dati:

nodi	$x_1$
condizioni	$y_1^0 = f(x_1)$ $y_1^1 = f'(x_1)$ $\vdots$ $y_1^k = f^{(k)}(x_1)$

Il polinomio di Taylor relativo alla funzione  $f$ , di grado  $k$  e di punto iniziale  $x_1$ :

$$P(x) = f(x_1) + f'(x_1)(x - x_1) + \dots + \frac{f^{(k)}(x_1)}{k!}(x - x_1)^k$$

è un polinomio di grado  $k$ , tale che:

$$P^{(j)}(x_1) = f^{(j)}(x_1) = y_1^j \quad j = 0, \dots, k$$

cioè soddisfa le condizioni di interpolazione di Hermite. Per l'unicità del polinomio interpolante esso è quindi il polinomio interpolante di Hermite.

In altre parole, il polinomio di Taylor risolve un particolare problema di interpolazione di Hermite, in cui sono note le  $k + 1$  condizioni su un unico punto. Tali condizioni sono espresse in termini delle derivate nel punto  $x_1$  della funzione  $f$ .



### 3.4 Interpolazione polinomiale a tratti

Uno dei requisiti principali di un modello matematico è, come già abbiamo visto nei paragrafi precedenti, che esso sia in grado di descrivere il problema in modo attendibile, cioè, ad esempio, che soddisfi i requisiti (i)  $\div$  (iii) illustrati nel primo paragrafo.

Purtroppo, il polinomio interpolante, se il numero dei punti di interpolazione è elevato, non fornisce in generale un modello accettabile. Infatti, al crescere del numero di punti aumenta il grado del polinomio interpolante e aumentano anche le oscillazioni del polinomio corrispondente ottenendo un modello non sempre coerente con l'andamento dei dati.

♣ **Esempio 3.19.** Assegnati i seguenti  $n = 24$  punti di interpolazione:

$n$	$x$	$y$
1	10.00	0.42
2	10.20	0.48
3	10.40	0.51
4	10.60	0.52
5	10.80	0.53
6	11.00	0.55
7	11.20	0.58
8	11.40	0.61
9	11.60	0.65
10	11.80	0.74
11	11.89	0.91
12	11.96	1.29

$n$	$x$	$y$
13	12.00	1.52
14	12.04	1.87
15	12.08	2.35
16	12.12	2.89
17	12.16	3.40
18	12.20	3.83
19	12.28	4.27
20	12.36	4.53
21	12.44	4.62
22	12.50	4.64
23	13.00	4.64
24	14.00	4.64

il polinomio interpolante di Lagrange  $p(x) \in \Pi_{23}$  ha il grafico riportato in Figura 3.18.

Dalla figura appare evidente che  $p$  oscilla in modo critico fra i dati e, quindi, piuttosto che ottenere nuove informazioni significative a partire dai dati assegnati, si perde gran parte delle informazioni in essi contenute.

Poichè tale comportamento dipende dal grado del polinomio interpolante, e questo, a sua volta, dipende dal numero di punti da interpolare, si può pensare di considerare solo una parte degli  $n$  punti (ad esempio eliminare un punto ogni due) e di costruire il corrispondente polinomio interpolante di Lagrange; in realtà questa strategia non sempre è consigliabile, perché si potrebbero eliminare dati significativi, e in ogni caso non sempre garantisce una soluzione accettabile. Ad esempio, riducendo successivamente il numero di punti di interpolazione, con  $n = 13, 9, 6$  punti si ottengono i polinomi interpolanti di Fig. 3.19, Fig. 3.20, Fig. 3.21.

Al diminuire del grado, il polinomio interpolante oscilla molto meno tra i dati, ma il modello può risultare ancora una volta poco attendibile perché ottenuto usando solo parte delle informazioni sul problema in esame, le quali evidentemente non riescono a riflettere in maniera adeguata l'andamento del fenomeno.

Probabilmente, se invece del polinomio interpolante congiungiamo a due a due i punti con un segmento di retta, può accadere che riusciamo ad avere una descrizione più soddisfacente. Ciò è quello che

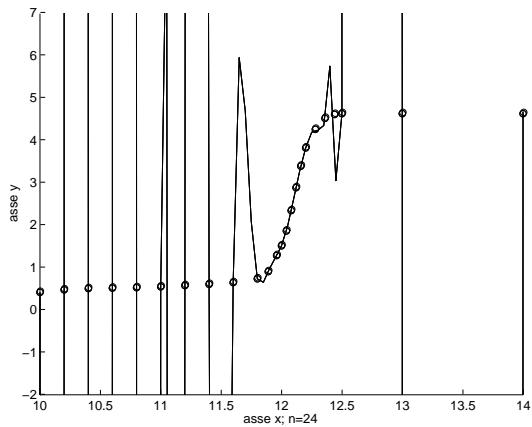


Figura 3.18: Andamento del polinomio interpolante di Lagrange di grado 23 relativo ai 24 punti assegnati

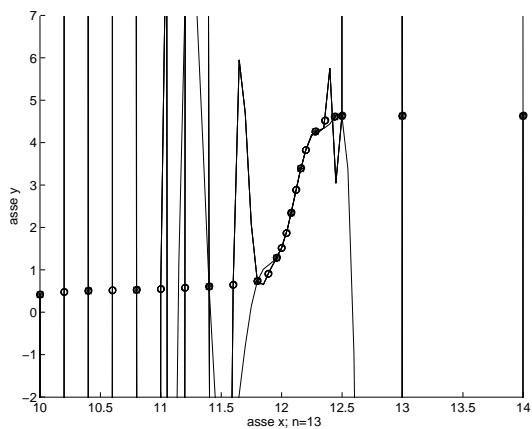


Figura 3.19: Andamento del polinomio interpolante di Lagrange di grado 12 relativo a 13 punti scelti tra i 24 assegnati.

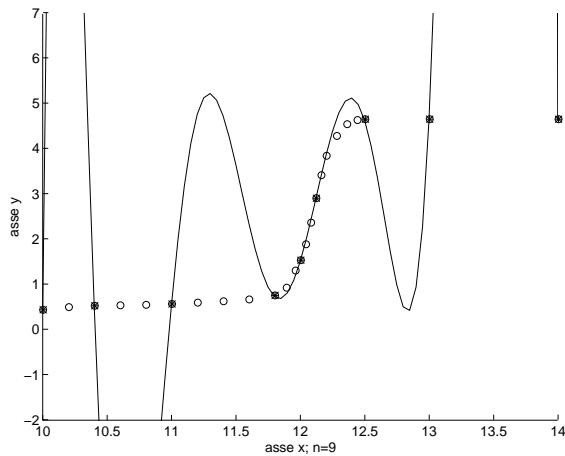


Figura 3.20: Andamento del polinomio interpolante di Lagrange di grado 8 relativo a 9 dei 24 punti assegnati.

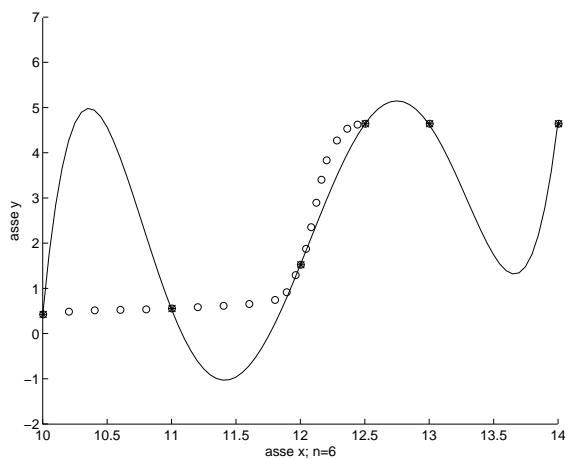


Figura 3.21: Polinomio interpolante di Lagrange di grado 5.

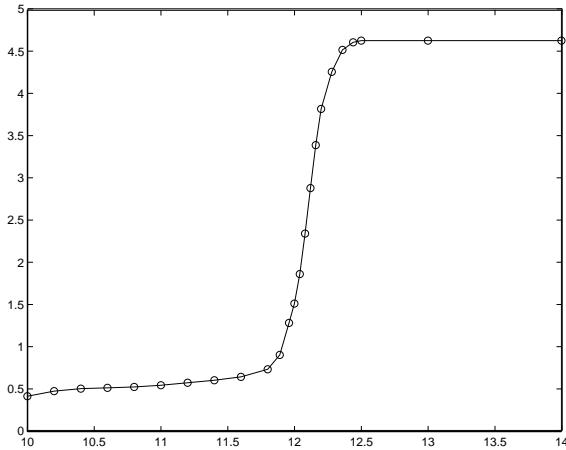


Figura 3.22: Un modello interpolante costruito congiungendo a due a due i punti con segmenti di rette.

viene mostrato in Fig. 3.22.



**Esempio 3.20.** Consideriamo i seguenti punti: e supponiamo che descrivano il profilo di un catamarano. Volendo disegnare la sezione del catamarano abbiamo bisogno di un modello che tra i punti interni non possegga delle oscillazioni.

Vediamo cosa succede se costruiamo il polinomio interpolante:

Chiaramente questo modello non è soddisfacente! Congiungendo a due a due i punti con segmenti di rette, si ottiene il modello riportato in Fig. 3.25, che sebbene segua i punti meglio del polinomio interpolante di grado 14, non è un modello soddisfacente a causa delle irregolarità nei punti di raccordo tra un tratto ed un altro che, in questo caso, non possono essere tollerate!



In molti casi per ridurre il grado del polinomio interpolante e le sue oscillazioni, una buona strategia è quella di suddividere l'intervallo dei nodi in sottointervalli contigui e costruire il modello interpolante localmente, cioè su ciascun sottointervallo. Ciò consente di determinare relativamente a ciascun sottointervallo un polinomio interpolante di grado più basso indipendentemente da quanti siano i nodi complessivi.

In particolare, raggruppando i nodi a due a due, il polinomio interpolante relativo a ciascun sottointervallo è un polinomio interpolante lineare a tratti cioè una funzione che

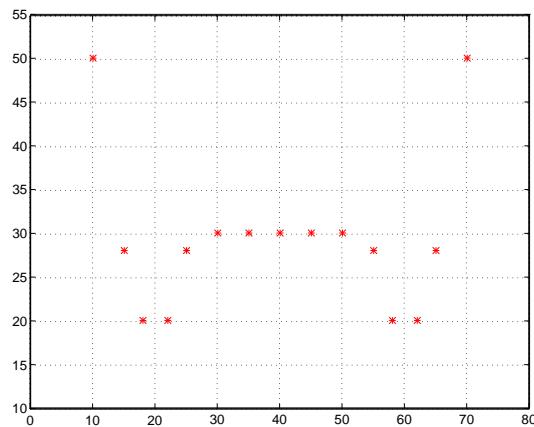


Figura 3.23: Dati rilevati per descrivere il profilo di un catamarano.

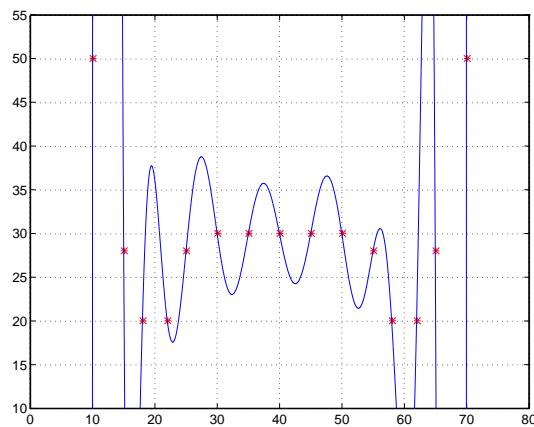


Figura 3.24: Polinomio interpolante di grado 14.

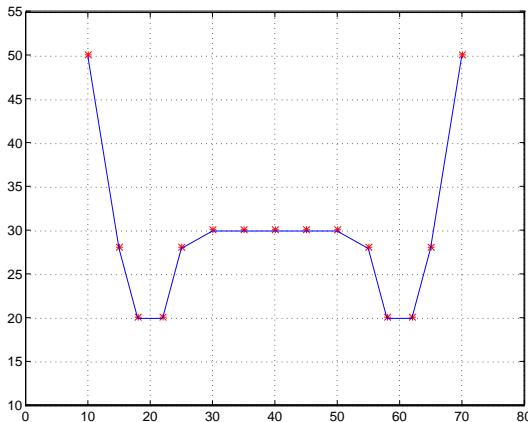


Figura 3.25: Un modello interpolante ottenuto congiungendo a due a due i punti con segmenti di rette.

in ogni intervallo dei nodi coincide con un polinomio di grado al più uno interpolante i punti estremi del sottointervallo. Diamo dunque la seguente:

**Definizione 3.12. (Funzione polinomiale interpolante lineare a tratti)**

Assegnato un insieme di nodi strettamente crescente  $K = \{x_1, \dots, x_n\}$ , e un insieme di  $n$  valori corrispondenti  $y_1, y_2, \dots, y_n$ , una **funzione polinomiale interpolante (di Lagrange) lineare a tratti** definita sull'insieme  $K$  è una funzione  $f$  tale che, in ogni sottointervallo  $[x_i, x_{i+1}]_{i=1, \dots, n-1}$  coincida con un polinomio di grado al più 1:

$$f(x) \equiv p_i(x) \in \Pi_1 \quad x \in [x_i, x_{i+1}], \quad i = 1, \dots, n-1.$$

ed inoltre  $f$  soddisfi alle condizioni di interpolazione di Lagrange sui nodi assegnati, ovvero:

$$f(x_i) = y_i \quad i = 1, \dots, n$$

Osserviamo che una funzione  $f$  polinomiale lineare a tratti è per definizione anche una funzione interpolante i punti di coordinate  $(x_i, f(x_i))$ .

Questa definizione si generalizza in modo naturale, richiedendo che ciascun polinomio  $p_i$  anziché essere lineare abbia un grado maggiore di 1. Ad esempio, supponendo che ciascun tratto sia costituito da un polinomio di grado  $p$  occorre raggruppare i nodi in gruppi da  $p+1$ . Supponiamo, per semplicità di avere  $n$  punti, con  $n-1$  divisibile per  $p$ , diamo la seguente:

**Definizione 3.13. (Funzione polinomiale interpolante a tratti di grado  $p$ )**

Assegnato un insieme di  $n$  nodi strettamente crescente  $K = \{x_1, \dots, x_n\}$ , e un insieme

di  $n$  valori corrispondenti  $y_1, y_2, \dots, y_n$ , una **funzione polinomiale interpolante a tratti di grado  $p$**  definita sull'insieme dei nodi  $K$  è una funzione  $f$  tale che, in ogni sottointervallo  $I_i = [x_{(i-1)p+1}, x_{ip+1}]$ , con  $i = 1, \dots, (n-1)/p$ , coincide con un polinomio di grado al più  $p$ :

$$f(x) \equiv p_i(x) \in \Pi_p \quad x \in I_i \quad i = 1, \dots, (n-1)/p$$

ed inoltre  $f$  soddisfa alle condizioni di interpolazione di Lagrange sui nodi assegnati, ovvero:

$$f(x_j) = p_i(x_j) = y_j \quad \text{se } x_j \in I_i$$

♣ **Esempio 3.21.** Costruire una funzione  $f$  polinomiale lineare a tratti che interpoli i punti:

$$(x_1, y_1) \equiv (0, 1), \quad (x_2, y_2) \equiv (1, 3), \quad (x_3, y_3) \equiv (3, 2), \quad (x_4, y_4) \equiv (5, 3).$$

Se  $K = \{x_1, x_2, x_3, x_4\} = \{0, 1, 3, 5\}$ ,  $f(x)$  può essere costruita *localmente* in ciascun intervallo dei nodi  $[x_1, x_2], [x_2, x_3], [x_3, x_4]$ .

Nel primo intervallo  $[x_1, x_2] = [0, 1]$ ,  $f$  deve essere un polinomio di grado al più uno e deve interpolare i punti  $(x_1, y_1)$  e  $(x_2, y_2)$ ; pertanto in tale intervallo  $f$  deve coincidere con l'unico polinomio di grado al più uno interpolante i suddetti punti, cioè con il segmento appartenente alla retta passante per  $(x_1, y_1), (x_2, y_2)$ :

$$f(x) \equiv p_1(x) = y_1 + y[x_1, x_2](x - x_1) = 1 + \frac{3-1}{1-0}(x-0) = 1 + 2x.$$

Analogamente, per  $x \in [x_2, x_3]$ :

$$f(x) \equiv p_2(x) = y_2 + y[x_2, x_3](x - x_2) = 3 + \frac{2-3}{3-1}(x-1) = 3 - \frac{1}{2}(x-1)$$

e per  $x \in [x_3, x_4]$ :

$$f(x) \equiv p_3(x) = y_3 + y[x_3, x_4](x - x_3) = 2 + \frac{3-2}{5-3}(x-3) = 2 + \frac{1}{2}(x-3).$$

Riassumendo, quindi, la funzione polinomiale lineare a tratti interpolante i punti assegnati è definita nel modo seguente:

$$f(x) \equiv \begin{cases} 1 + 2x & x \in [0, 1] \\ 3 - 0.5(x-1) & x \in [1, 3] \\ 2 + 0.5(x-3) & x \in [3, 5] \end{cases}$$



Assegnati i punti di coordinate:

$$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n), \quad \text{con } x_1 < x_2 < \dots < x_n,$$

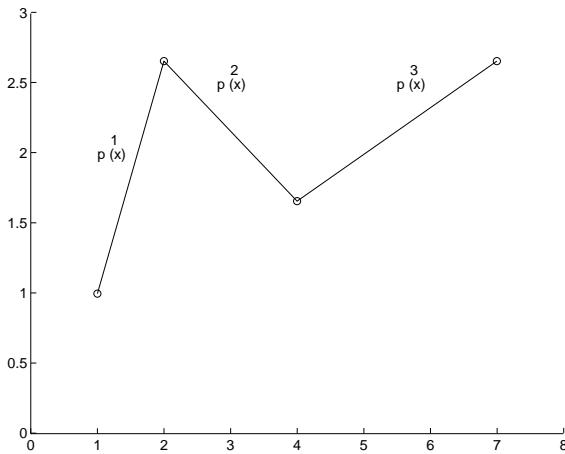


Figura 3.26: Funzione polinomiale lineare a tratti interpolante i punti  $(0,1)$ ,  $(1,3)$ ,  $(3,2)$ ,  $(5,3)$

la funzione polinomiale lineare a tratti  $f$  interpolante tali punti può essere espressa in ciascun intervallo  $[x_i, x_{i+1}]$ ,  $i = 1, \dots, n - 1$ , come:

$$f(x) \equiv p_i(x) = y_i + y[x_i, x_{i+1}](x - x_i) = a_i + b_i(x - x_i) \quad x \in [x_i, x_{i+1}].$$

Per costruire  $f$  occorre, dunque, calcolare i coefficienti  $a_i$  e  $b_i$ ,  $i = 1, \dots, n - 1$ .

Si osservi che in ciascun intervallo  $[x_i, x_{i+1}]$ ,  $i = 1, \dots, n - 1$ , i coefficienti  $a_i$  e  $b_i$  coincidono rispettivamente con le differenze divise di ordine zero e del primo ordine relative ai nodi  $x_i$  e  $x_{i+1}$ , e quindi

$$a_i = y_i \quad \text{e} \quad b_i = \frac{(y_{i+1} - y_i)}{(x_{i+1} - x_i)}.$$

Per valutare la funzione  $f$  così costruita in un fissato  $z$  occorre, poi:

1. **localizzare** l'intervallo dei nodi  $[x_i, x_{i+1}]$  tale che:

$$z \in [x_i, x_{i+1}]$$

2. **valutare**  $f(z) \equiv p_i(z) = y_i + b_i(z - x_i)$ .

Uno schema dell'algoritmo è mostrato nella procedura seguente.

```

procedure Pol_lin_tratti(in:  $x, y, z, n$  ; out:  $pz$ )
  /* SCOPO: costruzione della funzione polinomiale lineare a tratti
   interpolante un insieme assegnato di punti.
  /* Localizzazione dell'intervallo  $[x_i, x_{i+1}]$  (e quindi dell'indice  $i$ )
  /* al quale  $z$  appartiene, attraverso l'algoritmo di ricerca binaria.
  call ric_bin( $x, y, z, i, n$ )
  /* Calcolo dei coefficienti del segmento di retta che
  /* rappresenta la funzione polinomiale nell'intervallo  $[x_i, x_{i+1}]$ 
  call diffdiv( $i, y, b(i), x, n$ )
  /* Valutazione del polinomio lineare interpolante a tratti in  $z$ 
  call Horner( $i, b(i), z, pz, n$ )
  end Pol_lin_tratti

```

Procedura 3.8: Algoritmo per la costruzione e la valutazione  
della polinomiale lineare a tratti interpolante

### 3.4.1 Interpolazione mediante funzioni spline

#### Le funzioni spline

Si è visto che il polinomio interpolante non sempre fornisce una rappresentazione attendibile di una funzione descritta da un insieme di punti, a causa del fatto che, al crescere del numero di punti, cresce necessariamente il grado del polinomio e quindi le sue oscillazioni aumentano. Un modo per ovviare a questo inconveniente è quello di costruire localmente il polinomio interpolante, cioè suddividere l'intervallo dei nodi di interpolazione in sottointervalli raggruppando i nodi (a due a due oppure a tre a tre, etc.) ed in ciascuno di questi costruire il polinomio interpolante corrispondente al numero di nodi raggruppati. In ogni caso, però, non si riescono ad evitare eventuali discontinuità delle derivate nei punti di raccordo tra un polinomio e l'altro. Per determinare un modello più soddisfacente dei precedenti, è auspicabile poter costruire:

- un polinomio di grado basso;
- una funzione sufficientemente *dolce (smooth)* su tutto l'intervallo.

La soluzione a questo problema è fornita da particolari funzioni polinomiali a tratti dette *funzioni spline*. Una particolare funzione spline ha il grafico riportato in Fig. 3.27.

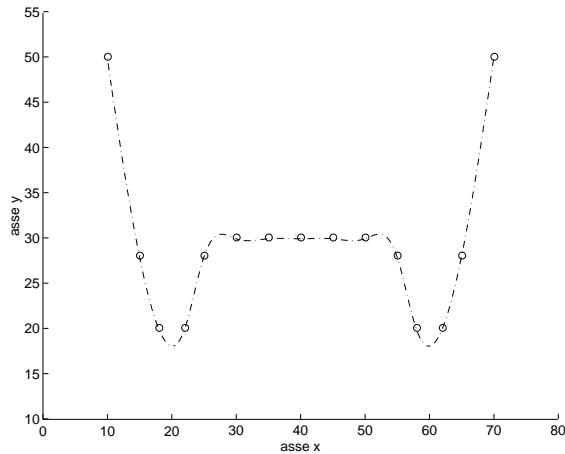


Figura 3.27: Una funzione spline

♣ **Esempio 3.22.** Un problema che si può presentare al disegnatore è il seguente: assegnati sul piano alcuni punti, unirli mediante una curva che non presenti spigoli, che cioè sia *abbastanza dolce*. Per risolvere questo problema si fa ricorso ad un leggero listello di legno (o di altro materiale flessibile) che, mediante gancetti sistemati opportunamente, unisce i punti assegnati nel modo desiderato. I ganci possono essere sistemati direttamente sui punti. In lingua inglese tale listello è noto col nome di *spline*<sup>28</sup> (Fig. 3.28).

<sup>28</sup> Per comprendere il passaggio dallo strumento spline alla definizione di *spline* come funzione matematica, introdotto un sistema di riferimento cartesiano, siano  $x_1, x_2, \dots, x_n$  le ascisse dei punti in cui sono disposti i pesi. Si indichi con  $y = y(x)$  l'equazione della curva descritta dal listello tra due pesi consecutivi, cioè per  $x \in [x_i, x_{i+1}]$ , in un fissato riferimento cartesiano. Si può dimostrare (teoria della elasticità) che, se  $M(x)$  indica il momento flettente del listello,  $E$  il modulo di Young del materiale ed  $I$  il momento di inerzia del segmento di spline considerato rispetto ad un opportuno punto, sussiste la relazione:

$$\frac{y''(x)}{(1 + (y'(x))^2)^{3/2}} = \frac{1}{EI} M(x) ; \quad x \in [x_i, x_{i+1}]$$

Tale espressione, nell'ipotesi che il listello sia soggetto a piccole deformazioni ( $y' \approx 0$ ) e che i pesi siano tra loro abbastanza vicini, può essere semplificata nella:

$$y''(x) = \frac{M}{EI} x \quad x \in [x_i, x_{i+1}]$$

dove  $M$  è una costante. Integrando l'espressione così ottenuta rispetto ad  $x$  due volte, si ricava che la  $y = y(x)$  è un polinomio di terzo grado tra due pesi consecutivi.

Poichè, a causa dei pesi, vi possono essere dei salti nella velocità di variazione della curvatura del listello, si può attribuire alla  $y(x)$  la continuità delle derivate prima e seconda, e al più una discontinuità della derivata terza nei punti in cui sono posizionati i pesi. Una formalizzazione matematica di quanto detto viene data successivamente (cfr. Teorema 3.5)

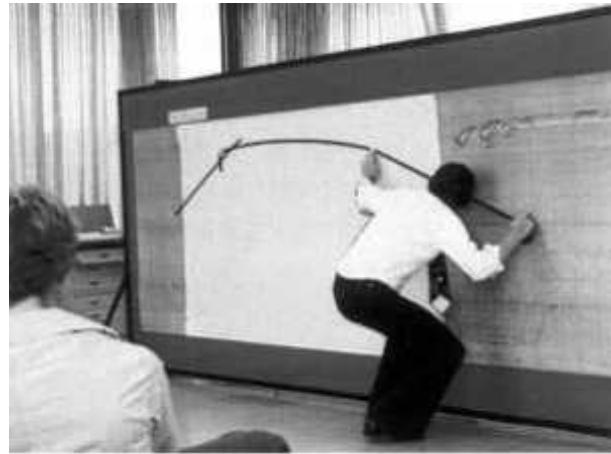


Figura 3.28: Lo strumento spline utilizzato da un disegnatore



Figura 3.29: Utilizzo dello strumento spline in un cantiere navale

Analizziamo ora quali sono le condizioni che definiscono una spline. Innanzitutto bisogna fissare l'insieme dei nodi :

$$K = \{x_1 < x_2 < \cdots < x_n\}$$



Figura 3.30: Dal *listello* alla realizzazione di un progetto

La prima caratteristica di una spline è che tra due *nodi* consecutivi essa sia rappresentata da un polinomio di un grado fissato. Nel caso di spline di primo grado, il polinomio deve essere al più di primo grado.

♣ **Esempio 3.23.** Assegnati i nodi:

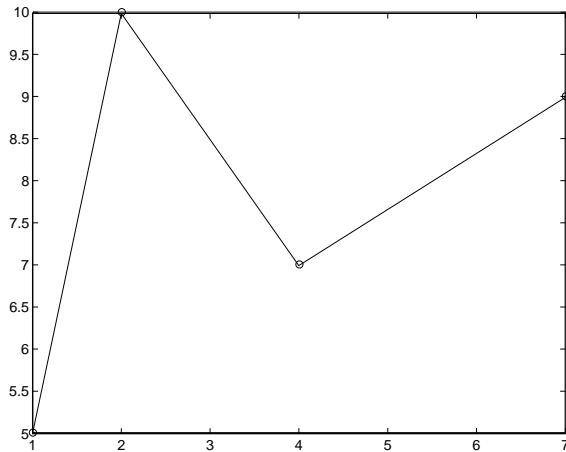
$$x_1 = 1, \quad x_2 = 2, \quad x_3 = 4, \quad x_4 = 7,$$

si vuole definire una funzione **spline di primo grado (del secondo ordine)**, relativa a tali nodi. Un modo è quello di procedere intervallo per intervallo. Nel caso di funzioni spline di primo grado, in ogni intervallo la funzione spline deve essere rappresentata da un polinomio di primo grado:

$$s(x) \equiv \begin{cases} p_1(x) & x \in [1, 2] \\ p_2(x) & x \in [2, 4] \\ p_3(x) & x \in [4, 7] \end{cases}$$

con  $p_1(x), p_2(x), p_3(x)$  polinomi di primo grado.

Un esempio di spline di primo grado è illustrato nella Figura 3.31.

Figura 3.31: Funzione spline lineare relativa ai nodi  $\{1, 2, 4, 7\}$ 

Quindi, una **spline lineare** è un **polinomio lineare a tratti** che congiunge i punti assegnati. ♣

### Definizione 3.14. (Spline lineare)

*Fissato l'insieme*

$$K = \{x_0 < x_1 < x_2 < \dots < x_n < x_{n+1}\},$$

*con gli  $x_i$ ,  $i = 1, \dots, n$  appartenenti all'asse reale e  $x_0 = -\infty$  e  $x_{n+1} = +\infty$ , una funzione  $s(x)$ , definita su tutto l'asse reale, è una **spline lineare** se:*

- $s(x) \equiv p_i(x) \in \Pi_1$  per  $x \in [x_i, x_{i+1}]$ ,  $i = 0, \dots, n$ ;
- $s(x)$  è una funzione continua in  $(x_0, x_{n+1})$  ( $s(x) \in C^0((-\infty, +\infty))$ ).

Indicato con  $S_1(K)$  l'insieme delle spline lineari costruite sull'insieme  $K$ , si ha:

$$S_1(K) \supset \Pi_1$$

ovvero un polinomio di primo grado è una spline lineare, viceversa non è vero che una spline lineare sia un polinomio di primo grado. Cosa si può dire sulla regolarità di questa funzione?

Come si osserva anche dal grafico in Figura 3.31, la regolarità di una spline lineare riguarda solo la continuità della funzione. Questa è la massima regolarità che possiamo ottenere perché dal momento che in ciascun tratto essa è rappresentata da un polinomio di primo grado la cui derivata prima fornisce il coefficiente angolare di ciascun segmento di retta e quindi la sua pendenza, volendo imporre la continuità della derivata prima nei nodi di raccordo, dovremmo imporre che la pendenza di ciascun segmento di retta nei

nodi di raccordo sia uguale, e cioè dovremmo imporre che i tratti di rette che definiscono la spline su ciascun sottointervallo appartengano alla medesima retta. Chiaramente, ciò non è possibile (a meno che i punti non siano tutti allineati).

Osserviamo, infine, che una spline lineare, nell'intervallo  $[x_1, x_n]$ , coincide con il polinomio interpolante a tratti, costruito sui nodi  $x_1, \dots, x_n$ .

♣ **Esempio 3.24.** Si vuole disegnare una **spline cubica** sullo stesso insieme di nodi dell'esempio 3.23. In questo caso, in ciascun sottointervallo  $[x_1, x_2] \equiv [1, 2]$ ,  $[x_2, x_3] \equiv [2, 4]$ ,  $[x_3, x_4] \equiv [4, 7]$  dobbiamo definire un polinomio di terzo grado, che indichiamo con  $p_i, i = 1, 2, 3$ , e la spline cubica è del tipo:

$$s(x) \equiv \begin{cases} p_1(x) & x \in [1, 2] \\ p_2(x) & x \in [2, 4] \\ p_3(x) & x \in [4, 7] \end{cases}$$

con  $p_1(x), p_2(x), p_3(x)$  polinomi di grado al più tre. Quanto detto garantisce che:

$$p_1(x_2) = p_2(x_2); \quad p_2(x_3) = p_3(x_3)$$

per definizione di funzione, cioè i polinomi che definiscono la spline in ciascun sottointervallo devono *raccordarsi* nei nodi comuni a due sottointervalli consecutivi. Questa condizione garantisce di conseguenza la continuità della funzione  $s(x)$  su tutto l'intervallo  $[x_1, x_4]$ . Vediamo se possiamo ottenere una regolarità maggiore dalla spline cubica. ♣

Definiamo, ora, una funzione spline cubica. Come vedremo una funzione spline cubica è una funzione polinomiale cubica a tratti che soddisfa ad opportune condizioni di regolarità nei nodi.

### Definizione 3.15. (Spline cubica)

Sia

$$K = \{x_0 < x_1 < x_2 < \dots < x_n < x_{n+1}\},$$

con gli  $x_i, i = 1, \dots, n$  appartenenti all'asse reale e  $x_0 = -\infty$  e  $x_{n+1} = +\infty$ , una funzione  $s(x)$ , definita su tutto l'asse reale, è una **spline cubica** se:

- $s(x) \equiv p_i(x) \in \Pi_3$  per  $x \in [x_i, x_{i+1}], i = 0, \dots, n$
- $s(x), s'(x), s''(x)$  sono funzioni continue in  $(-\infty, +\infty)$  ( $s(x) \in C^2((-\infty, +\infty))$ )<sup>29</sup>.

---

<sup>29</sup>In effetti la spline cubica è una caso particolare di *spline di grado m* (di ordine  $m + 1$ ).

### Definizione 3.16. (Spline di grado m (di ordine $m + 1$ ))

Queste funzioni sono quelle più utilizzate nelle applicazioni che comportano problemi di interpolazione.

Indicato con  $S_3(K)$  l'insieme delle spline cubiche relative all'insieme  $K$ , si ha che:

$$S_3(K) \supset \Pi_3$$

Osserviamo innanzitutto che dal momento che ogni tratto di curva è un polinomio di terzo grado, la derivata terza di  $s(x)$  in ciascun sottointervallo è una costante, in generale diversa tra un tratto e l'altro<sup>30</sup>.

Richiedere che la derivata terza sia una funzione continua su tutto l'intervallo dei nodi significa richiedere che il coefficiente del monomio di grado massimo, relativo a ciascun polinomio  $p_i$ , sia lo stesso per ogni polinomio  $p_i$ . D'altra parte, per la continuità della derivata prima e seconda, anche gli altri coefficienti di  $p_i$  risultano uguali tra loro e quindi dovrebbe esistere un unico polinomio di terzo grado passante per i nodi assegnati, qualunque sia il loro numero (ovvero  $p_i(x) \equiv p(x) \quad \forall i = 1, \dots, n$ ). Chiaramente, ciò in generale non è possibile, pertanto possiamo richiedere al più la continuità della derivata prima e della derivata seconda.

Nella Figura 3.32 è rappresentata una spline cubica relativa all'insieme dei nodi  $K = \{1, 2, 4, 7\}$ .

---

Sia

$$K = \{x_0 < x_1 < x_2 < \dots < x_n < x_{n+1}\},$$

con gli  $x_i$ ,  $i = 1, \dots, n$  appartenenti all'asse reale e  $x_0 = -\infty$  e  $x_{n+1} = +\infty$ , una funzione  $s(x)$ , definita su tutto l'asse reale, è una **spline di grado m** se:

- $s(x) \equiv p_i(x) \in \Pi_m$  per  $x \in [x_i, x_{i+1}]$ ,  $i = 0, \dots, n$ ;
- $s(x) \in C^{m-1}((-\infty, +\infty))$ : su tutto l'asse reale la funzione  $s(x)$  è continua con le sue derivate fino all'ordine  $m-1$ .

Indicato con  $S_m(K)$  l'insieme delle spline di grado  $m$  costruite sull'insieme  $K$ , si ha che:

$$S_m(K) \supset \Pi_m$$

<sup>30</sup>Se  $a_i$  indica il coefficiente del monomio di grado massimo nella rappresentazione del polinomio  $p_i$  di terzo grado che descrive la spline nel sottointervallo  $[x_i, x_{i+1}]$ , la derivata terza è  $6a_i$ .

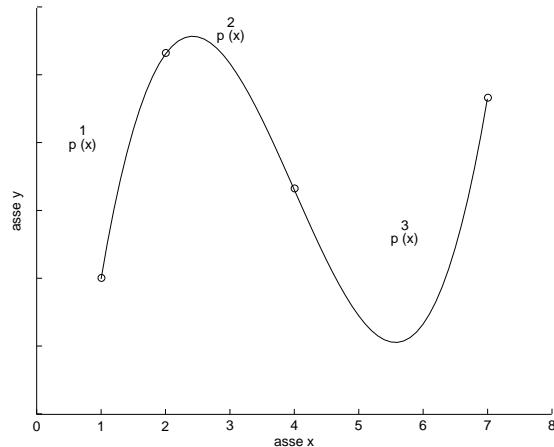


Figura 3.32: Funzione spline cubica sui nodi  $\{1, 2, 4, 7\}$

Affrontiamo ora il problema della costruzione di una spline, in particolare quello della spline cubica.

### 3.4.2 Un metodo costruttivo per la spline cubica naturale interpolante

In questo paragrafo vogliamo descrivere un metodo costruttivo per la spline cubica interpolante secondo Lagrange. Come vedremo, il problema di interpolazione ammette infinite soluzioni. Per garantire l'unicità della soluzione verrà introdotta la spline cubica *naturale*.

L'idea di base è costruire la spline intervallo per intervallo<sup>31</sup>. Si consideri, dunque, il problema di interpolazione seguente:

#### Problema di interpolazione mediante spline cubica

Fissati  $n$  nodi  $x_i$ ,  $i = 1, \dots, n$  e fissati  $n$  valori corrispondenti  $y_i$ ,  $i = 1, \dots, n$ , si vuole costruire una funzione spline cubica,  $s \in S_3(K)$ , con

<sup>31</sup>Esiste una motivazione strettamente computazionale nella scelta di costruire la spline intervallo per intervallo. Infatti, si può dimostrare che una funzione spline, su tutto il suo dominio di definizione, ammette una rappresentazione univoca in una opportuna base di funzioni. In tal caso, determinare la spline equivale a determinarne i coefficienti nella base. Purtroppo il sistema lineare a cui tale rappresentazione conduce è di tipo Vandermonde, quindi fortemente mal condizionato. È necessario, pertanto, derivare un metodo costruttivo alternativo, come quello che utilizza l'informazione che la spline in ogni sottointervallo è rappresentata da un polinomio.

$$K = \{x_0 < x_1 < x_2 < \dots < x_n < x_{n+1}\},$$

dove

$$x_0 = -\infty \quad \text{e} \quad x_{n+1} = +\infty,$$

tale che  $s(x_i) = y_i$ ,  $i = 1, \dots, n$ .

Si consideri l'intervallo  $[x_{i-1}, x_i]$  e sia  $p_{i-1}$  il polinomio che rappresenta la spline in tale intervallo. Se si conoscessero i valori della derivata prima di tale polinomio negli estremi del sottointervallo si potrebbe porre un problema di interpolazione di Hermite, del tipo:

nodi	$x_{i-1}$	$x_i$
condizioni	$y_{i-1}^0 = p_{i-1}(x_{i-1})$	$y_i^0 = p_{i-1}(x_i)$
	$y_{i-1}^1 = p'_{i-1}(x_{i-1})$	$y_i^1 = p'_{i-1}(x_i)$

e costruire il polinomio interpolante di Hermite, espresso dalla seguente formula di Newton:

$$\begin{aligned} p_{i-1}(x) = & y[x_{i-1}] + \lambda_{i-1}(x - x_{i-1}) + y[x_{i-1}, x_{i-1}, x_i](x - x_{i-1})^2 + \\ & + y[x_{i-1}, x_{i-1}, x_i, x_i](x - x_{i-1})^2(x - x_i) \end{aligned}$$

dove abbiamo posto:

$$\lambda_i = p'_{i-1}(x_i)$$

Si osservi che i coefficienti di  $p_{i-1}$  sono:

- $y_{i-1} = y[x_{i-1}]$ , quantità nota perché  $x_{i-1}$  è un nodo di interpolazione;
- $\lambda_{i-1} = y[x_{i-1}, x_{i-1}]$ , incognita da determinare;
- $y[x_{i-1}, x_{i-1}, x_i] = \frac{y[x_{i-1}, x_{i-1}] - y[x_{i-1}, x_i]}{x_{i-1} - x_i} = \frac{\lambda_{i-1} - y[x_{i-1}, x_i]}{x_{i-1} - x_i}$ , quantità calcolabile se noti i  $\lambda_i$ ,  $i = 1, \dots, n$ ;
- $y[x_{i-1}, x_{i-1}, x_i, x_i] = \frac{y[x_{i-1}, x_{i-1}, x_i] - y[x_{i-1}, x_i, x_i]}{x_{i-1} - x_i} = \frac{-y[x_{i-1}, x_i] + \lambda_{i-1}}{(x_{i-1} - x_i)^2} - \frac{y[x_{i-1}, x_i] - \lambda_i}{(x_{i-1} - x_i)^2}$ , quantità calcolabile se noti i  $\lambda_i$ ,  $i = 1, \dots, n$ .

In definitiva, i coefficienti di  $p_{i-1}$  sono tutti riconducibili alle quantità  $\lambda_i$ ,  $i = 1, \dots, n$ . Il problema è dunque calcolare queste quantità.

Dimostriamo la seguente:

**Proposizione 3.3.** Siano  $\xi$  ed  $\eta$  due numeri reali, distinti. Sia  $p \in \Pi_3$ . Allora:

$$p''(\xi) = \frac{2}{\xi - \eta} \{2p'(\xi) + p'(\eta) - 3y[\xi, \eta]\}$$

**Dimostrazione** Scriviamo il polinomio di Taylor di  $p$  di punto iniziale  $\eta$ , valutato in  $\xi$ :

$$p(\xi) = p(\eta) + p'(\eta)(\xi - \eta) + p''(\eta) \frac{(\xi - \eta)^2}{2} + c \cdot \frac{(\xi - \eta)^3}{6} \quad (c = \text{costante})$$

Allo stesso modo, scriviamo il polinomio di Taylor di  $p''$  di punto iniziale  $\xi$ , valutato in  $\eta$ :

$$p''(\eta) = p''(\xi) + (\eta - \xi) \cdot c$$

Sostituendo  $p''(\eta)$  nell'espressione del polinomio di Taylor di  $p$  segue :

$$y[\xi, \eta] = \frac{p(\xi) - p(\eta)}{\xi - \eta} = p'(\eta) + p''(\xi) \frac{(\xi - \eta)}{2} - c \cdot \frac{(\xi - \eta)^2}{2} + c \cdot \frac{(\xi - \eta)^2}{6}$$

Da cui, ricavando  $p''(\xi)$ , si ottiene:

$$p''(\xi) = \frac{2}{(\xi - \eta)} \{y[\xi, \eta] - p'(\eta) + \frac{1}{3}c \cdot (\xi - \eta)^2\}$$

Invertendo i ruoli di  $\xi$  e  $\eta$  si ha anche:

$$p(\eta) = p(\xi) + p'(\xi)(\eta - \xi) + p''(\xi) \frac{(\eta - \xi)^2}{2} + c \cdot \frac{(\eta - \xi)^3}{6}$$

da cui

$$y[\xi, \eta] = \frac{p(\eta) - p(\xi)}{\eta - \xi} = p'(\xi) + p''(\xi) \frac{(\eta - \xi)}{2} + c \cdot \frac{(\eta - \xi)^2}{6}$$

da cui:

$$2y[\xi, \eta] - 2p'(\xi) - p''(\xi)(\eta - \xi) = c \cdot \frac{(\eta - \xi)^2}{3}$$

Quindi sostituendo tale espressione in  $p''(\xi)$ , si ha la tesi, ovvero:

$$p''(\xi) = \frac{2}{\xi - \eta} \{2p'(\xi) + p'(\eta) - 3y[\xi, \eta]\}$$

■

Applichiamo la proposizione precedente al polinomio  $p_{i-1}$ , avendo scelto

$$\xi = x_i \quad \text{e} \quad \eta = x_{i-1}.$$

Si ha:

$$s''(x_i) = p''_{i-1}(x_i) = \frac{2}{h_i} \{2\lambda_i + \lambda_{i-1} - 3y[x_i, x_{i-1}]\} \quad i = 2, \dots, n$$

Applicando, di nuovo, la proposizione precedente con  $\xi = x_i$  e  $\eta = x_{i+1}$ , si ha:

$$s''(x_i) = p''_i(x_i) = \frac{-2}{h_{i+1}}\{2\lambda_i + \lambda_{i+1} - 3y[x_i, x_{i+1}]\} \quad i = 1, \dots, n-1$$

Ricordando che la spline cubica è una funzione di classe  $C^2$ , nell'intervallo dei nodi, segue che le derivate seconde devono essere continue nei nodi di raccordo cioè:

$$p''_{i-1}(x_i) = p''_i(x_i), \quad i = 2, \dots, n-1$$

ovvero:

$$\frac{2}{h_i}\{2\lambda_i + \lambda_{i-1} - 3y[x_i, x_{i-1}]\} + \frac{2}{h_{i+1}}\{2\lambda_i + \lambda_{i+1} - 3y[x_i, x_{i+1}]\} = 0$$

da cui, ponendo:

$$h_i = x_i - x_{i-1}, \quad i = 2, \dots, n$$

$$\frac{h_{i+1}}{h_i \cdot h_{i+1}}\{2\lambda_i + \lambda_{i-1} - 3y[x_i, x_{i-1}]\} + \frac{h_i}{h_i \cdot h_{i+1}}\{2\lambda_i + \lambda_{i+1} - 3y[x_i, x_{i+1}]\} = 0 \quad (3.41)$$

Posto:

$$\delta_i = \frac{h_i}{h_i + h_{i+1}}$$

e moltiplicando la (3.41) per

$$\frac{h_i \cdot h_{i+1}}{h_i + h_{i+1}}$$

si ha:

$$(1 - \delta_i)\lambda_{i-1} + 2\lambda_i + \delta_i\lambda_{i+1} = 3\{(1 - \delta_i)y[x_i, x_{i-1}] + \delta_i y[x_i, x_{i+1}]\} \\ i = 2, \dots, n-1$$

Si è ottenuto un sistema di  $n-2$  equazioni in  $n$  incognite,  $\lambda_i$ ,  $i = 1, \dots, n$ . Essendo il numero di incognite strettamente maggiore del numero di equazioni, possono esistere infinite soluzioni. In altri termini per costruire la spline interpolante relativa ad un insieme di nodi, le condizioni di regolarità non bastano ad individuarla univocamente. È necessario imporre condizioni aggiuntive che specializzino il tipo di spline da costruire (ossia a restringano l'insieme delle soluzioni ammissibili). Una delle spline più utilizzate nelle applicazioni è la **spline naturale**<sup>32</sup>.

---

<sup>32</sup>

### Definizione 3.17. (Spline naturale di grado m)

Una spline  $s$  di grado dispari  $m = 2j-1$ , relativa all'insieme dei nodi  $K$ , è una **spline naturale** se,

La denominazione *naturale*, assegnata a questo tipo particolare di funzione, è dovuta al fatto che lo strumento spline al di fuori dei pesi che lo vincolano, e cioè all'esterno dell'intervallo  $[x_1, x_n]$ , tende naturalmente ad assumere la forma di linea retta e, quindi, la sua derivata seconda deve essere ivi nulla.

Allora, per le condizioni aggiuntive che definiscono la spline cubica naturale interpolante sono:

$$\frac{d^2}{dx^2}s(x_1) = \frac{d^2}{dx^2}s(x_n) = 0,$$

da cui si ha:

$$s''(x_1) = 0 \rightarrow \frac{-2}{h_2}\{2\lambda_1 + \lambda_2 - 3y[x_1, x_2]\} = 0 \Leftrightarrow 2\lambda_1 + \lambda_2 = 3y[x_1, x_2]$$

e:

$$s''(x_n) = 0 \rightarrow \frac{2}{h_{n+1}}\{2\lambda_n + \lambda_{n-1} - 3y[x_n, x_{n-1}]\} = 0 \Leftrightarrow 2\lambda_n + \lambda_{n-1} = 3y[x_n, x_{n-1}]$$

Si è ottenuto il sistema:

$$\begin{cases} 2\lambda_1 + \lambda_2 = 3y[x_1, x_2] \\ (1 - \delta_i)\lambda_{i-1} + 2\lambda_i + \delta_i\lambda_{i+1} = 3\{(1 - \delta_i)y[x_i, x_{i-1}] + \delta_i y[x_i, x_{i+1}]\} \\ 2\lambda_n + \lambda_{n-1} = 3y[x_n, x_{n-1}] \end{cases} \quad i = 2, n-1 \quad (3.42)$$

che, risolto, fornisce il vettore  $\lambda_i$ ,  $i = 1, \dots, n$ .

Introdotta la matrice tridiagonale

$$A = \begin{pmatrix} 2 & 1 & 0 & 0 & \dots & 0 \\ (1 - \delta_2) & 2 & \delta_2 & 0 & \dots & 0 \\ 0 & (1 - \delta_3) & 2 & \delta_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & (1 - \delta_{n-1}) & 2 & \delta_{n-1} \\ 0 & 0 & 0 & \dots & 1 & 2 \end{pmatrix} \quad (3.43)$$

nonché la

$$B = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ (1 - \delta_2) & \delta_2 & 0 & \dots & 0 \\ 0 & (1 - \delta_3) & \delta_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & (1 - \delta_{n-1}) & \delta_{n-1} \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} \quad (3.44)$$

in ciascuno dei due intervalli

$$(-\infty, x_1), (x_n, +\infty)$$

coincide con un polinomio di grado minore o uguale a  $j-1$ . Ciò significa che

$$s^{(h)}(x_1) = s^{(h)}(x_n) = 0, \quad h = j, j+1, \dots, 2j-2.$$

La spline cubica naturale soddisfa tali condizioni con  $j=2$ .

ed il vettore

$$\Delta f = \begin{bmatrix} y[x_1, x_2] \\ y[x_2, x_3] \\ \vdots \\ y[x_{n-1}, x_n] \end{bmatrix} \quad (3.45)$$

il sistema (3.42) si traduce, in termini matriciali, in:

$$A\lambda = 3B\Delta f \quad (3.46)$$

Poiché la matrice  $A$  è tridiagonale, tale sistema può essere risolto utilizzando l'algoritmo di eliminazione di Gauss specializzato per matrici tridiagonali. Analizziamo, ora, le caratteristiche della matrice  $A$ .

Il determinante della matrice  $A$  risulta essere uguale al prodotto degli autovalori e, per il teorema di Gershgorin (Teorema B.6, paragrafo B.5.1), gli autovalori di  $A$  cadono tutti nel cerchio di centro 2 e raggio unitario; essi risultano, dunque, tutti diversi da zero, per cui è

$$\det(A) \neq 0$$

cioè esiste una ed una sola soluzione del sistema (3.46).

Per quanto riguarda il *condizionamento*, osservando che:

$$\mu_1(A) = \|A\|_1 \|A^{-1}\|_1 \leq n \cdot \max_{1 \leq i, j \leq n} |a_{ij}| \cdot \|A^{-1}\|_1$$

e che

$$\|A^{-1}\|_1 \leq \left( \min_{1 \leq i, j \leq n} |a_{ij}| \right)^{-1}$$

si ha

$$\begin{aligned} \mu_1(A) &\leq n \cdot \underbrace{\max_{1 \leq i, j \leq n} |a_{ij}|}_{2} \cdot \left( \min_{1 \leq i, j \leq n} |a_{ij}| \right)^{-1} = \\ &= 2 \cdot n \cdot \min_{1 \leq i, j \leq n} \left\{ \frac{h_i}{h_{i+1} + h_i}, \frac{h_{i+1}}{h_{i+1} + h_i} \right\}^{-1} \end{aligned}$$

Se

$$h_i = h \quad \forall i = 1, \dots, n$$

si ha

$$\begin{aligned} \mu_1(A) &\leq 2 \cdot n \cdot \left( \min \left\{ \frac{h}{2h}, \frac{h}{2h} \right\} \right)^{-1} = \\ &= 2 \cdot n \cdot \left( \frac{1}{2} \right)^{-1} \end{aligned}$$

ovvero

$$\mu_1(A) \leq 4n$$

Si può dedurre, allora, che la matrice  $A$  è ben condizionata, a meno che i nodi non siano "troppo" vicini o "troppo" lontani.

Inoltre, nell'applicazione dell'algoritmo di eliminazione di Gauss, *non è necessario il pivoting*. Infatti, essendo

$$\forall i = 1, \dots, n \quad 0 < \delta_i < 1 \quad \text{e} \quad 0 < 1 - \delta_i < 1$$

osserviamo, innanzitutto, che la matrice  $A$  risulta essere a diagonale dominante<sup>33</sup> Esaminiamo, poi, i passi del suddetto algoritmo, con riferimento alla matrice  $A$ . Indichiamo con

$$\underline{d} = (d_1, d_2, \dots, d_{n-1})$$

il vettore in cui memorizziamo la diagonale superiore di  $A$ , cioè

$$d_1 = 1, \quad d_2 = \delta_2, \dots, d_{n-1} = \delta_{n-1}$$

e con

$$\underline{e} = (e_2, e_3, \dots, e_n)$$

il vettore in cui memorizziamo la diagonale inferiore di  $A$ , cioè

$$e_2 = 1 - \delta_2, \quad e_3 = 1 - \delta_3, \dots, e_n = 1 - \delta_n$$

### • Passo 1

data la struttura tridiagonale della matrice  $A$ , l'unico moltiplicatore non nullo è quello relativo alla seconda riga:

$$m_{21} = \frac{e_2}{a_{11}} = \frac{1 - \delta_2}{2}$$

e, quindi,

$$0 < 1 - \delta_2 < 1 \Rightarrow 0 < m_{21} < \frac{1}{2}$$

inoltre

$$\begin{aligned} a_{22}^{(1)} &= a_{22} - m_{21}d_1 = 2 - \underbrace{\frac{1 - \delta_2}{2}}_{< \frac{1}{2}} \cdot 1 > 2 - \frac{1}{2} = \frac{3}{2} \\ d_2^{(1)} &= d_2 - m_{21} \underbrace{a_{13}}_{=0} = \delta_2 < 1 \end{aligned}$$

---

<sup>33</sup>Per la definizione di *matrice a diagonale dominante* si rimanda al Cap. 2

• **Passo 2**

l'unico moltiplicatore non nullo è quello relativo alla terza riga:

$$m_{32} = \frac{e_3^{(1)}}{a_{22}^{(1)}} = \frac{1 - \delta_3}{2 - \frac{1 - \delta_2}{2}}$$

ma

$$\begin{aligned} 0 < 1 - \delta_3 < 1 \\ 2 - \frac{1 - \delta_2}{2} > \frac{3}{2} \end{aligned} \Rightarrow 0 < \frac{1}{2 - \frac{1 - \delta_2}{2}} < \frac{2}{3} \quad \left. \right\} \Rightarrow 0 < m_{32} < \frac{2}{3}$$

ed, inoltre,

$$\begin{aligned} a_{33}^{(2)} &= a_{33}^{(1)} - m_{32} d_2^{(1)} > 2 - \delta_2 \frac{2}{3} > 2 - \frac{2}{3} = \frac{4}{3} > 1 \\ d_3^{(2)} &= d_3^{(1)} - m_{32} \underbrace{a_{24}^{(1)}}_{=0} = \delta_3 < 1 \end{aligned}$$

⋮

⋮

• **Passo  $n - 2$**

l'unico moltiplicatore non nullo è quello relativo alla  $(n - 1)$ -esima riga:

$$m_{n-1,n-2} = \frac{e_{n-1}^{(n-3)}}{a_{n-2,n-2}^{(n-3)}} = \frac{1 - \delta_{n-1}}{a_{n-2,n-2}^{(n-3)}}$$

ma

$$\begin{aligned} 0 < 1 - \delta_{n-1} < 1 \\ a_{n-2,n-2}^{(n-3)} > \frac{n-1}{n-2} \end{aligned} \Rightarrow 0 < \frac{1}{a_{n-2,n-2}^{(n-3)}} < \frac{n-2}{n-1} \quad \left. \right\} \Rightarrow 0 < m_{n-1,n-2} < \frac{n-2}{n-1}$$

ed, inoltre,

$$\begin{aligned} a_{n-1,n-1}^{(n-2)} &= a_{n-1,n-1}^{(n-3)} - m_{n-1,n-2} d_{n-2}^{(n-3)} = 2 - \underbrace{\left( \frac{1 - \delta_{n-1}}{a_{n-2,n-2}^{(n-3)}} \right)}_{< \frac{n-2}{n-1}} \delta_{n-2} > \\ 2 - \frac{n-2}{n-1} &> \frac{n}{n-1} \\ d_{n-1}^{(n-2)} &= d_{n-1}^{(n-3)} - m_{n-1,n-2} \underbrace{a_{n-2,n}^{(n-3)}}_{=0} = \delta_{n-1} < 1 \end{aligned}$$

• **Passo  $n - 1$**

l'unico moltiplicatore non nullo è quello relativo alla  $n$ -esima riga:

$$m_{n,n-1} = \frac{e_n^{(n-2)}}{a_{n-1,n-1}^{(n-2)}} = \frac{1}{a_{n-1,n-1}^{(n-2)}}$$

ma

$$a_{n-1,n-1}^{(n-2)} > \frac{n}{n-1} \Rightarrow \frac{1}{a_{n-1,n-1}^{(n-2)}} < \frac{n-1}{n}$$

e si ha

$$\begin{aligned} a_{n,n}^{(n-1)} &= 2 - m_{n,n-1} d_{n-1}^{(n-2)} = 2 - \underbrace{\left( \frac{1}{a_{n-1,n-1}^{(n-2)}} \right)}_{< \frac{n-1}{n}} \delta_{n-1} > 2 - \frac{n-1}{n} \cdot 1 \\ &= \frac{n+1}{n} \end{aligned}$$

Sostanzialmente, all' $i$ -esimo passo dell'algoritmo ( $i = 1, \dots, n-1$ ) viene effettuata una sola divisione, nel calcolo del moltiplicatore, in cui il divisore  $D_i$  risulta

$$2 \geq D_i > \frac{i+1}{i} > 1$$

Concludendo si può osservare che l'elemento diagonale, ad ogni passo, è sempre il massimo, in modulo, degli elementi appartenenti alla sua stessa colonna. Pertanto si può utilizzare l'algoritmo di eliminazione di Gauss senza strategia del pivoting.

Riepilogando quanto detto, il procedimento per la costruzione e valutazione, in un valore fissato  $\tilde{x}$ , della funzione spline cubica naturale interpolante un insieme di punti assegnati  $(x_i, y_i), i = 1, \dots, n$  consiste in:

- Passo 1):** costruzione della matrice  $A$  e del vettore dei termini noti  $3B\Delta f$ ;
- Passo 2):** risoluzione del sistema  $A\lambda = 3B\Delta f$ ;
- Passo 3):** determinazione dell'intervallo  $[x_i, x_{i+1}]$  a cui  $\tilde{x}$  appartiene;
- Passo 4):** calcolo dei coefficienti del polinomio interpolante di Hermite,  
nell' intervallo  $[x_i, x_{i+1}]$ ;
- Passo 5):** valutazione in  $\tilde{x}$ .

Uno schema della procedura per la costruzione della spline cubica naturale interpolante è il seguente:

```

procedure Spline_Nat_Int(in: n, x, y,  $\tilde{x}$ , out: s)

/# SCOPO: valutazione, in un fissato punto, della spline naturale cubica
interpolante, costruita su un insieme di punti assegnati

/# SPECIFICHE DEI PARAMETRI:
/# PARAMETRI DI INPUT:

var: n      : intero      { numero dei punti di }
                                { interpolazione }

var: x(n)  : array di reali { nodi di interpolazione }

var: y(n)  : array di reali { ordinate corrispondenti }
                                { ai nodi di interpolazione }

var:  $\tilde{x}$     : reale       { punto di valutazione }
                                { della spline }

/# PARAMETRI DI OUTPUT:

var: s      : reale       { valore della spline in  $\tilde{x}$  }

```

Procedura 3.9: Algoritmo per la costruzione e valutazione  
della spline naturale cubica interpolante - continua

```

/# INIZIO ISTRUZIONI:

call ordinamento( $n, x, y$ ) { ordinamento dei nodi  $x_i$  e }
{ dei valori corrispondenti  $y_i$ ; }

call costruzione( $n, x, y, dp, di, ds, df$ ) { costruzione del vettore dei termini noti}
{  $(3B\Delta f)$  e della matrice dei}
{ coefficienti del sistema  $A\lambda = 3B\Delta f$ ; }

call risoluzione( $n, dp, di, ds, df, lambda$ ) { risoluzione del sistema costruito}
{ mediante l'algoritmo di Gauss,}
{ specializzato per matrici tridiagonali; }

call valutazione( $n, x, y, lambda, \tilde{x}, s$ ) { ricerca binaria dell'intervallo di}
{ appartenenza del punto di valutazione}
{  $\tilde{x}$  e valutazione del }
{ polinomio che rappresenta la}
{ spline nell'intervallo  $[x_i, x_{i+1}]$ ,}
{ utilizzando la formula di Newton per il}
{ polinomio interpolante di Hermite}

end Spline_Nat_Int

```

Procedura 3.9: Algoritmo per la costruzione e valutazione  
della spline naturale cubica interpolante - fine

Nella procedura precedente i parametri  $dp, di, ds$  rappresentano, rispettivamente, la diagonale principale, la diagonale inferiore e la superiore di  $A$ ,  $df$  contiene il vettore delle differenze divise,  $\Delta f$  ed il vettore  $lambda$ , la soluzione del sistema  $A\lambda = 3B\Delta f$ . La formula di Newton per il polinomio interpolante di Hermite, di grado al più tre, tra due nodi consecutivi,  $x_i$  e  $x_{i+1}$ , da valutare in  $\tilde{x}$ , risulta la seguente:

$$\begin{aligned}
s &= a_0 + a_1(x - x_i) + a_2(x - x_i)^2 + a_3(x - x_i)^2(x - x_{i+1}) \\
a_0 &= y[x_i] \\
a_1 &= \lambda_i = y[x_i, x_i] \\
a_2 &= y[x_i, x_i, x_{i+1}] \\
a_3 &= y[x_i, x_i, x_{i+1}, x_{i+1}]
\end{aligned}$$

Detto  $\tilde{x}$  il punto di valutazione assegnato in input, se:

$\tilde{x} < x_1 \Rightarrow$  si valuta l'equazione della retta di coefficiente angolare  $\lambda_1$

$$s = y(1) + (\tilde{x} - x(1)) * \lambda_1$$

$\tilde{x} > x_n \Rightarrow$  si valuta l'equazione della retta di coefficiente angolare  $\lambda_n$

$$s = y(n) + (\tilde{x} - x(n)) * \lambda_n$$

dove  $\lambda_1$  e  $\lambda_n$  rappresentano i valori che la derivata prima assume, rispettivamente, nel primo e nell'ultimo nodo.

Infine, per quanto riguarda la complessità di tempo e, quindi l'*efficienza*, il numero di operazioni richieste dal metodo di eliminazione di Gauss per matrici tridiagonali è:

$$T(n) = \mathcal{O}(n) \text{ flop}$$

Concludendo, l'algoritmo per la costruzione e valutazione della spline cubica naturale interpolante un insieme di nodi assegnati, richiede una complessità di tempo che si ottiene, essenzialmente, stimando il numero di operazioni floating point coinvolte da ciascuna procedura, nonché il numero di confronti necessari per la ricerca binaria dell'intervallo di appartenenza del punto di valutazione:

- risoluzione di un sistema tridiagonale  $\Rightarrow T(n) = \mathcal{O}(n) \text{ flop}$
- ricerca binaria dell'intervallo di appartenenza del punto di valutazione  $\Rightarrow T(n) = \mathcal{O}(\log_2 n) \text{ confronti}$
- costruzione del polinomio interpolante  $\Rightarrow T(n) = \mathcal{O}(n^2) \text{ flop}$
- valutazione (algoritmo di Horner)  $\Rightarrow T(n) = \mathcal{O}(n) \text{ flop}$

♣ **Esempio 3.25.** Si vuole costruire una spline cubica interpolante 3 punti  $(x_i, y_i)$ ,  $i = 1, 2, 3$ . Costruiamo la spline intervallo per intervallo.

In ciascun intervallo fra due nodi consecutivi  $[x_1, x_2], [x_2, x_3]$  la spline  $s(x)$  deve essere un polinomio interpolante di Hermite, espresso dalla seguente formula di Newton:

$$\begin{aligned} s(x) \equiv p_i(x) &= \underbrace{y[x_i]}_{a_i} + \underbrace{\lambda_i}_{b_i} (x - x_i) + \underbrace{y[x_i, x_i, x_{i+1}]}_{c_i} (x - x_i)^2 + \\ &+ \underbrace{y[x_i, x_i, x_{i+1}, x_{i+1}]}_{d_i} (x - x_i)^2 (x - x_{i+1}) \in \Pi_3, \\ &x \in [x_i, x_{i+1}], \quad i = 1, 2 \end{aligned}$$

dove si pone:

$$\lambda_i = p'_i(x_i), \quad i = 1, 2, 3$$

In ciascuno dei due sottointervalli, dunque, occorre determinare i quattro coefficienti  $a_i, b_i, c_i, d_i$ , e quindi complessivamente, occorre determinare 8 coefficienti.

Applichiamo la proposizione 3.3 al polinomio  $p_{i-1}$ , avendo scelto  $\xi = x_i$  e  $\eta = x_{i-1}$ . Si ha:

$$s''(x_i) = p''_{i-1}(x_i) = \frac{2}{h_i} \{2\lambda_i + \lambda_{i-1} - 3y[x_i, x_{i-1}]\} \quad i = 2, 3$$

Applicando, di nuovo, la proposizione 3.3, con  $\xi = x_i$  e  $\eta = x_{i+1}$ , si ha:

$$s''(x_i) = p''_i(x_i) = \frac{-2}{h_{i+1}} \{2\lambda_i + \lambda_{i+1} - 3y[x_i, x_{i+1}]\} \quad i = 1, 2$$

Ricordando che la spline cubica è una funzione di classe  $C^2$ , nell'intervallo dei nodi, segue che le derivate seconde devono essere continue nei nodi di raccordo cioè:

$$p''_{i-1}(x_i) = p''_i(x_i), \quad i = 2$$

ovvero:

$$\frac{2}{h_i} \{2\lambda_i + \lambda_{i-1} - 3y[x_i, x_{i-1}]\} + \frac{2}{h_{i+1}} \{2\lambda_i + \lambda_{i+1} - 3y[x_i, x_{i+1}]\} = 0$$

da cui, ponendo:

$$\begin{aligned} h_i &= x_i - x_{i-1}, \quad i = 2, 3 \\ \frac{h_{i+1}}{h_i \cdot h_{i+1}} \{2\lambda_i + \lambda_{i-1} - 3y[x_i, x_{i-1}]\} + \frac{h_i}{h_i \cdot h_{i+1}} \{2\lambda_i + \lambda_{i+1} - 3y[x_i, x_{i+1}]\} &= 0 \end{aligned} \quad (3.47)$$

Posto:

$$\delta_i = \frac{h_i}{h_i + h_{i+1}}, \quad i = 2$$

e moltiplicando la (3.47) per

$$\frac{h_i \cdot h_{i+1}}{h_i + h_{i+1}}$$

si ha:

$$(1 - \delta_2)\lambda_1 + 2\lambda_2 + \delta_2\lambda_3 = 3\{(1 - \delta_2)y[x_2, x_1] + \delta_2y[x_2, x_3]\}, \quad i = 2$$

Si è ottenuto un sistema di *una* equazione in *tre* incognite,  $\lambda_i$ ,  $i = 1, 2, 3$ . Le due equazioni aggiuntive si ottengono imponendo che la spline sia una spline naturale, cioè:

$$\frac{d^2}{dx^2} s(x_1) = \frac{d^2}{dx^2} s(x_3) = 0,$$

da cui si ha:

$$s''(x_1) = 0 \rightarrow \frac{-2}{h_2} \{2\lambda_1 + \lambda_2 - 3y[x_1, x_2]\} = 0 \Leftrightarrow 2\lambda_1 + \lambda_2 = 3y[x_1, x_2]$$

e:

$$s''(x_3) = 0 \rightarrow \frac{2}{h_3} \{2\lambda_3 + \lambda_2 - 3y[x_3, x_2]\} = 0 \Leftrightarrow 2\lambda_3 + \lambda_2 = 3y[x_3, x_2]$$

Si è ottenuto il sistema

$$\begin{cases} 2\lambda_1 + \lambda_2 = 3y[x_1, x_2] \\ (1 - \delta_2)\lambda_1 + 2\lambda_2 + \delta_2\lambda_3 = 3\{(1 - \delta_2)y[x_2, x_1] + \delta_2y[x_2, x_3]\} \\ 2\lambda_3 + \lambda_2 = 3y[x_3, x_2] \end{cases} \quad (3.48)$$

che, risolto, fornisce il vettore  $\lambda_i$ ,  $i = 1, 2, 3$ .

Introdotta la matrice tridiagonale

$$A = \begin{pmatrix} 2 & 1 & 0 \\ (1 - \delta_2) & 2 & \delta_2 \\ 0 & 1 & 2 \end{pmatrix} \quad (3.49)$$

nonché la

$$B = \begin{pmatrix} 1 & 0 \\ (1 - \delta_2) & \delta_2 \\ 0 & 1 \end{pmatrix} \quad (3.50)$$

ed il vettore

$$\Delta f = \begin{bmatrix} y[x_1, x_2] \\ y[x_2, x_3] \end{bmatrix} \quad (3.51)$$

il sistema (3.48) si traduce, in termini matriciali, in:

$$A\lambda = 3B\Delta f \quad (3.52)$$

A partire dal vettore  $\lambda$ , soluzione del sistema (3.52), si determinano i coefficienti del polinomio interpolante di Hermite, nella formula di Newton, di grado al più tre, tra due nodi consecutivi,  $x_i$  e  $x_{i+1}$ ,

$$p_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^2(x - x_{i+1})$$

con

$$\begin{aligned} a_i &= y[x_i] \\ b_i &= \lambda_i = y[x_i, x_i] \\ c_i &= y[x_i, x_i, x_{i+1}] \\ d_i &= y[x_i, x_i, x_{i+1}, x_{i+1}] \end{aligned}$$

per  $i = 1, 2$ . ♣

### 3.4.3 Un altro metodo costruttivo per la spline cubica naturale interpolante

Assegnati i punti di coordinate  $(x_i, y_i)$ ,  $i = 1, \dots, n$ , per costruire<sup>34</sup> una spline cubica interpolante costruiamo il polinomio che rappresenta la spline intervallo per intervallo. In ciascun intervallo fra due nodi consecutivi

$$[x_1, x_2], \dots, [x_{n-1}, x_n]$$

---

<sup>34</sup>[7]

la spline  $s$  deve essere un polinomio di grado al più tre:

$$\begin{aligned} s(x) &\equiv p_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \in \Pi_3 \\ x &\in [x_i, x_{i+1}], \quad i = 1, \dots, n-1 \end{aligned}$$

In ciascun sottointervallo, dunque, occorre determinare i quattro coefficienti  $a_i, b_i, c_i, d_i$ , e quindi complessivamente occorre determinare  $4 \cdot (n-1)$  coefficienti.

Per come abbiamo definito la spline cubica, dobbiamo richiedere che  $s$  sia una funzione continua insieme con la sua derivata prima e seconda. Quindi dobbiamo imporre:

A)  $n-2$  condizioni di *regolarità* per  $s$  in  $x_i, i = 2, \dots, n-1$ :

$$p_{i-1}(x_i) = p_i(x_i), \quad i = 2, \dots, n-1 \quad (3.53)$$

B)  $n-2$  condizioni di *regolarità* per  $s'$  in  $x_i, i = 2, \dots, n-1$ :

$$\frac{d}{dx} p_{i-1}(x_i) = \frac{d}{dx} p_i(x_i), \quad i = 2, \dots, n-1 \quad (3.54)$$

C)  $n-2$  condizioni di *regolarità* per  $s''$  in  $x_i, i = 2, \dots, n-1$ :

$$\frac{d^2}{dx^2} p_{i-1}(x_i) = \frac{d^2}{dx^2} p_i(x_i), \quad i = 2, \dots, n-1 \quad (3.55)$$

Infine, affinchè  $s$  interpoli i punti assegnati, occorre imporre:

D)  $n$  condizioni di interpolazione:

$$s(x_i) = y_i, \quad i = 1, \dots, n \rightarrow \begin{cases} p_1(x_1) = y_1 \\ \vdots \\ p_{n-1}(x_{n-1}) = y_{n-1} \\ p_n(x_n) = y_n \end{cases} \quad (3.56)$$

In definitiva per determinare la spline cubica  $s$  interpolante gli  $n$  punti assegnati occorre determinare i  $4(n-1) = 4n-4$  coefficienti  $a_i, b_i, c_i, d_i, i = 1, \dots, n-1$  a partire dalle  $3 \cdot (n-2) + n = 4n-6$  condizioni (3.53, 3.54, 3.55, 3.56).

Tenendo conto che:

$$\frac{d}{dx} p_i(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2,$$

$$\frac{d^2}{dx^2} p_i(x) = 2c_i + 6d_i(x - x_i)$$

e che dalle condizioni di interpolazione (D) si ha:

$$\begin{cases} a_i = p_i(x_i) = y_i \quad i = 1, n-1 \\ a_n = a_{n-1} + b_{n-1}(x_n - x_{n-1}) + c_{n-1}(x_n - x_{n-1})^2 + d_{n-1}(x_n - x_{n-1})^3 \end{cases} \quad (3.57)$$

imponendo

$$h_i = x_{i+1} - x_i \quad i = 1, \dots, n-1$$

e rispettivamente le (A), (B) e (C) si ottiene:

$$\begin{cases} p_{i-1}(x_i) = p_i(x_i) \\ \frac{d}{dx}p_{i-1}(x_i) = \frac{d}{dx}p_i(x_i) \\ \frac{d^2}{dx^2}p_{i-1}(x_i) = \frac{d^2}{dx^2}p_i(x_i) \end{cases} \implies \begin{cases} a_{i-1} + b_{i-1}h_{i-1} + c_{i-1}h_{i-1}^2 + d_{i-1}h_{i-1}^3 = y_i \\ b_{i-1} + 2c_{i-1}h_{i-1} + 3d_{i-1}h_{i-1}^2 = b_i \\ 2c_{i-1} + 6d_{i-1}h_{i-1} = 2c_i \end{cases}$$

per  $i = 2, \dots, n-1$ ; da cui:

$$\begin{cases} b_{i-1}h_{i-1} + c_{i-1}h_{i-1}^2 + d_{i-1}h_{i-1}^3 = y_i - y_{i-1} \\ b_{i-1} - b_i + 2c_{i-1}h_{i-1} + 3d_{i-1}h_{i-1}^2 = 0 \\ 2c_{i-1} + 6d_{i-1}h_{i-1} - 2c_i = 0 \end{cases} \implies \begin{cases} b_{i-1} = \frac{y_i - y_{i-1}}{h_{i-1}} - c_{i-1}h_{i-1} - d_{i-1}h_{i-1}^2 \\ b_{i-1} - b_i + 2c_{i-1}h_{i-1} + 3d_{i-1}h_{i-1}^2 = 0 \\ d_{i-1} = \frac{1}{6h_{i-1}}(2c_i - 2c_{i-1}) \end{cases}$$

Per semplificare le notazioni poniamo:

$$\begin{aligned} \frac{d^2}{dx^2}p_i(x_i) &= \bar{y}_i, & i &= 1, \dots, n-1, \\ \frac{d^2}{dx^2}p_{n-1}(x_n) &= \bar{y}_n, \end{aligned} \tag{3.58}$$

con  $\bar{y}_i$  incognite da determinare. Poiché, si ha:

$$2c_i = \frac{d^2}{dx^2}p_i(x_i) = \bar{y}_i \quad i = 1, \dots, n-1$$

il sistema si riscrive nel modo seguente:

$$\begin{cases} b_{i-1} = \frac{y_i - y_{i-1}}{h_{i-1}} - \frac{1}{2}h_{i-1}\bar{y}_{i-1} - \frac{1}{6}h_{i-1}(\bar{y}_i - \bar{y}_{i-1}), \\ b_{i-1} + 2c_{i-1}h_{i-1} + 3d_{i-1}h_{i-1}^2 = b_i \\ d_{i-1} = \frac{1}{6h_{i-1}}(\bar{y}_i - \bar{y}_{i-1}), \end{cases}$$

In tale sistema, sostituendo le espressioni ottenute dei coefficienti  $b_i, b_{i-1}, c_{i-1}$  e  $d_{i-1}$  nella seconda equazione si ha:

$$h_{i-1}\bar{y}_{i-1} + 2(h_{i-1} + h_i)\bar{y}_i + h_i\bar{y}_{i+1} = 6 \left( \frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}} \right), \quad i = 2, \dots, n-1. \tag{3.59}$$

Si è ottenuto il sistema lineare di  $n-2$  equazioni nelle  $n$  incognite  $\bar{y}_i$ ,  $i = 1, \dots, n$ .

Essendo il numero di incognite strettamente maggiore del numero di equazioni, possono esistere infinite soluzioni. In altri termini per costruire la spline interpolante relativa ad un insieme di nodi, le condizioni di regolarità non bastano ad individuarla univocamente. È necessario imporre condizioni aggiuntive che specializzino il tipo di spline da costruire (ossia restringano l'insieme delle soluzioni ammissibili). Allora, per le condizioni aggiuntive che definiscono la spline cubica naturale interpolante:

$$\frac{d^2}{dx^2}s(x_1) = \frac{d^2}{dx^2}s(x_n) = 0,$$

si ha:

$$\bar{y}_1 = \bar{y}_n = 0,$$

per cui il sistema (3.59) si riduce ad un sistema lineare di  $n - 2$  equazioni in  $n - 2$  incognite del tipo:

$$F\bar{y} = g, \quad (3.60)$$

con:

$$F = \begin{bmatrix} 2(h_1 + h_2) & h_2 & 0 & \cdots & 0 \\ h_2 & 2(h_2 + h_3) & h_3 & 0 & \vdots \\ 0 & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & h_{n-2} \\ 0 & \cdots & 0 & h_{n-2} & 2(h_{n-2} + h_{n-1}) \end{bmatrix}$$

matrice dei coefficienti tridiagonale, simmetrica e a diagonale dominante,

$$\bar{y} = (\bar{y}_2, \dots, \bar{y}_{n-1})^T,$$

e

$$g = 6(y[x_2, x_3] - y[x_1, x_2], \dots, y[x_{n-1}, x_n] - y[x_{n-2}, x_{n-1}])^T.$$

Il sistema (3.60) può essere risolto utilizzando l'algoritmo di eliminazione di Gauss specializzato per matrici tridiagonali. Analizziamo, ora, le caratteristiche della matrice  $F$ . Per quanto riguarda il *condizionamento*, osservando che:

$$\mu_1(F) = \|F\|_1 \|F^{-1}\|_1 \leq (n-2) \cdot \max_{1 \leq i, j \leq n-2} |f_{ij}| \cdot \|F^{-1}\|_1$$

e che

$$\|F^{-1}\|_1 \leq \left( \min_{1 \leq i, j \leq n-2} |f_{ij}| \right)^{-1}$$

si ha

$$\mu_1(F) \leq (n-2) \cdot \max_{1 \leq i, j \leq n-2} |f_{ij}| \cdot \left( \min_{1 \leq i, j \leq n-2} |f_{ij}| \right)^{-1}.$$

Ma

$$\begin{aligned}\max_{1 \leq i, j \leq n-2} |f_{ij}| &= \max_{1 \leq i, j \leq n-2} \{2(h_i + h_{i+1})\} \\ \min_{1 \leq i, j \leq n-2} |f_{ij}| &= \min_{1 \leq i, j \leq n-2} h_i;\end{aligned}$$

se, allora

$$h_i = h \quad \forall i = 1, \dots, n-2$$

si ha

$$\begin{aligned}\mu_1(F) &\leq (n-2) \cdot \max_{1 \leq i, j \leq n-2} \{2(h_i + h_{i+1})\} \cdot \left( \min_{1 \leq i, j \leq n-2} h_i \right)^{-1} \\ &= (n-2) \cdot \left( \frac{4h}{h} \right)\end{aligned}$$

ovvero

$$\mu_1(F) \leq 4(n-2)$$

Si può dedurre, dunque, che la matrice  $F$  è ben condizionata, a meno che i nodi non siano "troppo" vicini o "troppo" lontani.

Inoltre, nell'applicazione dell'algoritmo di eliminazione di Gauss, *non è necessario il pivoting*. Infatti, poiché

$$\forall i = 1, \dots, n-2 \quad 2(h_i + h_{i+1}) > h_i \quad \text{e} \quad 2(h_i + h_{i+1}) > h_{i+1},$$

essendo

$$x_0 < x_1 < \dots < x_n$$

e, dunque,  $h_i > 0$ ,  $\forall i = 1, \dots, n-2$ , osserviamo, innanzitutto, che la matrice  $F$  risulta essere a diagonale dominante. Esaminiamo, poi, i passi del suddetto algoritmo, con riferimento alla matrice  $F$ . Indichiamo con

$$\underline{g} = (g_2, \dots, g_{n-2})$$

il vettore in cui memorizziamo la diagonale superiore di  $F$ , cioè

$$g_2 = h_2, \quad g_3 = h_3, \dots, g_{n-2} = h_{n-2}$$

e con

$$\underline{l} = (l_2, \dots, l_{n-2})$$

il vettore in cui memorizziamo la diagonale inferiore di  $F$ , cioè

$$l_2 = h_2, \quad l_3 = h_3, \dots, l_{n-2} = h_{n-2}$$

• **Passo 1**

data la struttura tridiagonale della matrice  $F$ , l'unico moltiplicatore non nullo è quello relativo alla seconda riga:

$$m_{21} = \frac{l_2}{f_{11}} = \frac{h_2}{2(h_1 + h_2)}$$

e, quindi,

$$m_{21} < 1 \quad (3.61)$$

inoltre

$$\begin{aligned} f_{22}^{(1)} &= f_{22} - m_{21}g_2 = 2(h_2 + h_3) - \underbrace{\frac{h_2}{2(h_1 + h_2)} \cdot h_2}_{<1} \\ &> 2h_2 + 2h_3 - h_2 = h_2 + 2h_3 \end{aligned}$$

$$g_3^{(1)} = \underbrace{g_3}_{=h_3} - m_{21} \underbrace{f_{13}}_{=0} = h_3$$

Ovviamente risulta

$$\begin{aligned} f_{22}^{(1)} &> h_2 \\ f_{22}^{(1)} &> h_3 \end{aligned}$$

• **Passo 2**

l'unico moltiplicatore non nullo è quello relativo alla terza riga:

$$m_{32} = \frac{l_3^{(1)}}{f_{22}^{(1)}} = \frac{h_3}{2(h_2 + h_3) - \frac{h_2}{2(h_1 + h_2)} \cdot h_2}$$

ma

$$f_{22}^{(1)} > h_3 \Rightarrow m_{32} < 1$$

ed, inoltre,

$$\begin{aligned} f_{33}^{(2)} &= f_{33}^{(1)} - m_{32}g_3^{(1)} = 2(h_3 + h_4) - m_{32} \cdot h_3 \\ &> 2h_3 + 2h_4 - h_3 = h_3 + 2h_4 \end{aligned}$$

$$g_4^{(2)} = \underbrace{g_4^{(1)}}_{=h_4} - m_{32} \underbrace{f_{24}^{(1)}}_{=0} = h_4$$

Ovviamente risulta

$$\begin{aligned} f_{33}^{(2)} &> h_3 \\ f_{33}^{(2)} &> h_4 \end{aligned}$$

⋮  
⋮

• **Passo  $n - 4$**

l'unico moltiplicatore non nullo è quello relativo alla  $(n - 3)$ -esima riga:

$$m_{n-3,n-4} = \frac{l_{n-3}^{(n-5)}}{f_{n-4,n-4}^{(n-5)}} = \frac{h_{n-3}}{2(h_{n-4} + h_{n-3}) - m_{n-4,n-5} \cdot h_{n-4}}$$

ma

$$f_{n-4,n-4}^{(n-5)} > h_{n-3} \Rightarrow m_{n-3,n-4} < 1$$

ed, inoltre,

$$\begin{aligned} f_{n-3,n-3}^{(n-4)} &= f_{n-3,n-3}^{(n-5)} - m_{n-3,n-4} g_{n-3}^{(n-5)} \\ &= 2(h_{n-3} + h_{n-2}) - m_{n-3,n-4} \cdot h_{n-3} > h_{n-3} + 2h_{n-2} \\ g_{n-2}^{(n-4)} &= \underbrace{g_{n-2}^{(n-5)}}_{=h_{n-2}} - m_{n-3,n-4} \underbrace{f_{n-4,n-2}^{(n-5)}}_{=0} = h_{n-2} \end{aligned}$$

Ovviamente risulta

$$\begin{aligned} f_{n-3,n-3}^{(n-4)} &> h_{n-3} \\ f_{n-3,n-3}^{(n-4)} &> h_{n-2} \end{aligned}$$

• **Passo  $n - 3$**

l'unico moltiplicatore non nullo è quello relativo alla  $(n - 2)$ -esima riga:

$$m_{n-2,n-3} = \frac{l_{n-2}^{(n-4)}}{f_{n-3,n-3}^{(n-4)}} = \frac{h_{n-2}}{2(h_{n-3} + h_{n-2}) - m_{n-3,n-4} \cdot h_{n-3}}$$

ma

$$f_{n-3,n-3}^{(n-4)} > h_{n-2} \Rightarrow m_{n-2,n-3} < 1$$

e si ha

$$\begin{aligned} f_{n-2,n-2}^{(n-3)} &= f_{n-2,n-2}^{(n-4)} - m_{n-2,n-3} g_{n-2}^{(n-4)} \\ &= 2(h_{n-2} + h_{n-1}) - m_{n-2,n-3} \cdot h_{n-2} > h_{n-2} + 2h_{n-1} \end{aligned}$$

Ovviamente risulta

$$\begin{aligned} f_{n-2,n-2}^{(n-3)} &> h_{n-2} \\ f_{n-2,n-2}^{(n-3)} &> h_{n-1} \end{aligned}$$

Concludendo si può osservare che l'elemento diagonale, ad ogni passo, è sempre il massimo, in modulo, degli elementi appartenenti alla sua stessa colonna. Pertanto si può utilizzare l'algoritmo di eliminazione di Gauss senza strategia del pivoting. Inoltre, il determinante della matrice  $F$  risulta essere uguale al prodotto degli elementi diagonali della matrice risultante all'ultimo passo del metodo di eliminazione di Gauss; ciascuno di essi, per quanto dimostrato, è positivo, essendo, infatti,

$$\begin{aligned} f_{11}^{(n-3)} &= 2(h_1 + h_2) \\ f_{ii}^{(n-3)} &= 2(h_i + h_{i+1}) - m_{i,i-1} \cdot h_i > h_i \quad \forall i = 2, \dots, n-2 \end{aligned}$$

e gli  $h_i > 0$ ,  $\forall i = 1, \dots, n-2$ , per definizione, nell'ipotesi

$$x_0 < x_1 < \dots < x_n.$$

Ne segue, dunque, che

$$\det(F) = \prod_{i=1}^{n-2} f_{ii}^{(n-3)} > 0$$

cioè esiste una ed una sola soluzione del sistema (3.60).

Riepilogando quanto detto, il procedimento per la costruzione e valutazione in un fissato valore  $\tilde{x}$  della funzione spline cubica naturale interpolante un insieme di punti assegnati  $(x_i, y_i), i = 1, \dots, n$  consiste in:

- Passo 1):** costruzione della matrice  $F$  e del vettore  $g$ ;
- Passo 2):** risoluzione del sistema  $F\bar{y} = g$ ;
- Passo 3):** determinazione dell'intervallo  $[x_i, x_{i+1}]$  a cui  $\tilde{x}$  appartiene;
- Passo 4):** calcolo dei coefficienti  $a_i, b_i, c_i, d_i$ ;
- Passo 5):** valutazione in  $\tilde{x}$ .

Uno schema della procedura per la costruzione della spline cubica naturale interpolante è il seguente:

```

procedure Spline_Nat_Int(in: n, x, y,  $\tilde{x}$ , out: s)

/# SCOPO: valutazione, in un fissato punto, della spline naturale cubica
interpolante, costruita su un insieme di punti assegnati

/# SPECIFICHE DEI PARAMETRI:
/# PARAMETRI DI INPUT:

var: n      : intero      { numero dei punti di }
                                { interpolazione }

var: x(n)  : array di reali { nodi di interpolazione }

var: y(n)  : array di reali { ordinate corrispondenti }
                                { ai nodi di interpolazione }

var:  $\tilde{x}$     : reale       { punto di valutazione }
                                { della spline }

/# PARAMETRI DI OUTPUT:

var: s      : reale       { valore della spline in  $\tilde{x}$  }

```

Procedura 3.10: Algoritmo per la costruzione e valutazione  
della spline naturale cubica interpolante - continua

```

/# INIZIO ISTRUZIONI:

call Coeff_F(n, x, e, f, m) { costruzione della matrice F }
call Coeff_g(n, x, y, g) { costruzione del vettore g }
call Gauss_Trid(n, e, f, m, g, ȳ) { risoluzione del sistema }
{ Fȳ = g mediante }
{ l'algoritmo di Gauss, }
{ specializzato per matrici }
{ tridiagonali }

call Ric_Bin(n, x, x̄, i) { determinazione }
{ dell'intervallo  $[x_i, x_{i+1}]$  }
{ a cui  $\tilde{x}$  appartiene }

call Coeff_Spline(xi, xi+1, yi, yi+1, ȳi, ȳi+1, ai, bi, ci, di) { calcolo dei coefficienti }
{ ai, bi, ci, di }

call Horner_Spline(xi, x̄, ai, bi, ci, di, s, n) { s è la valutazione della spline in  $\tilde{x}$  }

end Spline_Nat_Int

```

Procedura 3.10: Algoritmo per la costruzione e valutazione  
della spline naturale cubica interpolante - fine

Risolto il sistema, si determinano i coefficienti del polinomio che rappresenta la spline in ciascun intervallo  $[x_i, x_{i+1}]$  mediante le:

$$\left. \begin{array}{l} a_i = y_i \\ b_i = \frac{y_{i+1} - y_i}{h_i} - \frac{1}{6}(\bar{y}_{i+1} + 2\bar{y}_i)h_i \\ c_i = \frac{1}{2}\bar{y}_i \\ d_i = \frac{1}{6h_i}(\bar{y}_{i+1} - \bar{y}_i) \end{array} \right\} i = 1, \dots, n-1. \quad (3.62)$$

Infine, per quanto riguarda la complessità di tempo e, quindi l'*efficienza*, il numero di operazioni richieste dal metodo di eliminazione di Gauss per matrici tridiagonali è:

$T(n) = \mathcal{O}(n) \text{ flop}$

Concludendo, l'algoritmo per la costruzione e valutazione della spline cubica naturale interpolante un insieme di nodi assegnati, richiede una complessità di tempo che si ottiene, essenzialmente, stimando il numero di operazioni floating point coinvolte da ciascuna procedura, nonché il numero di confronti necessari per la ricerca binaria dell'intervallo di appartenenza del punto di valutazione:

- risoluzione di un sistema tridiagonale  $\Rightarrow T(n) = \mathcal{O}(n) \text{ flop}$
- ricerca binaria dell'intervallo di appartenenza del punto di valutazione  $\Rightarrow T(n) = \mathcal{O}(\log_2 n) \text{ confronti}$
- costruzione del polinomio interpolante  $\Rightarrow T(n) = \mathcal{O}(n^2) \text{ flop}$
- valutazione (algoritmo di Horner)  $\Rightarrow T(n) = \mathcal{O}(n) \text{ flop}$

♦ **Esempio 3.26.** Si vuole costruire una spline cubica interpolante 3 punti  $(x_i, y_i)$ ,  $i = 1, 2, 3$ . Costruiamo la spline intervallo per intervallo.

In ciascun intervallo fra due nodi consecutivi  $[x_1, x_2], [x_2, x_3]$  la spline  $s(x)$  deve essere un polinomio di grado al più tre:

$$\begin{aligned} s(x) &\equiv p_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \in \Pi_3, \\ x &\in [x_i, x_{i+1}], \quad i = 1, 2 \end{aligned}$$

In ciascuno dei due sottointervalli, dunque, occorre determinare i quattro coefficienti  $a_i, b_i, c_i, d_i$ , e quindi complessivamente occorre determinare 8 coefficienti.

Per come abbiamo definito la spline cubica, dobbiamo richiedere che  $s(x)$  sia una funzione continua insieme con la sua derivata prima e seconda nell'intervallo  $[x_1, x_3]$ . I due sottointervalli si raccordano nel nodo  $x_2$ , e quindi dobbiamo richiedere la regolarità della funzione  $s(x)$  e della sua derivata prima e seconda solo nel nodo  $x_2$ . Allora:

A) 1 condizione di *regolarità* per  $s(x)$  in  $x_2$ :

$$p_1(x_2) = p_2(x_2), \tag{3.63}$$

B) 1 condizione di *regolarità* per  $s'(x)$  in  $x_2$ :

$$\frac{d}{dx} p_1(x_2) = \frac{d}{dx} p_2(x_2), \tag{3.64}$$

C) 1 condizione di *regolarità* per  $s''(x)$  in  $x_2$ :

$$\frac{d^2}{dx^2} p_1(x_2) = \frac{d^2}{dx^2} p_2(x_2), \tag{3.65}$$

Infine, affinchè  $s(x)$  interpoli i punti assegnati, occorre imporre:

D) 3 condizioni di interpolazione:

$$s(x_i) = y_i, \quad i = 1, 2, 3 \rightarrow \begin{cases} p_1(x_1) = y_1 \\ p_2(x_2) = y_2 \\ p_2(x_3) = y_3 \end{cases} \quad (3.66)$$

In definitiva per determinare la funzione spline cubica  $s(x)$  interpolante i tre punti assegnati occorre determinare gli otto coefficienti  $a_i, b_i, c_i, d_i, i = 1, 2$  a partire dalle 6 condizioni (3.63-3.66).

Poniamo:

$$h_i = x_{i+1} - x_i \quad i = 1, 2$$

utilizzando le (3.63), (3.64), (3.65) si ha:

$$\begin{cases} a_1 + b_1 h_1 + c_1 h_1^2 + d_1 h_1^3 = a_2 \\ b_1 + 2c_1 h_1 + 3d_1 h_1^2 = b_2 \\ 2c_1 + 6d_1 h_1 = 2c_2 \end{cases} \quad (3.67)$$

e, dalle (3.66):

$$\begin{cases} a_i = y_i & i = 1, 2 \\ a_2 + b_2 h_2 + c_2 h_2^2 + d_2 h_2^3 = y_3 \end{cases} \quad (3.68)$$

Il sistema (3.67), insieme alle condizioni (3.68), diventa:

$$\begin{cases} b_1 h_1 + c_1 h_1^2 + d_1 h_1^3 = y_2 - y_1 \\ b_2 h_2 + c_2 h_2^2 + d_2 h_2^3 = y_3 - y_2 \\ b_1 - b_2 + 2c_1 h_1 + 3d_1 h_1^2 = 0 \\ 2c_1 + 6d_1 h_1 - 2c_2 = 0 \end{cases} \Leftrightarrow \begin{cases} b_1 = \frac{y_2 - y_1}{h_1} - c_1 h_1 - d_1 h_1^2 \\ b_2 = \frac{y_3 - y_2}{h_2} - c_2 h_2 - d_2 h_2^2 \\ b_1 - b_2 + 2c_1 h_1 + 3d_1 h_1^2 = 0 \\ d_1 = \frac{1}{6h_1}[2c_2 - 2c_1] \end{cases} \quad (3.69)$$

Quindi, sostituendo nella terza equazione le espressioni ottenute per  $b_1, b_2, d_1$ :

$$\frac{y_2 - y_1}{h_1} - \frac{y_3 - y_2}{h_2} + c_2(h_1 + h_2) - \frac{h_1}{6}[2c_2 - 2c_1] + d_2 h_2^2 = 0$$

da cui semplificando:

$$\frac{y_2 - y_1}{h_1} + \frac{y_3 - y_2}{h_2} + \frac{2}{3}h_1 c_2 + c_2 h_2 + \frac{1}{3}h_1 c_1 + d_2 h_2^2 = 0 \quad (3.70)$$

Quest'ultima equazione coinvolge tre incognite:  $c_1, c_2$  e  $d_2$ ; questo vuol dire che possono esistere infinite soluzioni al problema. Per garantire l'unicità della soluzione dobbiamo aggiungere altre due condizioni. In genere si impongono condizioni sulla derivata prima o seconda di  $s(x)$  nei nodi estremi  $x_1$  e  $x_3$ ; in base al tipo di condizioni richieste di solito si distinguono vari tipi di spline; ad esempio si parla di **spline cubica naturale** se si impone che:

$$\frac{d^2}{dx^2} s(x_1) = 0 \quad \frac{d^2}{dx^2} s(x_3) = 0;$$

Costruiamo quindi la spline cubica naturale. In questo caso dobbiamo imporre che:

$$\begin{aligned} \frac{d^2}{dx^2} s(x_1) &= \frac{d^2}{dx^2} p_1(x_1) = 0, \\ \frac{d^2}{dx^2} s(x_3) &= \frac{d^2}{dx^2} p_2(x_3) = 2c_2 + 6d_2 h_2 = 0 \end{aligned} \quad (3.71)$$

Dalla prima delle (3.71), poiché:

$$\frac{d^2}{dx^2} p_1(x_1) = c_1$$

ricaviamo:

$$c_1 = 0$$

Dalla seconda delle (3.71) ricaviamo  $d_2$ :

$$d_2 = \frac{-c_2}{3h_2}$$

Sostituendo nella (3.70) si ha:

$$\frac{y_2 - y_1}{h_1} - \frac{y_3 - y_2}{h_2} + \frac{2}{3}c_2(h_1 + h_2) = 0$$

Da cui si ricava  $c_2$ :

$$c_2 = \frac{2}{3} \left( \frac{y_1 - y_2}{h_1} - \frac{y_2 - y_3}{h_2} \right) / (h_1 + h_2)$$

e successivamente  $d_2$ . Infine dalle (3.69) si ottengono  $b_1$ ,  $b_2$  e  $d_1$ .



Come è stato osservato la funzione spline cubica descrive matematicamente lo strumento *spline* che sin dall'antichità i costruttori delle carene delle navi usavano per tracciare il profilo. La posizione assunta dallo strumento flessibile e di legno, quando veniva fissato in certi punti, era quella corrispondente ad una situazione di equilibrio. Ci aspettiamo quindi che in quella posizione sia minima l'energia del sistema “*spline + pesi*”.

In effetti l'equivalenza tra lo strumento meccanico e la funzione matematica sussiste ancora una volta. La funzione spline cubica naturale interpolante è quella funzione che, tra tutte le funzioni interpolanti un prefissato insieme di nodi, continue con le derivate prime e seconde, minimizza un funzionale energia del tipo:

$$E(f) = \int_{x_1}^{x_n} [f''(x)]^2 dx \quad f \in C^2[x_1, x_n]$$

Indicato con  $\eta_3(K)$  l'insieme delle spline naturali cubiche relative all'insieme

$$K = \{x_0, x_1, \dots, x_n, x_{n+1}\},$$

con  $x_0 = -\infty$  e  $x_{n+1} = +\infty$ , sussiste il seguente:

**Teorema 3.5.** Assegnati i nodi:

$$a \leq x_1 < x_2 < \dots < x_n \leq b,$$

sia  $g \in C^2([a, b])$ , una funzione interpolante i punti  $(x_i, y_i)$ ,  $i = 1, \dots, n$ :

$$g(x_i) = y_i, \quad i = 1, \dots, n$$

Sia  $s \in \eta_3(K)$  la spline cubica naturale interpolante, costruita sull'insieme

$$K = \{x_0, x_1, \dots, x_n, x_{n+1}\},$$

con  $x_0 = -\infty$  e  $x_{n+1} = +\infty$ , tale che

$$s(x_i) = y_i, \quad i = 1, \dots, n$$

Allora<sup>35</sup>

$$E(s) = \int_a^b [s''(x)]^2 dx \leq E(g) = \int_a^b [g''(x)]^2 dx$$

$\forall g \in C^2([a, b])$  ed, in particolare, l'uguaglianza vale se, e solo se,  $g \equiv s$  in  $[a, b]$ .<sup>36</sup>

**Dimostrazione** Posto  $f = g - s$  si ha:

$$\begin{aligned} \int_a^b [g''(x)]^2 dx &= \int_a^b [s''(x) + f''(x)]^2 dx = \\ &= \int_a^b [s''(x)]^2 dx + \int_a^b [f''(x)]^2 dx + 2 \int_a^b [s''(x) \cdot f''(x)] dx \end{aligned}$$

Dimostriamo, ora, che

$$\int_a^b [s''(x) \cdot f''(x)] dx = 0$$

da cui si avrà, necessariamente, che

$$\int_a^b [g''(x)]^2 dx \geq \int_a^b [s''(x)]^2 dx$$

ovvero la tesi.

Integrando per parti l'integrale

$$\int_a^b [s'' \cdot f''] dx$$

si ottiene

$$\int_a^b [s''(x) \cdot f''(x)] dx = [f'(x) \cdot s''(x)]_a^b - \int_a^b f'(x) s'''(x) dx$$

---

<sup>35</sup>Per semplicità viene riportato solo il risultato relativo alla spline cubica. Il teorema sussiste, più in generale, per una qualunque spline di grado dispari  $k = 2j + 1$  e in tal caso il ruolo svolto dalla derivata seconda in  $E$  viene svolto dalla derivata di ordine  $j$ , laddove il minimo del funzionale  $E(s)$  viene definito tra tutte le funzioni  $g \in C^j(\mathfrak{R})$ , cioè continue con le derivate fino all'ordine  $j$ .

<sup>36</sup>Si osserva che, nell'ipotesi:

$$|s'(x)| < 1 \Rightarrow |s'(x)|^2 \ll 1$$

il funzionale  $E(s)$  può essere interpretato come il funzionale energia relativo alla posizione di equilibrio dello strumento *spline*, in cui, ricordando il significato assunto nella nota (28),  $M(x)$  indica il momento flettente del listello,  $E$  il modulo di Young del materiale ed  $I$  il momento di inerzia del segmento di spline considerato rispetto ad un opportuno punto:

$$E(s) \approx \int_{x_1}^{x_n} \frac{s''(x)}{(1 + (s'(x))^2)^{3/2}} dx = \int_{x_i}^{x_n} \frac{1}{EI} M(x) dx$$

Risulta verificato che lo strumento spline, una volta costretto a passare per certi punti fissi, è effettivamente quello la cui posizione di equilibrio, minimizza un'energia.

Essendo  $s$  una spline naturale cubica, il primo termine è nullo, dall'annullarsi di  $s''$  in  $a$  ed in  $b$ ; applicando la proprietà additiva degli integrali in  $[a, x_1], [x_1, x_2], \dots, [x_n, b]$  al secondo termine, segue:

$$\int_a^b [s'' \cdot f''] dx = - \left[ \int_a^{x_1} f' \cdot s''' dx + \int_{x_1}^{x_2} f' \cdot s''' dx + \dots + \int_{x_n}^b f' \cdot s''' dx \right] \quad (3.72)$$

Poiché  $s$  è una spline naturale cubica, in ogni intervallo  $[x_i, x_{i+1}]$  essa è rappresentata da un polinomio di terzo grado, la cui derivata terza è una costante; in particolare si deduce che:

$$x \in [x_i, x_{i+1}], \quad i = 0, \dots, n \Rightarrow s'''(x) = q_i = s'''(x_i)_+ = s'''(x_{i+1})_-$$

dove  $s'''(x_i)_+$  è la derivata destra calcolata in  $x_i$  ed, analogamente,  $s'''(x_{i+1})_-$  è la derivata sinistra in  $x_{i+1}$ . Segue, dunque, dalla (3.72):

$$\begin{aligned} &= - \left[ q_0 \int_a^{x_1} f'(x) dx + q_1 \int_{x_1}^{x_2} f'(x) dx + \dots \right] = \\ &= - [-f(a)s'''(a)_+ + f(x_1)[s'''(x_1)_- - s'''(x_1)_+] + \\ &\quad + f(x_2)[s'''(x_2)_- - s'''(x_2)_+] + \dots] = \\ &= - \left[ \sum_{i=1}^n f(x_i) [s'''(x_i)_- - s'''(x_i)_+] + \right. \\ &\quad \left. + \underbrace{[f(b)s'''(b)_- - f(a)s'''(a)_+]}_{=0} \right] \end{aligned} \quad (3.73)$$

laddove l'ultimo termine si annulla poiché:

$$\text{se } b \equiv x_n \text{ allora } f(x_n) = g(x_n) - s(x_n) = 0$$

$$\text{se } a \equiv x_1 \text{ allora } f(x_1) = g(x_1) - s(x_1) = 0$$

$$\text{se } b < x_n \text{ allora } s'''(b) = 0 \text{ (spline naturale cubica)}$$

$$\text{se } a > x_1 \text{ allora } s'''(a) = 0 \text{ (spline naturale cubica)}$$

Si consideri, ora, la differenza

$$s'''(x_i)_- - s'''(x_i)_+;$$

siano  $p_i$  e  $p_{i+1}$  i due polinomi rappresentanti la spline nell'intervallo  $(x_{i-1}, x_i)$  e  $(x_i, x_{i+1})$  rispettivamente. Poiché

$$\begin{array}{rcl} p_i(x_i) &=& p_{i+1}(x_i) \\ p'_i(x_i) &=& p'_{i+1}(x_i) \\ p''_i(x_i) &=& p''_{i+1}(x_i) \end{array} \Rightarrow p_i(x) - p_{i+1}(x) \text{ ha, in } x_i, \text{ uno zero di molteplicità 3}$$

Segue, poi, dalla formula di Taylor, che

$$p_i(x) - p_{i+1}(x) = a_i(x - x_i)^3$$

e, dunque,

$$p'''_i(x_i) - p'''_{i+1}(x_i) = 6a_i$$

con

$$\begin{array}{rcl} p'''_i(x_i) &=& s'''(x_i)_- \\ p'''_{i+1}(x_i) &=& s'''(x_i)_+ \end{array}$$

da cui

$$s'''(x_i)_- - s'''(x_i)_+ = 6a_i$$

La (3.72) si riduce, quindi, a:

$$\int_a^b [s'' \cdot f''] dx = - \sum_{i=1}^n 6a_i f(x_i),$$

poiché  $f(x_1) = f(x_2) = \dots = f(x_n) = 0$ , si ha

$$\int_a^b s''(x) \cdot f''(x) dx = - \sum_{i=1}^n 6a_i f(x_i) = 0$$

cioè la tesi.

L'uguaglianza

$$\int_a^b [g''(x)]^2 dx = \int_a^b [s''(x)]^2 dx$$

vale se

$$\int_a^b [f''(x)]^2 dx = 0 \Rightarrow f''(x) = 0 \Rightarrow f \in \Pi_1 \quad \text{in } [a, b];$$

e, poiché  $f$  ha  $n$  zeri, con  $n > 2$ ,  $f$  è il polinomio identicamente nullo in  $[a, b]$  e, dunque,

$$g(x) \equiv s(x), \quad x \in [a, b]$$

ovvero l'asserto. ■

Come si può interpretare il risultato del Teorema 3.5?

Osserviamo che, per una qualsiasi funzione  $g \in C^2(\mathfrak{R})$ , si definisce la **curvatura** in un punto  $P(x, g(x))$ , appartenente ad una curva di equazione  $y = g(x)$ , come:

$$curv_g(P) = \frac{g''(x)}{(1 + (g'(x))^2)^{3/2}}$$

Nell'ipotesi che  $|g'(x)| \ll 1$ , risulta:

$$\frac{g''(x)}{(1 + (g'(x))^2)^{3/2}} \approx g''(x)$$

In questa ipotesi, dunque l'essere  $E(s)$  minimo corrisponde ad una richiesta di curvatura media minima per la funzione  $s$  e quindi ad una richiesta di oscillazioni regolari (smooth). Per questo motivo si usa spesso associare alla funzione spline un andamento mediamente regolare (privo di picchi), anche se va comunque notato che tale comportamento sussiste solo se la derivata prima della funzione spline è sufficientemente piccola in valore assoluto.

## 3.5 L'approssimazione dei minimi quadrati

In questo paragrafo affrontiamo il problema della rappresentazione di dati nel caso in cui essi siano affetti da un errore non trascurabile (derivante dagli strumenti di misura utilizzati, oppure da opportune semplificazioni). Come già detto nel §3.1, per evitare di esaltare l'errore presente nei dati, è più ragionevole richiedere che la funzione non debba assumere i valori assegnati ma che invece si scosti poco da questi in modo da non perdere completamente le informazioni in essi contenute e, allo stesso tempo, fornire una rappresentazione attendibile. In questi casi si parla di **modello approssimante**.

In base alla scelta della misura dello scostamento della funzione approssimante dai dati e ai vincoli che imponiamo sullo scostamento, si caratterizza il tipo di approssimazione. In particolare, se sceglio come misura dello scostamento la somma dei quadrati delle distanze dei punti assegnati dal grafico della funzione approssimante, e imponiamo che tale scostamento sia il minimo possibile, allora la funzione che si ottiene viene detta **migliore approssimazione nel senso dei minimi quadrati**.

Come per il modello interpolante, anche per quello approssimante è opportuno fissare *la forma* della funzione. In particolare, in questa sezione, prendiamo in esame soltanto le funzioni approssimanti di tipo **polinomiale**. In questo caso, si parla di **approssimazione polinomiale**.

### 3.5.1 La migliore approssimazione nel senso dei minimi quadrati

♣ **Esempio 3.27.** Riprendiamo l'esempio 3.5; in questo esempio si è visto che la quantità di ossigeno respirata da un individuo in movimento, in base alle misure effettuate mediante lo spirometro, ha un andamento lineare. I punti però non sono allineati, cioè il modello interpolante non può essere lineare e pertanto non è attendibile. Assumiamo come modello per descrivere i dati registrati attraverso lo spirometro, una retta.

Sia

$$y = f(x) = a + bx$$

la sua equazione. Si vuole calcolare il valore dei coefficienti  $a$  e  $b$ .

La retta disegnata in Fig. 3.33 è quella che realizza la migliore approssimazione nel senso dei minimi quadrati dei punti assegnati.

Consideriamo, per ogni punto, la distanza (verticale) del punto dalla retta. Assumiamo questa distanza come *misura* dello scostamento del generico punto dalla funzione  $f(x) = a + bx$ . Per ogni

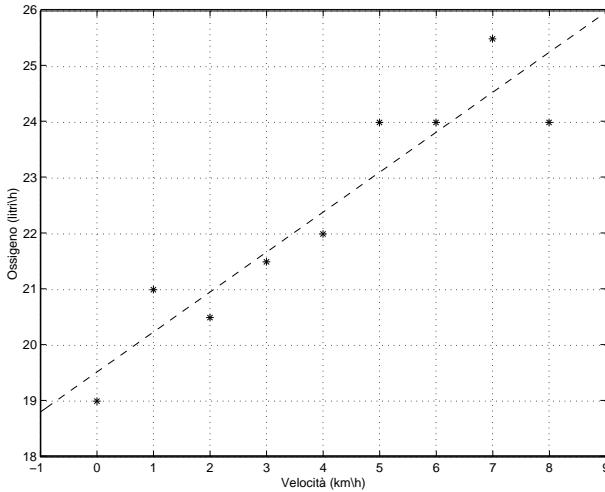


Figura 3.33: Un modello approssimante che descrive i dati

punto  $P_i = (x_i, y_i)$ , la distanza verticale è il segmento che congiunge l'ordinata del punto con l'ordinata del punto di ascissa  $x_i$  appartenente alla retta, in particolare tale valore è  $f(x_i) = a + bx_i$ . Indichiamo, quindi, con  $d_i$  tale distanza cioè:  $d_i = |y_i - a - b \cdot x_i|$ .

Per misurare lo scostamento di tutti i punti dalla retta consideriamo la somma dei quadrati di queste distanze<sup>37</sup>, e imponiamo che esso sia minimo. Imporre che lo scostamento sia minimo equivale a minimizzare la funzione:

$$S(a, b) = \sum_{i=1}^9 d_i^2 = \sum_{i=1}^9 [y_i - (a + b \cdot x_i)]^2$$

Osserviamo che, al variare di  $a$  e di  $b$  la funzione  $z = S(a, b)$  descrive un paraboloide (mostrato in Figura 3.35). Consideriamo le sezioni piane<sup>38</sup> di questo paraboloide ottenute riguardando la funzione  $S(a, b)$  come funzione della sola variabile  $a$  e pensando  $b$  come costante, oppure come funzione della sola variabile  $b$  e pensando  $a$  come costante. Indichiamo rispettivamente:  $f_1(a) = S(a, \cdot)$ , e  $f_2(b) = S(\cdot, b)$  le sezioni piane relative al primo e al secondo caso. In Figura 3.36 è mostrata una delle funzioni del tipo  $f_1(a) = S(a, \cdot)$ , per un fissato valore di  $b$ . Come si può notare, la sezione piana di un paraboloide è una parabola. Analogamente in Figura 3.37, è mostrata una delle funzioni del tipo  $f_2(b) = S(\cdot, b)$ , cioè una delle sezioni paraboliche ottenute considerando la funzione  $S(a, b)$  come funzione di una sola variabile, in particolare della variabile  $b$ . Anche in questo caso si può notare che si tratta di una parabola. Nel calcolare il punto di minimo di  $f_1(a)$ , basta imporre che la derivata prima di  $f_1(a)$  sia nulla. Avendo

<sup>37</sup>Più precisamente, questo equivale a considerare il quadrato della norma euclidea del vettore:  $d = (d_1, d_2, \dots, d_9)$ , cioè:

$$\|d\|^2 = d_1^2 + d_2^2 + \dots + d_9^2$$

<sup>38</sup>Le sezioni piane sono curve ottenute intersecando il paraboloide con il piano  $y = y_0$  cioè con un piano parallelo al piano  $xz$  ad una distanza  $y_0$  dal piano  $xz$ , e con il piano  $x = x_0$  cioè con il piano parallelo al piano  $yz$  e ad una distanza  $x_0$  dal piano  $yz$ .

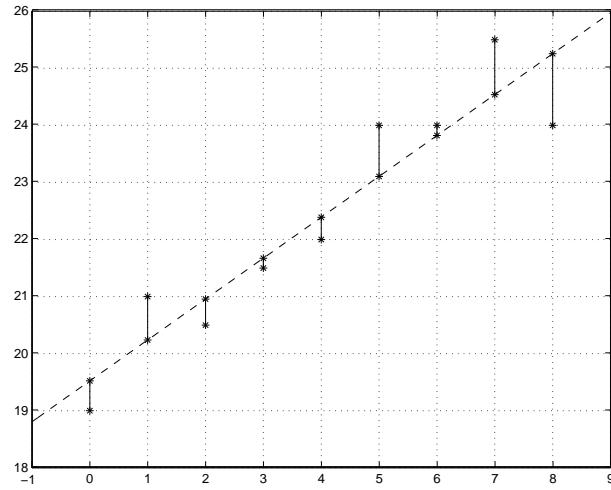
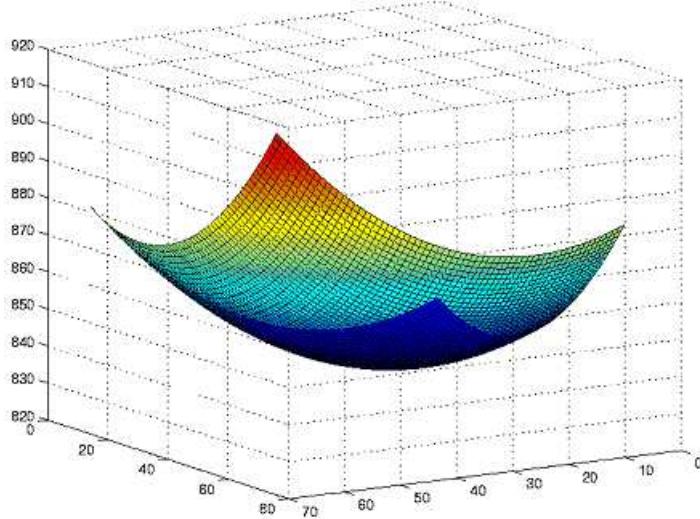


Figura 3.34: Distanza verticale dei punti dalla retta

riguardato  $b$  come parametro, ne risulta una equazione che dipende da  $b$  oltre che da  $a$ . Analogamente, per la funzione  $f_2(b)$ , avendo riguardato  $a$  come parametro, imponendo che la derivata prima sia nulla si ottiene una equazione che dipende da  $a$  oltre che da  $b$ .

Figura 3.35: Il Paraboloide  $S(a,b)$ 

Il punto di minimo del paraboloide, in quanto punto di minimo per tutte le sezioni paraboliche, è determinato in corrispondenza di quei valori di  $a$  e di  $b$  che soddisfano sia la prima che la seconda equazione.

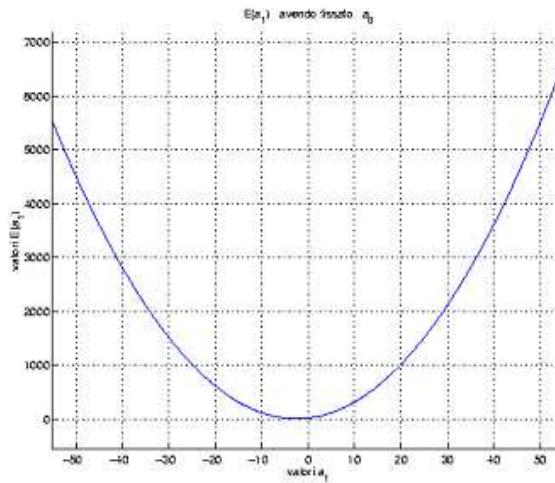


Figura 3.36: Sezione  $f_1(a) = S(a, \cdot)$  del Paraboloide ottenuta fissando b

Quindi <sup>39</sup>

$$\begin{cases} \frac{\partial S}{\partial a} = 0 \\ \frac{\partial S}{\partial b} = 0 \end{cases}$$

<sup>39</sup>Questa condizione in generale non è sufficiente a garantire che il punto sia un punto di minimo, ma assicura soltanto che tale punto è un **punto estremale**. La condizione necessaria che determina la presenza di un minimo o di un massimo (relativo) (cioè di un punto **estremante**) è che la matrice *Hessiana*, cioè la matrice costituita dalle derivate seconde di  $S(a, b)$  sia semidefinita positiva. Inoltre, se la derivata seconda rispetto alla prima variabile è positiva allora il punto è di minimo, altrimenti se è negativa il punto è un punto di massimo. Una condizione affinché una matrice sia definita positiva è fornita dal **criterio di Sylvester** che asserisce che tutti i minori principali di una matrice definita positiva devono essere non negativi. Nel caso in esame la matrice Hessiana è la seguente:

$$\begin{pmatrix} 9 & 36 \\ 36 & 204 \end{pmatrix}$$

i cui minori principali sono rispettivamente 9 e 540. In particolare, il coefficiente di posto (1,1), è positivo (in generale coincide con il numero di punti).

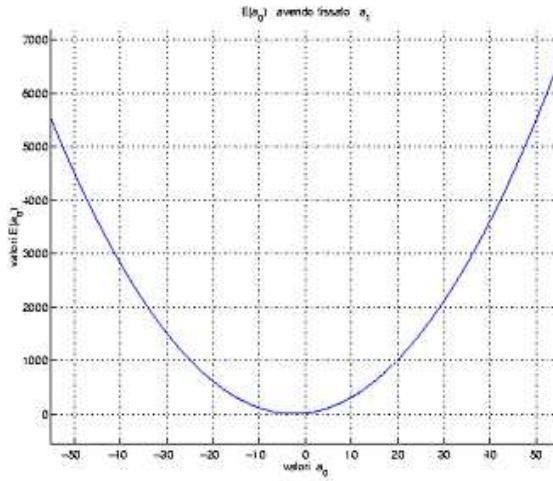


Figura 3.37: Sezione  $f_2(b) = S(\cdot, b)$  del Paraboloide ottenuta fissando a

cioè:

$$\begin{cases} \frac{df_1(a, \cdot)}{da} = \frac{\partial \sum_{i=1}^9 [y_i - (a + b \cdot x_i)]^2}{\partial a} = 0 \\ \frac{df_2(\cdot, b)}{db} = \frac{\partial \sum_{i=1}^9 [y_i - (a + b \cdot x_i)]^2}{\partial b} = 0 \end{cases} \quad (3.74)$$

quindi:

$$\begin{cases} \frac{\partial \sum_{i=1}^9 [y_i - (a + b \cdot x_i)]^2}{\partial a} = 0 \\ \frac{\partial \sum_{i=1}^9 [y_i - (a + b \cdot x_i)]^2}{\partial b} = 0 \end{cases}$$

da cui:

$$\begin{cases} 2 \sum_{i=1}^9 [y_i - (a + b \cdot x_i)] (-x_i) = 0 \\ 2 \sum_{i=1}^9 [y_i - (a + b \cdot x_i)] (-1) = 0 \end{cases}$$

cioè, effettuando i prodotti:

$$\begin{cases} 9a + \sum_{i=1}^9 x_i b = \sum_{i=1}^9 y_i \\ \sum_{i=1}^9 x_i a + \sum_{i=1}^9 x_i^2 b = \sum_{i=1}^9 x_i y_i \end{cases} \quad (3.75)$$

Si è ottenuto un sistema lineare di 2 equazioni e 2 incognite. La sua soluzione fornisce i coefficienti della retta  $y = f(x) = a + bx$  che tra tutte le rette realizza la minima distanza dai punti assegnati. Nell'esempio considerato i punti  $P_i$  hanno le seguenti coordinate:

$$\begin{aligned} x &= [0, 1, 2, 3, 4, 5, 6, 7, 8]; \\ y &= [19, 21, 20.5, 21.5, 22, 24, 24, 25.5, 24]; \end{aligned}$$

il sistema (3.75) diventa:

$$\begin{cases} 9 \cdot a + \sum_{i=1}^9 x_i \cdot b = \sum_{i=1}^9 y_i \\ \sum_{i=1}^9 x_i \cdot a + \sum_{i=1}^9 x_i^2 \cdot b = \sum_{i=1}^9 y_i \cdot x_i \end{cases} \iff \begin{cases} 9a + 36b = 201.5 \\ 36a + 204b = 849 \end{cases}$$

La soluzione di tale sistema è la coppia  $(19.52, 0.7167)$ , rispettivamente i coefficienti  $a$  e  $b$  della retta  $y = f(x) = a + b \cdot x$ . La retta così costruita si dice **retta dei minimi quadrati**.

Descrivere i punti  $(x_i, y_i)$  mediante la retta dei minimi quadrati, significa assumere come valore *attendibile* per ogni misura  $x_i$ , il valore

$$f(x_i) = a + bx_i.$$

Un modo per stimare l'attendibilità del valore  $f(x_i)$  rispetto alla misura  $y_i$  consiste nel calcolare la *deviazione standard*,  $\sigma$ , delle misure  $y_1, y_2, \dots, y_n$  rispetto ai corrispondenti valori assunti *veri*  $f(x_i)$ , cioè calcolare:

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^9 (d_i)^2} = \sqrt{\frac{1}{9} \sum_{i=1}^9 (y_i - f(x_i))^2}, \quad (3.76)$$

Questa quantità viene anche caratterizzata imponendo che la probabilità di ottenere queste stesse misure sia massima.<sup>40</sup>

In altre parole, la deviazione standard è (a meno del farore  $\sqrt{(1/N)}$ ) la radice quadrata della somma dei quadrati delle distanze dei punti dalla retta approssimante e quindi, possiamo concludere che la retta di migliore approssimazione nel senso dei minimi quadrati è stata ottenuta minimizzando la deviazione standard.

Calcoliamo, dunque, la deviazione standard relativa alle misure  $y_1, \dots, y_n$  dell'esempio 3.19:

$$\sigma = \sqrt{\frac{1}{9} \sum_{i=1}^9 (y_i - 19.52 \cdot x_i - 0.7167)^2} = 0.712$$

Questo risultato consente di affermare che su ciascuna misura  $y_i$  c'è un'incertezza media pari a 0.712.

Questa quantità rappresenta, inoltre, il valore medio della somma dei quadrati delle distanze (verticali) di tutti i punti dalla retta dei minimi quadrati.




---

<sup>40</sup>Se le misure sono affette da errori casuali ma di ordine di grandezza sufficientemente piccolo rispetto a quello dei valori stessi, allora la distribuzione limite di ciascuna delle misure è la distribuzione normale:

$$\exp(-Y^2/2\sigma^2)$$

centrata nel valore *vero*  $Y$ , mentre  $\sigma$  è la deviazione standard, o *parametro di larghezza*. Quanto più  $\sigma$  è grande tanto meno velocemente la funzione tende a zero al crescere di  $x$  cioè tanto più larga è la "campana", caratteristica forma della distribuzione normale. In queste ipotesi la probabilità di ottenere  $N$  misure ( $y_i$ ) è proporzionale al prodotto delle probabilità di ottenere ciascuna misura  $y_i$  e quindi a:

$$\frac{1}{\sigma^N} \exp(-\sum (y_i - Y_i)^2 / 2\sigma^2)$$

Il valore di  $\sigma$  che rende massima tale probabilità è proprio la deviazione standard definita dalla (3.76).

♣ **Esempio 3.28.** Determinare una stima dell'accelerazione di gravità in una certa regione della superficie terrestre

Consideriamo un corpo che cade liberamente e misuriamo la sua altezza in una successione di istanti di tempo. Supponendo che il moto sia uniformemente accelerato con accelerazione data dall'accelerazione di gravità, l'equazione di moto del corpo è:

$$y(t) = y_0 + v_0 \cdot t + \frac{1}{2} \cdot g \cdot t^2 \quad (3.77)$$

avendo indicato con  $y_0$  la posizione iniziale,  $v_0$  la velocità iniziale e con  $g$  l'accelerazione di gravità.

Si effettuano le seguenti misure:

tempo	0	1	2	3	4	5
altezza	65	131	113	89	51	7

avendo misurato il tempo in decimi di secondi e l'altezza in centimetri. A partire da questi dati ci aspettiamo un andamento come quello descritto dall'equazione in (3.77).

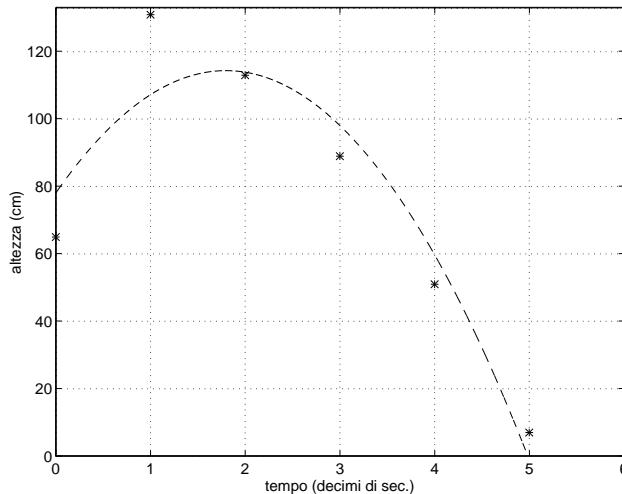


Figura 3.38: Traiettoria percorsa da un corpo in caduta soggetto all'accelerazione di gravità

Invece i punti non appartengono ad una parabola (cfr. Figura 3.38). Anche in questo caso, l'errore introdotto nell'effettuare le misure è tale che rappresentando in un piano cartesiano i punti di coordinate

$(t_i, h_i)$ , dove  $t_i$  indica l'istante di tempo in cui è stata misurata l'altezza  $h_i$ , questi punti non descrivono una parabola. Pertanto, anche in questo caso il modello interpolante non può essere attendibile. Costruiamo allora un modello approssimante.

Considerata la funzione di equazione (3.77) calcoliamo i suoi coefficienti  $y_0, v_0, g$  richiedendo che la curva descritta dal grafico di tale funzione sia quella che si scosti il meno possibile dai dati. Ancora una volta, assumiamo come criterio per valutare lo scostamento tra il generico punto e il grafico della funzione (3.77) la distanza (verticale) del punto dalla curva.

Indicata con  $d_i$  la distanza del generico punto  $P_i$  dalla curva, cioè la differenza in valore assoluto tra l'ordinata del punto e l'ordinata del punto sulla curva con la medesima ascissa, lo scostamento della funzione approssimante dai dati è misurato dalla somma dei quadrati delle distanze di ciascun punto:

$$S = d_1^2 + d_2^2 + d_3^2 + d_4^2 + d_5^2 + d_6^2$$

Quindi, per calcolare i coefficienti  $y_0, v_0$  e  $g$  della parabola approssimante minimizziamo la funzione:

$$S(y_0, v_0, g) = \sum_{i=1}^6 \left[ y_i - (y_0 + v_0 \cdot t_i + \frac{1}{2}g \cdot t_i^2) \right]^2$$

Per calcolare il punto di minimo della funzione  $S(y_0, v_0, g)$  imponiamo che le derivate parziali di  $S$  rispetto a  $y_0, v_0$  e  $g$  siano nulle, cioè richiediamo che:

$$\begin{cases} \frac{\partial S}{\partial y_0} = 0 \\ \frac{\partial S}{\partial v_0} = 0 \\ \frac{\partial S}{\partial g} = 0 \end{cases}$$

quindi:

$$\begin{cases} \frac{\partial \sum_{i=1}^6 [y_i - (y_0 + v_0 \cdot t_i + \frac{1}{2}g \cdot t_i^2)]^2}{\partial y_0} = 0 \\ \frac{\partial \sum_{i=1}^6 [y_i - (y_0 + v_0 \cdot t_i + \frac{1}{2}g \cdot t_i^2)]^2}{\partial v_0} = 0 \\ \frac{\partial \sum_{i=1}^6 [y_i - (y_0 + v_0 \cdot t_i + \frac{1}{2}g \cdot t_i^2)]^2}{\partial g} = 0 \end{cases}$$

da cui:

$$\begin{cases} 2 \sum_{i=1}^6 \left[ y_i - (y_0 + v_0 \cdot t_i + \frac{1}{2}g \cdot t_i^2) \right] (-1) = 0 \\ 2 \sum_{i=1}^6 \left[ y_i - (y_0 + v_0 \cdot t_i + \frac{1}{2}g \cdot t_i^2) \right] (-t_i) = 0 \\ 2 \sum_{i=1}^6 \left[ y_i - (y_0 + v_0 \cdot t_i + \frac{1}{2}g \cdot t_i^2) \right] (-\frac{1}{2}t_i^2) = 0 \end{cases}$$

Effettuando i prodotti:

$$\left\{ \begin{array}{l} 6 \cdot y_0 + \sum_{i=1}^6 t_i \cdot v_0 + \frac{1}{2} \sum_{i=1}^6 t_i^2 \cdot g = \sum_{i=1}^6 y_i \\ \sum_{i=1}^6 t_i \cdot y_0 + \sum_{i=1}^6 t_i^2 \cdot v_0 + \frac{1}{2} \sum_{i=1}^6 t_i^3 \cdot g = \sum_{i=1}^6 y_i \cdot t_i \\ \sum_{i=1}^6 t_i^2 \cdot y_0 + \sum_{i=1}^6 t_i^3 \cdot v_0 + \frac{1}{2} \sum_{i=1}^6 t_i^4 \cdot g = \sum_{i=1}^6 y_i \cdot t_i^2 \end{array} \right.$$

Siamo pervenuti ad un sistema di 3 equazioni nelle 3 incognite  $y_0$ ,  $v_0$  e  $g$ . La soluzione di tale sistema è la terna dei coefficienti della parabola di equazione (3.77) che realizza la migliore approssimazione nel senso dei minimi quadrati dai punti misurati.

La parabola così costruita si dice **parabola dei minimi quadrati**. In questo caso, effettuando i prodotti il sistema lineare diventa:

$$\left\{ \begin{array}{l} 6 \cdot y_0 + 15 \cdot v_0 + 27.5 \cdot g = 456 \\ 15 \cdot y_0 + 55 \cdot v_0 + 112.5 \cdot g = 863 \\ 55 \cdot y_0 + 225 \cdot v_0 + 489.5 \cdot g = 2375 \end{array} \right.$$

La soluzione di tale sistema è la terna  $(78.0714, 40.4214, -22.5)$ , cioè la terna dei coefficienti della parabola di equazione (3.77) che realizza la migliore approssimazione nel senso dei minimi quadrati dai punti misurati. In particolare  $g = -22.5$ . ♣

Assegnati i punti:

$$P_i = (x_i, y_i) \quad i = 1, \dots, m$$

si vuole costruire il **polinomio di migliore approssimazione nel senso dei minimi quadrati**. Indicato con:

$$p(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \dots + a_n \cdot x^n$$

tale polinomio, imponiamo che sia minima la somma delle distanze (verticali) di ciascun punto  $P_i$  dal grafico del polinomio  $p$ . Consideriamo la funzione:

$$S(a_0, a_1, \dots, a_n) = \sum_{i=1}^m [y_i - (a_0 + a_1 \cdot x_i + a_2 \cdot x_i^2 + \dots + a_n \cdot x_i^n)]^2$$

che rappresenta la misura dello scostamento, nel senso dei minimi quadrati, di  $p$  dai punti  $P_i$ . Calcoliamo il punto di minimo imponendo che le derivate (parziali) di  $S(a_0, a_1, \dots, a_n)$ ,

rispetto alle incognite  $a_0, a_1, \dots, a_n$  siano nulle<sup>41</sup>. Dunque imponendo che:

$$\left\{ \begin{array}{l} \frac{\partial S}{\partial a_0} = 0 \\ \frac{\partial S}{\partial a_1} = 0 \\ \vdots \\ \frac{\partial S}{\partial a_n} = 0 \end{array} \right.$$

ed effettuando i prodotti, si ottiene un sistema lineare di  $n + 1$  equazioni nelle  $n + 1$  incognite  $a_0, a_1, \dots, a_n$ :

$$\left\{ \begin{array}{lcl} ma_0 + \sum_{i=1}^m x_i \cdot a_1 \dots + \sum_{i=1}^m x_i^n \cdot a_n & = & \sum_{i=1}^m y_i \\ \sum_{i=1}^m x_i \cdot a_0 + \sum_{i=1}^m x_i^2 \cdot a_1 \dots + \sum_{i=1}^m x_i^{n+1} \cdot a_n & = & \sum_{i=1}^m x_i y_i \\ \dots & \dots & \dots \end{array} \right. \quad (3.78)$$

Tale sistema prende il nome di **sistema di equazioni normali**. Come vedremo nel prossimo paragrafo, si può dimostrare che questo sistema ammette un'unica soluzione. La soluzione fornisce il vettore dei coefficienti del polinomio di grado al più  $n$  che tra tutti i polinomi di grado al più  $n$  realizza la **migliore approssimazione nel senso dei minimi quadrati**.

♣ **Esempio 3.29.** Un biologo misura una popolazione di batteri per tre giorni successivi con i risultati mostrati nella tabella seguente:

t=tempo	0	1	2
p= popolaz. ( $\times 10^3$ )	153	137	128

A partire da questi dati il biologo vuole determinare la stima migliore per la *vita media* della popolazione.

---

<sup>41</sup>Questa condizione in generale non basta a garantire che tale punto sia di minimo. In generale garantisce solo che si tratta di un punto estremale. L'eventuale presenza di massimi o di minimi (relativi) (cioè di punti estremanti) è stabilita dalla matrice Hessiana. Nel caso specifico si può verificare che la matrice Hessiana, cioè la matrice costituita dalle derivate seconde della funzione  $S(a_0, a_1, \dots, a_n)$ , è definita positiva; per cui il punto stazionario sarà sicuramente un punto estremante cioè un punto in cui siamo in presenza di un minimo o di un massimo. D'altra parte, inoltre, il coefficiente di prima riga e prima colonna della matrice Hessiana, cioè la derivata seconda rispetto alla prima variabile  $a_0$ , è uguale a  $2m$  pertanto è positivo. Il punto quindi sarà sicuramente di minimo.

Molte popolazioni (di batteri, di nuclei radioattivi) tendono a variare esponenzialmente nel tempo. Se una popolazione  $P$  sta diminuendo esponenzialmente, allora la legge che regola tale fenomeno è del tipo:

$$P(t) = P_0 e^{-t/\tau}$$

dove  $\tau$  è detta *vita media* della popolazione. Si deve determinare, quindi, il valore di  $\tau$ .

Poiché l'equazione:

$$P(t) = P_0 e^{-t/\tau} \quad (3.79)$$

non è lineare, prima di procedere cerchiamo di *linearizzarla* cioè di trasformarla in una equazione lineare equivalente. Poiché la non linearità è introdotta unicamente dalla funzione esponenziale un modo per eliminarla è quello di esprimere anche il primo membro in termini della funzione esponenziale, cioè:

$$P(t) \equiv e^{\log(P(t))}$$

e quindi la (3.79) si scrive:

$$e^{\log(P(t))} = P_0 e^{-t/\tau} = e^{\log(P_0)} e^{-t/\tau} = e^{\log(P_0) - t/\tau}$$

Questa equazione è verificata se e solo se:

$$\log(P(t)) = \log(P_0) - t/\tau \quad (3.80)$$

cioè posto:

$$z(t) = \log(P(t)), \quad z_0 = \log P_0$$

se e solo se

$$z(t) = z_0 - t/\tau$$

Abbiamo ricondotto l'equazione (3.79) ad una equazione lineare nella incognita  $t$  di cui si devono calcolare i coefficienti  $z_0$  e  $\tau$  in modo che sia minima la distanza del grafico della funzione  $y = z(t)$  dai punti assegnati. Tenendo conto che si è effettuata la trasformazione  $z(t) = \log(P(t))$ , trasformiamo allo stesso modo anche le coordinate dei punti assegnati:

$$\begin{aligned} z(0) &= \log(P(0)) = 11.9382 \\ z(1) &= \log(P(1)) = 11.8277 \\ z(2) &= \log(P(2)) = 11.7598 \end{aligned}$$

Considerata la funzione:

$$S(z_0, \tau) = \sum_{i=1}^3 [z_i - (z_0 - t_i/\tau)]^2$$

il sistema delle equazioni normali è il seguente:

$$\begin{cases} 3 \cdot a + 3 \cdot b = 35.5257 \\ 3 \cdot a + 5 \cdot b = 35.3473 \end{cases}$$

dove  $a = z_0$  e  $b = -1/\tau$ . La soluzione è la coppia  $(11.9311, -0.0892) = (z_0, -\frac{1}{\tau})$ , da cui deriva che la vita media della popolazione in esame,  $\tau$ , è  $\frac{1}{0.0892} = 11.21$  giorni. ♣

### 3.5.2 Esistenza ed unicità

In tutti gli esempi che abbiamo descritto la costruzione della funzione approssimante conduce alla risoluzione di un sistema di equazioni lineari. L'esistenza e l'unicità della funzione approssimante dipende, pertanto, dalla esistenza ed unicità della soluzione del sistema delle equazioni normali e quindi dalla matrice dei coefficienti del sistema.

♣ **Esempio 3.30.** Consideriamo l'esempio 3.27 e determiniamo la matrice dei coefficienti del sistema lineare (3.75). Tale sistema è del tipo:

$$\begin{cases} 9 \cdot a + \sum_{i=1}^9 x_i \cdot b = \sum_{i=1}^9 y_i \\ \sum_{i=1}^9 x_i \cdot a + \sum_{i=1}^9 x_i^2 \cdot b = \sum_{i=1}^9 x_i \cdot y_i \end{cases} \quad (3.81)$$

per cui la matrice dei coefficienti è:

$$C = \begin{pmatrix} 9 & \sum_{i=1}^9 x_i \\ \sum_{i=1}^9 x_i & \sum_{i=1}^9 x_i^2 \end{pmatrix} \quad (3.82)$$

Introdotta la matrice:

$$A = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \\ 1 & x_5 \\ 1 & x_6 \\ 1 & x_7 \\ 1 & x_8 \\ 1 & x_9 \end{pmatrix}$$

e considerata la sua trasposta:

$$A^T = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 \end{pmatrix}$$

la matrice  $C$  si può ottenere dal prodotto righe per colonne della trasposta di  $A$  per  $A$  cioè:

$$C = \begin{pmatrix} 9 & \sum_{i=1}^9 x_i \\ \sum_{i=1}^9 x_i & \sum_{i=1}^9 x_i^2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_9 \end{pmatrix} \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \\ 1 & x_5 \\ 1 & x_6 \\ 1 & x_7 \\ 1 & x_8 \\ 1 & x_9 \end{pmatrix} = A^T \cdot A$$

Analogamente, il termine noto del sistema (3.81) si può esprimere come prodotto della trasposta di  $A$

per il vettore  $y$  delle ordinate corrispondenti ai nodi. In altre parole si ha:

$$\begin{pmatrix} \sum_{i=1}^9 y_i \\ \sum_{i=1}^9 x_i y_i \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \end{pmatrix} = A^T y$$

e quindi posto  $c = (a, b)^T$ , il sistema (3.81):

$$\begin{cases} m \cdot a + \sum_{i=1}^m x_i \cdot b = \sum_{i=1}^m y_i \\ \sum_{i=1}^m x_i \cdot a + \sum_{i=1}^m x_i^2 \cdot b = \sum_{i=1}^m x_i y_i \end{cases}$$

in termini matriciali si riscrive:

$$A^T \cdot Ac = A^T y$$

L'esistenza e l'unicità della soluzione dipende dal determinante della matrice  $A^T A$ . ♣

Riscriviamo il sistema di equazioni normali (3.78). Introdotta la matrice:

$$A = \begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ \vdots & \dots & \dots & \dots & \vdots \\ 1 & x_i & x_i^2 & \dots & x_i^{n-1} \\ \vdots & \dots & \dots & \dots & \vdots \\ 1 & x_m & x_m^2 & \dots & x_m^{n-1} \end{pmatrix}$$

si verifica facilmente che la matrice dei coefficienti del sistema (3.78) si esprime come:

$$C = A^T A$$

e, analogamente, indicato con  $z$  il vettore dei termini noti nel sistema (3.78) si verifica che:

$$z = A^T y$$

Pertanto, il sistema di equazioni normali (3.78), è espresso in forma matriciale nel modo seguente:

$$A^T Ac = A^T y$$

Vogliamo vedere sotto quali condizioni esso ammette un'unica soluzione.

♣ **Esempio 3.31.** Consideriamo il sistema omogeneo associato al sistema di equazioni normali relativo all'esempio 3.30:

$$\begin{cases} m \cdot a + \sum_{i=1}^m x_i \cdot b = 0 \\ \sum_{i=1}^m x_i \cdot a + \sum_{i=1}^m x_i^2 \cdot b = 0 \end{cases} \iff A^T \cdot Ac = 0 \quad (3.83)$$

Moltiplicando la prima equazione per  $a$  e la seconda per  $b$ , si ha:

$$\begin{cases} a \cdot (m \cdot a + \sum_{i=1}^m x_i \cdot b) = 0 \\ b \cdot (\sum_{i=1}^m x_i \cdot a + \sum_{i=1}^m x_i^2 \cdot b) = 0 \end{cases} \quad (3.84)$$

Sommmando ambo i membri

$$a \cdot \left( m \cdot a + \sum_{i=1}^m x_i \cdot b \right) + b \cdot \left( \sum_{i=1}^m x_i \cdot a + \sum_{i=1}^m x_i^2 \cdot b \right) = 0 \quad (3.85)$$

Osserviamo che in questo modo si ottiene un'equazione equivalente al sistema (3.83). Infatti, chiaramente ogni soluzione del sistema (3.83) è soluzione della (3.85), viceversa data l'equazione (3.85) con  $a$  e  $b$  arbitrari, se assumiamo che questa equazione sia soddisfatta qualunque siano i valori di  $a$  e di  $b$ , necessariamente devono annullarsi le quantità racchiuse in parentesi, cioè  $a$  e  $b$  devono soddisfare le due equazioni del sistema (3.83).

Osserviamo che la (3.85), sviluppando i calcoli, si può esprimere come:

$$a^2 m + 2 \sum_{i=1}^m x_i ab + \sum_{i=1}^m x_i^2 b^2 = \sum_{i=1}^m (a^2 + 2x_i ab + x_i^2 b^2) = \sum_{i=1}^m (a + b \cdot x_i)^2 = 0 \quad (3.86)$$

Perché sia soddisfatta la (3.86) è necessario che sia:

$$p(x_i) = a + bx_i = 0, \quad i = 1, \dots, n.$$

Si possono verificare due eventualità: il polinomio  $p(x) = a + bx$  si annulla nei punti  $x_i$ , oppure  $p$  è il polinomio identicamente nullo. Escludiamo la prima possibilità perché un polinomio di primo grado può avere al più uno zero e invece i punti sono  $m > 1$ .

Ne deriva che  $p$  è il polinomio identicamente nullo cioè  $a = b = 0$ , pertanto l'equazione (3.85), e quindi il sistema (3.83), ammette solo la soluzione identicamente nulla, e quindi il suo determinante è diverso da zero. ♣

Consideriamo il sistema

$$A^T Ac = 0$$

omogeneo associato al sistema  $A^T A c = A^T y$  e moltiplichiamo la  $k - ma$  equazione per il coefficiente  $a_k$  del polinomio approssimante  $p(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^m$ . Sommando le  $m$  equazioni si ha:

$$\sum_{i=1}^m p(x_i)^2 = 0 \quad (3.87)$$

Perché sia verificata la (3.87) è necessario che sia  $p(x_i) = 0$  per  $i = 1, \dots, n$ .

Si possono verificare due possibilità: il polinomio  $p$  si annulla in questi punti, oppure il polinomio  $p$  è identicamente nullo. Se supponiamo che sia  $n > m$ , dal teorema fondamentale dell'algebra si deduce che un polinomio di grado al più  $m$  non può avere  $n$  zeri se  $n > m$  e, pertanto, escludiamo la prima possibilità; rimane, allora, l'altra, cioè  $p(x)$  è il polinomio identicamente nullo. L'equazione (3.76) e, quindi, il sistema omogeneo associato alle equazioni normali, ammette solo la soluzione identicamente nulla e ciò implica che il determinante del sistema delle equazioni normali è diverso da zero.

Osserviamo che l'ipotesi che ha consentito di concludere che il determinante debba essere necessariamente non nullo e quindi che il sistema lineare che fornisce i coefficienti del polinomio di migliore approssimazione nel senso dei minimi quadrati ammette un'unica soluzione, è che il numero di punti  $n$  sia maggiore del grado del polinomio  $m$ .

Possiamo dunque concludere che:

**Teorema 3.6.** *Assegnati  $n$  nodi distinti ed  $n$  valori corrispondenti, il sistema di equazioni normali relativo alla costruzione del polinomio  $p(x) \in \Pi_m$  di migliore approssimazione nel senso dei minimi quadrati, ammette una ed una sola soluzione se e solo se i nodi sono a due a due distinti e se  $n > m + 1$ .<sup>42</sup>*

♦ **Esempio 3.32.** Assegnati i punti:

$$P_1 = (1, 1), P_2 = (2, 5),$$

si determini la retta di migliore approssimazione nel senso dei minimi quadrati.

Se

$$A = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \end{pmatrix}$$

in questo caso risulta:

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$$

---

<sup>42</sup>In effetti il teorema risulta vero anche nell'ipotesi  $n > m$  ma, poiché in questo caso può accadere che  $n = m + 1$  cioè che la soluzione coincida con il polinomio interpolante, e dato che in linea di principio nella costruzione del polinomio di migliore approssimazione non siamo interessati a trovare il polinomio interpolante, si preferisce in questa sede lasciare la condizione  $n > m + 1$ .

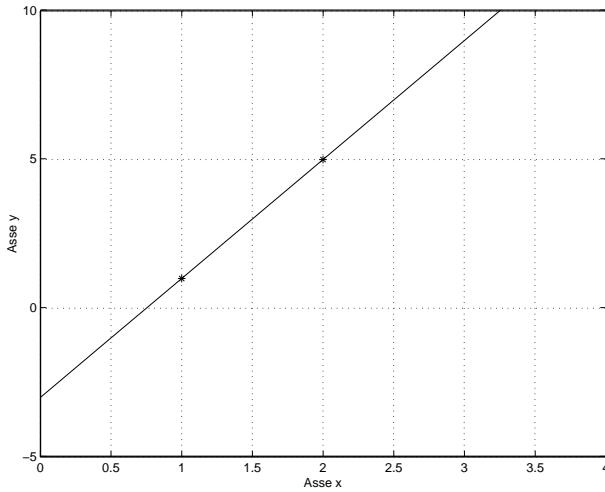


Figura 3.39: Approssimazione nel senso dei minimi quadrati con un polinomio di primo grado

il sistema delle equazioni normali è del tipo:

$$A^T A c = A^T y \quad (3.88)$$

dove con  $c$  abbiamo indicato il vettore dei coefficienti  $a, b$  della retta  $y = a + bx$  di minimi quadrati. Quindi, si ha:

$$\begin{cases} 2a + 3b = 6 \\ 3a + 5b = 11 \end{cases}$$

La soluzione di questo sistema è:

$$a = -3, \quad b = 4$$

cioè la retta di minimi quadrati ha equazione:

$$y = f(x) = -3 + 4x$$

Rappresentiamo in un sistema di assi cartesiani i punti e la retta:

Come si osserva dalla figura, il polinomio di primo grado di migliore approssimazione nel senso dei minimi quadrati è la retta interpolante. In altre parole, in questo caso lo scostamento dei punti dalla funzione di minimi quadrati è nullo. Vediamo di spiegare cosa è successo nel risolvere questo problema.

Osserviamo innanzitutto che, poiché il determinante di  $A$  è dato da:

$$\det(A) = (x_2 - x_1)$$

se  $x_1 \neq x_2$  tale matrice è non singolare, inoltre  $n = 2 > 1 = m$ . Pertanto esiste l'inversa  $A^{-1}$ , il sistema (3.88) è quindi equivalente al seguente sistema:

$$Ac = y \quad (3.89)$$

ottenuto moltiplicando il precedente a sinistra per la matrice  $(A^T)^{-1}$ .

Il sistema (3.89) ammette un'unica soluzione. L'unica soluzione è quella che fornisce i coefficienti del polinomio interpolante. Esso infatti, coincide con il sistema che si ottiene dal metodo dei coefficienti



indeterminati costruito per il calcolo dei coefficienti della retta interpolante. Pertanto si può affermare che in questo caso la retta dei minimi quadrati è la retta interpolante, cioè è la retta in corrispondenza della quale  $y_i = f(x_i)$  cioè la distanza dei punti dalla retta è nulla.<sup>43</sup>

In questo caso il modello interpolante è anche un modello approssimante perché i coefficienti del polinomio interpolante rendono nullo lo scostamento S. Ma in un problema di rappresentazione di dati, affetti da un errore non trascurabile, non ha senso costruire il modello interpolante.

Tenendo conto che fissati  $n$  punti il polinomio interpolante ha grado  $n - 1$ , bisogna escludere che il grado del polinomio approssimante,  $m$ , possa essere uguale a  $n - 1$ .

### **3.5.3 Un algoritmo per la costruzione e valutazione del polinomio di migliore approssimazione nel senso dei minimi quadrati**

Per calcolare i coefficienti del polinomio di migliore approssimazione nel senso dei minimi quadrati bisogna risolvere il sistema delle equazioni normali. Tale sistema può essere risolto utilizzando l'algoritmo di Gauss<sup>44</sup>. In conclusione, una procedura per la determinazione del polinomio di migliore approssimazione nel senso dei minimi quadrati deve prevedere:

1. la costruzione della matrice dei coefficienti  $A^T A$  del sistema;
2. la costruzione del termine noto  $A^T y$  del sistema;
3. la risoluzione del sistema:  $A^T A c = A^T y$ ;
4. la valutazione del polinomio approssimante.

Utilizzando una procedura per ciascuna delle richieste, una versione dell'algoritmo è la seguente:

---

<sup>43</sup>Si osservi che in termini matriciali questo significa che  $y = Ac$

<sup>44</sup>In generale, non è consigliabile l'algoritmo di Gauss perché la matrice dei coefficienti del sistema delle equazioni normali è mal condizionata. In tal caso si preferiscono opportuni algoritmi di fattorizzazione che trasformano il sistema in uno meglio condizionato.

```

procedure polinapprox(in: n, m, x, y,  $\tilde{x}$  ; out: p)

/# SCOPO: valutazione in un punto del polinomio di grado al più m
di migliore approssimazione nel senso dei minimi quadrati,
relativo ad un insieme di punti assegnati

/# SPECIFICHE DEI PARAMETRI:
/# PARAMETRI DI INPUT:
var: n : intero { numero dei punti }
var: m : intero { grado del polinomio }
var: x(n) : reale { nodi di approssimazione }
var: y(n) : reale { ordinate corrispondenti }
{ ai nodi di approssimazione }
var:  $\tilde{x}$  : reale { punto di valutazione }

/# PARAMETRI DI OUTPUT:
var: p : reale { p è il valore del polinomio }
{ nel punto di valutazione }

/# INIZIO ISTRUZIONI:
call Sist(n, x, B) { costruzione della matrice del sistema }
{ di equazioni normali }
call noto(n, y, b) { calcolo del termine noto del sistema }
{ di equazioni normali }
call Gauss(n, B, c, b) { risoluzione del sistema }
{ di equazioni normali }
call Horner(c,  $\tilde{x}$ , p, n) { p è la valutazione del polinomio}
{ nel punto assegnato }

end polinapprox

```

Procedura 3.11: Migliore approssimazione nel senso dei minimi quadrati

### 3.6 MATLAB e la rappresentazione di dati

È possibile utilizzare alcune funzioni predefinite in ambiente MATLAB per rappresentare un insieme di punti nel piano o nello spazio.

La funzione `polyfit` costruisce il polinomio interpolante di Lagrange e più in generale il polinomio di migliore approssimazione nel senso dei minimi quadrati. Funzioni polinomiali a tratti di primo e terzo grado possono essere costruite utilizzando la funzione `interp1`. Spline cubiche si ottengono, infine, richiamando la funzione built-in `spline`. Vediamone l'uso attraverso alcuni esempi.

♣ **Esempio 3.33.** Supponiamo di voler costruire il polinomio interpolante un insieme assegnato di punti. Il seguente programma, dopo aver calcolato i coefficienti del polinomio interpolante di Lagrange, ne disegna il grafico:

```
>> x=[1:1:10];
>> y=[ 2 4 6 7 9 10 23 24 48 56 ];
>> xi=[0:0.1:10];
>> c=polyfit(x,y,9);
>> yi = polyval(c,xi);
>> plot(x,y,'o',xi,yi)
```

Analizziamo il programma istruzione per istruzione.

L'istruzione:

```
>> x=[1:1:10];
```

definisce l'array `x` contenente i nodi di interpolazione. In particolare `x` contiene i valori da 1 a 10 equispaziati con passo 1.

L'istruzione:

```
>> y=[ 2 4 6 7 9 10 23 24 48 56 ];
```

definisce l'array `y` contenente le ordinate corrispondenti ai nodi di interpolazione.

L'istruzione:

```
>> xi=[0:0.1:10];
```

definisce l'array **xi** dei punti necessari alla valutazione del polinomio interpolante. L'array **xi** viene definito dall'insieme dei valori da 0 a 10 equispaziati con passo 0.1.

La funzione **polyfit** :

```
>> c=polyfit(x,y,9);
```

restituisce il vettore dei coefficienti del polinomio, di grado al più 9, interpolante i punti assegnati.

La funzione **polyval**:

```
>> yi = polyval(c,xi);
```

valuta, nei punti **xi**, il polinomio i cui coefficienti sono definiti dalle componenti dell'array **c**.

L'istruzione:

```
>> plot(x,y,'o',xi,yi)
```

consente di disegnare sia i punti di interpolazione (in corrispondenza dei quali verrà disegnata una 'o'), sia il grafico del polinomio interpolante.

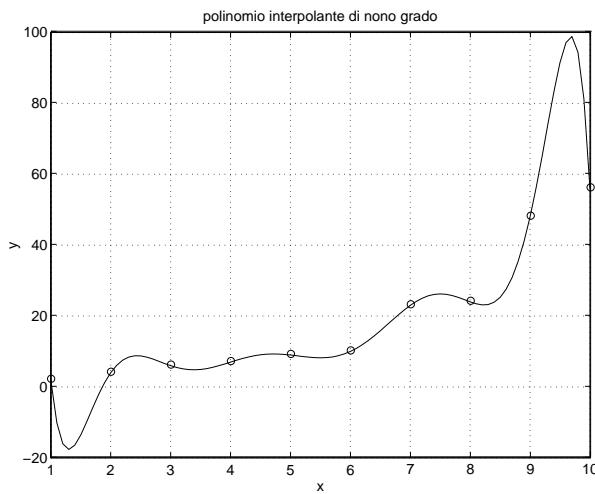


Figura 3.40: Polinomio interpolante

La stessa funzione `polyfit` può essere utilizzata per costruire il polinomio di migliore approssimazione nel senso dei minimi quadrati.

In particolare le istruzioni:

```
>> c1=polyfit(x,y,1);  
  
>> yi1 = polyval(c1,xi);
```

consentono, rispettivamente, di costruire e valutare il polinomio di primo grado, di migliore approssimazione nel senso di minimi quadrati.

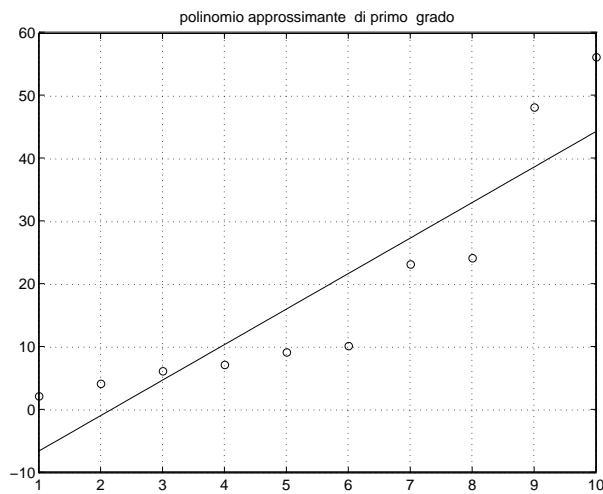


Figura 3.41: Polinomio di approssimazione nel senso dei minimi quadrati

La Figura 3.42 mostra un confronto tra il polinomio interpolante e quello approssimante, ottenuto eseguendo l'istruzione:

```
>> plot(x,y,'o',xi,yi,x1,yi1)
```

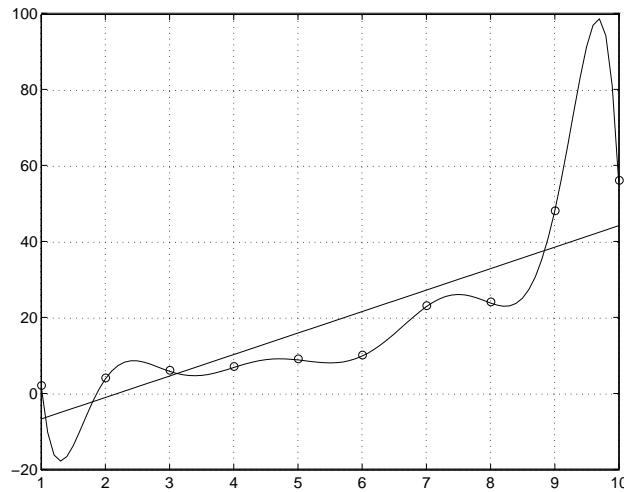


Figura 3.42: Confronto tra la retta dei minimi quadrati e il polinomio interpolante

Analogamente, se  $m < n - 1$  la funzione:

```
>> c=polyfit(x,y,m);
```

calcola i coefficienti del polinomio di grado  $m$ , di migliore approssimazione nel senso dei minimi quadrati.



**♣ Esempio 3.34.** Le seguenti istruzioni MATLAB costruiscono rispettivamente il polinomio interpolante a tratti di primo e terzo grado, sullo stesso insieme di punti considerato nell'esempio precedente:

```
>> zi= interp1(x,y,xi);
>> zi= interp1(x,y,xi,cubic);
```

Dopo aver calcolato i coefficienti si possono valutare i polinomi utilizzando la funzione `polyval`, mentre per disegnare il loro grafico si può utilizzare la funzione `plot`. Infine, l'istruzione seguente valuta, nei nodi `xi`, la spline cubica interpolante:

```
>> zi= spline(x,y,xi);
```



♣ **Esempio 3.35.** L'interpolazione lineare a tratti è alla base dell'algoritmo utilizzato per disegnare in ambiente MATLAB il grafico di una qualsiasi funzione. Supponiamo di voler disegnare il grafico della funzione  $y = \sin(x)$  nell'intervallo  $[0, \pi]$ . Vediamo cosa succede se eseguiamo l'istruzione seguente:

```
>> x=[0:0.9:2π];
>> plot(x,sin(x))
```

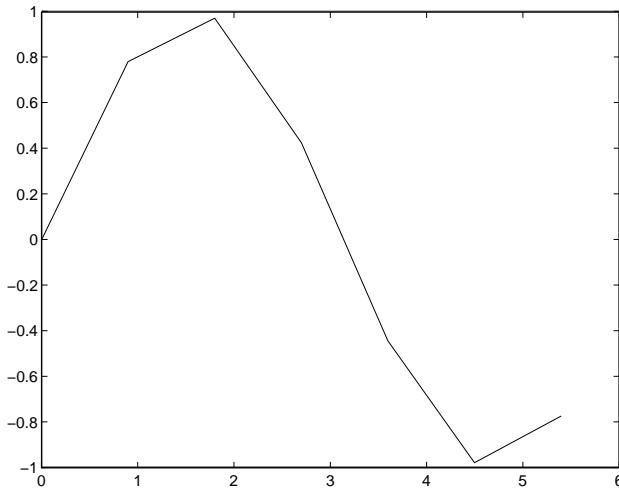


Figura 3.43: Grafico della funzione  $\sin(x)$ , ottenuto suddividendo l'intervallo  $[0, 2\pi]$  in sottointervalli di ampiezza 0.9

Come si può osservare dalla Figura 3.43, per rappresentare questi punti sul piano, essi vengono congiunti a due a due con segmenti di retta. Chiaramente la rappresentazione sarà tanto più attendibile quanto più numerosi sono i nodi nell'intervallo fissato. Ad esempio, eseguendo le istruzioni:

```
>> x=[0:0.1:2π];
>> plot(x,sin(x))
```

si utilizza una distribuzione di punti, per mezzo della quale la rappresentazione di  $y=\sin(x)$  risulta piuttosto soddisfacente, come si può osservare in Figura 3.44.



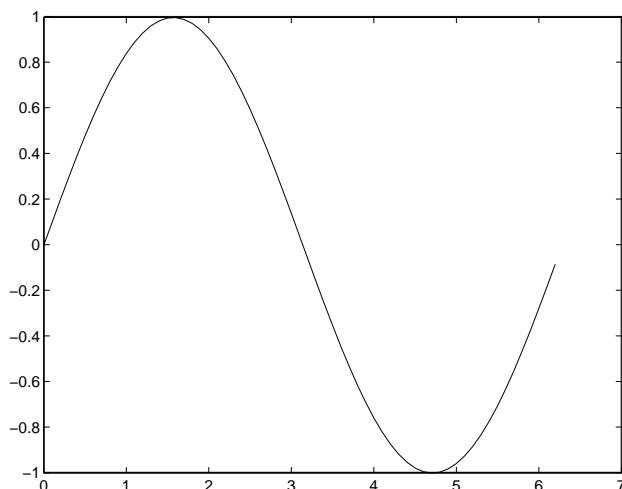


Figura 3.44: Grafico della funzione  $\sin(x)$ , ottenuto suddividendo l'intervallo  $[0, 2\pi]$  in sottointervalli di ampiezza 0.1

### 3.7 Il software disponibile per la rappresentazione di dati

Prima di descrivere seppur brevemente il software disponibile per la rappresentazione di dati, richiamiamo la classificazione del software matematico introdotta da *J. Rice* in *Numerical methods, Software and Analysis*, Academic Press, 1993. Tale classificazione consiste in:

- routines individuali (eventualmente raccolte in collezioni),
- packages di routines che risolvono problemi in una specifica area computazionale,
- packages di routines di base,
- librerie *general - purpose*,
- Ambienti di risoluzione di problemi (PSE).

Descriviamo, brevemente per ciascuna delle classi individuate, il software disponibile per la rappresentazione di dati.

- ACM-TOMS (*Collected Algorithms of the Association for Computing Machinery*), una collezione di più di 650 elementi di software matematico, distribuita dall'ACM Algorithms Distribution Service.
- PPPACK (*Package of Piecewise Polynomial and spline Routines*), raccolta dei programmi del libro di C. De Boor, *A practical Guide to Splines*, del 1979.
- LAPACK (*Linear Algebra Package*), il package di routines di algebra lineare, ottimizzate per calcolatori a memoria gerarchica.
- le librerie del NAG e la IMSL.

Tranne le librerie, tutto il software citato è di pubblico dominio ed è reperibile tramite la rete web attraverso il sito di Netlib <http://www.netlib.org>. Tra le routines dell'ACM-TOMS ve ne sono 20 per problemi di interpolazione e 25 per quelli di approssimazione. In particolare, si utilizza l'interpolazione polinomiale (nella formula di Newton), polinomiale a tratti, mediante spline cubiche.

PPPACK è un package di 58 routines in Fortran 77 per la costruzione, valutazione, derivazione ed integrazione di spline cubiche interpolanti e approssimanti.

La libreria NAG (Mark 17) contiene due capitoli dedicati alla rappresentazione di dati, rispettivamente E01 (interpolazione) e E02 (Fitting di curve e superfici). Il capitolo E01 contiene 24 routines per l'interpolazione di Lagrange in due e tre dimensioni, mediante polinomi, funzioni razionali, polinomi a tratti, spline cubiche. Ad esempio, nell'ambito dell'interpolazione polinomiale, la routine E01AEF fornisce il polinomio interpolante costruito attraverso la formula di Newton, mentre se si vuole solo la valutazione in un preassegnato valore si può utilizzare la routine E01AAF basata sullo schema di Aitken.

La routine E01BAF costruisce la spline cubica interpolante.

Per quanto riguarda l' approssimazione di dati, nel senso dei minimi quadrati, i modelli utilizzati dalle routine della NAG si basano sui polinomi e sulle spline cubiche.

La libreria dell'IMSL contiene circa 60 routine in Fortran 77 per la rappresentazione di curve e superfici mediante modelli interpolanti o approssimanti. In tutti i casi si utilizzano solo spline cubiche.

Le routines sono raggruppate in due sottoinsiemi: nel primo la spline cubica viene costruita intervallo per intervallo, nel secondo si utilizzano spline di ordine al più 15 e la rappresentazione utilizza opportune funzioni di base (le B-spline).

Infine tra gli ambienti di risoluzione di problemi, quello più utilizzato è sicuramente il MATLAB .

## 3.8 Esercizi sulla rappresentazione di dati

### 3.8.1 Alcuni esercizi sull'interpolazione polinomiale

**Esercizio 1** Costruire il polinomio di terzo grado che assume i seguenti valori:

x	1	3	4	6
y	-7	5	8	14

utilizzando la Formula di Lagrange.

**Esercizio 2** Utilizzando un opportuno polinomio interpolante, calcolare una stima del valore della funzione esponenziale  $f(x) = e^x$  in  $x = 0.9$  a partire dai seguenti valori tabulati:

x	0.6	0.7
y	1.82212	2.01375

Utilizzare, successivamente i seguenti punti:

x	0.6	0.7	0.8
y	1.82212	2.01375	2.22554

ed infine i punti:

x	0.6	0.7	0.8	1.0
y	1.82212	2.01375	2.22554	2.71828

Confrontare i valori ottenuti.

**Esercizio 3** Calcolare una stima del valore assunto dalla funzione esponenziale in  $x = .0075$ , utilizzando la tavola seguente:

x	0.006	0.007	0.008
y	1.00601	1.00702	1.00803

**Esercizio 4** Determinare il polinomio interpolante espresso nella formula di Lagrange e nella formula di Newton, relativo ai seguenti dati:

$$\begin{array}{c|ccc} x & 7 & 1 & 2 \\ \hline y & 146 & 2 & 1 \end{array}$$

e dimostrare che sono identicamente uguali.

**Esercizio 5** Calcolare una stima del valore assunto dalla funzione  $\sin(x)$  in  $x = 1.375$  utilizzando il polinomio di primo grado nell'intervallo  $[1.35, 1.40]$ , e il polinomio di secondo grado nell'intervallo  $[1.35, 1.45]$ .

**Esercizio 6** A partire dai seguenti dati:

anno	caffè
1961	1052
1965	1204
1969	1556
1973	2010
1977	1815
1981	2253
1985	2812

costruire il polinomio interpolante di settimo grado.

**Esercizio 7** A partire dai seguenti dati:

anno	laureati
1961/62	23
1967/68	40
1970/71	57
1973/74	63
1976/77	72
1979/80	76
1982/83	74

costruire il polinomio interpolante di sesto grado, utilizzando la formula di Newton.

**Esercizio 8** Costruire il polinomio interpolante i dati seguenti:

A)

$$\begin{array}{c|cc} x & 3 & 7 \\ \hline y & 5 & -1 \end{array}$$

B)

$$\begin{array}{c|ccc} x & 7 & 1 & 2 \\ \hline y & 146 & 2 & 1 \end{array}$$

C)

$$\begin{array}{c|cccc} x & 3 & 7 & 1 & 2 \\ \hline y & 10 & 146 & 2 & 1 \end{array}$$

D)

$$\begin{array}{c|cccccc} x & 1.5 & 2.7 & 3.1 & -2.1 & -6.6 & 11.0 \\ \hline y & 0 & 0 & 0 & 1 & 0 & 0 \end{array}$$

### 3.8.2 Alcuni esercizi sull'approssimazione nel senso dei minimi quadrati

**Esercizio 1** Se il volume di un campione di gas ideale è tenuto costante allora la sua temperatura  $T$  è una funzione lineare della sua pressione  $p$ :

$$T(p) = a + bp$$

calcolare la migliore approssimazione nel senso dei minimi quadrati a partire dalle seguenti misure:

$$\begin{array}{c|ccccc} p & 65 & 75 & 85 & 95 & 105 \\ \hline T & -20 & 17 & 42 & 94 & 127 \end{array}$$

Valutare l'affidabilità delle misure  $T_i$  utilizzando la deviazione standard.

**Esercizio 2** Un carrello viaggia a velocità costante e viene cronometrato il suo passaggio in 4 differenti posizioni, in successione di 4 istanti di tempo:

$$\begin{array}{c|cccc} t & 17.6 & 40.4 & 67.7 & 90.1 \\ \hline s & 0 & 3000 & 6000 & 9000 \end{array}$$

calcolare una stima della sua velocità.

Valutare, inoltre, l'affidabilità della misure  $s_i$  utilizzando la deviazione standard.

**Esercizio 3** Assegnati i punti:

x	-4	-2	-1	0	1	3	4	6
y	-35.1	15.1	15.9	8.9	.1	.1	.21.1	135

determinare il polinomio di terzo grado di migliore approssimazione nel senso dei minimi quadrati.

Calcolare, inoltre, l'indice di condizionamento del sistema delle equazioni normali.

**Esercizio 4** Calcolare la parabola e la retta di migliore approssimazione relativamente ai punti:

x	0	.5	1	1.5
y	0	.6	1	1.5

Calcolare, inoltre, l'indice di condizionamento del sistema delle equazioni normali, sia nel primo che nel secondo caso.



# Bibliografia

- [1] Davis P. J. - *Interpolation and approximation* - Blaisdell, New York, 1963.
- [2] Gauss F. C. - *Theoria Combinationis Observationum Erroribus Minimis Obnoxiae* - tradotto in inglese da G.W. Stewart, Classics in applied Mathematics, n. 11, SIAM, 1995
- [3] Hayes J.G. - *Numerical approximation to Functions and Data* - The Athlone Press, 1970.
- [4] Meinardus G. - *Introductory Lectures on some problems in approximation theory* - manoscritto, 1974 .
- [5] Severi F. - *Geometria elementare* - 1926
- [6] Shoenberg I.J. - *Contribution to the problem of approximation of equidistant data by analytical functions* - Vol 4, N.1, 1945
- [7] Spath H. - *Spline Algorithms for Curves and Surfaces* - UTILITAS MATHEM. PUBLISH. INC., 1973
- [8] Ueberhuber C.W. - *Numerical Computation 1: Methods, Software and Analysis* - Springer, 1997.

# Capitolo 4

## La quadratura

### 4.1 Generalità

La misura delle aree è uno dei problemi scientifici più antichi. In epoca babilonese ed egiziana, ad esempio, venivano calcolate delle buone approssimazioni dell'area di un cerchio con il cosiddetto **metodo di esaustione**<sup>1</sup>. Problema analogo era la costruzione di un quadrato con area uguale a quella di un cerchio assegnato mediante l'ausilio di riga e compasso in un numero finito di passi. Tale problema era noto come **quadratura del cerchio**<sup>2</sup> e, più in generale, il termine **quadratura** veniva utilizzato per riferirsi a qualunque problema di calcolo dell'area o del volume di un oggetto di forma nota. Il problema del calcolo delle aree e dei volumi è tuttora di interesse in molte applicazioni di varia natura come, ad esempio, nella determinazione del volume di terra da rimuovere per la costruzione di strade, o nel calcolo dell'area della superficie della carrozzeria di un'automobile. In genere i problemi sono relativi a figure geometriche di forma irregolare, per cui non è sempre possibile applicare formule elementari per il calcolo delle aree (o dei volumi).

♣ **Esempio 4.1.** Si supponga di voler calcolare l'area della regione piana chiusa mostrata in Figura 4.1. Tale regione può essere divisa in cinque sottoregioni tracciando delle linee orizzontali e verticali come mostrato in Figura 4.2.

---

<sup>1</sup>Il metodo consisteva nel dimostrare che due grandezze sono uguali mostrando che la loro differenza è inferiore a qualsiasi quantità data, per quanto piccola. In particolare, per determinare l'area del cerchio, si inscrive al suo interno una successione di poligoni regolari con un numero di lati crescente. La superficie del poligono diventa sempre più grande ad ogni passo, ma rimane sempre inferiore a quella del cerchio. Inoltre la differenza tra le due aree diventa sempre più piccola. In tal modo l'area del cerchio veniva calcolata con la massima precisione possibile nonostante non si conoscesse esattamente. Utilizzando tale metodo Archimede fornì anche un'approssimazione per il numero irrazionale  $\pi$  (il rapporto tra area del cerchio e quadrato del raggio), stimando che esso appartiene all'intervallo chiuso [223/71, 22/7] commettendo un errore dell'ordine di  $10^{-4}$ .

<sup>2</sup>Tale problema non ha soluzione. F. Lindemann, nel 1882 mostrò infatti che il numero  $\pi$  è un numero irrazionale.

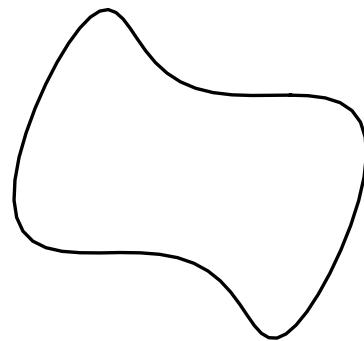


Figura 4.1: Regione piana chiusa

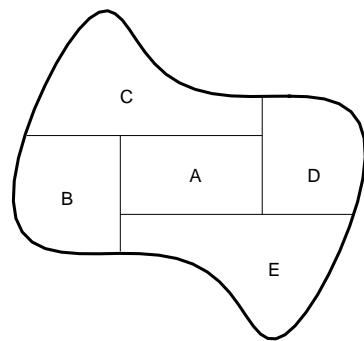


Figura 4.2: Suddivisione della regione in sottoregioni

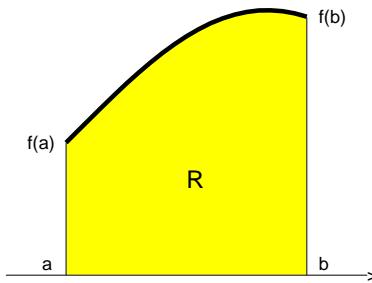
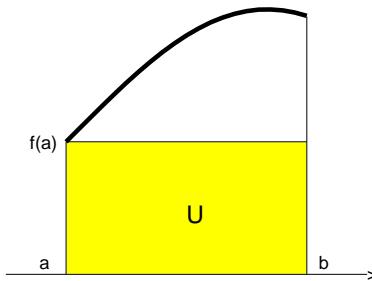


Figura 4.3: Area di un rettangoloide

Figura 4.4: Approssimazione di  $\mathcal{A}(R)$  mediante la formula rettangolare

La sottoregione  $A$  è un rettangolo la cui area può essere calcolata mediante il prodotto delle sue dimensioni. Le sottoregioni  $B, C, D, E$  hanno forma irregolare ed il calcolo della loro area può essere ricondotto al calcolo dell'area di una regione piana delimitata da una curva.



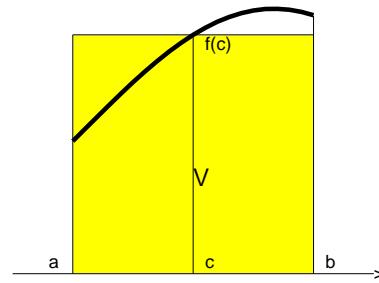
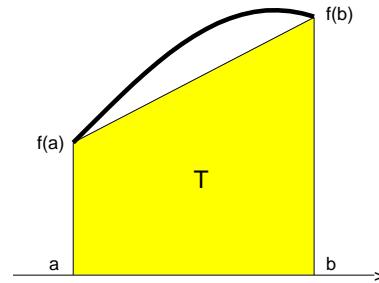
Se si indica con  $y = f(x)$  la funzione (non negativa) che rappresenta la curva, alla regione  $R$  in Figura 4.3 si dà il nome di **rettangoloide** di base l'intervallo  $[a, b]$ , relativo alla funzione  $y = f(x)$ , la cui area  $\mathcal{A}(R)$  è detta **integrale definito di  $f(x)$  tra  $a$  e  $b$**  ed è indicata con

$$\mathcal{A}(R) \equiv I[f] = \int_a^b f(x) dx \quad (4.1)$$

## 4.2 Formule di quadratura: primi esempi

Il metodo più naturale per approssimare l'area del rettangoloide  $R$  è quello di sovrapporre ad esso delle figure elementari e di calcolare l'area di ognuna di esse. Sia, ad esempio,  $R$  il rettangoloide di base l' intervallo  $[a, b]$ , relativo alla funzione  $y = f(x)$  mostrato in Figura 4.3.

Un primo modo per approssimare l'area  $\mathcal{A}(R)$  del rettangoloide  $R$  è quello di utilizzare un rettangolo  $U$  con lati di misura  $(b - a)$  (la base) e  $f(a)$  (l'altezza). Detta allora

Figura 4.5: Approssimazione di  $\mathcal{A}(R)$  mediante la formula del punto medioFigura 4.6: Approssimazione di  $\mathcal{A}(R)$  mediante la formula trapezoidale

$\mathcal{A}(U)$  l'area del rettangolo  $U$  si ha che

$$I[f] \equiv \mathcal{A}(R) \simeq \mathcal{A}(U) = f(a)(b-a) \quad (4.2)$$

Tale formula è detta **formula rettangolare** (Figura 4.4). Inoltre è possibile scegliere il rettangolo  $V$  con lati di misura  $(b-a)$  e  $f(c)$  dove  $c = (a+b)/2$  è il punto medio dell'intervallo  $[a, b]$ , ottenendo così la **formula del punto medio** (Figura 4.5):

$$I[f] \equiv \mathcal{A}(R) \simeq \mathcal{A}(V) = f(c)(b-a) \quad (4.3)$$

Un terzo metodo per approssimare l'area di  $R$  è quello di utilizzare un trapezio  $T$  con basi di misura  $f(a)$  e  $f(b)$  ed altezza  $(b-a)$ , cioè

$$I[f] \equiv \mathcal{A}(R) \simeq \mathcal{A}(T) = \frac{(b-a)}{2} [f(a) + f(b)] \quad (4.4)$$

Quest'ultima formula è detta **formula trapezoidale** ed è indicata anche con il simbolo  $T[f]$ . Come mostrato in Figura 4.6, la formula trapezoidale equivale a calcolare l'area al di sotto del segmento di retta passante per la coppia di punti  $(a, f(a))$ ,  $(b, f(b))$ .

Analogamente è possibile ricavare altre formule. Infatti la parabola passante per i punti  $(a, f(a))$ ,  $(b, f(b))$  e  $(c, f(c))$  (dove  $c$  è il punto medio di  $[a, b]$ ) ha equazione:

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

per cui calcolando l'area al di sotto di tale parabola si ottiene:

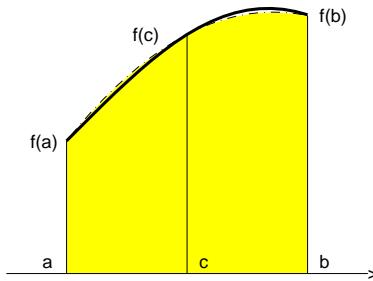
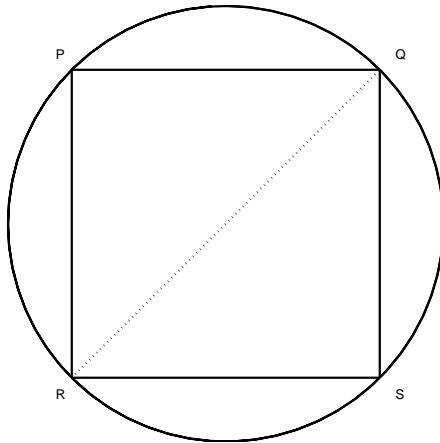
Figura 4.7: Approssimazione di  $\mathcal{A}(R)$  mediante la formula di Simpson

Figura 4.8: Decomposizione del cerchio in 5 sottoregioni

$$I[f] \equiv \mathcal{A}(R) \simeq \frac{(b-a)}{2} \left( \frac{f(a)}{3} + \frac{4}{3}f(c) + \frac{f(b)}{3} \right) \quad (4.5)$$

Tale formula è nota come **formula di Simpson**<sup>3</sup> (Figura 4.7) ed è indicata con  $S[f]$ .

♣ **Esempio 4.2.** Si voglia calcolare l'area del cerchio  $\Omega$  di raggio 1 in un sistema aritmetico floating point con base  $\beta = 10$  e precisione  $t = 5$  (si ricorda che il valore di tale area è  $\mathcal{A}(\Omega) = \pi = 3.1415926\dots$ ).

Il cerchio può essere diviso in 5 sottoregioni (Figura 4.8). Una di queste è un quadrato che ha come diagonale il segmento  $\overline{QR}$  lungo 2 e come lato un segmento la cui lunghezza è  $\sqrt{2}$ . L'area del quadrato è quindi 2. Le restanti sottoregioni sono calotte circolari uguali tra loro e la relativa area può essere calcolata con la formula trapezoidale. Una di tali sottoregioni è mostrata in Figura 4.9.

Se si considera il punto medio  $M$  di  $[P, Q]$  come origine del riferimento cartesiano si ha che i vertici  $P$  e  $Q$  hanno rispettivamente coordinate  $P \equiv (-\frac{\sqrt{2}}{2}, 0)$  e  $Q \equiv (\frac{\sqrt{2}}{2}, 0)$  e la curva che sottende il segmento  $\overline{PQ}$  è il grafico della funzione:

$$f(x) = \sqrt{1-x^2} - \frac{\sqrt{2}}{2}$$

<sup>3</sup>Nonostante il nome, tale formula fu proposta per primo da Cavalieri nel 1639 con il nome di *regola parabolica* e ripresa da James Gregory nel 1668. Solo nel 1743 Thomas Simpson rese famosa tale formula. Per tale motivo essa è nota anche con il nome di **formula di Cavalieri-Simpson**.

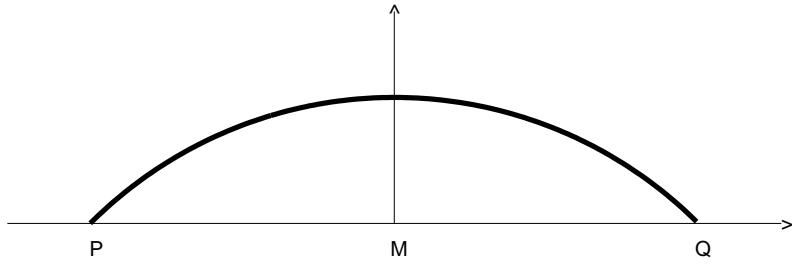


Figura 4.9: Area della calotta circolare

Tale regione è composta da due sottoregioni simmetriche relative ai due sottointervalli  $[P, M]$  e  $[M, Q]$ . È sufficiente quindi calcolare solo l'area della sottoregione relativa al segmento  $[M, Q]$ . Cioè

$$I[f] = \int_0^{\sqrt{2}/2} \left( \sqrt{1-x^2} - \frac{\sqrt{2}}{2} \right) dx = \frac{\pi - 2}{8} = 0.142699\dots$$

Utilizzando la formula trapezoidale (4.4) sull'intervallo  $[M, Q]$  risulta  $(b-a) = \sqrt{2}/2$  e

$$I[f] \simeq \mathcal{A}(T) = \frac{\sqrt{2}}{4} \left( 1 - \frac{\sqrt{2}}{2} \right) = 0.10355 \times 10^0$$

mentre l'errore assoluto commesso nell'approssimazione di  $I[f]$  con  $\mathcal{A}(T)$  nella sottoregione relativa al segmento  $[M, Q]$  è

$$|(\pi - 2)/8 - 0.10355 \times 10^0| \simeq 0.39149 \times 10^{-1}$$

Un'approssimazione dell'area del cerchio è quindi:

$$\mathcal{A}(\Omega) \simeq 2 + 8(0.10355 \times 10^0) = .28284 \times 10^1$$

e poiché l'area del cerchio di raggio 1 è  $\pi = 3.141592\dots$ , l'errore assoluto commesso nell'approssimare  $\mathcal{A}(\Omega)$  con il valore ottenuto, è

$$|\pi - .28284 \times 10^1| \simeq 0.31315 \times 10^0$$



Uno degli obiettivi che si pone la quadratura è quindi quello di determinare formule accurate per il calcolo di un integrale definito di una funzione  $y = f(x)$  (non solo non negativa). Le formule finora proposte (4.2), (4.3), (4.4), (4.5) per il calcolo di (4.1) sono una combinazione lineare di valori della funzione integranda  $f(x)$ , con coefficienti dipendenti dall'ampiezza dell'intervallo di integrazione. Si può dare quindi la seguente definizione:

#### Definizione 4.1. (Formula di quadratura)

Fissata una funzione integrabile nel senso di Riemann<sup>4</sup> ed un intero  $n$ , dati  $n$  punti  $x_i \in [a, b]$  detti **nodi** e  $n$  valori  $A_i$  detti **pesi**, la combinazione lineare:

$$Q[f] = A_1 f(x_1) + \cdots + A_n f(x_n)$$

prende il nome di **formula di quadratura**.

Una formula di quadratura rappresenta un'approssimazione dell'integrale  $I[f]$ , per cui è possibile dare anche la seguente definizione:

#### Definizione 4.2. (Errore di discretizzazione)

La differenza:

$$E[f] = I[f] - Q[f] = \int_a^b f(x) dx - \sum_{i=1}^n A_i f(x_i)$$

è l'**errore di discretizzazione della formula di quadratura**  $Q[f]$ .

In Figura 4.10 è fornita la rappresentazione grafica dell'errore di discretizzazione della formula trapezoidale per il calcolo di (4.1).

♣ **Esempio 4.3.** Si calcoli l'errore della formula trapezoidale relativo all'integrale:

$$I[f] = \int_1^2 (3x + 1) dx = 5.5$$

Applicando la formula trapezoidale si ha:

$$T[f] = (4 + 7)/2 = 5.5$$

In tal caso l'errore di discretizzazione della formula trapezoidale è nullo. Si può facilmente osservare che tale proprietà è vera per ogni polinomio  $p(x) \in \Pi_1$ , cioè  $E[p] = 0 \forall p \in \Pi_1$ . Per tale motivo si dice che la formula trapezoidale è **esatta per polinomi di primo grado**. ♣

---

<sup>4</sup>Si ricorda che una funzione è detta *integrabile nel senso di Riemann in  $[a,b]$*  se, detta  $P_1, P_2, \dots, P_m, \dots$  una generica successione di partizioni dell'intervallo  $[a, b]$  in sottointervalli  $[t_{j-1}, t_j]$ , tale che la successione dei sottointervalli di massima ampiezza  $\Delta_1, \Delta_2, \dots, \Delta_m, \dots$  converga a zero al crescere di  $m$ , le corrispondenti successioni di somme  $S_1[f], S_2[f], \dots, S_m[f], \dots$ :

$$S_m[f] = \sum_{j=1}^m f(\xi_j)(t_j - t_{j-1}) \quad \xi_j \in [t_{j-1}, t_j]$$

(chiamate somme di Riemann), hanno un limite comune  $S$  (che è proprio l'integrale definito della funzione, detto anche integrale di Riemann) che è indipendente dalla scelta della partizione e dei punti  $\xi_j$ , cioè:

$$\lim_{m \rightarrow \infty} S_m[f] = S = \int_a^b f(x) dx.$$

In particolare sono funzioni integrabili nel senso di Riemann le funzioni continue e le funzioni limitate con un numero finito di discontinuità. Nel seguito diremo semplicemente funzioni integrabili per indicare le funzioni integrabili nel senso di Riemann.

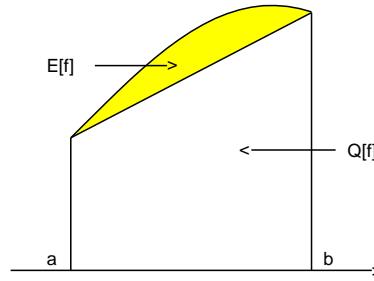


Figura 4.10: Rappresentazione grafica dell'errore di discretizzazione

### 4.2.1 Le formule di quadratura composite

Dalla Figura 4.10 è possibile notare che in generale l'errore di una formula trapezoidale dipende anche dall'ampiezza dell'intervallo  $[a, b]$ . Si può pensare allora di dividere l'intervallo  $[a, b]$  in  $m$  sottointervalli (talvolta chiamati *pannelli*) di uguale ampiezza  $\tau = (b - a)/m$  mediante i punti equidistanziati:

$$a = t_0 < t_1 < \dots < t_m = b \quad \text{con} \quad t_j = a + j\tau, \quad j = 0, \dots, m \quad (4.6)$$

e, poiché:

$$\int_a^b f(x) dx = \sum_{j=1}^m \int_{t_{j-1}}^{t_j} f(x) dx$$

si può utilizzare la formula trapezoidale in ognuno dei sottointervalli  $[t_{j-1}, t_j]$ <sup>5</sup>.

♣ **Esempio 4.4.** Si utilizzi la formula trapezoidale con 4 sottointervalli per calcolare l'integrale:

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

In questo caso l'ampiezza dei sottointervalli coincide con la distanza tra i nodi in cui viene valutata la funzione integranda, ed è  $h = \tau = t_j - t_{j-1} = \frac{b-a}{4}$  e per la proprietà additiva degli integrali risulta:

$$\int_a^b f(x) dx = \sum_{j=1}^4 \int_{t_{j-1}}^{t_j} f(x) dx \simeq \sum_{j=1}^4 \frac{(t_j - t_{j-1})}{2} [f(t_{j-1}) + f(t_j)] =$$

<sup>5</sup>L'idea di ridurre l'errore di una formula di quadratura applicando la stessa formula su più sottointervalli dell'intervallo di integrazione è basata sull'osservazione che, scelti  $m$  nodi  $t_j$  nell'intervallo  $[a, b]$  la somma di Riemann è una formula rettangolare composita in cui l'altezza del rettangolo non coincide con il valore della funzione nell'estremo inferiore dell'intervallo.

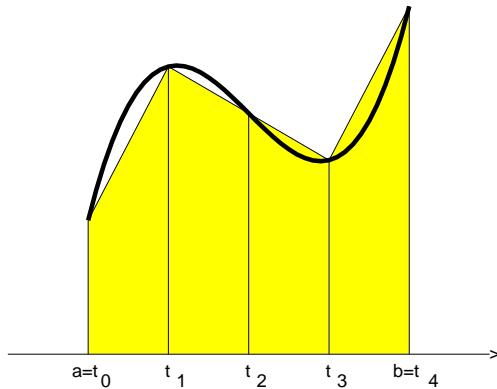


Figura 4.11: Formula trapezoidale composita su 4 sottointervalli

$$h \left[ \frac{f(t_0)}{2} + f(t_1) + f(t_2) + f(t_3) + \frac{f(t_4)}{2} \right]$$

In Figura 4.11 è fornita la rappresentazione grafica della formula di quadratura ottenuta utilizzando la formula trapezoidale nei 4 sottointervalli  $[t_0, t_1]$ ,  $[t_1, t_2]$ ,  $[t_2, t_3]$  e  $[t_3, t_4]$ , dell'intervallo  $[a, b]$ . ♣

♣ **Esempio 4.5.** Calcolare l'integrale (vedi esempio 4.2):

$$I[f] = \int_0^{\sqrt{2}/2} \left( \sqrt{1-x^2} - \frac{\sqrt{2}}{2} \right) dx$$

dividendo l'intervallo  $[0, \sqrt{2}/2]$  rispettivamente in 2 e 4 sottointervalli in cui si applica la formula trapezoidale. Facendo riferimento alla Figura 4.12, mediante la formula trapezoidale in un solo sottointervallo,  $T_1[f]$ , si è avuta un'approssimazione dell'area del rettangoloide relativo all'intervallo  $[M, Q]$  ed un errore  $E_1[f]$  rispettivamente di:

$$T_1[f] = 0.10355 \times 10^0,$$

$$|E_1[f]| = |I[f] - T_1[f]| \simeq 0.39149 \times 10^{-1}.$$

Se si utilizza invece la formula trapezoidale sui due sottointervalli  $[M, R]$  e  $[R, Q]$  con  $R$  punto medio di  $[M, Q]$  si ha un'approssimazione

$$T_2[f] = \frac{\sqrt{2}}{4} \left[ \frac{1}{2} \left( 1 - \frac{\sqrt{2}}{2} \right) + \left( \sqrt{\frac{7}{8}} - \frac{\sqrt{2}}{2} \right) \right] = 0.13250 \times 10^0$$

con un errore

$$|E_2[f]| = |I[f] - T_2[f]| = |(\pi - 2)/8 - 0.13250 \times 10^0| \simeq 0.10199 \times 10^{-1}.$$

Dimezzando ancora l'ampiezza dei sottointervalli  $[M, R]$  e  $[R, Q]$ , applicando la formula trapezoidale sui quattro sottointervalli,  $[M, S]$ ,  $[S, R]$ ,  $[R, T]$  e  $[T, Q]$ , con  $S$  punto medio di  $[M, R]$  e  $T$  punto medio

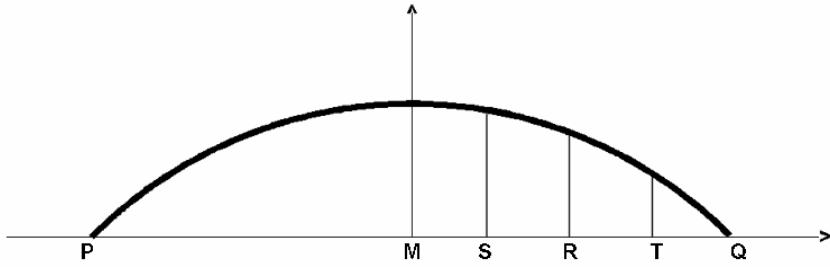


Figura 4.12: Area della calotta circolare

di  $[R, Q]$ , si ha:

$$\begin{aligned} T_4[f] &= \frac{\sqrt{2}}{8} \left[ \frac{1}{2} \left( 1 - \frac{\sqrt{2}}{2} \right) + \left( \sqrt{\frac{31}{32}} - \frac{\sqrt{2}}{2} \right) + \left( \sqrt{\frac{7}{8}} - \frac{\sqrt{2}}{2} \right) + \left( \sqrt{\frac{23}{32}} - \frac{\sqrt{2}}{2} \right) \right] \\ &= 0.14014 \times 10^0, \end{aligned}$$

$$|E_4[f]| = |I[f] - T_4[f]| = |(\pi - 2)/8 - 0.140140 \times 10^0| \simeq 0.25591 \times 10^{-2}.$$

Dall'esempio si può notare che dimezzando l'ampiezza dei sottointervalli l'errore si riduce di circa 4 volte. ♣

### Definizione 4.3. (Formula trapezoidale composita)

Suddiviso l'intervallo  $[a, b]$  in  $m$  sottointervalli di ampiezza

$$\tau = \frac{(b - a)}{m},$$

mediante i punti equidistanziati:

$$a = x_0 < x_1 < \dots < x_m = b \quad \text{con} \quad x_i = a + i\tau, \quad i = 0, \dots, m \quad (4.7)$$

la formula di quadratura  $T_m[f]$  che si ottiene applicando, in ognuno degli intervalli  $[x_{j-1}, x_j]$ , con  $j = 1, \dots, m$ , la formula trapezoidale  $T[f]$ , è detta **formula trapezoidale composita**. In questo caso la distanza tra i nodi  $h$  coincide con  $\tau$ , per cui, posto

$$x_i = a + ih, \quad i = 0, \dots, m - 1, \quad x_m = b,$$

si ha:

$$\int_a^b f(x) dx \simeq T_m[f] = h \left( \frac{f(x_0)}{2} + f(x_1) + \dots + f(x_{m-1}) + \frac{f(x_m)}{2} \right) \quad (4.8)$$

È possibile definire altre formule composite, come, ad esempio, la formula di Simpson composita  $S_m[f]$  su  $m$  sottointervalli  $[x_{j-1}, x_j]$  di ampiezza  $\tau = (b - a)/m$ . In tal caso, dalla (4.5) si ha che la distanza tra i nodi è  $h = \tau/2 = (b - a)/2m$ , per cui posto

$$x_0 = a \quad x_i = a + ih \quad i = 1, \dots, 2m$$

la formula di Simpson composita risulta:

$$S_m[f] = h \left( \frac{f(x_0)}{3} + \frac{4}{3}f(x_1) + \frac{2}{3}f(x_2) + \cdots + \frac{2}{3}f(x_{2m-2}) + \frac{4}{3}f(x_{2m-1}) + \frac{f(x_{2m})}{3} \right)$$

In generale si ha:

**Definizione 4.4. (Formula di quadratura composita)**

Fissata una formula di quadratura

$$Q[f] = \sum_{i=1}^n A_i f(x_i)$$

ed un intervallo  $[a, b]$ , si divida tale intervallo in  $m$  sottointervalli

$$[t_{j-1}, t_j] \quad (j = 1, \dots, m).$$

Si definisce  $Q_m[f]$  **formula di quadratura composita** la formula che si ottiene applicando  $Q[f]$  in ogni sottointervallo  $[t_{j-1}, t_j]$  di ampiezza  $\tau_j = (t_j - t_{j-1})$ , cioè:

$$Q_m[f] = \sum_{j=1}^m Q^{(j)}[f] = \sum_{j=1}^m \sum_{i=1}^n A_i^{(j)} f(x_i^{(j)}). \quad (4.9)$$

dove con  $Q^{(j)}[f]$  si è indicata la formula  $Q[f]$  nel  $j$ -mo intervallo  $[t_{j-1}, t_j]$ , con  $x_i^{(j)}$  si è indicato l' $i$ -mo nodo di  $Q^{(j)}[f]$  e con  $A_i^{(j)}$  l' $i$ -mo peso di  $Q^{(j)}[f]$ .

Si noti il ruolo differente che giocano gli  $n$  nodi  $x_i$  della Definizione 4.1 e gli  $m+1$  punti  $t_j$  della Definizione 4.4. I primi rappresentano gli  $n$  nodi di una formula di quadratura  $Q[f]$  mentre i secondi sono utilizzati per suddividere l'intervallo  $[a, b]$  in  $m$  sottointervalli in ognuno dei quali viene applicata la formula di quadratura fissata. In generale  $Q_m[f]$  ha  $mn$  nodi, tranne nel caso in cui il primo e l'ultimo nodo di  $Q[f]$  coincidano con gli estremi di ogni intervallo  $[t_{j-1}, t_j]$  (come ad esempio accade per la formula trapezoidale e per la formula di Simpson). In tal caso  $Q_m[f]$  possiede  $m(n - 1) + 1$  nodi.

**Definizione 4.5. (Formule convergenti)**

Data una funzione integrabile  $y = f(x)$ , la successione di formule composite  $\{Q_m[f]\}$  è detta **convergente** se

$$\lim_{m \rightarrow \infty} E_m[f] = 0 \Leftrightarrow \lim_{m \rightarrow \infty} Q_m[f] = \int_a^b f(x) dx$$

dove  $E_m[f]$  è l'errore dell' $m$ -ma formula di quadratura  $Q_m[f]$ .

L'esempio 4.5 ha messo in luce che l'errore di discretizzazione della formula trapezoidale composita si riduce quando si dimezza l'ampiezza dei sottointervalli  $h$ .

**Teorema 4.1.** Sia  $T[f]$  la formula trapezoidale e  $f(x)$  una funzione integrabile nell'intervallo  $[a, b]$ . Detta  $T_m[f]$  la formula composita su  $m$  sottointervalli di ampiezza  $\tau = (b - a)/m$  definita dalla (4.8) si ha:

$$\lim_{m \rightarrow \infty} E_m[f] = 0$$

dove  $E_m[f] = I[f] - T_m[f]$  è l'errore della formula trapezoidale composita  $T_m[f]$ .

**Dimostrazione** La formula trapezoidale  $T[f]$  applicata all'intervallo  $[t_{j-1}, t_j] \subseteq [a, b]$  di ampiezza  $\tau$  è:

$$\int_{t_{j-1}}^{t_j} f(x) dx \simeq \frac{(t_j - t_{j-1})}{2} [f(t_{j-1}) + f(t_j)]$$

La formula trapezoidale composita  $T_m[f]$  è allora:

$$\begin{aligned} T_m[f] &= \frac{1}{2} \sum_{j=1}^m (t_j - t_{j-1}) (f(t_{j-1}) + f(t_j)) \\ &= \frac{1}{2} \left[ \sum_{j=1}^m (t_j - t_{j-1}) f(t_{j-1}) + \sum_{j=1}^m (t_j - t_{j-1}) f(t_j) \right] \end{aligned}$$

Ma per l'integrabilità della funzione  $f(x)$  si ha

$$\lim_{m \rightarrow \infty} \sum_{j=1}^m (t_j - t_{j-1}) f(t_{j-1}) = \lim_{m \rightarrow \infty} \sum_{j=1}^m (t_j - t_{j-1}) f(t_j) = \int_a^b f(x) dx = I[f]$$

da cui:

$$\lim_{m \rightarrow \infty} E_m[f] = I[f] - \lim_{m \rightarrow \infty} T_m[f] = I[f] - \frac{1}{2} [2I[f]] = 0$$

cioè la tesi. ■

Questo tipo di comportamento di riduzione dell'errore dimostrato per la formula trapezoidale composita al crescere del numero di sottointervalli è comune ad altre formule composite<sup>6</sup>.

Un aspetto centrale della quadratura è fornire, oltre al valore dell'integrale  $I[f]$ , anche una stima dell'errore  $E[f]$  basata su quantità calcolabili.

♣ **Esempio 4.6.** Si determini una stima calcolabile dell'errore della formula trapezoidale composita. Preliminarmente si dimostra il:

**Teorema 4.2.** Sia  $f \in C^2[a, b]$  e

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

---

<sup>6</sup>La dimostrazione di ciò è data nel **Teorema 1.7** del §1.2.3, Capitolo 1 del Volume 2.

la formula trapezoidale. Detto  $E[f] = I[f] - T[f]$  l'errore della formula trapezoidale, si ha:

$$|E[f]| \leq \frac{5M(b-a)^3}{12} \quad M = \max_{x \in [a,b]} |f''(x)|.$$

**Dimostrazione** Siano

$$\begin{aligned} f(x) &= f(a) + f'(a)(x-a) + f''(\xi_1)(x-a)^2/2 & \xi_1 \in [a, x] \\ f(x) &= f(b) + f'(b)(x-b) + f''(\xi_2)(x-b)^2/2 & \xi_2 \in [b, x] \end{aligned}$$

le formule di Taylor per la funzione  $f(x)$  rispettivamente di punti iniziali  $a$  e  $b$ . Sommando membro a membro, dividendo per 2 e integrando termine a termine si ha:

$$\begin{aligned} I[f] &= \int_a^b f(x) dx = \int_a^b \frac{[f(a) + f(b)]}{2} dx + \\ &+ \int_a^b \frac{[f'(a)(x-a) + f'(b)(x-b)]}{2} dx + \int_a^b \frac{[f''(\xi_1)(x-a)^2 + f''(\xi_2)(x-b)^2]}{4} dx \end{aligned} \quad (4.10)$$

da cui, applicando la formula trapezoidale  $T[f]$  ai primi due integrali al secondo membro (le cui funzioni integrande sono polinomi di grado zero e di primo grado per cui la formula trapezoidale è esatta), si ha:

$$\begin{aligned} I[f] &= \int_a^b f(x) dx = \frac{(b-a)}{2}[f(a) + f(b)] + \\ &+ \frac{[f'(a)(b-a) + f'(b)(a-b)](b-a)}{4} + \int_a^b \frac{[f''(\xi_1)(x-a)^2 + f''(\xi_2)(x-b)^2]}{4} dx \end{aligned}$$

Da cui si ha:

$$E[f] = I[f] - T[f] = \frac{[f'(a) - f'(b)](b-a)^2}{4} + \int_a^b \frac{[f''(\xi_1)(x-a)^2 + f''(\xi_2)(x-b)^2]}{4} dx \quad (4.11)$$

Per il Teorema di Lagrange applicato alla funzione  $f'(x)$  è possibile trovare un punto  $\xi \in [a, b]$  tale che:

$$f'(b) - f'(a) = f''(\xi)(b-a)$$

da cui, sostituendo nella (4.11), si ha

$$E[f] = -\frac{f''(\xi)(b-a)^3}{4} + \int_a^b \frac{[f''(\xi_1)(x-a)^2 + f''(\xi_2)(x-b)^2]}{4} dx$$

Posto  $M = \max_{x \in [a,b]} |f''(x)|$  è allora possibile scrivere:

$$|E[f]| \leq \left| \frac{M(b-a)^3}{4} \right| + \frac{M}{4} \left| \int_a^b [(x-a)^2 + (x-b)^2] dx \right| \quad (4.12)$$

Per l'integrale a secondo membro della (4.12) con facili calcoli si ha:

$$\int_a^b [(x-a)^2 + (x-b)^2] dx = \frac{2(b-a)^3}{3}$$

e quindi:

$$|E[f]| \leq \frac{M(b-a)^3}{4} + \frac{M(b-a)^3}{6} = \frac{5M(b-a)^3}{12}$$

cioè la tesi. ■

**Corollario 4.1.** *Nelle stesse ipotesi del teorema precedente, detta  $T_m[f]$  la formula trapezoidale composita su  $m$  sottointervalli  $[t_{j-1}, t_j]$  di ampiezza  $h = \tau = (b-a)/m$  si ha che l'errore  $E_m[f] = I[f] - T_m[f]$  è:*

$$|E_m[f]| \leq \frac{5M(b-a)h^2}{12} \quad \text{con} \quad M = \max_{x \in [a,b]} |f''(x)|$$

**Dimostrazione** Per il Teorema 4.2, in ogni sottointervallo  $[t_{j-1}, t_j]$  di ampiezza  $h = \tau = (b-a)/m$  si ha:

$$|E[f]| \leq \frac{5Mh^3}{12}$$

e per ottenere una maggiorazione di  $E_m[f]$  basta sommare il contributo dell'errore in ogni sottointervallo  $[t_{j-1}, t_j]$ , cioè:

$$|E_m[f]| \leq \sum_{j=1}^m \frac{5Mh^3}{12} = m \frac{5Mh^3}{12} = \frac{5M(b-a)h^2}{12} \quad (4.13)$$

da cui la tesi. ■

Osserviamo che, dalla (4.13), se si dimezza la distanza tra i nodi  $h$ , l'errore si riduce di circa quattro volte. Detti allora  $E_m[f]$  ed  $E_{2m}[f]$  rispettivamente gli errori delle due formule trapezoidali composite  $T_m[f]$  e  $T_{2m}[f]$  assumiamo per il corollario del Teorema 4.2 che  $|E_m[f]| \simeq 4|E_{2m}[f]|$ . Approssimando  $I[f]$  con le due formule considerate si ha:

$$\begin{aligned} I[f] - T_m[f] &= E_m[f] \\ I[f] - T_{2m}[f] &= E_{2m}[f], \end{aligned}$$

da cui, sottraendo membro a membro e passando ai valori assoluti, supponendo che  $E_m$  ed  $E_{2m}$  abbiano lo stesso segno, si ha

$$|T_{2m}[f] - T_m[f]| = |E_{2m}[f] - E_m[f]| \simeq 3|E_{2m}[f]|$$

e quindi

$$|E_{2m}[f]| \simeq |T_{2m}[f] - T_m[f]|/3.$$

L'errore  $E_{2m}[f]$  può essere, quindi, stimato utilizzando la differenza tra le due formule  $T_m[f]$  e  $T_{2m}[f]$ . ♣

È utile osservare che, sfruttando la proprietà di convergenza delle formule composite, se ad esempio è stata già calcolata la formula trapezoidale composita  $T_m[f]$  per avere una stima più accurata dell'integrale (mantenendo l' equispaziatura tra i nodi) è possibile dimezzare la distanza tra i nodi,  $h$ , ottenendo così la formula  $T_{2m}[f]$  i cui nodi hanno distanza  $h/2$ . Tale scelta comporta da un lato un miglioramento dell'accuratezza del risultato e dall'altro l'utilizzo del valore di  $T_m[f]$  nel calcolo di  $T_{2m}[f]$ , richiedendo la valutazione della funzione integranda solo nei punti medi degli  $m$  sottointervalli  $[x_{j-1}, x_j]$  di  $[a, b]$ :

$$T_{2m}[f] = \frac{1}{2}T_m[f] + \frac{h}{2} \left[ f\left(a + \frac{h}{2}\right) + f\left(a + \frac{3h}{2}\right) + \dots + f\left(b - \frac{h}{2}\right) \right]. \quad (4.14)$$

In questo caso le due formule trapezoidali composite  $T_m[f]$  e  $T_{2m}[f]$  si dicono innestate.

### Definizione 4.6. (Formule innestate)

Due formule  $Q'[f]$  e  $Q''[f]$ , relative ad uno stesso intervallo, tali che l'insieme dei nodi di  $Q'[f]$  è contenuto nell'insieme dei nodi di  $Q''[f]$ , costituiscono una **coppia di formule innestate**.

Data quindi una coppia di formule innestate  $Q'[f]$  e  $Q''[f]$  l'errore  $E''[f] = I[f] - Q''[f]$  può essere stimato mediante la differenza:

$$|E''[f]| \simeq \alpha |Q'[f] - Q''[f]| \quad \alpha > 0 \quad (4.15)$$

dove  $\alpha$  è una costante che dipende dalle formule di quadratura utilizzate.

La disponibilità di formule innestate è particolarmente utile per ridurre la complessità computazionale delle formule di quadratura (e più in generale di un algoritmo per la quadratura). Infatti essa è determinata attraverso **il numero di valutazioni della funzione integranda** nei nodi  $x_i$ , perché il numero di operazioni relative a tale calcolo è predominante (la funzione integranda può infatti essere combinazione di polinomi algebrici, funzioni razionali, irrazionali, trigonometriche e così via).

## 4.3 Esempio di sviluppo di software matematico per la quadratura

### 4.3.1 L'algoritmo

I dati di un problema di quadratura sono la funzione integranda  $f$ , l'intervallo di integrazione  $[a, b]$  ed una tolleranza  $Tol$  che rappresenta l'accuratezza richiesta. Il modo più naturale per progettare un algoritmo di quadratura è quindi quello di prevedere come dati di input tali valori<sup>7</sup> e di produrre come output un valore  $Res$  (che sia un'approssimazione di  $I[f]$ ) con un errore  $Err$  (che rappresenti una stima attendibile dell'errore  $E[f]$ ) minore di  $Tol$ . Si noti quindi che un buon algoritmo di quadratura deve essere in grado di fornire, oltre ad una stima del valore dell'integrale, anche una stima dell'errore commesso.

Caratteristiche importanti di un tale algoritmo devono essere l'**affidabilità**, cioè la garanzia che il risultato soddisfi i vincoli della tolleranza richiesta, l'**efficienza**, cioè la capacità di calcolare il risultato con il minor numero possibile di valutazioni della funzione integranda e la **robustezza**, cioè la capacità dell'algoritmo di far fronte a situazioni anomale come ad esempio il caso di funzioni non integrabili.

**Strategia implementativa.** Al fine di soddisfare i vincoli richiesti sull'errore, un algoritmo per la quadratura può essere basato su uno schema iterativo che calcoli una successione convergente ad  $I[f]$ . Tale successione può essere costruita in base al

---

<sup>7</sup>La funzione integranda  $f$  va fornita all'algoritmo sotto forma di procedura.

Teorema 4.1, sfruttando cioè la proprietà di convergenza delle formule composite al crescere del numero di sottointervalli in cui si divide l'intervallo di integrazione  $[a, b]$ .

**Scelta della famiglia di formule.** Poiché l'efficienza maggiore si raggiunge quando l'algoritmo calcola un risultato soddisfacente i requisiti di accuratezza con il minor numero di valutazioni di  $f(x)$ , le formule composite utilizzate in algoritmi di quadratura sono tali che, ad ogni passo dell'algoritmo, è possibile riutilizzare tutte, o quasi, le informazioni ottenute nei passi precedenti, in modo da ridurre il numero di valutazioni della funzione integranda. In particolare le formule più utilizzate sono quelle di tipo innestato come, ad esempio, le formule trapezoidali composite

$$\{T_m[f]\}_{m=2^k}, \quad k = 0, 1, 2, \dots \quad (T_1[f] \equiv T[f])$$

ottenute raddoppiando ad ogni passo il numero di intervalli.

**Criterio di arresto.** La scelta del criterio di arresto di una successione convergente al valore dell'integrale è legata alla disponibilità di stime ragionevoli ed affidabili dell'errore associato alle formule di quadratura utilizzate. Si è detto che una stima per la formula trapezoidale composita è fornita dalla (4.15) con  $\alpha = 1/3$ . In questo caso un criterio di arresto può essere basato sulla relazione<sup>8</sup>:

$$|T_m[f] - T_{2m}[f]|/3 < \text{Tol} \quad (4.16)$$

Oltre a verificare se il risultato soddisfa la tolleranza (assoluta) richiesta `Tol`, il criterio di arresto dell'algoritmo deve prevedere un controllo sul numero di valutazioni della funzione integranda, il quale deve essere minore di una costante `Maxval` fornita in input, in quanto, come in tutti gli algoritmi iterativi, è possibile che un criterio di arresto basato solo sulla stima dell'errore sia soddisfatto solo dopo un numero eccessivo di passi. Una variabile di output `Iflag` segnala tale eventualità.

Tali considerazioni sono alla base della seguente Procedura 4.1.

---

<sup>8</sup>In generale i criteri di arresto delle procedure implementate nelle librerie di software matematico sono più sofisticati della (4.16) e spesso richiedono che l'errore relativo, o una combinazione dell'errore relativo e dell'errore assoluto sia minore di una tolleranza assegnata, cioè, ad esempio

$$\alpha|Q_m[f] - Q_{2m}[f]| < \max(tol_{abs}, tol_{rel}|Q_{2m}[f]|)$$

dove  $tol_{abs}$  e  $tol_{rel}$  sono rispettivamente la tolleranza richiesta per l'errore assoluto e per l'errore relativo. Controlli ulteriori possono essere effettuati anche sull'ampiezza degli intervalli. Infatti nel caso in cui la funzione integranda presenti delle singolarità, l'algoritmo può suddividere indefinitamente il dominio di integrazione senza generare un risultato soddisfacente la tolleranza richiesta.

```

procedure Trapez1(in:  $a, b, Tol, f, Maxval$  ; out:  $Int, Err, Iflag$ )
  /# SCOPO: calcolo di un integrale definito mediante la formula
  trapezoidale composita con stima dell' errore di
  discretizzazione

  /# SPECIFICHE PARAMETRI:
  /# PARAMETRI DI INPUT:
    var:  $a$  : reale {primo estremo dell' intervallo di integrazione}
    var:  $b$  : reale {secondo estremo dell' intervallo di integrazione}
    var:  $Tol$  : reale {tolleranza da soddisfare}
    var:  $f$  : reale {funzione integranda}
    var:  $Maxval$  : intero {massimo numero di valutazioni}

  /# PARAMETRI DI OUTPUT:
    var:  $Int$  : reale {stima dell' integrale}
    var:  $Err$  : reale {stima dell' errore di discretizzazione}
    var:  $Iflag$  : intero {indicatore d' errore}

```

Procedura 4.1: Algoritmo per la formula trapezoidale composita - continua

```

/# VARIABILI LOCALI:
var: h      : reale   {distanza tra i nodi}
var: x      : reale   {nodo attuale per il calcolo della formula}
                           {di quadratura}
var: oldint : reale   {formula trapezoidale composita}
                           {al passo precedente, necessaria}
                           {per il calcolo dell' errore}
var: sum     : reale   {somma parziale per la costruzione della}
                           {formula trapezoidale composita}
var: m       : intero  {numero dei nodi per la formula}
                           {trapezoidale composita}
var: fval    : intero  {numero di valutazioni della}
                           {funzione integranda}

/# INIZIO ISTRUZIONI:
m := 2;
h := (b - a)/m;
x := (a + b)/m;
oldint := h * (f(a) + f(b));
Int := oldint/2 + h * f(x);
fval := 3;
Err := |Int - oldint|/3;
while (Err>Tol and
           fval<Maxval) do          {ciclo principale di}
                           {iterazione con test sull'errore}
m := 2 * m;
h := h/2;
sum := 0.;

for k = 1, m, 2 do      {calcolo della formula trapezoidale}
                           {composita su m punti medi}
x := a + k * h;
sum := sum + f(x);

```

Procedura 4.1: Algoritmo per la formula trapezoidale composita - continua

```

 $fval := fval + 1;$ 
endfor
 $oldint := Int;$ 
 $Int := oldint/2 + h * sum;$ 
 $Err := |Int - oldint|/3$  {stima dell'errore }
endwhile
if ( $fval < Maxval$ ) then {definizione della variabile Iflag}
     $Iflag := 0;$ 
else
     $Iflag := 1;$ 
endif
end Trapez1

```

Procedura 4.1: Algoritmo per la formula trapezoidale composita - fine

### 4.3.2 Il software

Lo scopo di una routine per il calcolo di un integrale definito su un intervallo  $[a, b]$  è quello di rendere trasparente all'utente i vari dettagli relativi all'algoritmo implementato e di fornire il risultato a partire da un numero minimo di informazioni indispensabili da precisare in input. In particolare:

#### Definizione 4.7. (Integratore automatico)

*Si dice integratore automatico (Figura 4.13) una routine basata su un algoritmo che richieda almeno i parametri di input:*

- una routine per il calcolo di  $f(x)$ ;
- gli estremi dell'intervallo  $a$  e  $b$ ;
- una tolleranza  $Tol$  da soddisfare;
- un limite di valutazioni massime,  $Maxval$  per la funzione integranda;

*e fornisca almeno i parametri di output:*

- una stima  $Int$  di  $I[f]$  ed  $Err$  di  $E[f]$
- un indicatore  $Iflag$  che avvisi l'utente se è stata raggiunta la tolleranza  $Tol$  oppure il motivo per cui non è stata raggiunta.

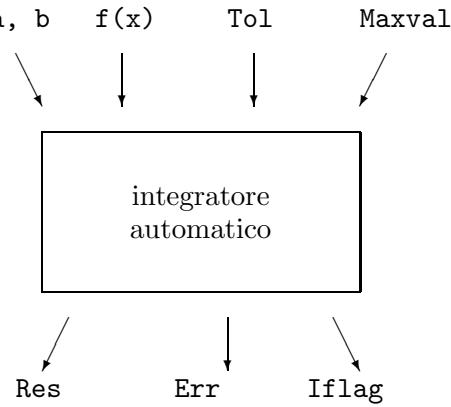


Figura 4.13: Schema di integratore automatico

A partire dalla procedura `Trapez1` è possibile costruire un elemento di software matematico per la quadratura. L'utente deve fornire un programma chiamante ed una routine `F(X)` che restituisca il valore della funzione integranda in ogni nodo previsto dalla formula di quadratura. Se, ad esempio, si utilizza il linguaggio di programmazione Fortran, la testata di tale subroutine può essere:

```
SUBROUTINE TRAPEZ1(A,B,F,TOL,MAXVAL,INT,ERR,IFLAG,FVAL)
```

Il programma chiamante `INTEG` illustra l'uso della subroutine `TRAPEZ1` nel caso si voglia integrare la funzione  $f(x) = \sqrt{x(1-x)}$  nell'intervallo  $[0, 1]$  con una tolleranza `TOL=0.00001` e `MAXVAL=10000`. Il parametro, `FVAL`, fornisce in output il numero di valutazioni di funzioni effettuate. Separatamente, nella `FUNCTION F(X)`, è implementata la funzione integranda. Tale unità di programma deve essere fornita dall'utente insieme al programma chiamante e nel caso si voglia calcolare l'integrale di una diversa funzione è sufficiente modificare solo l'istruzione `F=SQRT(X*(1.-X))` che definisce la funzione.

```

PROGRAM INTEG
C
C dichiarazione delle variabili
C
REAL A, B, TOL, INT, ERR, F
INTEGER FVAL, IFLAG, MAXVAL
EXTERNAL F
C
C inizializzazione delle variabili di input
C
A=0.
B=1.
TOL=.00001
MAXVAL=10000
C
C chiamata di TRAPEZ1
C
CALL TRAPEZ1(A,B,F,TOL,MAXVAL,INT,ERR,IFLAG,FVAL)
C
C stampa dei risultati
C
IF (IFLAG.eq.0) THEN
    PRINT*, 'Integrale   =' ,INT
    PRINT*, 'Errore      =' ,ERR
    PRINT*, 'Val. fun.   =' ,FVAL
    PRINT*, 'Iflag       =' ,IFLAG
ELSE
    PRINT*, 'tolleranza non raggiunta'
ENDIF
STOP
END
C
C FUNCTION F(X)
C
REAL FUNCTION F(X)
REAL X
F=SQRT(X*(1.-X))
RETURN
END

```

Esempio di programma chiamante FORTRAN per la subroutine TRAPEZ1

Un programma FORTRAN, che implementa la versione in singola precisione della subroutine TRAPEZ1 e del programma chiamante precedente, è stato compilato ed ese-

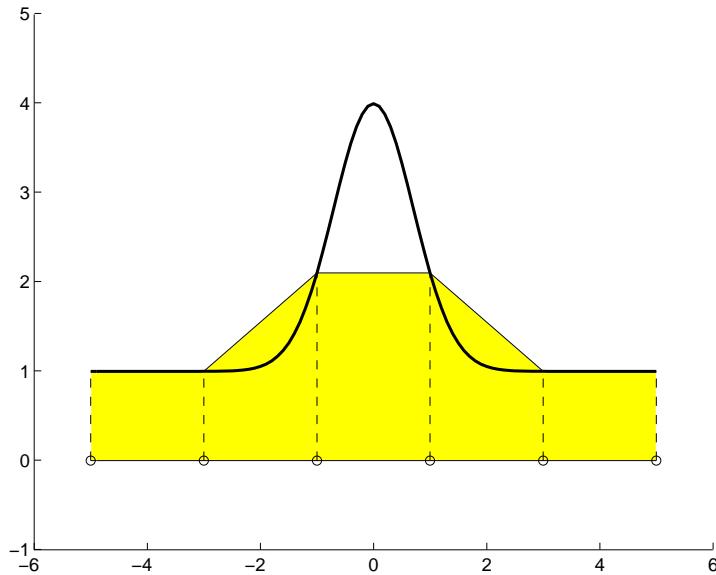


Figura 4.14: Approssimazione di  $I[f]$  mediante la formula trapezoidale composita su 5 sottointervalli dell' intervallo  $[-5, 5]$

guito su un calcolatore che utilizza il sistema aritmetico standard IEEE. L'esecuzione ha dato come risultati:

Integrale	= 0.3926973641
Errore	= 0.3695487976E-05
Val. Fun.	= 4097
Iflag	= 0

### 4.3.3 Gli algoritmi adattativi per la quadratura

Nel paragrafo 4.3.1 è stato sviluppato un algoritmo per la quadratura basato sulla famiglia di formule trapezoidaliali composite. Tale algoritmo dimezza successivamente l'ampiezza dei sottointervalli finché una stima dell'errore commesso non risulti minore di una tolleranza assegnata.

Qualunque sia la funzione integranda, l'algoritmo utilizza sempre la stessa distribuzione dei nodi nell'intervalllo  $[a, b]$  e ciò lo rende, in alcuni casi, poco efficiente.

♣ **Esempio 4.7.** Si calcoli

$$I[f] = \int_{-5}^5 (3e^{-x^2} + 1) dx$$

mediante la formula trapezoidale composita con 5 e 10 intervalli.

Nella Figura 4.14 si nota come la prima scelta dei nodi non tiene ben conto dell'andamento della funzione nell'intervallo  $[-1, 1]$ , mentre in Figura 4.15 si vede come raddoppiando il numero di intervalli non si migliora significativamente l'approssimazione di  $I[f]$  negli intervalli  $[-5, -2]$  e  $[2, 5]$ . ♣

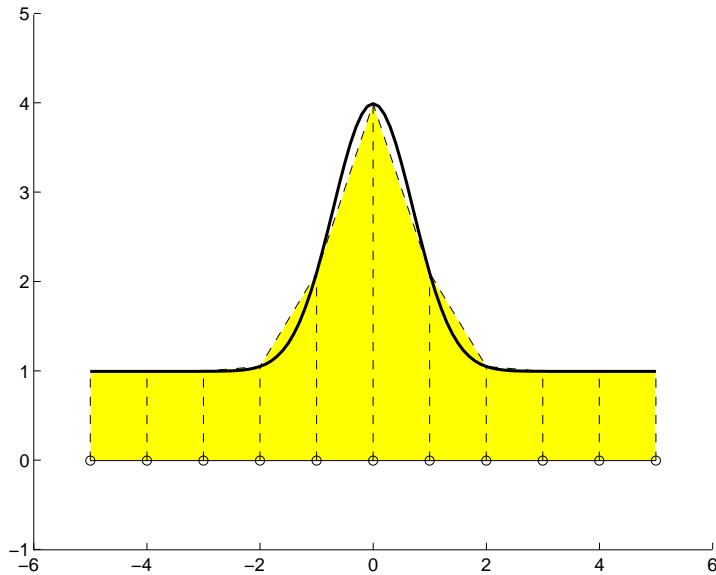


Figura 4.15: Approssimazione di  $I[f]$  mediante la formula trapezoidale composita su 10 sottointervalli dell' intervallo  $[-5, 5]$

L'esempio precedente mostra che la strategia di raddoppiare il numero degli intervalli della partizione di  $[a, b]$  è in genere poco efficiente, in quanto non tiene conto dell'andamento della funzione integranda.

Una valida alternativa consiste nel cercare di adattare la distribuzione dei nodi alle proprietà della funzione integranda.

#### Definizione 4.8. (Algoritmo adattativo)

*Un algoritmo adattativo per la quadratura è un algoritmo che sceglie dinamicamente (cioè durante l'esecuzione) la distribuzione dei nodi, in maniera da adattare il partizionamento dell' intervallo di integrazione al particolare andamento della funzione integranda. Un algoritmo in cui l'insieme dei nodi è scelto secondo uno schema fissato, indipendentemente dalla funzione integranda, è detto algoritmo non adattativo.*

Gli algoritmi non adattativi possono essere utilizzati efficacemente nel caso in cui sia noto l'andamento della funzione integranda, scegliendo preventivamente la distribuzione più adatta alla funzione da integrare. Viceversa gli algoritmi adattativi sono più utili nel caso in cui non si hanno informazioni sufficienti sull'andamento della funzione, poiché, in questo caso, è l'algoritmo stesso a scegliere in maniera opportuna i nodi. All'interno della classe degli algoritmi adattativi sono possibili due strategie: la strategia **adattativa globale** e la strategia **adattativa locale**.

La **strategia adattativa globale** opera come segue. Sia  $\varepsilon$  la tolleranza richiesta sull'intervallo iniziale  $\Delta = [a, b]$ ; vengono calcolati, in tale intervallo, l'integrale  $Q(\Delta)$  (ad esempio mediante la formula trapezoidale e la trapezoidale composita  $T_2[f]$ ) e l'errore  $E(\Delta)$  (mediante la (4.16) con  $m = 1$ ). Se  $E(\Delta) < \varepsilon$  allora l'algoritmo termina e  $Q(\Delta)$  è il risultato. In caso contrario si divide  $\Delta$  in due sottointervalli  $\Delta_1$  e  $\Delta_2$  nei quali vengono calcolati  $Q(\Delta_i)$  ed  $E(\Delta_i)$ ,  $i = 1, 2$ . Se  $E = E(\Delta_1) + E(\Delta_2) < \varepsilon$  allora si termina e  $Q = Q(\Delta_1) + Q(\Delta_2)$  è il risultato. In caso contrario si divide a metà il sottointervallo in cui  $E(\Delta_i)$  è maggiore. In generale al  $k$ -mo passo dell'algoritmo, l'intervallo  $[a, b]$  è diviso in  $k+1$  intervalli disgiunti di differente ampiezza. Se  $\sum E(\Delta_i) < \varepsilon$  l'algoritmo termina e  $\sum Q(\Delta_i)$  è il risultato. Altrimenti, viene diviso a metà il sottointervallo  $\overline{\Delta}$  con massima stima dell'errore, cioè:

$$|E(\overline{\Delta})| = \max_{i=1,\dots,k+1} |E(\Delta_i)|$$

La strategia adattativa globale può essere così riassunta:

```

Posto l' intervallo corrente  $\Delta = [a, b]$ , inizializza  $\varepsilon$ 
calcola un' approssimazione dell' integrale,  $Q(\Delta)$ ,
mediante una formula di quadratura opportuna e la formula di
quadratura composita
calcola  $E(\Delta)$  (mediante la (4.15), con  $\alpha$  opportuno)
while ( $E(\Delta) > \varepsilon$ )
    cerca il sottodomino  $\overline{\Delta}$  con stima dell'errore massima
    dividi  $\overline{\Delta}$  in 2 parti uguali
    calcola in tali parti una stima dell'integrale e dell'errore
    aggiorna  $Q(\Delta)$  e  $E(\Delta)$ 
endwhile
```

### Algoritmo adattativo globale

Nella **strategia adattativa locale** al primo passo vengono calcolati  $E(\Delta)$  e  $Q(\Delta)$  analogamente alla strategia globale. Se  $E(\Delta) < \varepsilon$  allora l'algoritmo termina e  $Q(\Delta)$  è il risultato. In caso contrario si divide  $\Delta$  in due sottointervalli  $\Delta_1$  e  $\Delta_2$  nei quali vengono calcolati  $Q(\Delta_i)$  ed  $E(\Delta_i)$ ,  $i = 1, 2$ . Si esamina dapprima l'intervallo  $\Delta_1$ . Fissata una tolleranza locale  $\varepsilon_1$  (generalmente  $\varepsilon_1 = \text{amp}(\Delta_1)\varepsilon/(b-a)$  con  $\text{amp}(\Delta_1)$  ampiezza dell'intervallo  $\Delta_1$ ), se  $E(\Delta_1) < \varepsilon_1$  si passa ad esaminare l'intervallo  $\Delta_2$ , altrimenti si divide ulteriormente  $\Delta_1$ . In generale, al passo  $k$ -mo l'intervallo  $\Delta = [a, b]$  è diviso in  $k+1$  sottointervalli. Detto  $\Delta_k$  l'intervallo in esame, se  $E(\Delta_k) < \varepsilon_k$  si passa ad esaminare  $\Delta_{k+1}$ , altrimenti si suddivide ulteriormente  $\Delta_k$ . Ogni volta che viene soddisfatto il criterio di tolleranza locale relativo al sottointervallo  $\Delta_k$  viene aggiunto  $Q(\Delta_k)$  ad una somma parziale che, al termine della procedura, fornirà il valore di  $I[f]$ . L'algoritmo termina quando l'ultimo intervallo soddisfa il criterio di tolleranza locale.

La strategia adattativa locale può essere così riassunta:

```

Posto l'intervallo corrente  $\Delta = [a, b]$ , inizializza  $\varepsilon$ 
calcola  $\overline{Q}(\Delta)$  mediante una formula di quadratura opportuna
repeat
    dividi l'intervallo corrente in 2 parti
    calcola  $Q(\Delta)$  mediante la formula di quadratura composita
    calcola  $E(\Delta)$  (mediante la (4.15), con  $\alpha$  opportuno)
    if ( $E(\Delta) > \varepsilon_{loc}$ ) then
        poni l'intervallo corrente  $\Delta =$  intervallo piu' a sinistra
    else
        somma  $Q(\Delta)$  al valore della somma parziale dell'integrale
        poni l'intervallo corrente  $\Delta =$  successivo intervallo
    endif
until ( intervalli tutti esaminati )

```

### Algoritmo adattativo locale

La Figura 4.16 mostra come la routine di MATLAB quad per la quadratura, basata su un algoritmo adattativo, scelga i nodi per la funzione

$$f(x) = 3e^{-x^2} + 1 \quad \text{nell'intervallo } [-2, 8].$$

Si nota che dove la funzione integranda presenta un picco, l'algoritmo utilizza dei sottointervalli di ampiezza inferiore rispetto ai sottointervalli in cui la funzione è pressoché costante.

Dal punto di vista implementativo la strategia globale può essere realizzata mediante l'uso di una lista ordinata<sup>9</sup> secondo le stime crescenti degli errori  $E(\Delta_i)$ , in cui vengono conservate le informazioni relative a tutti gli intervalli esaminati (estremi degli intervalli e valori di  $E(\Delta_i)$  e  $Q(\Delta_i)$ ). In questo modo l'intervallo da suddividere (cioè quello con massima stima dell'errore) è sempre in testa alla lista. Viceversa la strategia locale può essere realizzata mediante l'uso di una pila<sup>10</sup>, costruita inserendo durante ogni suddivisione i due intervalli così ottenuti nella testa della pila, prima quello di destra poi quello di sinistra; in tal modo l'intervallo da esaminare (quello più a sinistra) si trova sempre in testa alla pila. Entrambe le strategie presentano vantaggi e svantaggi, anche se le routine più efficienti attualmente esistenti sono basate sulla strategia globale.

<sup>9</sup>Una lista ordinata è una struttura dati dinamica (nel senso che la sua dimensione può variare durante l'esecuzione del programma). Tale struttura è composta da un insieme di nodi, ognuno dei quali formato da due campi: l'informazione e un puntatore al successivo nodo nella lista. Mediante il campo puntatore è possibile accedere ad ogni nodo della lista ed è possibile modificare l'ordine degli elementi solo intervenendo su tale campo.

<sup>10</sup>Una pila è una struttura dati dinamica in cui è possibile inserire ed estrarre gli elementi solo da un estremo della struttura dati. Tale estremo è detto *testa della pila*. Una struttura dati con tali caratteristiche è detta di tipo *Last In First Out* (LIFO).

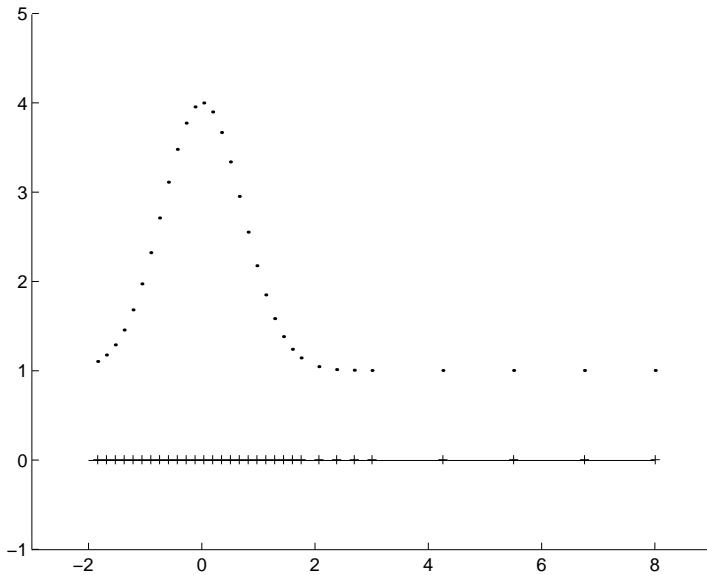


Figura 4.16: Scelta dei nodi da parte di una routine adattativa

Quest'ultima, infatti, a parità di tolleranza richiesta, utilizza generalmente un numero minore di valutazioni della funzione integranda rispetto alla strategia locale. Inoltre, se l'algoritmo termina per aver raggiunto il massimo numero di valutazioni della funzione integranda, è sempre disponibile una stima di  $I[f]$ . Diversamente, la strategia locale produce una stima dell'integrale della funzione solo fino all'intervallo che ha esaminato per ultimo procedendo dall'estremo inferiore dell'intervallo  $[a, b]$  verso quello superiore.

Tuttavia la strategia locale richiede un'occupazione minore di memoria per la pila, in quanto, se essa contiene ad esempio 50 sottointervalli, l'intervallo in testa alla pila ha ampiezza al più  $2^{-50}(b-a) \simeq 10^{-15}(b-a)$  che, per usuali valori di  $(b-a)$ , è un numero minore della precisione relativa dei sistemi floating point utilizzati dalla maggior parte dei calcolatori. In tal caso si ha che la pila non conterrà mai più di 50 intervalli. Per tale motivo la strategia locale ha avuto un maggior successo rispetto alla strategia globale negli anni '70, quando la capacità limitata della memoria dei calcolatori costituiva un serio vincolo alla dimensione dei programmi eseguibili. Con la tecnologia odierna tale problema è superato e la maggior parte degli algoritmi adattativi sono basati sulla strategia globale.

Esaminiamo ora due algoritmi adattativi per la quadratura. Il primo algoritmo, chiamato **Qlocal** (Procedura 4.2), fa uso della strategia locale. In particolare esso utilizza una procedura **Trap(c,d,Q)** che calcola l'approssimazione **Q** di  $I[f]$  in un generico intervallo  $[c, d]$  mediante la formula trapezoidale. L'algoritmo utilizza tre pile per conservare rispettivamente i valori di **Q** e dei relativi estremi degli intervalli. Nei linguaggi di programmazione dove non sono previste<sup>11</sup>, le tre pile possono essere realizzate con

<sup>11</sup>Ad esempio nel linguaggio FORTRAN 77 non è prevista l'allocazione dinamica della memoria e quindi tutte le strutture dati dinamiche come la pila, la lista e la coda non possono essere definite. Tale problema è stato superato nel linguaggio FORTRAN 90 che permette l'allocazione dinamica della

array monodimensionali, dove il primo elemento dell'array rappresenta il fondo della pila e l' $n$ -mo ne rappresenta la testa. L'algoritmo effettua inoltre un controllo sul massimo numero di valutazioni della funzione integranda e sulla minima ampiezza di ogni intervallo<sup>12</sup>.

---

memoria come accade nel linguaggio C.

<sup>12</sup>Si tenga presente che, lavorando in un sistema aritmetico a precisione finita, l'ampiezza dell'intervallo, in cui determinare una stima dell'errore e dell'integrale, è, almeno, uguale all'epsilon macchina relativo ad  $inf$ ,  $\epsilon \cdot |inf|$ , con  $inf$  estremo inferiore dell'intervallo corrente ed  $\epsilon$  l'epsilon macchina.

Si osserva, comunque, che i controlli descritti non sono gli unici possibili. Numerose routine prevedono ad esempio controlli sul tempo massimo di esecuzione del programma, sull'accumulo degli errori di round-off o sulla dimensione massima delle aree di lavoro utilizzate per le strutture dati.

```

procedure Qlocal(in:  $a, b, Tol, f, Maxvalminlen;$  out:  $Int, Err, Iflag$ )
  /# SCOPO: calcolo di un integrale definito mediante un algoritmo
  adattativo locale

  /# SPECIFICHE PARAMETRI:
  /# PARAMETRI DI INPUT:
    var:  $a$  : reale {primo estremo dell' intervallo di integrazione}
    var:  $b$  : reale {secondo estremo dell' intervallo di integrazione}
    var:  $Tol$  : reale {tolleranza da soddisfare}
    var:  $f$  : reale {funzione integranda}
    var:  $Maxval$  : intero {massimo numero di valutazioni}
    var:  $minlen$  : reale {minima ampiezza dell' intervallo}

  /# PARAMETRI DI OUTPUT:
    var:  $Int$  : reale {stima dell' integrale}
    var:  $Err$  : reale {stima dell' errore di discretizzazione}
    var:  $Iflag$  : intero {indicatore d' errore}

  /# VARIABILI LOCALI:
    var:  $Q1$  : reale {stima dell'integrale mediante la formula}
      {trapezoidale}
    var:  $Q2$  : reale {stima dell'integrale mediante la formula}
      {trapezoidale}
    var:  $Locerr$  : reale {errore di discretizzazione della formula}
      {trapezoidale composita}
    var:  $p1(100)$  : reale {array monodimensionale per la memorizzazione}
      {degli estremi sinistri degli intervalli}

```

Procedura 4.2: Algoritmo adattativo locale - continua

```

var: p2(100) : reale   {array monodimensionale per la memorizzazione}
                           {degli estremi destri degli intervalli}
var: p3(100) : reale   {array monodimensionale per la memorizzazione}
                           {dei risultati parziali}
var: inf      : reale   {estremo sinistro dell' intervallo corrente}
var: sup      : reale   {estremo destro dell' intervallo corrente}
var: res      : reale   {stima corrente dell' integrale definito}
var: newres   : reale   {stima dell' integrale mediante la formula}
                           {trapezoidale composita}
var: n        : intero  {dimensione delle liste}
var: nval     : intero  {numero di valutazioni di funzione}

 $\text{/}\# INIZIO ISTRUZIONI:}$ 
  Int := 0.;
  Err := 0.;

  call Trap(a, b, Q1)           {chiamata della routine}
  n := 1;                      {per la formula trapezoidale}
  nval := 3;
  p1(n) := a;                  {inizializzazione della pila}
  p2(n) := b;
  p3(n) := Q1;

  repeat                      {ciclo principale di iterazione}
    inf := p1(n);              {il sottointervallo in testa}
    sup := p2(n);              {alla pila viene suddiviso}
    res := p3(n);              {in due parti}
    n := n - 1;
    call Trap(inf, (inf + sup)/2, Q1);
    call Trap((inf + sup)/2, sup, Q2);
    newres := Q1 + Q2;
    Locerr:=|res - newres|/3;
    if (Locerr < Tol(sup - inf)/(b - a))          {test sull'}

```

Procedura 4.2: Algoritmo adattativo locale - continua

```

 $or(sup - inf) \leq minlen)$ then {accuratezza locale}
   $Int := Int + newres;$ 
   $Err := Err + Locerr;$ 
else
   $p1(n + 1) := (inf + sup)/2;$  {test sull'accuratezza}
   $p2(n + 1) := sup;$  {fallito: i due nuovi}
   $p3(n + 1) := Q2;$  {sottodomini vengono}
   $p1(n + 2) := inf;$  {inseriti nella pila}
   $p2(n + 2) := (inf + sup)/2;$ 
   $p3(n + 2) := Q1;$ 
   $n =: n + 2;$ 
endif
   $nval := nval + 6;$ 
until( $n = 0$  or  $nval \geq Maxval$ )
if( $n = 0$ )then {definizione della variabile}
   $Iflag := 0;$  {Iflag}
else
   $Iflag := 1;$ 
endif
return  $Int, Err, Iflag$ 
end Qlocal

```

## Procedura 4.2: Algoritmo adattativo locale - fine

Il secondo algoritmo, chiamato **Qglobal** (Procedura 4.3), è invece basato su una strategia globale. Esso utilizza una procedura **quadra(a,b,Q,E)** per la stima del valore di  $I[f]$  e dell'errore nell'intervallo  $[a, b]$  che può essere realizzata mediante una qualunque coppia di formule di quadratura. Per conservare i sottodomini ancora da esaminare, sono state utilizzate 4 liste contenenti gli estremi degli intervalli (**alist** e **blist**), le stime dell'integrale nei vari sottodomini (**qlist**) e le stime dell'errore (**elist**). Tali liste sono ordinate secondo i valori dell'errore presenti in **elist**, in maniera che le informazioni relative all'intervallo con massima stima dell'errore si trovino sempre nella testa della lista. Anche in questo caso, nei linguaggi dove non sono previste, le liste ordinate possono essere realizzate mediante array monodimensionali dove la testa è rappresentata dall' $n$ -mo elemento dell'array. Una procedura **sort(alist,blist,qlist,elist,n)**

è dedicata all'ordinamento. In realtà poichè si è interessati solo all'errore di massimo modulo, la procedura `sort` esegue solo la ricerca del massimo e lo posiziona nell' $n$ -mo elemento delle liste. L'algoritmo termina quando si è raggiunto il massimo numero di valutazioni della funzione integranda o quando la stima dell'errore globale è minore di `Tol`.

```

procedure Qglobal(in: a, b, Tol, f, Maxval ; out: Int, Err, Iflag)

/# SCOPO: calcolo di un integrale definito mediante un algoritmo
adattativo globale

/# SPECIFICHE PARAMETRI:
/# PARAMETRI DI INPUT:
var: a : reale {primo estremo dell' intervallo di integrazione}
var: b : reale {secondo estremo dell'intervallo di integrazione}
var: Tol : reale {tolleranza da soddisfare}
var: f : reale {funzione integranda}
var: Maxval : intero {massimo numero di valutazioni}

/# PARAMETRI DI OUTPUT:
var: Int : reale {stima dell' integrale}
var: Err : reale {stima dell' errore di discretizzazione}
var: Iflag : intero {indicatore d' errore}

/# VARIABILI LOCALI:
var: Q1 : reale {prima approssimazione dell'integrale}
{mediante la formula trapezoidale}
var: Q2 : reale {stima dell'integrale mediante}
{la formula trapezoidale}
var: c : reale {estremo sinistro dell' intervallo corrente}
var: d : reale {estremo destro dell' intervallo corrente}
var: Iold : reale {stima dell' integrale definito nel sottointervallo}
{in cui è massima la stima dell' errore}
var: Eold : reale {massima stima dell' errore}

```

Procedura 4.3: Algoritmo adattativo globale - continua

```

var: E           : reale   {errore di discretizzazione della formula}
                  {trapezoidale, sull' intervallo di integrazione}
var: E1          : reale   {errore di discretizzazione della formula}
                  {trapezoidale, sul sottintervallo sinistro}
var: E2          : reale   {errore di discretizzazione della formula}
                  {trapezoidale, sul sottintervallo destro}
var: alist(100)  : reale   {array monodimensionale per la memorizza-}
                  {zione degli estremi sinistri degli intervalli}
var: blist(100)  : reale   {array monodimensionale per la memorizza-}
                  {zione degli estremi destri degli intervalli}
var: qlist(100)  : reale   {array monodimensionale per la memorizza-}
                  {zione dei risultati parziali}
var: elist(100)  : reale   {array monodimensionale per la memorizza-}
                  {zione degli errori parziali}
var: n           : intero  {dimensione delle liste}
var: nval         : intero  {numero di valutazioni di funzione}

/# INIZIO ISTRUZIONI:

call quadra(a, b, Q1, E)  {prima approssimazione dell'integrale con la}
                           {formula trapezoidale mediante la}
                           {procedura quadra}
nval := 3;                 {Si suppone che la complessita'}
                           {computazionale di quadra sia}
                           {di tre valutazioni di funzione}
n := 1;                   {inizializzazione delle liste}
alist(n) := a;

```

Procedura 4.3: Algoritmo adattativo globale - continua

```

 $blist(n) := b;$ 
 $qlist(n) := Q1;$ 
 $elist(n) := E;$ 
 $Int := Q1;$ 
 $Err := E;$ 
while( $Err > Tol$  and  $nval < Maxval$ )do
     $c := alist(n);$ 
     $d := blist(n);$ 
     $Iold := qlist(n);$ 
     $Eold := elist(n);$ 
     $n := n - 1;$ 
    call quadra( $c, (c + d)/2, Q1, E1$ )
    call quadra( $(c + d)/2, d, Q2, E2$ )
     $Int := Int - Iold + Q1 + Q2;$            {aggiornam. dei valori dell'integrale}
     $Err := Err - Eold + E1 + E2;$           {e dell'errore}
     $nval := nval + 6;$ 
     $alist(n + 1) := c;$                    {i due nuovi sottointervalli}
     $blist(n + 1) := (c + d)/2;$           {vengono aggiunti alla lista}
     $qlist(n + 1) := Q1;$ 
     $elist(n + 1) := E1;$ 
     $alist(n + 2) := (c + d)/2;$ 
     $blist(n + 2) := d;$ 
     $qlist(n + 2) := Q2;$ 
     $elist(n + 2) := E2;$ 
     $n := n + 2;$ 

```

Procedura 4.3: Algoritmo adattativo globale - continua

```

call sort(alist, blist, qlist, elist, n)           {le liste vengono ordinate}
                                                {in base agli errori; il}
                                                {sottointervallo con massimo}
                                                {errore si trova nell'n-mo nodo}

endwhile

if(nval < Maxval)then                      {definizione della variabile Iflag}
    Iflag := 0;
else
    Iflag := 1;
endif
return Int, Err, Iflag
end Qglobal

```

Procedura 4.3: Algoritmo adattativo globale - fine

#### 4.3.4 MATLAB e la quadratura

MATLAB è un *Problem Solving Environment* (PSE), per il calcolo numerico e la visualizzazione di dati.

Originariamente scritto per la risoluzione di problemi di algebra lineare<sup>13</sup>, si è evoluto negli anni ed attualmente copre tutti i principali campi della matematica computazionale.

Per l'integrazione numerica, MATLAB mette a disposizione due function: `quad` e `quad8`. Entrambe le function sono basate su un algoritmo automatico di tipo adattativo con strategia locale. Come nucleo computazionale `quad` utilizza la formula di Simpson, mentre `quad8` utilizza la formula di Newton-Cotes con 9 nodi<sup>14</sup>.

Per far uso della funzione `quad` è possibile utilizzare una delle funzioni seguenti:

```
>> res = quad('fun',a,b)
>> res = quad('fun',a,b,tol)
>> res = quad('fun',a,b,tol,graf)
```

Tale funzione restituisce nella variabile `res` il valore calcolato di un integrale definito tra i limiti `a` e `b` della funzione `fun`. Quest'ultima deve essere resa disponibile a `quad` mediante un file di nome `fun.m`.

♣ **Esempio 4.8.** Si calcoli:

$$I[f] = \int_0^{2\pi} |\sin(x)| dx = 4$$

mediante la funzione `quad` e si valuti il numero di operazioni floating point effettuate. È necessario creare il file `fun.m` contenente le operazioni che definiscono la funzione integranda:

```
function y=fun(x)
y = abs(sin(x));
end
```

e quindi eseguire i comandi:

```
>> a=0.;
>> b=2*pi;
>> res = quad('fun',a,b)
res =
4.00000207024612
>> flops
ans =
840
```

<sup>13</sup>Il nome sta infatti per MATrix LABoratory.

<sup>14</sup>Le formule di Newton-Cotes sono trattate nel §1.1.3 del Cap. 1 del Volume 2.



In generale **quad** calcola il valore dell'integrale con una tolleranza pari a  $10^{-3}$ , a meno che non venga specificato il parametro opzionale **tol**.

♣ **Esempio 4.9.** Si calcoli l'integrale precedente, a meno di una tolleranza  $10^{-4}$ . In tal caso è necessario eseguire i comandi:

```
>> a=0. ;
>> b=2*pi;
>> tol=10^(-4);
>> res = quad('fun',a,b,tol)
res =
    4.00000013081371
>> flops
ans =
    1688
```

Si noti come l'accuratezza maggiore richieda un maggior numero di operazioni floating point. ♣

Infine, se si utilizza il parametro opzionale **graf**, assegnando ad esso un valore diverso da 0, **quad** visualizza un grafico con i valori della funzione nei nodi utilizzati dalla routine per il calcolo dell'integrale. L'utilizzo di **quad8** è analogo a quello di **quad**. Entrambe le funzioni hanno un limite di 10 suddivisioni per ogni sottointervallo onde evitare di raffinare indefinitamente un sottointervallo nel caso in cui esso contenga un'eventuale singolarità. In tal caso viene prodotto il messaggio:

Recursion level limit reached in quad. Singularity likely.

che indica la presenza di tale situazione. Conseguenza di ciò è che sia **quad** sia **quad8** non sono in grado di calcolare con efficienza, ad esempio, integrali del tipo:

$$I[f] = \int_0^1 \frac{1}{\sqrt{x}} dx$$

♣ **Esempio 4.10.** Si calcoli l'integrale dell'esempio 4.8 nell'intervallo  $[0, 5]$  con le due funzioni **quad** e **quad8**. Si valutino il numero di operazioni floating point e si mostrino anche i nodi utilizzati dalla funzione **quad8**. Per tale esempio si ha:

```

>> a=0.;
>> b=5;
>> tol=10^ (-4);
>> graf = 1;
>> flops(0);
>> res = quad('fun',a,b,tol)
Recursion level limit reached
in quad. Singularity likely.
Recursion level limit reached
in quad. Singularity likely.
res =
3.28366224139591
>> flops
ans =
1116
>> flops(0);
>> res = quad8('fun',a,b,tol,graf)
res =
3.28366221766898
>> flops
ans =
1903

```

La presenza del parametro `graf` nella chiamata di `quad8` fa sì che, oltre al risultato venga visualizzata anche la Figura 4.17.

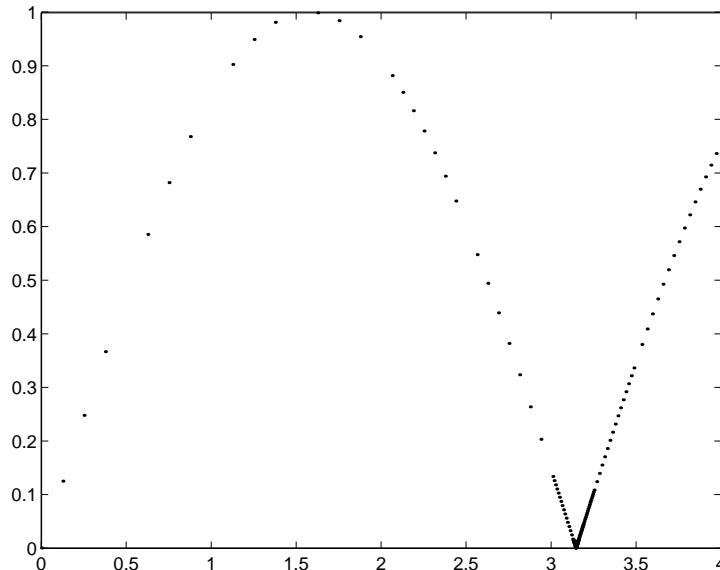


Figura 4.17: Generazione dei nodi da parte della funzione `quad8`

In questo caso la funzione `quad8` utilizza un maggior numero di nodi proprio in corrispondenza di  $\pi$  dove la derivata prima è singolare. Inoltre da un punto di vista computazionale la funzione `quad8` effettua un maggior numero di valutazioni di funzione rispetto a `quad`, ma è in grado di integrare con più accuratezza ed efficienza funzioni aventi discontinuità nella derivata prima. ♣



## 4.4 Computer problems sulla quadratura

**Esercizio 1** Sviluppare una subroutine Fortran TRAP1 per il calcolo dell'area sottesa da una funzione data  $f(x)$  nell'intervallo  $[a, b]$ , basata sulla formula trapezoidale composita su  $m$  sottointervalli, i cui parametri di input siano gli estremi dell'intervallo  $a$  e  $b$ , la funzione  $f(x)$  ed il numero di sottointervalli  $m$  ed il cui parametro di output sia il valore dell'area  $T$ .

Dato un cerchio di raggio  $r = 1$ , si utilizzi la subroutine per il calcolo dell'area della calotta circolare delimitata dalla circonferenza e da un lato del quadrato inscritto del cerchio. Sapendo che l'area della calotta è:

$$\text{Area} = \frac{\pi - 2}{4}$$

si utilizzino di seguito  $m = 1, 2, 4, 8, \dots, 1024$  sottointervalli e si osservi come si riduce l'errore ad ogni passo.

**Esercizio 2** Data una parabola intersecante l'asse delle ascisse nei punti  $P_1 = (0, 0)$  e  $P_2 = (4, 0)$  ed avente come vertice il punto  $P_2 = (2, 8)$ , ripetere l'esercizio 1 calcolando l'area racchiusa tra la parabola e l'asse delle ascisse. Si ricordi, che Archimede (287 a.c.) ha dimostrato che l'area di un segmento parabolico è uguale a  $4/3$  dell'area di un triangolo avente la stessa base e la stessa altezza del segmento parabolico.

**Esercizio 3** Si ripetano gli esercizi 1 e 2 sostituendo alla formula trapezoidale la formula rettangolare e la formula del punto medio. Si commentino i risultati.

**Esercizio 4** Si determini una stima calcolabile dell' errore relativo e si modifichi la subroutine TRAP1 dell'esercizio 1 in una subroutine TRAP2 in modo tale da sostituire il parametro di input relativo al numero di sottointervalli  $m$  con un parametro di input  $tol$  che rappresenti la tolleranza sull'errore relativo commesso ed aggiungendo i parametri di output  $E$ , che rappresenta una stima dell'errore relativo commesso, e  $V$ , che rappresenta il numero di valutazioni di funzioni effettuate.

Si calcoli quindi l'area della calotta circolare dell'esercizio 1 e l'area del segmento parabolico dell'esercizio 2 con tolleranze 0.001, 0.0001 e col la massima accuratezza del sistema aritmetico utilizzato, specificando il numero di valutazioni della funzione integranda.

**Esercizio 5** Mediante la subroutine TRAP2 sviluppata nell'esercizio 4, si calcoli l'area

al di sotto delle funzioni di equazione

- i)  $y = e^x \quad x \in [0, 1]$  (val. esatto = 1.718281...)
- ii)  $y = x^2 e^{-x^2} \quad x \in [0, 4]$  (val. esatto = 0.4431132...)
- iii)  $y = \log(x) \quad x \in [1, 10]$  (val. esatto = 14.02585...)
- iv)  $y = \sin(x^2) \quad x \in [0, 5]$  (val. esatto = 0.5279172...)
- v)  $y = \sqrt{(1-x^2)} \quad x \in [0, 1]$  (val. esatto = 1.570796...)
- vi)  $e^{-x^2} \quad x \in [-1, 1]$  (val. esatto = 1.493648...)
- vii)  $1/x^2 \quad x \in [0.1, 1]$  (val. esatto = 9.000000...)
- viii)  $\frac{\sin(e^x + \cos(x))}{(1+(1+x)\log(1+x))} \quad x \in [0.1, 1]$  (val. esatto = 1.5782736...)

con tolleranze 0.001, 0.0001 e con la massima accuratezza del sistema aritmetico floating-point utilizzato. Si fornisca anche una stima dell'errore relativo commesso e del numero di valutazioni di funzioni effettuate.

**Esercizio 6** Si sviluppi un integratore automatico **ADALOC** basato su un algoritmo adattativo con strategia locale che faccia uso della formula trapezoidale e si ripeta l'esercizio 5, confrontando il numero di valutazioni di funzioni effettuate dalle subroutine **TRAP2** e **ADALOC**.

**Esercizio 7** Si sviluppi un integratore automatico **ADAGLO** basato su un algoritmo adattativo, con strategia globale, che faccia uso della formula trapezoidale e si ripeta l'esercizio 5, confrontando il numero di valutazioni di funzioni effettuate dalle subroutine **TRAP2**, **ADALOC** e **ADAGLO**.

**Esercizio 8** Utilizzare le subroutine **ADALOC** e **ADAGLO** per calcolare l'area al di sotto delle funzioni nell'intervallo  $[0, 1]$ :

$$\begin{aligned} f_1(x) &= x^2(1.2 - x)(1 - e^{2(x-1)}) & \text{val. esatto} &= 0.00954996... \\ f_2(x) &= x^{0.1}(1.2 - x)(1 - e^{20(x-1)}) & \text{val. esatto} &= 0.602297... \end{aligned}$$

utilizzando la tolleranza  $tol = 0.001$  e la massima accuratezza del sistema aritmetico utilizzato. Si confronti il numero di valutazioni di funzioni effettuate.

**Esercizio 9** Ripetere l'esercizio 8, fornendo in output, la dimensione massima della pila utilizzata dalla subroutine **ADALOC** e della lista utilizzata dalla subroutine **ADAGLO**.

**Esercizio 10** Mediante la subroutine **ADALOC** si ripeta l'esercizio 8 fornendo, in output, gli estremi degli intervalli in cui la stima dell'errore soddisfa il criterio di tolleranza locale. Si raffronti il risultato con il grafico delle funzioni.

**Esercizio 11** Si ripeta l'esercizio 5 utilizzando le funzioni matlab **QUAD** e **QUAD8**. Si valuti la complessità computazionale con la funzione **FLOPS**

**Esercizio 12** Si scriva una funzione MATLAB **ADAGLO** basata sulla strategia adattativa globale che utilizzi la formula del punto medio. Tale funzione, oltre a fornire una stima del valore dell'area al di sotto della funzione data e dell'errore relativo commesso, disegni anche il grafico della funzione con i nodi utilizzati dall'algoritmo. Si calcolino nuovamente le aree dell'esercizio 8.

# Bibliografia

- [1] Chawla M. - *Convergence of Newton-Cotes quadratures for analytic functions* - BIT, vol. 11 (1971), pp. 159–167
- [2] Davis P. e Rabinowitz P. - *Methods of numerical integration* - Academic Press Inc. (1984)
- [3] Delves L. and Mohamed J. - *Computational methods for integral equations* - Cambridge University Press (1985)
- [4] Engels P. - *Numerical quadrature and cubature* - Academic Press Inc. (1980)
- [5] Evans G. - *Practical Numerical Integration* - Wiley (1993)
- [6] Fox J. et al. - *Solving problems on concurrent processors* - vol. 1, Prentice Hall, 1988.
- [7] Gautschi W. e Inglese G. - *Lower bounds for the condition number of Vandermonde matrices* - Numer. Math. vol. 52 (1988), pp. 241-250
- [8] Genz A. e Malik A. - *An inbedded family of fully symmetric numerical integration rule* - SIAM J. Num. Anal., vol. 20, No.3, June 1983, pp. 580–588
- [9] Hammersley J. e Handscomb D. - *Monte Carlo methods* - Methuen, 1964
- [10] Krommer A. e Ueberhuber C. - *Numerical Integration on advanced computer systems* - Springer Verlag (1994)
- [11] Krommer A.R. e Ueberhuber Ch.W. - *Computational integration* - SIAM (1998)
- [12] Kronrod A. - *Nodes and weights of quadrature formulas* - Consultants bureau, 1965
- [13] Krylov V. - *Approximate calculation of integrals* - Mc Millan Comp. (1962)
- [14] Malcom L. e Simpson R. - *Local versus Global strategies for adaptive quadrature* - ACM Trans. Math. Soft., vol. 1 (1975), pp. 129–146
- [15] Math Works Inc. - *MATLAB - Reference guide* - 1992
- [16] Monegato G. - *Stieltjes polynomials and related quadrature rules* - SIAM Review, vol. 24 (1982), pp. 137–158

- [17] Monegato G. - *Positivity of weights of extended Gauss-Legendre quadrature rules* - Math. Comp (1978)
- [18] Numerical Algorithm Group - *NAG library, Mark 17* - NAG - Oxford (1996)
- [19] Numerical Algorithm Group - *NAG Numerical Parallel library, release 2* - NAG - Oxford (1997)
- [20] Piessens R. et al - *QUADPACK, a subroutine package for automatic integration* - Springer Verlag (1983)
- [21] Sloan I. e Joe S. - *Lattice Methods for Multiple Integration* - Clarendon Press 1994
- [22] Szegö G. - *Orthogonal polynomials* - American Math. Soc., 1939
- [23] Stroud A.H. - *Approximate calculation of multiple integrals* - Prentice Hall (1971)
- [24] Visual Numerics - *IMSL Math Library version 2* - 1993
- [25] Ueberhuber Ch.W. - *Numerical Computation 2, Methods Software and Analysis* - Springer (1997)

# Capitolo 5

## Una introduzione alla risoluzione numerica di equazioni non lineari

### 5.1 Introduzione

Questo capitolo è dedicato allo studio ed all'analisi di algoritmi numerici per la risoluzione di equazioni non lineari. In generale, assegnata una funzione non lineare  $f : A \subseteq \mathbb{R} \rightarrow \mathbb{R}$ , una equazione non lineare è un'equazione del tipo:

$$f(x) = 0 . \quad (5.1)$$

Risolvere l'equazione (5.1) equivale a calcolare per quali valori di  $x \in [a, b] \subseteq A$  risulta  $f(x) = 0$ . Ogni valore di  $x$  per cui è soddisfatta la (5.1) è detto **radice dell'equazione** oppure **zero della funzione**  $f$ .

Spesso il modello matematico utilizzato per descrivere e risolvere problemi applicativi è costituito da equazioni non lineari e, generalmente, la difficoltà nel risolvere tali problemi si trasferisce direttamente alla difficoltà nella risoluzione dell'equazione non lineare associata. Si descrivono, nel seguito, alcuni esempi di problemi o applicazioni la cui risoluzione riconduce ad una equazione del tipo (5.1).

♣ **Esempio 5.1.** La legge che regola la crescita di una popolazione è del tipo

$$p(t) = \frac{p_M}{1 + (p_M/p_0)e^{-kp_M t}} \quad (5.2)$$

dove  $p(t)$  fornisce il numero di individui presenti al tempo  $t$ ,  $p_0$  rappresenta la popolazione all'istante iniziale  $p(t_0) = p_0$ ,  $p_M$  è il valore massimo raggiungibile dalla popolazione, mentre  $k$  è un prefissato fattore di crescita. Se si desidera conoscere in quale istante la popolazione raggiungerà un livello uguale a metà del massimo consentito, occorrerà risolvere l'equazione

$$\frac{p_M}{1 + (p_M/p_0)e^{-kp_M t}} - \frac{p_M}{2} = 0$$

nella variabile  $t$ .



♣ **Esempio 5.2.** Si supponga di disporre di un capitale di 1000 euro da investire. Si assuma, inoltre, che il tipo di investimento prescelto frutti un interesse netto dell'otto per cento annuo e non sia consentito per frazioni di anno. Si vuol conoscere il tempo di investimento necessario (in anni) perché il capitale raddoppi.

Si osservi che al termine del primo anno il capitale sarà cresciuto dell'otto per cento e quindi sarà uguale a  $1080 = 1000 \cdot 1.08$ . Tale capitale, investito durante il secondo anno frutterà a sua volta un interesse dell'otto per cento; dunque, al termine del secondo anno, il capitale sarà uguale a  $(1000 \cdot 1.08) \cdot 1.08 = 1000 \cdot (1.08)^2$ . In generale, al termine dell' $n$ -mo anno di investimento, il capitale ammonterà a  $1000 \cdot (1.08)^n$ . La risoluzione del problema si riconduce, quindi, a determinare il più piccolo intero  $\bar{n}$  per il quale si verifichi:

$$1000 \cdot 1.08^{\bar{n}} \geq 2000.$$

ovvero a determinare la soluzione  $\bar{x}$  dell'equazione

$$1000 \cdot 1.08^x - 2000 = 0$$

e calcolare successivamente  $\bar{n}$  come minimo intero maggiore o uguale a  $\bar{x}$ . <sup>1</sup>



♣ **Esempio 5.3.** In meccanica celeste, la posizione  $x$  di un pianeta nella sua orbita può essere determinata risolvendo un'equazione del tipo:

$$x - E \sin x - M = 0 \quad (5.3)$$

con  $E$  ed  $M$  costanti positive minori di 1.



♣ **Esempio 5.4.** Il problema della determinazione del minimo [massimo] di una funzione convessa [concava]  $f \in C[a, b]$  è equivalente a quello della risoluzione di un'equazione non lineare:

$$f'(x) = 0. \quad (5.4)$$

Pertanto un capitolo fondamentale dei metodi numerici di ottimizzazione è costituito dalla risoluzione di sistemi di equazioni non lineari.




---

<sup>1</sup>Si osservi che da continuità, crescenza e codominio della funzione  $f(x) = 1000 \cdot 1.08^x - 2000$  segue banalmente l'esistenza e unicità della soluzione del problema.

In tutti gli esempi precedenti l'equazione coinvolta è scalare. Nel caso generale, ovvero di una funzione di più variabili, cioè:

$$F : A \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^n, \quad n \geq 2,$$

l'equazione  $F(x) = 0$  rappresenta un sistema di equazioni non lineari, ed in tal caso la soluzione, quando esiste, è un vettore di  $n$  componenti.

♣ **Esempio 5.5.** Un esempio di sistema di equazioni non lineari è il seguente:

$$F(x_1, x_2) = \begin{pmatrix} x_1^2 + x_2^2 - 5 \\ x_1 + x_2 + 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Una soluzione di tale sistema, di due equazioni non lineari, definita su tutto  $\mathbb{R}^2$ , è  $x^* = (1, -2)$ . ♣

In seguito sarà considerato solo il caso scalare <sup>2</sup>.

L'esistenza ed il numero delle radici di equazioni non lineari scalari dipende non solo dalla funzione, ma anche dall'intervallo considerato. Ovvero, i dati del problema del calcolo delle radici di un'equazione non lineare sono la funzione  $f$  e l'intervallo di ricerca  $[a, b]$ .

♣ **Esempio 5.6.** Si consideri la seguente equazione non lineare scalare:

$$f(x) = x^4 - 12x^3 + 47x^2 - 60x = 0.$$

Le radici reali di tale equazione, definita su tutto  $\mathbb{R}$ , sono 4 e precisamente  $x^* = 0, 3, 4, 5$  (si osservi che la funzione considerata è un polinomio algebrico di quarto grado), tuttavia se si considera il problema di determinare le radici nell'intervallo  $[-1, +1]$  l'equazione ha una sola soluzione ( $x^* = 0$ ). Considerando invece l'intervallo  $[-1, 3.5]$  l'equazione ha due radici ( $x^* = 0, 3$ ). Mentre nell'intervallo  $[1, 2]$ , l'equazione non ha soluzioni. ♣

È importante osservare che anche quando si dispone dell'espressione analitica delle soluzioni, spesso non è possibile valutarla con un numero finito di operazioni.

---

<sup>2</sup>Tale scelta è giustificata dal fatto che la maggior parte dei metodi per la risoluzione numerica di equazioni non lineari scalari è estendibile al caso di sistemi.

♣ **Esempio 5.7.** L'equazione  $\cos(\log(x)) = 0$  ha infinite radici nell'intervallo  $(0, \infty)$ , precisamente:

$$x^* = e^{\pi/2 + k\pi}, \quad k \in \mathbb{N}.$$

Tuttavia una valutazione delle stesse comporta delle approssimazioni. ♣

♣ **Esempio 5.8.** Si consideri il polinomio di quinto grado

$$f(x) = x^5 - 2x^4 - x^3.$$

Per il Teorema Fondamentale dell'Algebra, l'equazione (algebrica, in questo caso) non lineare  $f(x) = 0$  ha 5 radici, e cioè i 5 zeri del polinomio. Per determinare tali soluzioni si può fattorizzare il polinomio nel prodotto di 2 polinomi, e precisamente  $f(x) = x^3(x^2 - 2x - 1)$ . L'equazione  $f(x) = 0$  è quindi decomposta in due equazioni non lineari,  $x^3 = 0$  e  $x^2 - 2x - 1 = 0$ , le cui soluzioni sono:  $x_1 = 0$ , con molteplicità 3,  $x_4 = 1 - \sqrt{2}$ ,  $x_5 = 1 + \sqrt{2}$ . Anche se nel caso del polinomio considerato si è riusciti a determinare gli zeri mediante una fattorizzazione in polinomi di grado più basso, in generale non esistono formule per calcolare gli zeri di polinomi di grado maggiore o uguale a 5 (Teorema di Abel). ♣

♣ **Esempio 5.9.** Si analizzi la funzione seguente:

$$f(x) = xe^{3x} + x^3 + \log(x+1) - 3.$$

Considerato l'intervallo  $[0, 1.5]$ , si ha che  $f(0) = -3$  e  $f(1.5) = 136.317$ , e quindi la funzione assume agli estremi valori di segno opposto. Inoltre, si ha che  $f'(x) = 3x \exp 3x + \exp 3x + 3x^2 + 1/(x+1) > 0$ ,  $x \in [0, 1.5]$ , cioè la funzione è strettamente crescente e, quindi, ha un unico zero nell'intervallo considerato. ♣

In generale, non esistono metodi che consentono di calcolare gli zeri di funzioni non lineari con un numero finito di operazioni. La soluzione può essere approssimata solo numericamente mediante la progettazione di opportuni algoritmi.

## 5.2 Il Metodo di Tabulazione

Un modo "naturale" di calcolare le soluzioni di  $f(x) = 0$  è di indagare sui valori che la funzione  $f$  assume in un insieme finito di punti nell'intervallo. Scelti ad esempio  $n$  punti  $t_1, \dots, t_n$ , analizzando i valori  $f(t_i)$ , si individuano i sottointervalli  $[t_i, t_{i+1}]$  in cui la funzione assume agli estremi valori di segno opposto. Tali sottointervalli possono contenere uno zero della funzione.

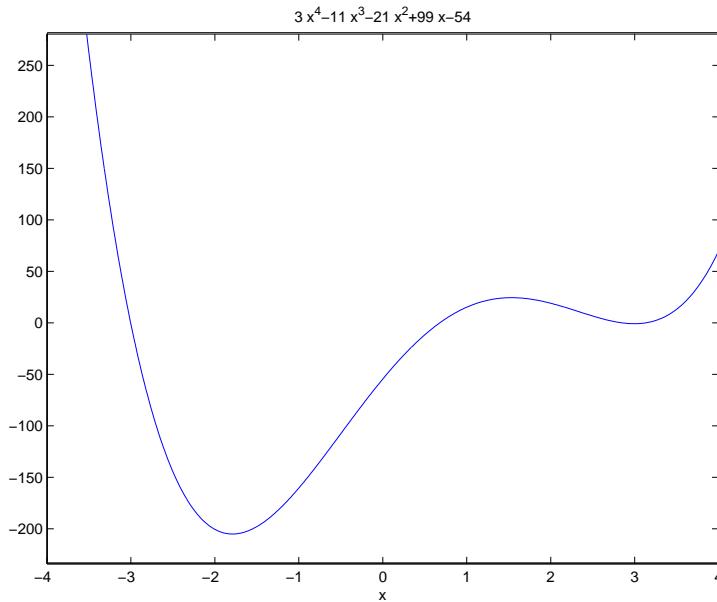


Figura 5.1: Grafico di  $f(x) = 3x^4 - 11x^3 - 21x^2 + 99x - 54$  in  $[-4, 4]$

♣ **Esempio 5.10.** Si consideri l'equazione seguente:

$$3x^4 - 11x^3 - 21x^2 + 99x - 54 = 0, \quad (5.5)$$

il cui grafico è mostrato in Fig. 5.1.

Le radici sono  $x^* = -3$ ,  $x^* = 2/3$  e  $x^* = 3$  (quest'ultima è radice doppia). Si vogliono trovare le radici nell'intervallo  $[-2, 2]$ . Si considerano  $n = 11$  punti distinti ed equidistanziati, cioè i punti

$$t_i = -2 + (i - 1) \cdot h, \quad i = 1, \dots, 11, \quad \text{con} \quad h = 0.4,$$

e si valuta la funzione  $f$  in tali punti (Fig. 5.2).

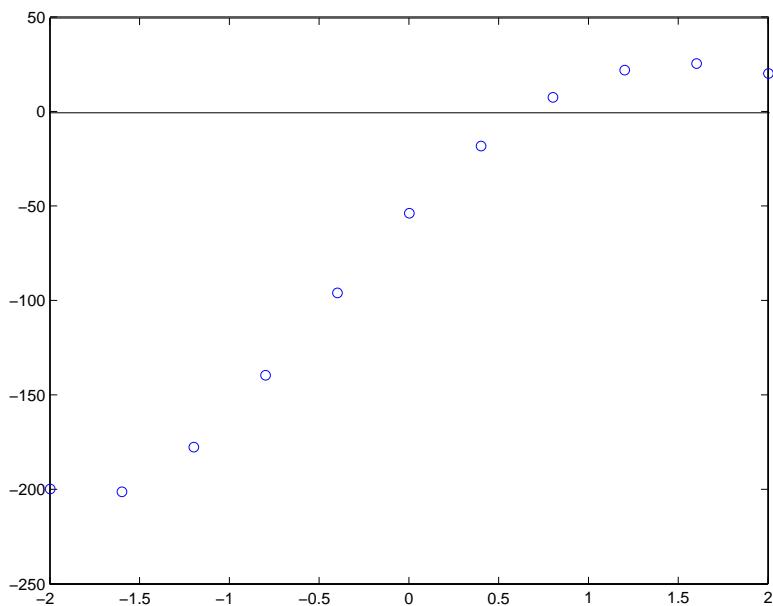
In Tabella 5.1 sono riportati gli 11 punti considerati nell'intervallo  $[-2, 2]$  e i valori corrispondenti della funzione, approssimati alla quarta cifra significativa. Osservando tali valori possiamo dire che l'equazione (5.5) potrebbe avere una radice nell'intervallo  $[0.4, 0.8]$  (qui la funzione assume valori di segno opposto agli estremi) (Fig. 5.3). Pertanto il punto medio dell'intervallo  $[t_7, t_8]$ ,  $x_1 = 0.6$ , può essere assunto come approssimazione della radice  $x^* = 0.666\dots$ .

Si può applicare di nuovo il procedimento appena descritto all'intervallo  $[0.4, 0.8]$ , considerando gli 11 punti

$$t_i = 0.4 + (i - 1) \cdot h, \quad i = 1, \dots, 11, \quad \text{con} \quad h = 0.04.$$

I valori che la funzione assume in tali punti, riportati in Tabella 5.2, mostrano che essa potrebbe avere uno zero nell'intervallo  $[0.64, 0.68]$ .

Se si arresta il procedimento a questo passo, si può assumere come approssimazione della radice dell'equazione (5.5) il punto medio dell'intervallo  $[t_7, t_8]$ , cioè  $x_2 = 0.66$ , ottenendo, così un'accuratezza

Figura 5.2: 11 valori calcolati di  $f(x) = x^4 - 11x^3 - 21x^2 + 99x - 54$  in  $[-2, 2]$ 

$x$	$f(x)$
-2.000e+01	-2.000e+02
-1.600e+01	-2.014e+02
-1.200e+01	-1.778e+02
-8.000e+00	-1.397e+01
-4.000e+00	-9.617e+01
0.000e+00	-5.400e+01
4.000e+00	<b>-1.838e+00</b>
8.000e+00	<b>7.35e+01</b>
1.200e+01	2.177e+01
1.600e+01	2.524e+01
2.000e+01	2.000e+01

Tabella 5.1: Tabulazione di  $f(x) = x^4 - 11x^3 - 21x^2 + 99x - 54$  in  $[-2, 2]$

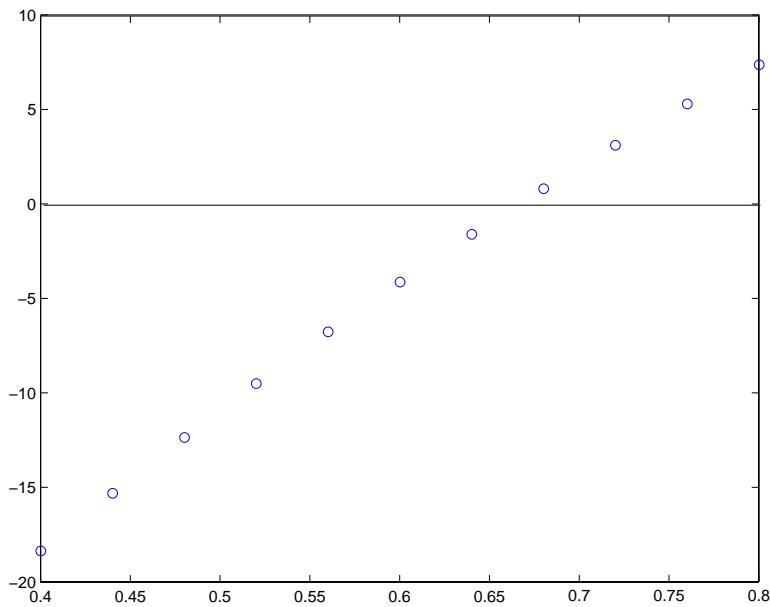


Figura 5.3: 11 valori calcolati di  $f(x) = 3x^4 - 11x^3 - 21x^2 + 99x - 54$  in  $[0.4, 0.8]$

$x$	$f(x)$
.4000e+00	-1.838e+01
.4400e+00	-1.5339e+01
.4800e+00	-1.237e+01
.5200e+00	-9.525e+00
.5600e+00	-6.782e+00
.6000e+00	-4.147e+00
.6400e+00	<b>-1.621e+00</b>
.6800e+00	<b>7.922e-01</b>
.7200e+00	3.094e+00
.7600e+00	5.282e+00
.8000e+00	7.356e+00

Tabella 5.2: Tabulazione di  $f(x) = 3x^4 - 11x^3 - 21x^2 + 99x - 54$  in  $[0.4, 0.8]$

$x$	$f(x)$
0.6400e+01	-1.621e+00
0.6440e+01	-1.375e+00
0.6480e+01	-1.130e+00
0.6520e+00	-8.858e-01
0.6560e+00	-6.427e-01
0.6600e+01	-4.008e-01
0.6640e+01	<b>-1.599e-01</b>
0.6680e+00	<b>7.978e-02</b>
0.6720e+01	3.184e-01
0.6760e+01	5.559e-01
0.6800e+01	7.922e-01

**Tabella 5.3:** Tabulazione di  $f(x) = 3x^4 - 11x^3 - 21x^2 + 99x - 54$  in  $[0.64, 0.68]$ 

di 2 cifre significative. Se si desidera una accuratezza maggiore si può iterare il procedimento a partire dall'intervallo  $[0.64, 0.68]$  (vedi Tabella 5.3).



Il procedimento utilizzato nell'esempio è detto **Metodo di Tabulazione** e rappresenta il modo più semplice di calcolare una approssimazione dello zero di una funzione in un intervallo.

♣ **Esempio 5.11.** Si consideri l'equazione seguente:

$$x^2 - 0.09 = 0, \quad (5.6)$$

il cui grafico è mostrato in Fig. 5.4.

Le radici sono  $x^* = -0.3$  e  $x^* = 0.3$ . Si vogliono trovare le radici nell'intervallo  $[-4, 2]$ . Si considerano  $n = 5$  punti distinti ed equidistanziati, cioè i punti

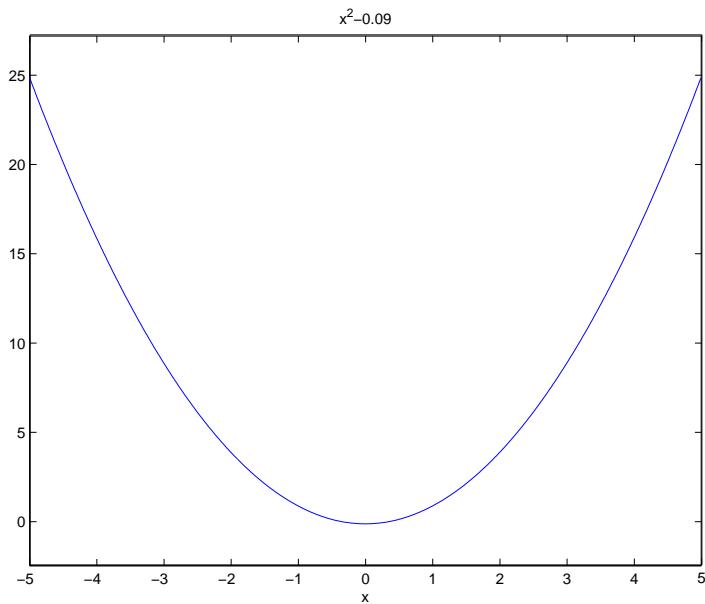
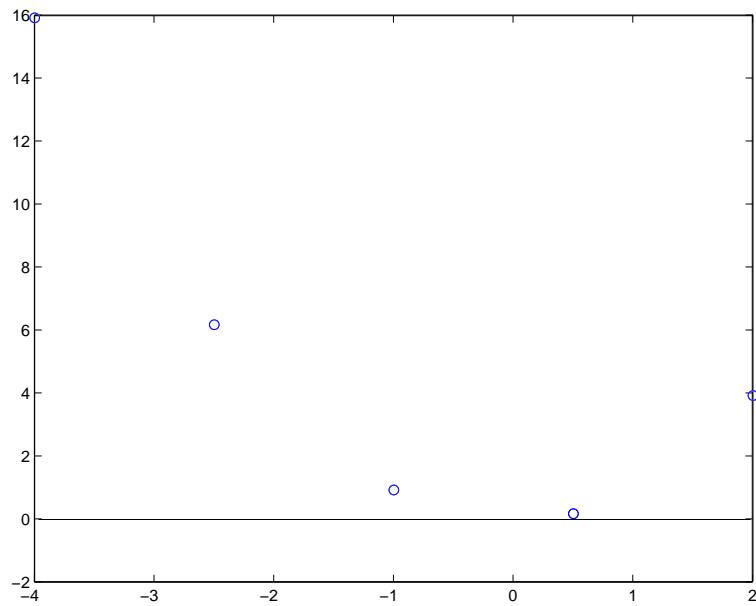
$$t_i = -4 + (i - 1) \cdot h, \quad i = 1, \dots, 5, \quad \text{con} \quad h = 1.5,$$

e si valuta la funzione  $f$  in tali punti (Fig. 5.5).

In Tabella 5.4 sono riportati i 5 punti considerati nell'intervallo  $[-4, 2]$  ed i valori corrispondenti della funzione, approssimati alla quarta cifra significativa. Osservando tali valori possiamo dire che il Metodo di Tabulazione non rileva alcun intervallo in cui la funzione assume valori di segno opposto agli estremi.

In questo caso si può applicare di nuovo il Metodo di Tabulazione aumentando ad esempio il numero di punti di tabulazione, ovvero considerando gli  $n = 9$  punti

$$t_i = -4 + (i - 1) \cdot h, \quad i = 1, \dots, 9, \quad \text{con} \quad h = 0.75.$$

Figura 5.4: Grafico di  $f(x) = x^2 - 0.09$  in  $[-5, 5]$ Figura 5.5: 5 valori calcolati di  $f(x) = x^2 - 0.09$  in  $[-4, 2]$

$x$	$f(x)$
-4.000e+01	1591e+02
-2.500e+01	6160e+01
-1.000e+01	9100e+00
5.000e+00	1600e+00
2.000e+01	3910e+01

**Tabella 5.4:** Tabulazione di  $f(x) = x^2 - 0.09$  in  $[-4, 2]$  (5 punti)

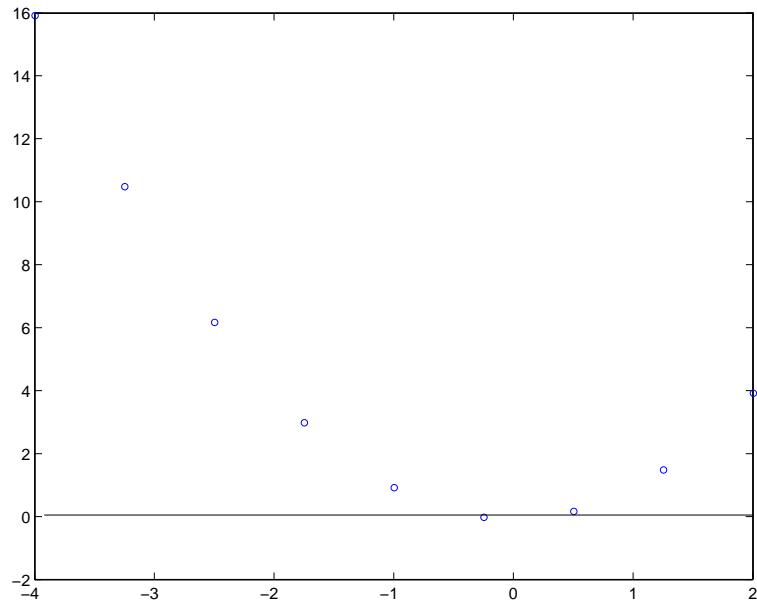


Figura 5.6: 9 valori calcolati di  $f(x) = x^2 - 0.09$  in  $[-4, 2]$

$x$	$f(x)$
-4000e+01	.1591e+02
-3250e+01	.1047e+02
-2500e+01	.6160e+01
-1750e+01	.2972e+01
-1000e+01	.9100e+00
-2500e+00	-0.2750e-01
.5000e+00	. 1600e+00
.1250e+01	.1472e+00
.2000e+01	.3910e+01

**Tabella 5.5: Tabulazione di  $f(x) = x^2 - 0.09$  in  $[-4, 2]$  (9 punti)**

I valori che la funzione assume in tali punti, riportati in Tabella 5.5, mostrano che il metodo individua due intervalli dove la funzione assume segno discorde agli estremi, ed in essi potrebbe avere uno zero (Fig. 5.6).

Se si arresta il procedimento a questo passo, si può assumere, come approssimazione della radice dell'equazione (5.6), sia il punto medio dell'intervallo  $[t_5, t_6]$ , cioè  $x_2 = -0.625$ , sia il punto medio dell'intervallo  $[t_6, t_7]$ , cioè  $x_2 = 0.125$ . A questo punto, se si desidera un'accuratezza maggiore, a partire da uno dei due intervalli, ad esempio l'intervallo più a sinistra  $[t_5, t_6]$ , si può iterare il procedimento. ♣

L'esempio 5.11 ha evidenziato che il Metodo di Tabulazione non sempre è in grado di determinare al primo passo un intervallo che contiene zeri della funzione. In tal caso può essere opportuno effettuare una tabulazione più fitta. Di seguito si riporta una sintesi del Metodo di Tabulazione che tiene conto di questa possibilità.

Ad ogni passo  $k$ , indicato con  $[a_k, b_k]$  il generico intervallo corrente, si ha:

- scelta di  $n$  punti distinti in  $[a_k, b_k]$ , in cui valutare  $f$ :

$$a_k = t_1 < \dots < t_n = b_k$$

ovvero il **processo di tabulazione**;

- scelta, tra i sottointervalli  $[t_i, t_{i+1}]$  (per  $i = 1, \dots, n-1$ ), dell'intervallo (più a sinistra<sup>3</sup>) dove si verifica una variazione di segno, cioè

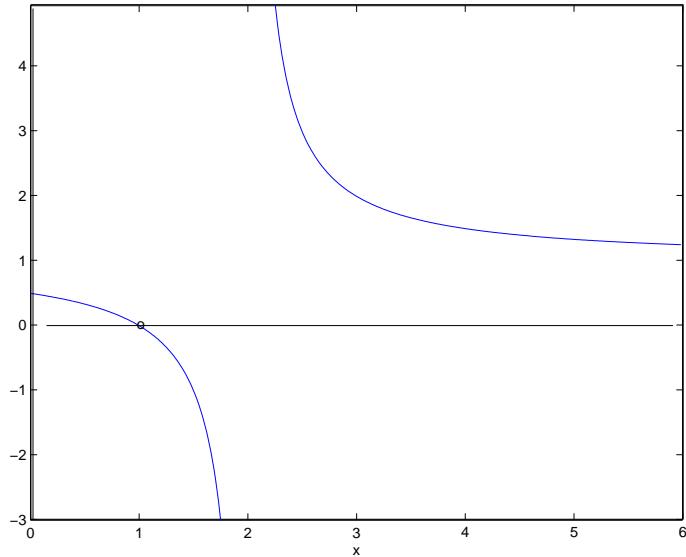
– se  $s = \min\{i \in \{1, \dots, n-1\} \text{ t.c. } f(t_i) * f(t_{i+1}) \leq 0\}$ , allora

$$[a_{k+1}, b_{k+1}] \equiv [t_s, t_{s+1}]$$

e si assume come approssimazione dello zero  $x_{k+1} = (a_{k+1} + b_{k+1})/2$

---

<sup>3</sup>Può essere scelto, in modo arbitrario, un qualsiasi altro intervallo.

Figura 5.7: Grafico di  $f(x) = \frac{x-1}{x-2}$  in  $[0, 6]$ 

- altrimenti, se la funzione non cambia segno in nessun intervallo, si effettua una nuova tabulazione in  $[a_k, b_k]$  con  $m > n$  punti.

### 5.2.1 Applicabilità del Metodo di Tabulazione

In questo paragrafo siamo interessati ad individuare le condizioni che garantiscono al metodo di fornire un risultato (a prescindere dall'accuratezza dello stesso) ovvero che garantisca l'applicabilità del metodo. Si osserva che il metodo in esame può essere sempre applicato a funzioni che siano valutabili su tutto l'intervallo di ricerca. Se tale condizione non è verificata, nulla si può dire sulla sua applicabilità.

♣ **Esempio 5.12.** Consideriamo l'equazione:

$$f(x) = \frac{x-1}{x-2} = 0$$

La funzione  $f$  ha uno zero in  $x^* = 1$  e non è definita in  $\hat{x} = 2$ . Applicando un passo del Metodo di Tabulazione in  $[0, 6]$ , con  $n = 5$ , si ottengono i risultati mostrati nella Tabella 5.6.

Se analizziamo i risultati in tabella si deduce che la funzione cambia segno in  $[0, 1.5]$ , ed una prima approssimazione della soluzione è data dal punto medio di tale intervallo,  $x_1 = 0.75$ .

Tuttavia se vogliamo applicare il Metodo di Tabulazione alla stessa funzione  $f$ , cambiando da 5 a 7 il numero di punti di tabulazione, già alla prima iterazione il metodo diventa non applicabile. Infatti i punti della tabulazione sono  $0, 1, 2, 3, 4, 5, 6$  e la funzione non è definita in uno dei punti della tabulazione.



$x$	$f(x)$
.0000e+00	.5000e+00
.1500e+00	-.1000e+01
.3000e+00	.2000e+01
.4500e+00	.1400e+01
.6000e+01	.1250e+01

**Tabella 5.6:** Un passo del Metodo di Tabulazione, con  $n = 5$ , applicato alla funzione in Fig. 5.7

### 5.2.2 Convergenza

Poichè il Metodo di Tabulazione è un metodo iterativo, ovvero a partire da valori iniziali,  $a$  e  $b$ , genera una successione di valori  $\{x_k\}_{k \in N}$ , è importante studiare le condizioni che assicurano che tale successione di valori "si avvicini sempre di più" alla soluzione del problema  $x^*$ , ovvero si vuole studiare la **convergenza** del metodo<sup>4</sup>.

♣ **Esempio 5.13.** Consideriamo la funzione:

$$f(x) = x^2 - \frac{1}{2},$$

essa ha due zeri nei punti  $x^* = -\sqrt{2}/2 \simeq -0.7071$  ed  $\hat{x} = \sqrt{2}/2 \simeq 0.7071$ ,

Si vuole determinare uno zero di  $f$  in  $[0, 1]$  utilizzando il Metodo di Tabulazione. Scegliendo 5 punti equidistanziati, cioè i punti  $t_i = (i - 1) \cdot h$ ,  $i = 1, \dots, 5$ , con  $h = 0.4$ , si ottengono i valori riportati in Tabella 5.7.

Si osserva che la funzione  $f$  assume segno discorde agli estremi dell'intervallo  $[0.5, 0.75]$ , il cui punto medio  $x_1 = 0.625$  è la prima approssimazione dello zero di  $f$ . Procedendo con il Metodo di Tabulazione,

---

<sup>4</sup>Più in generale, per un metodo iterativo sussiste la seguente definizione:

**Definizione 5.1.** Considerata l'equazione

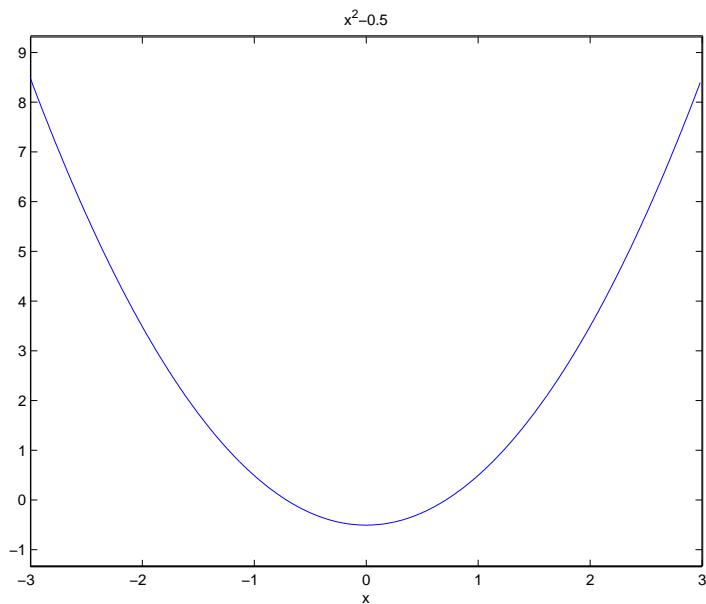
$$f(x) = 0$$

e supposto che la funzione abbia almeno uno zero nell'intervallo  $[a, b]$ , sia  $\{x_k\}_{k \in N}$  la successione generata da un metodo iterativo.

Si dice che il metodo converge se:

$$\lim_{k \rightarrow \infty} x_k = x^*, \quad (5.7)$$

dove  $x^*$  è uno zero della funzione nell'intervallo  $[a, b]$

Figura 5.8: Grafico di  $f(x) = x^2 - \frac{1}{2}$  in  $[-3, 3]$ 

$x$	$f(x)$
.0000e+00	-.5000e+00
.2500e+00	-.4375e+00
.5000e+00	<b>-.2500e+00</b>
.7500e+00	<b>.6250e-01</b>
.1000e+01	.5000e+00

Tabella 5.7: Primo passo del Metodo di Tabulazione, con  $n = 5$ , applicato alla funzione in Fig. 5.8

$k$	$x_k$	$f(x)$
1	.6250e+00	<b>-.1093e+00</b>
2	.7188e+00	<b>.1667e-01</b>
3	.7109e+00	<b>.5378e-02</b>
4	.7089e+00	<b>.2658e-02</b>
5	.7075e+01	<b>.5562e-03</b>

**Tabella 5.8:** 5 iterazioni del Metodo di Tabulazione, con  $n = 5$ , applicato alla funzione in Fig. 5.8

$x$	$f(x)$
.0000e+00	<b>-.1000e+01</b>
.1000e+00	<b>.2000e+01</b>
.2000e+00	.1100e+02
.3000e+00	.9000e+01
.4000e+01	.1200e+02

**Tabella 5.9:** Primo passo del Metodo di Tabulazione, con  $n = 5$ , applicato alla funzione in Fig. 5.9

si determinano via via nuovi sottointervalli di  $[0, 1]$  dove la funzione cambia segno agli estremi. Indicato quindi con  $x_k$  il punto medio dell'intervallo corrente  $[a_k, b_k]$  dopo 5 iterazioni si ottengono i risultati mostrati in Tabella 5.8.



♣ **Esempio 5.14.** Consideriamo la funzione:

$$f(x) = \begin{cases} 3x^2 - 1, & \text{se } x \leq 2.5 \\ 3x & x > 2.5 \end{cases}. \quad (5.8)$$

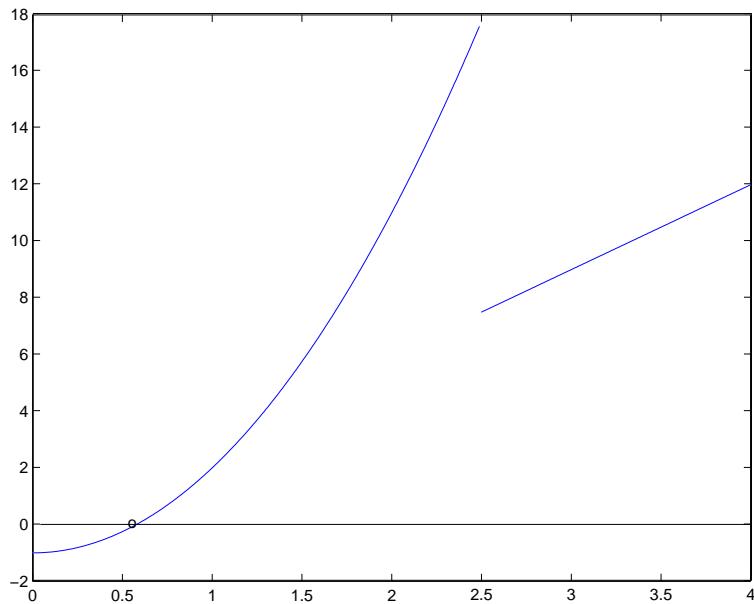
Essa ha due zeri nei punti  $x^* = -\sqrt{3}/3 \simeq -0.57735$  ed  $x^* = \sqrt{3}/3 \simeq 0.57735$ , ed un punto di discontinuità in 2.5,

Si vuole determinare uno zero di  $f$  in  $[0, 4]$  utilizzando il Metodo di Tabulazione. Scegliendo 5 punti equidistanziati, cioè i punti  $t_i = i - 1$  con  $i = 1, \dots, 5$ , si ottengono i valori riportati in Tabella 5.9.

Si osserva che la funzione  $f$  assume segno discorde agli estremi dell'intervallo  $[0, 1]$ , il cui punto medio  $x_1 = 0.5$  è la prima approssimazione dello zero di  $f$ . Dopo 5 iterazioni del Metodo di Tabulazione si ottengono i risultati riportati in Tabella 5.10.



Nell'esempio 5.13 le approssimazioni determinate dal Metodo di Tabulazione sem-

Figura 5.9: Grafico della funzione (5.8) in  $[0, 4]$ 

$k$	$x_k$	$f(x_k)$
1	.5000e+00	<b>-.2500e+00</b>
2	.6250e+00	<b>.1718e+00</b>
3	.5937e+00	<b>.5761e-01</b>
4	.5703e+00	<b>-.2423e-01</b>
5	.5761e+00	<b>-.4078e-02</b>

Tabella 5.10: 5 iterazioni del Metodo di Tabulazione, con  $n = 5$ , applicato alla funzione in Fig. 5.9

brano avvicinarsi sempre più alla zero  $\hat{x} = \sqrt{2}/2 \simeq 0.7071$ , ovvero il metodo sembra convergere. Una giustificazione di tale risultato si può dare osservando che il Metodo di Tabulazione è essenzialmente basato sulla considerazione che se una funzione continua assume valori di segno opposto agli estremi di un intervallo, la funzione si annulla almeno in un punto interno a tale intervallo. Si ricorda, infatti, il seguente:

**Teorema 5.1. [Teorema degli zeri]**

*Sia  $f$  una funzione continua in un intervallo  $[a, b]$ . Se  $f(a) \cdot f(b) < 0$ , allora esiste  $x_0 \in (a, b)$  tale che  $f(x_0) = 0$ .*

Da un punto di vista grafico tale risultato può essere espresso dicendo che se i due punti del grafico di  $y = f(x)$ , di ascissa  $a$  e  $b$ , si trovano rispettivamente al di sopra ed al di sotto (o viceversa) dell'asse delle  $x$ , allora il grafico della funzione interseca tale asse in almeno un punto.

Tuttavia, è bene precisare, e se ne ha conferma nell'esempio 5.14, che nel Metodo di Tabulazione si può avere la convergenza anche senza le ipotesi del Teorema degli zeri. Al fine di determinare condizioni sufficienti alla convergenza introduciamo l'**errore di troncamento analitico** alla  $k$ -ma iterazione, definito come:

$$e_k = x^* - x_k, \quad (5.9)$$

con  $x^*$  uno zero della funzione nell'intervallo  $[a, b]$ . Si osservi che la convergenza del metodo ad uno zero della funzione è equivalente alla convergenza a zero della successione dei valori assoluti degli errori, cioè:

$$\lim_{k \rightarrow \infty} x_k = x^* \quad \Leftrightarrow \quad \lim_{k \rightarrow \infty} |e_k| = 0 \quad (5.10)$$

Sussiste il seguente:

**Teorema 5.2.** *Sia  $f$  una funzione continua nell'intervallo chiuso e limitato  $[a, b]$ . Allora, condizione sufficiente affinché il Metodo di Tabulazione con  $n$  punti equidistanziati<sup>5</sup>, converga a  $x^*$ , dove  $x^*$  è uno zero della funzione nell'intervallo  $[a, b]$  è che  $f(a) \cdot f(b) < 0$ .*

**Dimostrazione** Il Metodo di Tabulazione genera, a partire dall'intervallo  $[a_0, b_0] = [a, b]$ , una successione di intervalli tutti contenenti uno zero  $x^*$  della funzione (Teorema degli zeri). Inoltre essendo i punti della tabulazione equidistanziati, ognuno di questi intervalli ha ampiezza uguale a  $1/(n-1)$  dell'ampiezza dell'intervallo precedente; si ha quindi:

$$x^* \in [a_k, b_k], \quad \forall k \geq 0; \quad (5.11)$$

$$b_k - a_k = \frac{1}{n-1}(b_{k-1} - a_{k-1}), \quad \forall k \geq 1. \quad (5.12)$$

Applicando l'uguaglianza (5.12) ripetutamente, si ottiene che:

$$b_k - a_k = \frac{1}{(n-1)^k}(b_0 - a_0), \quad \forall k \geq 1. \quad (5.13)$$

---

<sup>5</sup>Un risultato analogo continua a valere se i punti della tabulazione sono arbitrari

Poichè  $x_k = (a_k + b_k)/2$  è il punto medio dell'intervallo  $[a_k, b_k]$ , dalla (5.11) si ha:

$$|e_k| \leq \frac{1}{2}(b_k - a_k) \quad (5.14)$$

da cui, in base alla (5.13) si ha:

$$|e_k| \leq \frac{1}{2(n-1)^k}(b_0 - a_0), \quad \forall k \geq 0. \quad (5.15)$$

Quindi, la successione dei valori assoluti degli errori converge a zero perchè è limitata superiormente da una successione che converge a zero. Di conseguenza, la successione  $\{x_k\}$  converge a  $x^*$ . ■

Dalla (5.14) si ricava che una stima dell'errore è fornita dalla metà dell'ampiezza dell'intervallo corrente che rappresenta il massimo errore, in valore assoluto, che si può avere in quella iterazione, cioè

$$e_k = \mathcal{O}((n-1)^{-k})$$

da cui si può dedurre la relazione che esiste tra l'errore alla  $k+1$ -ma iterazione e quello dell'iterazione precedente:

$$|e_{k+1}| \simeq \frac{1}{n-1} |e_k| \quad (5.16)$$

La relazione mostra che, ad ogni iterazione del Metodo di Tabulazione, l'errore diminuisce rispetto a quello all'iterazione precedente del fattore  $n-1$ . In tal caso, si dice che il metodo **converge in modo lineare**. In generale sussiste la seguente:

**Definizione 5.2.** Si dice che una successione  $\{x_k\}_{k \in \mathbb{N}}$  convergente a  $x^*$  ha ordine di convergenza  $p$  se esiste  $\bar{n} \in \mathbb{N}$  ed un numero reale positivo  $C$  ( $< 1$  se  $p = 1$ ) tale che, per ogni  $n > \bar{n}$  si abbia

$$|x_{n+1} - x^*| \leq C|x_n - x^*|^p$$

In particolare quando  $p = 1$  si parla di **convergenza lineare**, per  $p = 2$  **convergenza quadratica** ed, in generale, per  $p > 1$  **convergenza superlineare**; la costante  $C$  prende il nome di **raggio di convergenza**. Il teorema precedente conferma che, per il Metodo di Tabulazione, la convergenza è assicurata purché si conosca un intervallo  $[a_0, b_0]$  tale che  $f(a_0)f(b_0) < 0$  (sotto le ipotesi di continuità per  $f$ ). Metodi iterativi per la risoluzione di equazioni non lineari che hanno la suddetta proprietà di convergenza sono detti **globali**, i quali si distinguono dai metodi **locali**, ovvero da quei metodi la cui convergenza ad uno zero della funzione dipende non solo dall'intervallo di ricerca, ma anche dalla scelta opportuna dei punti iniziali del procedimento iterativo.

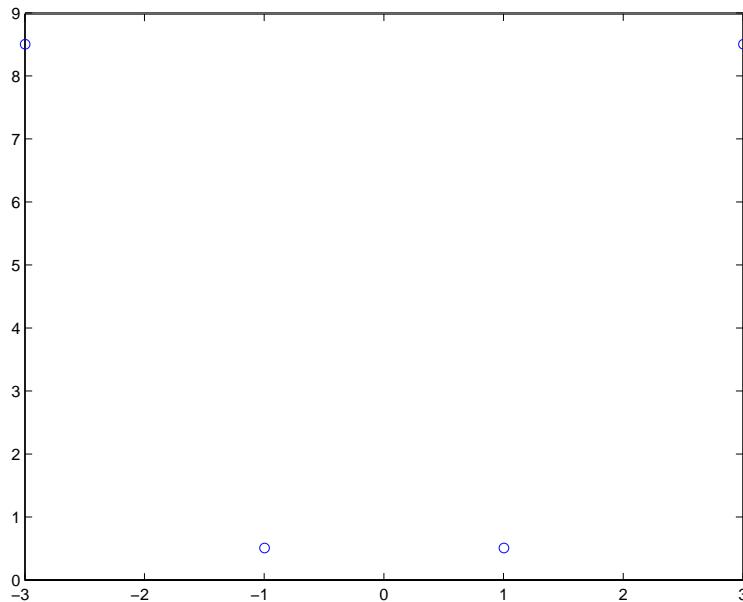


Figura 5.10: Tabulazione di  $f(x) = x^2 - \frac{1}{2}$  in  $[-3, 3]$  con  $n = 4$

### 5.3 Il Metodo di Bisezione

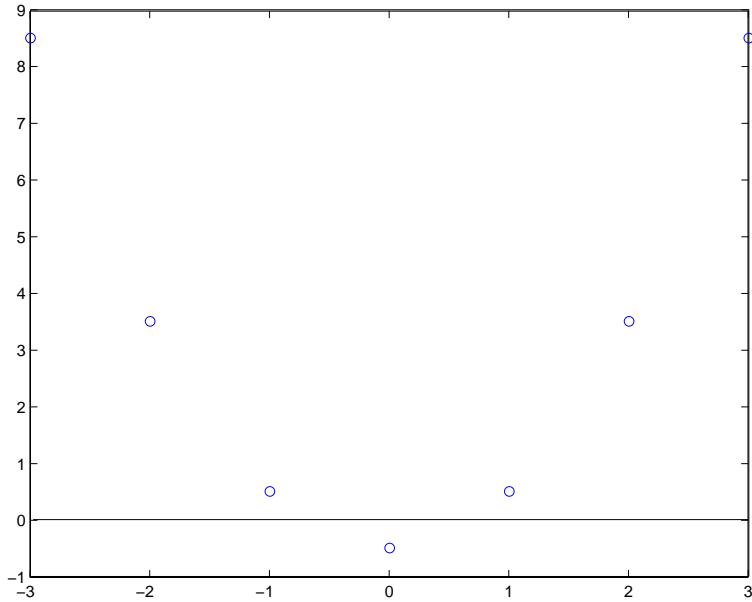
♣ **Esempio 5.15.** Consideriamo la funzione dell'esempio 5.13:

$$f(x) = x^2 - \frac{1}{2},$$

essa ha due zeri nei punti  $x^* = -\sqrt{2}/2 \simeq -0.7071$  e  $x^* = \sqrt{2}/2 \simeq 0.7071$ . Se tabuliamo la funzione in  $[-3, 3]$  con 4 punti di tabulazione non troviamo alcun intervallo dove la funzione cambia segno, come mostrato in Figura 5.10. Quindi, volendo applicare un altro passo, infittiamo la tabulazione, scegliendo  $n = 7$  punti di tabulazione della funzione in  $[-3, 3]$ . Dalla Figura 5.11 si può evincere la presenza di almeno due zeri della funzione, e precisamente il primo nell'intervallo  $[-1, 0]$  ed il secondo nell'intervallo  $[0, 1]$ .



Nell'esempio 5.15 si osserva che l'individuazione di un intervallo che potrebbe contenere uno zero dipende dal numero di punti di tabulazione, nel senso che una tabulazione con pochi punti può non rivelare la presenza di uno o più zeri. D'altra parte per individuare intervalli contenenti delle soluzioni potrebbe essere necessario scegliere troppi punti di tabulazione, e quindi possono essere necessarie numerose valutazioni della funzione, la maggior parte delle quali spesso "superflue". Da queste considerazioni si deduce che il Metodo di Tabulazione è di utilità pratica solo per una "rozza" localizzazione delle

Figura 5.11: Tabulazione di  $f(x) = x^2 - \frac{1}{2}$  in  $[-3, 3]$  con  $n = 7$ 

radici di una equazione, cioè per la determinazione di intervalli in cui potrebbe esservi una radice. Un metodo più efficiente per la risoluzione di equazioni non lineari è il **Metodo di Bisezione**.

♣ **Esempio 5.16.** Consideriamo l'equazione dell'esempio 5.10:

$$f(x) = 3x^4 - 11x^3 - 21x^2 + 99x - 54 = 0 \quad (5.17)$$

si consideri l'intervallo  $[0, 2]$ , in cui la funzione ha un solo zero,  $x^* = 2/3$  e valori estremi di segno opposto. Si consideri il punto medio dell'intervallo iniziale, cioè  $x_1 = 1$ . Si generano in tal modo due sottointervalli,  $[0, 1]$  e  $[1, 2]$ , entrambi di ampiezza uguale alla metà di quello iniziale, e  $x_1$  costituisce un'approssimazione dello zero della funzione. A questo punto, osservando che la funzione assume valori di segno opposto agli estremi dell'intervallo  $[0, 1]$ , si può applicare il procedimento appena descritto a tale intervallo, ottenendo una nuova approssimazione della soluzione,  $x_2 = 0.5$ , ed altri due sottointervalli,  $[0, 0.5]$  e  $[0.5, 1]$ . Iterando il procedimento è possibile determinare una successione di intervalli, ciascuno di ampiezza uguale alla metà del precedente e contenente una soluzione di (5.17). Il procedimento descritto è noto come **Metodo di Bisezione**.

In Tabella 5.11 sono riportati i valori  $x_k$ , e i corrispondenti valori della funzione, per  $k = 1, \dots, 19$ , ottenuti applicando il Metodo di Bisezione alla risoluzione di (5.5). Si osservi che con 5 iterazioni ( $k = 5$ ), si ottiene una approssimazione della soluzione migliore di quella che si ottiene tabulando la funzione in 11 punti equidistanziati nell'intervallo (vedi Tabella 5.1). Ciò conferma la maggiore efficienza del Metodo di Bisezione rispetto al Metodo di Tabulazione generale. In Figura 5.12 è rappresentata graficamente l'applicazione del metodo alla risoluzione di (5.17).



$k$	$x_k$	$f(x_k)$
1	1.0000e+00	1.8768e+01
2	5.0000e-01	-1.0938e+01
3	7.5000e-01	4.7461e+00
4	6.2500e-01	-2.5559e+00
5	6.8750e-01	1.2325e+00
6	6.5625e-01	-6.2764e-01
7	6.7188e-01	3.1097e-01
8	6.6406e-01	-1.5620e-01
9	6.6797e-01	7.7921e-02
10	6.6602e-01	-3.9005e-02
11	6.6699e-01	1.9491e-02
12	6.6650e-01	-9.7485e-03
13	6.6675e-01	4.8735e-03
14	6.6663e-01	-2.4369e-03
15	6.6669e-01	1.2184e-03
16	6.6666e-01	-6.0922e-04
17	6.6667e-01	3.0461e-04
18	6.6666e-01	-1.5231e-04
19	6.6667e-01	7.6153e-05

**Tabella 5.11:** Metodo di Bisezione applicato all'equazione  $3x^4 - 11x^3 - 21x^2 + 99x - 54 = 0$  in  $[0, 2]$  (prime 19 iterazioni)

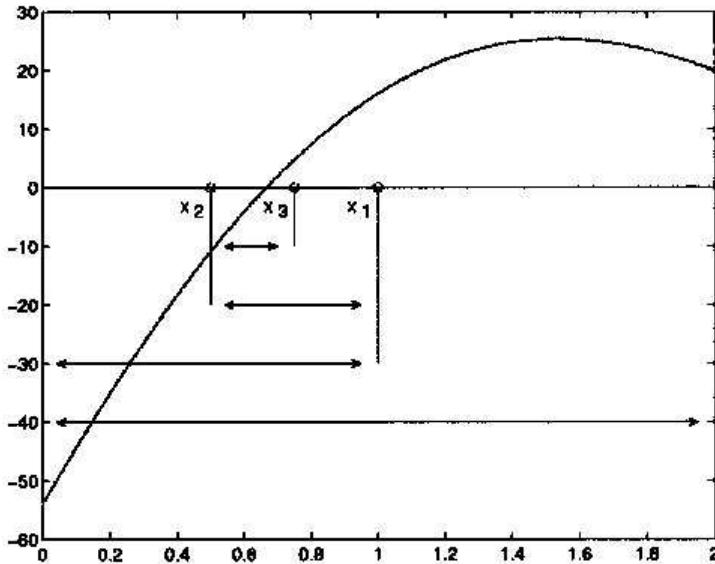


Figura 5.12: Metodo di Bisezione applicato all'equazione  $3x^4 - 11x^3 - 21x^2 + 99x - 54 = 0$  in  $[0, 2]$  (prime 3 iterazioni)

♣ **Esempio 5.17.** Consideriamo la funzione:

$$f(x) = \begin{cases} x^3 - 1/3, & \text{se } x \leq 2/3 \\ x/3 & x > 2/3 \end{cases} \quad (5.18)$$

il cui grafico è riportato in Figura 5.13.

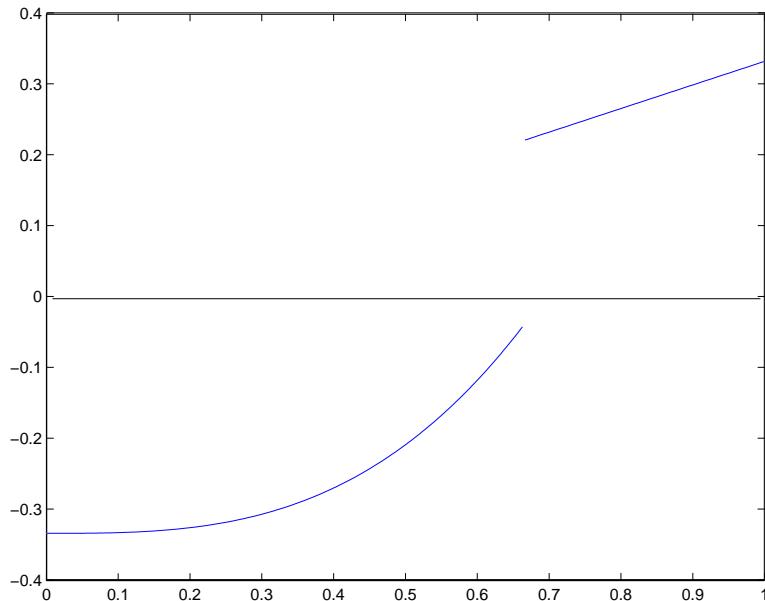
Applicando il Metodo di Bisezione nell'intervallo  $[0, 1]$  si ottengono i valori riportati in Tabella 5.12.



Il Metodo di Bisezione è un particolare metodo iterativo che, a partire dall'intervallo di ricerca  $[a, b]$ , genera una successione di valori  $\{x_k\}_{k=1,2,\dots}$ , come descritto di seguito: Sia  $[a_0, b_0] \equiv [a, b]$  e sia  $[a_k, b_k]$  ( $k > 0$ ) il generico intervallo:

- calcolo del punto medio di  $[a_k, b_k]$ :

$$x_{k+1} = a_k + \frac{b_k - a_k}{2}$$

Figura 5.13: Grafico della funzione (5.18) in  $[0, 1]$ 

$n$	$x_n$	$f(x_n)$
1	1.0000e+00	1.8768e+01
2	5.0000e-01	-0.20833e+00
3	7.5000e-01	0.25000e+00
4	6.2500e-01	-0.89192e-01
5	6.8750e-01	0.22916e+00
6	6.5625e-01	-0.50710e-01
7	6.7188e-01	0.22396e+00
8	6.6406e-01	-0.40499e-01
9	6.6797e-01	0.22656e+00

Tabella 5.12: Metodo di Bisezione applicato alla (5.18) in  $[0, 2]$  (prime 9 iterazioni)

- scelta dell'intervallo successivo:
  - se  $f(x_{k+1})f(a_k) \leq 0$ , allora  $[a_{k+1}, b_{k+1}] \equiv [a_k, x_{k+1}]$ ;
  - altrimenti  $[a_{k+1}, b_{k+1}] \equiv [x_{k+1}, b_k]$ .

Di seguito è riportata una prima versione dell'algoritmo, scritto utilizzando il linguaggio Pascal-like, basato sul Metodo di Bisezione. Si osservi che in tale versione è fissato il numero di iterazioni (rappresentato dalla variabile `itmax`) da effettuare. Le variabili `a`, `b` e `c` rappresentano rispettivamente gli estremi dell'intervallo corrente ed il suo punto medio, mentre `fa`, `fb`, `fc` sono i valori che la funzione, rappresentata da `f`, assume nei tre punti considerati. Si osservi, infine, che gli estremi dell'intervallo iniziale, `a` e `b`, come pure la funzione `f`, sono i dati di input dell'algoritmo. Le variabili `c` e `fc` rappresentano i dati di output, cioè, rispettivamente, l'approssimazione dello zero e del valore di `f` in `c`.

```

procedure fxbis(input: a,b,f,itmax, out: c,fc)

/# SCOPO: calcolo di un'approssimazione dello zero

/# di una funzione mediante il Metodo di Bisezione

/# SPECIFICHE PARAMETRI:

var: a : reale {primo estremo dell'intervallo di ricerca}
var: b : reale {secondo estremo dell'intervallo di ricerca}
var: f : funzione esterna {funzione di cui si cerca lo zero}
var: itmax : reale {numero massimo di iterazioni}
var: c : reale {approssimazione dello zero}
var: fc : reale {valore di f in c}

/# INIZIO ISTRUZIONI:

fa := f(a);
fb := f(b);
for i = 1, itmax
/# calcolo del punto medio e del valore della funzione
c := (a + b)/2;
fc := f(c);
if (fc * fa ≤ 0) then {test sul segno di f}
    b := c;
    fb := fc;
else

```

Procedura 5.1: Algoritmo di Bisezione (prima versione) - continua

```

    a := c;
    fa := fc;
  endif
endfor
end fxbis

```

Procedura 5.1: Algoritmo di Bisezione (prima versione) - fine

### 5.3.1 Applicabilità del Metodo di Bisezione

Si osservi che il Metodo di Bisezione è un procedimento di "tabulazione nel punto medio", più precisamente di tabulazione con  $n = 3$  punti equidistanti. Di conseguenza il Metodo di Bisezione eredita le caratteristiche del Metodo di Tabulazione. Si deduce così che può essere sempre applicato a funzioni che sono valutabili su tutto l'intervallo di ricerca, mentre se tale condizione non è verificata, nulla si può dire sulla sua applicabilità. L'esempio che segue mostra che, nel caso di funzioni non ovunque definite, l'applicabilità del Metodo di Bisezione dipende dall'intervallo di ricerca, ovvero dalla scelta dei punti di tabulazione.

♣ **Esempio 5.18.** Consideriamo l'equazione:

$$f(x) = \frac{1}{x^2} - 4 = 0$$

La funzione  $f$  ha due zeri in  $x^* = -0.5$  e  $\hat{x} = 0.5$ . Applichiamo il Metodo di Bisezione in  $[-2, 1.5]$ . Indicato con  $c$  il punto medio dell'intervallo corrente  $[a_k, b_k]$ , dopo 5 iterazioni si ottengono i risultati mostrati nella Tabella 5.13.

Il metodo, in questo caso, sembra avvicinarsi ad una delle radici dell'equazione ( $-0.5$ ). Tuttavia, se vogliamo applicare il Metodo di Bisezione alla stessa funzione  $f$ , cambiando però l'intervallo in  $[-2, 2]$ , già alla prima iterazione il metodo diventa non applicabile. Infatti il valore del punto medio dell'intervallo al primo passo è  $x_1 = 0$  e qui la funzione non è definita.



### 5.3.2 Convergenza

Come abbiamo fatto per l'applicabilità, anche la convergenza può essere dedotta dai medesimi risultati del Metodo di Tabulazione. Dal Teorema 5.2 segue, quindi, il seguente:

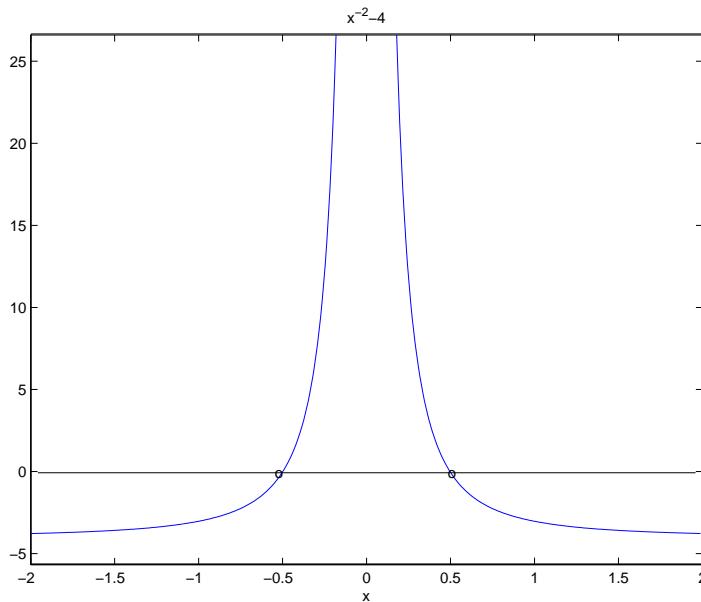


Figura 5.14: Grafico di  $f(x) = \frac{1}{x^2} - 4$  in  $[-2, 2]$

$k$	$c$	$f(c)$
1	-0.2500	12.000
2	-1.1250	-3.2099
3	-0.6875	-3.4711
4	-0.4688	0.5511
<b>5</b>	<b>-0.5781</b>	<b>-1.0080</b>

Tabella 5.13: 5 iterazioni del Metodo di Bisezione applicato alla funzione in Fig. 5.14

**Corollario 5.1.** *Sia  $f$  una funzione continua nell'intervallo chiuso e limitato  $[a, b]$ , con  $f(a) \cdot f(b) < 0$ .*

*Allora, condizione sufficiente affinché il Metodo di Bisezione converga a  $x^*$ , dove  $x^*$  è uno zero della funzione nell'intervallo  $[a, b]$  è che  $f(a) \cdot f(b) < 0$ .*

Il corollario precedente conferma che il Metodo di Bisezione converge sotto opportune ipotesi; inoltre trattandosi di un metodo di Tabulazione, anche esso converge linearmente. In particolare dalla relazione (5.16) con  $n = 3$ , si ottiene

$$|e_{k+1}| \simeq \frac{1}{2} |e_k| \quad (5.19)$$

da cui si ha che il raggio di convergenza del Metodo di Bisezione è  $1/2$ .

Si osservi, infine, che analogamente al Metodo di Tabulazione, anche per il Metodo di Bisezione le ipotesi di continuità della funzione, a volte, non sono necessarie per la convergenza del metodo. A tal fine si consideri nuovamente l'esempio 5.16. In tale esempio la funzione non ha radici, tuttavia la successione determinata dal Metodo di Bisezione (Tabella 5.12) sembra comunque convergere al valore  $2/3$ , cioè il punto di discontinuità della funzione. Ovvero il metodo riesce in ogni caso a fornire una soluzione<sup>6</sup>.

## 5.4 Il Metodo di Newton

Nel Metodo di Bisezione la successione di valori  $\{x_k\}$  è generata tenendo conto solo delle variazioni di segno della funzione negli estremi degli intervalli  $[a_k, b_k]$ . Non è utilizzata alcuna informazioni sull'andamento della funzione; ad esempio potrebbe essere utile avere informazioni sulla "pendenza" della funzione nell'intervallo di ricerca, che consentano di selezionare lo stesso in modo più efficiente.

♣ **Esempio 5.19.** Consideriamo l'equazione non lineare:

$$f(x) = 0.2x - \log(x) = 0$$

Se consideriamo la funzione  $f$  in  $[1, 7]$  essa ha uno zero in  $x^* = 1.2959$ , ovvero ha uno zero vicino all'estremo sinistro dell'intervallo. In questo caso, il Metodo di Bisezione è abbastanza "lento". Osserviamo, infatti, la Figura 5.16

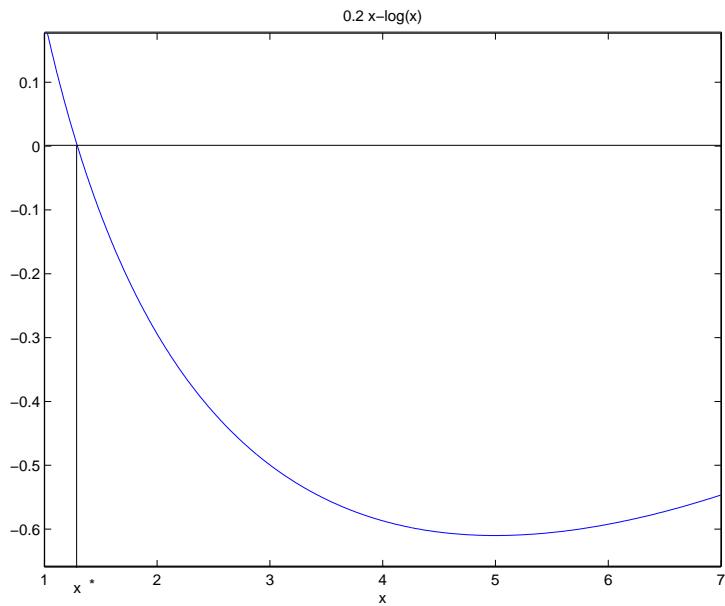
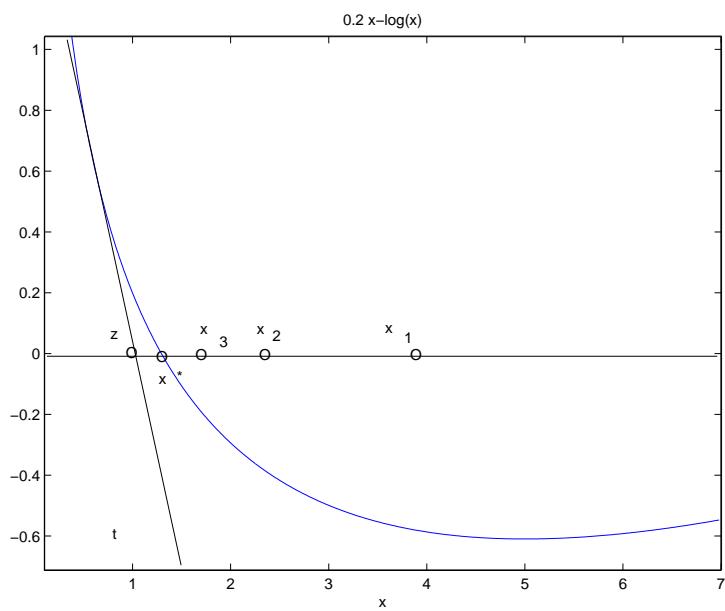
dove sono rappresentati i 3 passi del Metodo di Bisezione,  $x_3$  è ancora "abbastanza" lontano dalla soluzione  $x^*$ , come si può osservare anche dai valori numerici riportati nella Tabella 5.14.

Se indichiamo con  $t$  la retta tangente a  $f$  nel punto  $(1, f(1)) = 0.2$ , detta  $z$  l'ascissa della sua intersezione con l'asse delle ascisse, tale punto risulta molto più vicino ad  $x^*$ , come si evince dalla Tabella 5.14.




---

<sup>6</sup>In questo senso si parla di robustezza del Metodo di Bisezione. Più in generale la **robustezza** è una caratteristica dei metodi globalmente convergenti.

Figura 5.15: Grafico di  $f(x) = 0.2x - \log(x)$  in  $[1, 7]$ Figura 5.16: Metodo di Bisezione applicato all'equazione  $0.2x - \log(x) = 0$  in  $[1, 7]$  (prime 3 iterazioni)

$x_1 = 4$	$f(x_1) = -0.5863\dots$
$x_2 = 2.5$	$f(x_2) = -0.4163\dots$
$x_3 = 1.75$	$f(x_3) = -0.2096\dots$
$z = 1.25$	$f(z) = 0.0269\dots$

**Tabella 5.14: 3 iterazioni del Metodo di Bisezione applicato alla funzione in Fig. 5.16**

Un metodo che tiene conto, in qualche modo, dell'andamento della funzione è il **Metodo di Newton**.

♣ **Esempio 5.20.** Considerata l'equazione:

$$f(x) = 3x^4 - 11x^3 - 21x^2 + 99x - 54 = 0 \quad (5.20)$$

si scelga il punto  $x_0 = 1$  e si consideri l'equazione della retta tangente alla curva  $y = f(x)$  in questo punto:

$$m_0(x) = f(x_0) + f'(x_0)(x - x_0) = 16 + 36(x - 1).$$

Il punto di intersezione di tale retta con l'asse delle ascisse (che si ottiene risolvendo  $m_0(x) = 0$ ) è:

$$x_1 = 20/36 \simeq 0.55556,$$

Si può procedere in modo analogo a partire da  $x_1$ , ottenendo:

$$m_1(x) = f(x_1) + f'(x_1)(x - x_1) \simeq -7.0815 + 67.539(x - 0.55556);$$

da cui:

$$x_2 = \simeq 0.66041.$$

In Tabella 5.15 sono riportati i valori ottenuti nelle prime 4 iterazioni del Metodo di Newton applicato all'equazione (5.20), con  $x_0 = 1$ .

Confrontando tali risultati con quelli ottenuti con il Metodo di Bisezione, riportati nella Tabella 5.11, si osserva che, per calcolare un'approssimazione  $\tilde{x} = 0.66667$  con quattro cifre significative dello zero  $x^* = 0.66666\dots$ :

- il *Metodo di Bisezione* impiega 19 iterazioni calcolando:

$$\tilde{x} = x_{19} = 0.66667 |f(x_{19})| = 7.6153 \times 10^{-5}$$

- il *Metodo di Newton* impiega 4 iterazioni calcolando:

$$\tilde{x} = x_4 = 0.66667 |f(x_4)| = 1.8001 \times 10^{-8}$$

$k$	$x_k$	$f(x_k)$
1	5.5556e-01	-7.0818e+00
2	6.6041e-01	-3.7601e-01
3	6.6664e-01	-1.3582e-03
4	6.6667e-01	-1.8001e-08

**Tabella 5.15:** 4 iterazioni del Metodo di Newton applicato all'equazione  $3x^4 - 11x^3 - 21x^2 + 99x - 54 = 0$  con  $x_0 = 1$ .

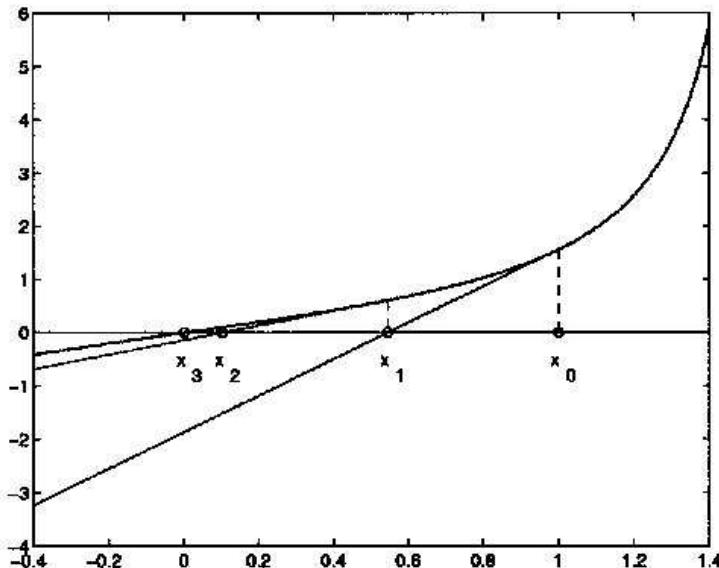


Figura 5.17: 3 iterazioni del Metodo di Newton applicato all'equazione  $\tan x = 0$  con  $x_0 = 1$

Si può affermare quindi che, per l'equazione considerata, il Metodo di Newton fornisce valori vicini alla soluzione più velocemente rispetto al Metodo di Bisezione. ♣

♣ **Esempio 5.21.** Consideriamo l'equazione:

$$\tan x = 0,$$

In Fig. 5.17 è illustrata graficamente l'applicazione di 3 iterazioni del Metodo di Newton alla risoluzione di tale equazione, con approssimazione iniziale dello zero  $x_0 = 1$ . ♣

Il procedimento appena descritto è noto come **Metodo di Newton**. Tale metodo costruisce una successione di valori  $\{x_k\}_{k=1,2,\dots}$  nel modo seguente:

- $\forall k = 0, 1, 2, \dots$ , a partire da un valore iniziale  $x_0$ , si considera la tangente alla curva  $y = f(x)$  nel punto  $(x_k, f(x_k))$ , e si assume come approssimazione dello zero della funzione il valore,  $x_{k+1}$ , ottenuto intersecando tale retta con l'asse delle ascisse, cioè:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \quad (5.21)$$

Di seguito è riportata una prima versione dell'algoritmo di Newton per risolvere l'equazione:

$$f(x) = 0, \quad x \in [a, b] \subseteq A, \quad (5.22)$$

dove  $f : A \subseteq \mathbb{R} \rightarrow \mathbb{R}$  è una funzione con derivata  $f'$  definita in  $[a, b]$ .

```

procedure fxnew(input:  $x_0, f, f', itmax$ , out:  $x_{new}, fx_{new}$ )
  /* SCOPO: calcolo di un'approssimazione dello zero
  /* di una funzione mediante il Metodo di Newton
/* SPECIFICHE PARAMETRI:
  var:  $x_0$  : reale {approssimazione iniziale dello zero}
  var:  $f$  : funzione esterna {funzione di cui si cerca lo zero}
  var:  $f'$  : funzione esterna {derivata di  $f$ }
  var:  $itmax$  : reale {numero massimo di iterazioni}
  var:  $x_{new}$  : reale {approssimazione dello zero}
  var:  $fx_{new}$  : reale {valore di  $f$  in  $x_{new}$ }

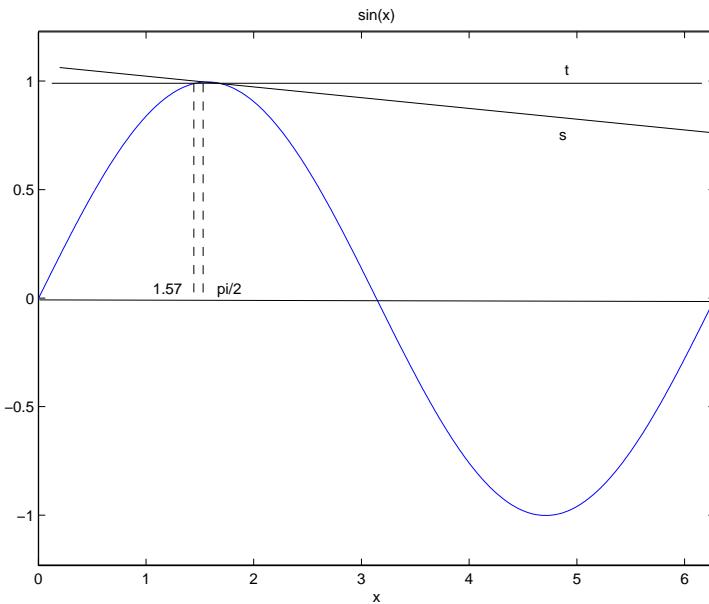
/* INIZIO ISTRUZIONI:
   $k := 0$ 
  /* generazione della successione delle approssimazioni
  repeat
     $x_{k+1} := x_k - f(x_k)/f'(x_k)$  {calcolo}
     $k := k + 1$  {dell'approssimazione}
  until (condizione di arresto verificata)
end fxnew

```

Procedura 5.2: Algoritmo di Newton (prima versione)

### 5.4.1 Applicabilità del Metodo di Newton

Abbiamo visto che, per applicare il Metodo di Bisezione, è sufficiente che la funzione  $f$  sia valutabile nell'intervallo di ricerca. Per il Metodo di Newton invece è richiesto l'uso della derivata prima della funzione. Inoltre, nel Metodo di Newton l'approssimazione  $x_{k+1}$  può essere calcolata solo se è  $f'(x_k) \neq 0$ . La Fig. 5.18 mostra un'interpretazione geometrica di tale condizione.

Figura 5.18: Metodo di Newton applicato all'equazione  $\sin(x) = 0$ 

♣ **Esempio 5.22.** Se si considera l'equazione:

$$\sin(x) = 0$$

in  $]0, 2\pi[$  e si vuole calcolare un'approssimazione dello zero  $\pi$ , con il Metodo di Newton, in tale intervallo, scegliendo come approssimazione iniziale  $x_0 = \pi/2$ , la retta tangente  $t$  non ha intersezione con l'asse delle  $x$ , essendo ad esso parallela. Infatti, in questo caso,  $f'(x_0) = \cos(\pi/2) = 0$

Purtroppo però, anche se come punto iniziale si sceglie un punto "vicino" a  $\pi/2$ , ad esempio, si pone  $x_0 = 1.58$  per cui  $f'(x_0) \neq 0$ , la retta tangente  $s$  non ha intersezione con l'asse delle  $x$  nell'intervallo di ricerca dello zero.

È quindi intuitivamente giustificabile il fatto che il Metodo di Newton non sia applicabile non solo se  $f'(x_k)$  è uguale a zero, ma anche se è molto piccolo in modulo.



### 5.4.2 Convergenza

Si vogliono analizzare le caratteristiche di convergenza del Metodo di Newton, ovvero si vuole studiare sotto quali condizioni il Metodo di Newton converge ad una radice di  $f$ .

♣ **Esempio 5.23.** Applichiamo il Metodo di Newton all'equazione:

$$f(x) = \arctan(x) = 0$$

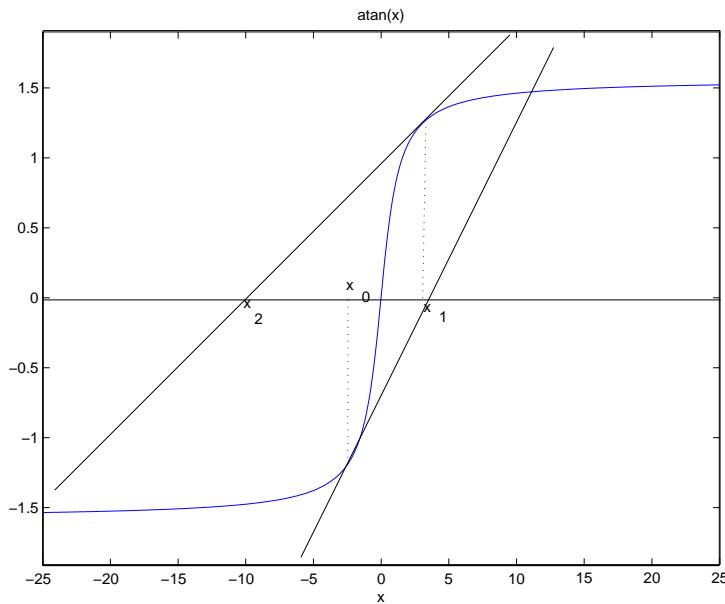


Figura 5.19: Metodo di Newton applicato all'equazione  $\arctan(x) = 0$  con  $x_0 = -2$  (prime 3 iterazioni) (esempio in cui l'algoritmo diverge)

$k$	$x_k$	$f(x_k)$
0	-2	-1.1071
1	3.5357	1.2952
2	-13.9510	-1.4992

**Tabella 5.16: Metodo di Newton applicato all'equazione  $\arctan(x) = 0$  con  $x_0 = -2$**

In Fig. 5.19 sono illustrate le prime 3 iterazioni del Metodo di Newton, applicato alla funzione  $f(x) = \arctan(x)$  nell'intervallo  $[-25, 25]$ . Dalla Tabella 5.16 si evince che, partendo da  $x_0 = -2$ , la successione di approssimazioni generata si allontana dallo zero  $x^* = 0$  dell'equazione, e pertanto il metodo diverge per tale scelta del punto iniziale.

Se applichiamo, invece, il Metodo di Newton scegliendo come punto iniziale  $x_0 = 1$ , dalla Tabella 5.17 si evince che dopo 4 iterazioni si ottiene una buona approssimazione dello zero

$$x_4 = -0.001061012\dots \quad f(x_4) = -0.001061017\dots$$

ovvero con la scelta di tale punto iniziale il metodo converge. ♣

Come è stato evidenziato dall'esempio 5.23, per il Metodo di Newton la sola continuità della funzione non è sufficiente per la convergenza, in quanto quest'ultima dipende

$k$	$x_k$	$f(x_k)$
0	1	0.78
1	-0.507...	-0.511...
2	0.1168...	0.1168...
3	-0.00106...	-0.00106..

**Tabella 5.17:** Metodo di Newton applicato all'equazione  $\arctan(x) = 0$  con  $x_0 = 1$  (prime 4 iterazioni)

dalla scelta del punto iniziale  $x_0$ : se tale punto è "abbastanza vicino" alla soluzione, allora il metodo genera una successione di valori convergente alla soluzione. Da ciò si deduce che, a differenza dei metodi di Tabulazione e Bisezione, il Metodo di Newton è un metodo "a convergenza locale". Condizioni sufficienti per la convergenza del Metodo di Newton sono fornite dal:

**Teorema 5.3.** *Sia  $f \in C^2([a, b])$  con  $x^* \in [a, b]$  zero di  $f$ . Si assume che  $|f'(x)| \geq m$ , per ogni  $x \in [a, b]$ , con  $m > 0$ . Allora esiste un  $\delta > 0$  tale che, se  $x_0 \in [a, b]$  è scelto in modo che  $|x_0 - x^*| < \delta$ , la successione  $\{x_k\}$  generata dal Metodo di Newton converge a  $x^*$ . Inoltre, si ha<sup>7</sup>:*

$$|x_{k+1} - x^*| = \frac{1}{2} \frac{|f''(\xi_k)|}{|f'(x_k)|} |x_k - x^*|^2, \quad \xi_k \in \text{int}[x_k, x^*], \quad k = 0, 1, \dots \quad (5.23)$$

**Dimostrazione** Posto  $M = \max|f''(x)|$ , definiamo:

$$\delta = \tau(2m/M), \quad \text{con } 0 < \tau < 1. \quad (5.24)$$

Considerata la formula di Taylor di  $f$  di punto iniziale  $x_0$ , si ha che:

$$f(x^*) = f(x_0) + f'(x_0)(x^* - x_0) + \frac{f''(\xi_0)}{2}(x^* - x_0)^2, \quad \xi_0 \in [x_0, x^*]. \quad (5.25)$$

Poiché  $x^*$  è uno zero di  $f$ , il primo membro di (5.25) è uguale a zero. Dividendo ambo i membri di (5.25) per  $f'(x_0)$  (che è non nullo per ipotesi) si ha

$$0 = \frac{f(x_0)}{f'(x^*)} + (x^* - x_0) + \frac{f''(\xi_0)}{2f'(x_0)}(x^* - x_0)^2, \quad \xi_0 \in [x_0, x^*]. \quad (5.26)$$

Dalla (5.21) segue

$$|x_1 - x^*| = \frac{1}{2} \frac{|f''(\xi_0)|}{|f'(x_0)|} |x_0 - x^*|^2 \leq \frac{1}{2} \frac{M}{m} |x_0 - x^*|^2. \quad (5.27)$$

Poichè per ipotesi  $|x_0 - x^*| < \delta = \tau(2m/M)$ , dalla (5.27) segue che:

$$|x_1 - x^*| \leq \tau |x_0 - x^*| < |x_0 - x^*| < \delta. \quad (5.28)$$

---

<sup>7</sup>Dati due numeri reali distinti  $a$  e  $b$ , con  $\text{int}[a, b]$  indicheremo l'intervallo i cui estremi sono rispettivamente il minimo e il massimo fra i due numeri

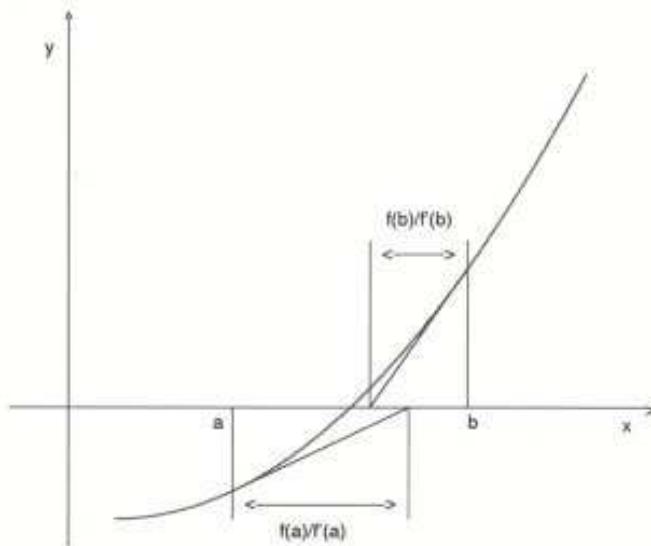


Figura 5.20: Illustrazione grafica del Teorema 5.4

Ciò dimostra che il nuovo punto  $x_1$  è anch'esso distante dallo zero  $x^*$  meno della quantità  $\delta$ . Applicando il ragionamento precedente a partire da  $x_1$  si ottiene:

$$|x_2 - x^*| = \frac{1}{2} \frac{|f''(\xi_1)|}{|f'(x_1)|} |x_1 - x^*|^2 \leq \frac{1}{2} \frac{M}{m} |x_1 - x^*|^2, \quad (5.29)$$

con  $\xi_1 \in \text{int}[x_1, x^*]$ . Inoltre:

$$|x_2 - x^*| \leq \tau |x_1 - x^*| < \tau^2 |x_0 - x^*|. \quad (5.30)$$

In generale,abbiamo che:

$$|x_{k+1} - x^*| = \frac{1}{2} \frac{|f''(\xi_k)|}{|f'(x_k)|} |x_k - x^*|^2 \leq \frac{1}{2} \frac{M}{m} |x_k - x^*|^2, \quad (5.31)$$

e quindi la (5.23). Inoltre, dalla (5.30):

$$|x_k - x^*| \leq \tau^k |x_0 - x^*|. \quad (5.32)$$

Poichè  $0 < \tau < 0$ , si ha  $\lim_{k \rightarrow \infty} \tau^k = 0$ , e quindi  $\lim_{k \rightarrow \infty} |x_k - x^*| = 0$ , che è equivalente alla convergenza a  $x^*$  della successione generata da Metodo di Newton. ■

Il teorema appena dimostrato conferma che la convergenza del Metodo di Newton è garantita se il punto iniziale è sufficientemente vicino alla soluzione.

Il teorema seguente, che ci limiteremo ad enunciare, fornisce una condizione sufficiente per la convergenza del Metodo di Newton indipendentemente dalla scelta del punto iniziale. Chiaramente in questo caso le ipotesi sulla funzione  $f$  sono più forti.

**Teorema 5.4.** *Sia  $f \in C^2[a, b]$ ; si supponga inoltre che sia  $f(a)f(b) < 0$  e nell'intervallo  $[a, b]$  sia  $f' \neq 0$  e  $f''$  di segno costante. Se*

$$\left| \frac{f(a)}{f'(a)} \right| < b - a, \quad \left| \frac{f(b)}{f'(b)} \right| < b - a, \quad (5.33)$$

*allora il Metodo di Newton converge comunque si scelga il punto iniziale  $x_0$  nell'intervallo  $[a, b]$ .*

Le condizioni espresse dal teorema ora enunciato possono essere geometricamente (vedi Fig. 5.20) interpretate come segue: la funzione  $f$  è strettamente monotona nell'intervallo  $[a, b]$ , non cambia concavità in tale intervallo, assume valori discordi agli estremi e le tangenti alla curva di equazione  $y = f(x)$  nei punti di ascissa  $a$  e  $b$  intersecano l'asse  $x$  in punti interni all'intervallo  $[a, b]$ .

Per quanto riguarda la velocità di convergenza del Metodo di Newton, si osservi, dalla (5.32), che l'errore ad una certa iterazione è circa il quadrato dell'errore alla iterazione precedente. Ovvero il Metodo di Newton converge quadraticamente, come mostra il:

**Teorema 5.5. [Velocità di convergenza del Metodo di Newton]**

*Sia  $f \in C^2([a, b])$  con  $x^* \in [a, b]$  ed  $f'(x^*) \neq 0$ . Se  $x_0$  è scelto sufficientemente vicino a  $x^*$  la successione  $\{x_n\}$  generata dal Metodo di Newton converge quadraticamente a  $x^*$  ed, inoltre,*

$$e_{n+1} = \frac{1}{2} \frac{f''(\xi_n)}{f'(x_n)} e_n^2, \quad \xi_n \in [x_n, x^*], \quad (5.34)$$

dove  $e_h = x_h - x^*$

Informazioni sul raggio di convergenza si hanno dal seguente:

**Teorema 5.6. [Raggio di convergenza del Metodo di Newton]**

*Nelle ipotesi del Teorema 5.5, il Metodo di Newton converge alla soluzione  $x^*$  se, posto  $|x_0 - x^*| = \rho$  e*

$$K(\rho) = \frac{1}{2} \frac{\max_{|x-x^*|<\rho} |f''(x)|}{\min_{|x-x^*|<\rho} |f'(x)|},$$

si ha

$$K(\rho)\rho < 1$$

Si può dimostrare che se  $x^*$  è una radice di molteplicità  $n > 1$ , allora il Metodo di Newton converge (solo) linearmente. Si osservi che  $K(\rho)\rho \rightarrow 0$  se  $\rho \rightarrow 0$  e, quindi, i due teoremi ora dimostrati assicurano che, se  $x^*$  è radice semplice di  $f \in C^2[a, b]$ , allora il Metodo di Newton converge quadraticamente purchè l'approssimazione iniziale sia scelta sufficientemente vicina a  $x^*$ . La convergenza quadratica implica che (almeno in vicinanza della soluzione) le cifre corrette della soluzione approssimata raddoppiano ad ogni iterazione; una conferma sperimentale di questo fatto si ha nell'esempio 5.20.

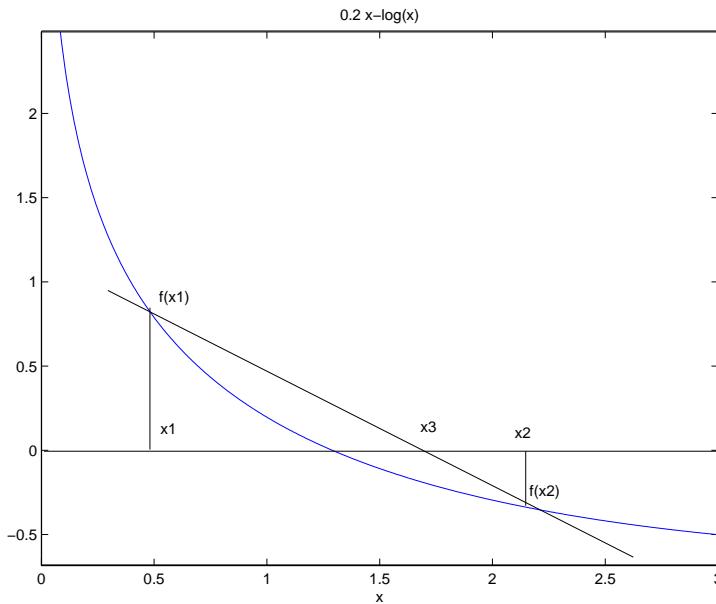


Figura 5.21: Approssimazione della soluzione

## 5.5 Il Metodo delle Secanti

♣ **Esempio 5.24.** Consideriamo la funzione:

$$f(x) = 0.2x - \log(x)$$

assegnati i punti  $x_1$  ed  $x_2$  ed i valori corrispondenti  $f(x_1)$  e  $f(x_2)$  è possibile costruire la retta secante  $y = r(x)$  passante per i punti  $(x_1, f(x_1))$  ed  $(x_2, f(x_2))$ , dove:

$$r(x) = f(x_2) + \frac{f(x_1) - f(x_2)}{x_1 - x_2}(x - x_1) \quad (5.35)$$

La Figura 5.21 mostra che la radice  $x^* = 1.2959\dots$  si può approssimare con  $x_3$  ottenuto come intersezione di  $y = r(x)$  con l'asse delle ascisse.

Tale idea è alla base del metodo numerico detto **Metodo delle Secanti**.



Il **Metodo delle Secanti** può essere interpretato come una variante del Metodo di Newton in cui, al passo  $(n+1)$ -mo anziché considerare la tangente alla curva di equazione  $y = f(x)$  nel punto di ascissa  $x_n$  si costruisce la secante alla curva nei punti di ascissa  $x_n$  e  $x_{n-1}$  rispettivamente; in altre parole,  $x_{n+1}$  è calcolata come intersezione di tale secante con l'asse delle ascisse.

Come il Metodo di Newton, anche il Metodo delle Secanti procede in modo iterativo: dati due valori iniziali  $x_0$  e  $x_1$  si calcola  $x_2$ , quindi noti  $x_1$  e  $x_2$  si calcola  $x_3$  e così via.

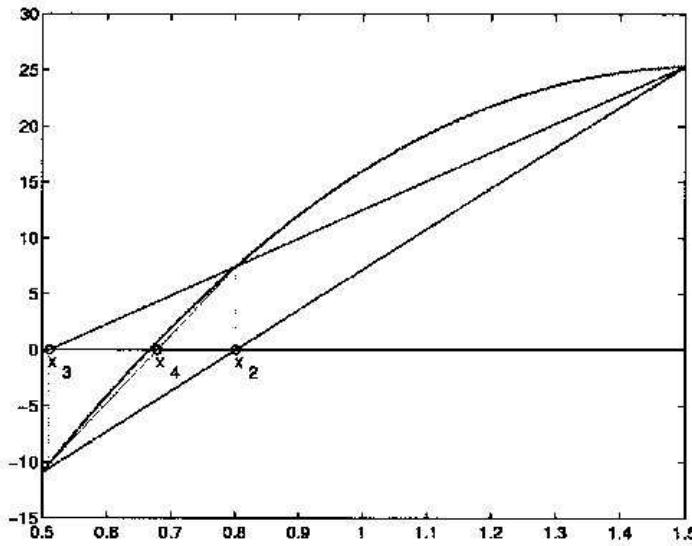


Figura 5.22: Metodo delle Secanti applicato alla risoluzione di  $3x^4 - 11x^3 - 21x^2 + 99x - 54 = 0$  con  $x_0 = 0.5$  e  $x_1 = 1.5$  (prime tre iterazioni)

La generica iterazione  $k$  può essere sintetizzata come segue:

- si considera la retta secante alla curva  $y = f(x)$  nei punti  $(x_{k-1}, f(x_{k-1}))$  e  $(x_k, f(x_k))$ ,

$$m_{k+1}(x) = f(x_k) + \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(x - x_k),$$

e si assume come approssimazione dello zero della funzione il valore,  $x_{k+1}$ , ottenuto intersecando tale retta con l'asse delle ascisse, cioè:

$$x_{k+1} = x_k - f(x_k) \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}. \quad (5.36)$$

In Fig. 5.22 è illustrata graficamente l'applicazione del Metodo delle Secanti.  
Di seguito riportiamo l'algoritmo del Metodo delle Secanti.

```

procedure fxsec(input:  $x_0, f, itmax$ , out:  $xsec, fxsec$ )
  /# SCOPO: calcolo di un'approssimazione dello zero
  /# di una funzione mediante il Metodo delle Secanti
  /# SPECIFICHE PARAMETRI:
    var:  $x_0$  : reale           {approssimazione iniziale dello zero}
    var:  $f$  : funzione esterna   {funzione di cui si cerca lo zero}
    var:  $itmax$  : reale         {numero massimo di iterazioni}
    var:  $xsec$  : reale          {approssimazione dello zero}
    var:  $fxsec$  : reale          {valore di f in xsec}
  /# INIZIO ISTRUZIONI:
     $k := 1$ 
    /# generazione della successione delle approssimazioni
    repeat
       $x_{k+1} := x_k - f(x_k)(x_k - x_{k-1})/(f(x_k) - f(x_{k-1}))$            {calcolo}
      {dell'approssimazione}
       $k := k + 1$ 
    until (condizione di arresto verificata)
  end fxsec

```

Procedura 5.3: Algoritmo del Metodo delle Secanti

### 5.5.1 Applicabilità del Metodo delle Secanti

Il Metodo delle Secanti offre, rispetto al Metodo di Newton il vantaggio di non richiedere la valutazione della derivata della funzione. Quindi, a differenza di quest'ultimo, è applicabile anche quando la funzione non ha derivata nota a priori. Tuttavia, proprio in vicinanza della soluzione, alcuni problemi computazionali possono derivare dal calcolo di:

$$\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \quad (5.37)$$

n	$x_n - x^*$ (Secanti)	$x_n - x^*$ (Newton)
1	1.0227333e+00	1.0274087e+00
2	7.9795980e-01	6.7160857e-01
3	5.7919392e-01	4.2182169e-01
4	4.1961536e-01	2.4137373e-01
5	2.8766862e-01	1.1571221e-01
6	1.8486823e-01	3.9039751e-02
7	1.0634089e-01	6.2030114e-03
8	5.1321407e-02	1.8844420e-04
9	1.8362718e-02	1.8063821e-07
10	3.9135545e-03	1.6620039e-13
11	3.4139206e-04	0.0000000e+00
12	6.7124537e-06	0.0000000e+00
13	1.1657860e-08	0.0000000e+00
14	3.9857007e-13	0.0000000e+00
15	0.0000000e+00	0.0000000e+00

**Tabella 5.18: Metodo di Newton e Metodo delle Secanti applicati alla risoluzione di  $x^4(\sin x - \cos x) = 0$ , con  $x_0 = 2.4$  e  $x_1 = 2.6$  (Secanti) e  $x_0 = 2.5$  (Newton)  $x^* = \pi/4$**

che è rapporto fra due quantità che possono diventare numericamente nulle. Da queste considerazioni è ragionevole affermare che il Metodo di Newton è più "affidabile" di quello delle Secanti; quest'ultimo (con un calcolo accurato di (5.37)) è, comunque, da preferire quando il calcolo della derivata risulta computazionalmente molto costoso o addirittura impossibile.

### 5.5.2 Convergenza

♣ **Esempio 5.25.** Applichiamo il Metodo di Newton e quello delle Secanti per risolvere in  $[0.5, 2]$ :

$$x^4(\cos x - \sin x) = 0$$

che ha uno zero in  $x^* = \pi/4$ . In Tabella 5.18 è mostrata l'applicazione di 15 iterazioni dei due metodi. Si osserva che il Metodo delle Secanti converge meno velocemente di quello di Newton. ♣

Fra Metodo di Newton e Metodo delle Secanti, non è possibile affermare la superiorità dell'uno rispetto all'altro. I risultati in Tabella 5.18 mostrano che per l'esempio in esame, almeno nelle vicinanze della soluzione, i due metodi hanno comportamenti simili (si tenga conto che il costo computazionale di Newton è circa il doppio rispetto a quello

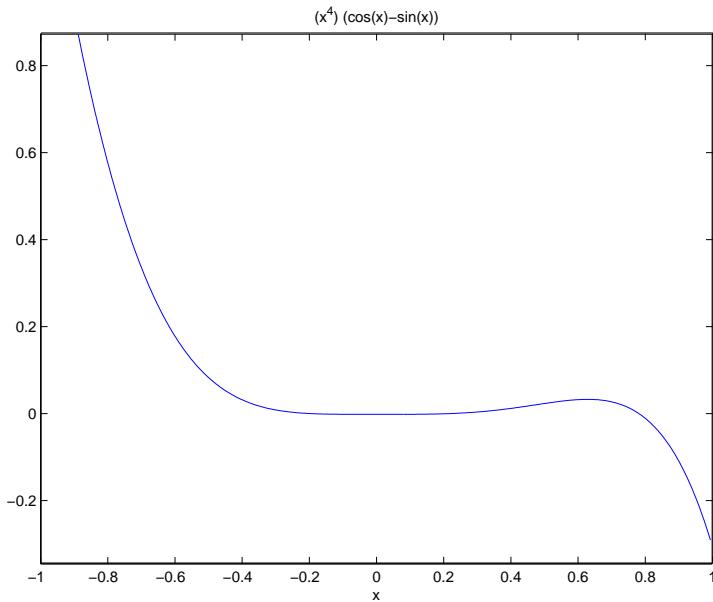


Figura 5.23: Grafico di  $f(x) = x^4(\sin x - \cos x)$  in  $[-1,1]$

delle Secanti); ciò non deve meravigliare visto che, se  $x_n$  e  $x_{n-1}$  sono vicini, allora la secante è una buona approssimazione della tangente. Il teorema che segue, ottenuto con argomenti del tutto analoghi a quelli utilizzati per la dimostrazione del Teorema 5.3, fornisce condizioni sufficienti alla convergenza del Metodo delle Secanti.

### Teorema 5.7. [Convergenza del Metodo delle Secanti]

*Si supponga  $f \in C^2([a, b])$ ,  $x^*$  sia soluzione di*

$$f(x) = 0, \quad x \in [a, b] \subseteq A,$$

*dove  $f : A \subseteq \mathbb{R} \rightarrow \mathbb{R}$  è una funzione con derivata  $f'$  definita in  $[a, b]$ , tale che  $f'(x^*) \neq 0$ . Se  $x_0$  è scelto sufficientemente vicino a  $x^*$  la successione  $\{x_n\}$  generata dal Metodo delle Secanti converge a  $x^*$  ed, inoltre*

$$e_{n+1} = -\frac{1}{2} \frac{f''(\xi_n)}{f'(\zeta_n)} e_n e_{n-1}, \quad \xi_n, \zeta_n \in \text{int}[x_n, x^*] \quad (5.38)$$

*dove  $e_h = x_h - x^*$  è l'errore locale di troncamento.*

**Dimostrazione** Sia

$$e_{k+1} = x^* - x_{k+1}.$$

Consideriamo la retta secante la curva  $y = f(x)$  nei punti  $(x_{k-1}, f(x_{k-1}))$  e  $(x_k, f(x_k))$ ,

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

Si ha

$$\begin{aligned} f(x) &= m_{k+1}(x) + f[x_{k-1}, x_k, x](x - x_{k-1})(x - x_k) = \\ &= f(x_k) + \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(x - x_k) + f[x_{k-1}, x_k, x](x - x_{k-1})(x - x_k). \end{aligned} \quad (5.39)$$

Per  $x = x_{k+1}$ ,  $m_{k+1}(x) = 0$ , ovvero

$$f(x_k) + \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(x_{k+1} - x_k) = 0. \quad (5.40)$$

Ponendo  $x = x^*$  nella (5.39) si ha

$$f(x^*) = 0 = f(x_k) + \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(x^* - x_k) + f[x_{k-1}, x_k, x^*](x^* - x_{k-1})(x^* - x_k). \quad (5.41)$$

Sottraendo membro a membro, la (5.40) dalla (5.41), si ottiene

$$0 = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(x^* - x_{k+1}) + f[x_{k-1}, x_k, x^*](x^* - x_{k-1})(x^* - x_k)$$

per cui, essendo  $e_{k+1} = x^* - x_{k+1}$  si ha:

$$e_{k+1} = -\frac{f[x_{k-1}, x_k, x^*]}{f[x_{k-1}, x_k]}(x^* - x_{k-1})(x^* - x_k),$$

avendo posto

$$f[x_{k-1}, x_k] = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}.$$

Essendo, per ipotesi,  $f \in C^2([a, b])$ , se  $x_{k-1}$  e  $x_k$  sono sufficientemente vicini a  $x^*$ , cioè se:

$$\begin{aligned} x_{k-1} &\rightarrow x^* \\ x_k &\rightarrow x^*, \end{aligned}$$

per la definizione di differenza divisa di ordine due su tre nodi coincidenti, si ha:

$$f[x_{k-1}, x_k, x^*] = \frac{f''(\xi_k)}{2} \quad \text{con } \xi_k \in (x^*, x_{k-1}, x_k).$$

Inoltre, essendo  $f'(x^*) \neq 0$  per ipotesi, esiste un intorno di  $x^*$  in cui  $|f'| > 0$  e, dunque, un punto  $\zeta_k$  che vi appartiene, per il quale  $f'(\zeta_k) \neq 0$ ; inoltre, se  $\zeta_k \in \text{int}[x_{k-1}, x_k]$ , per il Teorema del valor medio,

$$f[x_{k-1}, x_k] = f'(\zeta_k),$$

da cui

$$e_{k+1} = -\frac{1}{2} \frac{f''(\xi_k)}{f'(\zeta_k)} e_k e_{k-1}$$

e, dunque, l'asserto. ■

Si osservi come da (5.38) non segue immediatamente l'ordine di convergenza del metodo.

Supponiamo che il Metodo delle Secanti sia convergente di ordine  $p$ , ovvero, posto  $e_n = x^* - x_n$ , supponiamo che:

$$|e_{n+1}| \approx C \cdot |e_n|^p$$

Dal Teorema 5.7 abbiamo ottenuto una stima dell'errore del tipo

$$|e_{n+1}| \approx K \cdot |e_n| \cdot |e_{n-1}|$$

con  $K = \frac{1}{2} \frac{f''(\xi_n)}{f'(\zeta_n)}$ . Quindi

$$C \cdot |e_n|^p \approx K \cdot |e_n| \cdot C^{-\frac{1}{p}} \cdot |e_n|^{\frac{1}{p}}$$

da cui segue la relazione

$$C \cdot |e_n|^p \approx K \cdot C^{-\frac{1}{p}} \cdot |e_n|^{\frac{1}{p}+1}$$

che sussiste solo se

$$p = \frac{1}{p} + 1 \Rightarrow p^2 - p - 1 = 0 \Rightarrow p = \frac{1 \pm \sqrt{5}}{2} \quad (5.42)$$

e se

$$K \approx C^{1+\frac{1}{p}} \approx C^p.$$

Poiché si può dimostrare che la radice in valore assoluto minore, dell'equazione nella (5.42), può essere trascurata, allora

$$|e_{n+1}| \approx K^{\frac{1}{p}} \cdot |e_n|^p, \quad \text{con } p = \frac{1 + \sqrt{5}}{2} = 1.618 \dots (k \gg 1).$$

Concludendo, il Metodo delle Secanti ha ordine di convergenza uguale a:

$$(1 + \sqrt{5})/2 \simeq 1.62$$

e quindi converge più lentamente del Metodo di Newton (che ha ordine di convergenza 2). Si deve tenere conto, per un equo raffronto fra i due metodi, del costo computazionale del Metodo di Newton maggiore rispetto a quello delle Secanti; in particolare, se si assume che la complessità computazionale per il calcolo di  $f'$  è circa uguale a quella per il calcolo di  $f$  (assunzione ragionevole in molti casi), un'iterazione nel primo metodo ha un costo circa doppio di una iterazione che del secondo; poiché per il Metodo delle Secanti

$$|e_{n+2}| \simeq |e_{n+1}|^{1.62} \simeq |e_n|^{1.62^2} \simeq |e_n|^{2.62},$$

quest'ultimo, a parità di valutazioni complessive di funzione, può risultare più veloce del Metodo di Newton.

## 5.6 I metodi ibridi. Il Metodo di Dekker-Brent

I metodi ibridi si propongono di coniugare la robustezza (capacità di fornire sempre una soluzione) dei metodi globalmente convergenti con l'efficienza (veloce convergenza) dei metodi locali. Un esempio di tali metodi è il Metodo di Dekker Brent che rappresenta una combinazione dei metodi di Bisezione e Secanti.

Nel Metodo di Dekker-Brent, a partire da un intervallo di ricerca  $[a, b]$ :

- al fine di garantire una "sufficiente" velocità di convergenza, la scelta della approssimazione successiva avviene mediante il Metodo delle Secanti. Se il valore così calcolato cade all'esterno dell'intervallo di ricerca, si utilizza il Metodo di Bisezione.
- al fine di garantire la convergenza si individua un intervallo contenuto nel precedente come per il Metodo di Bisezione.

Nel dettaglio, il Metodo di Dekker-Brent si può descrivere nel modo seguente:

- **inizializzazione:**  $x_0 = a$ ,  $x_1 = b$ ,  $y_1 = a$ ,  $y_{-1} \equiv y_0 = a + 1$ .

- **$k + 1$ -mo passo** ( $k \geq 1$ ): viene calcolato, con il Metodo delle Secanti, il valore:

$$x_s = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})} \quad (5.43)$$

Se  $x_s$  appartiene all'intervallo che ha per estremi  $x_k$  e  $y_k$ , e  $y_k \neq y_{k-2}$ , allora

$$x_{k+1} \equiv x_s ,$$

altrimenti  $x_{k+1}$  è calcolato con il Metodo di Bisezione, cioè

$$x_{k+1} = \frac{x_k + y_k}{2}.$$

Il valore di  $y_{k+1}$ , l'altro estremo dell'intervallo di ricerca, viene calcolato come:

$$y_{k+1} = \begin{cases} x_k & \text{se } f(x_k)f(x_{k+1}) < 0 \\ y_k & \text{altrimenti} \end{cases} \quad (5.44)$$

Osserviamo che se  $y_k = y_{k-2}$  vuol dire che, per almeno 2 iterazioni, l'estremo dell'intervallo di ricerca di estremi  $x_k$  e  $y_k$  è rimasto fisso, ovvero la soluzione potrebbe trovarsi, in prossimità di questo estremo. In tal caso si utilizza la Bisezione. Infine, la (5.44) assicura che la funzione assume segno discorde nei punti  $x_{k+1}$  e  $y_{k+1}$ , pertanto l'intervallo che ha tali estremi (che quindi contiene una soluzione) ha ampiezza minore di quella dell'intervallo precedente.

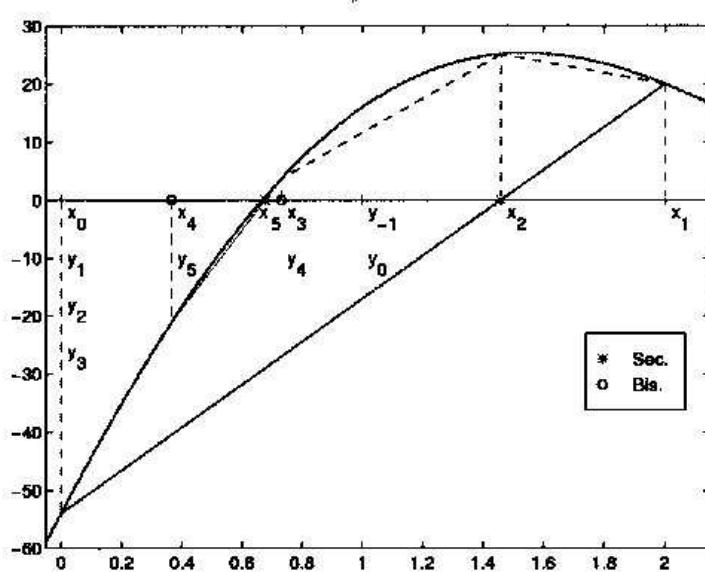


Figura 5.24: Metodo di Dekker-Brent applicato all'equazione  $f(x) = 3x^4 - 11x^3 - 21x^2 + 99x - 54$  con  $x_0 = 0$  e  $x_1 = 2$  (prime 4 iterazioni)

♣ **Esempio 5.26.** Applichiamo il Metodo di Dekker-Brent alla risoluzione del problema:

$$3x^4 - 11x^3 - 21x^2 + 99x - 54 = 0$$

In Figura 5.24 è illustrata l'applicazione del Metodo di Dekker-Brent alla risoluzione dell'equazione in  $[0, 2]$ . In Tabella 5.19 sono riportati i risultati ottenuti. Confrontando tali risultati con quelli riportati in Tabella 5.11, ottenuti applicando il Metodo di Bisezione, risulta evidente la superiorità del metodo ibrido.



Il Metodo di Dekker-Brent eredita la proprietà di metodo globalmente convergente dal Metodo di Bisezione; inoltre, per  $k$  sufficientemente grande,  $x_k$  viene calcolato secondo il Metodo delle Secanti (5.43) da cui, quindi, eredita l'ordine di convergenza.

Di seguito riportiamo l'algoritmo di Dekker-Brent, scritto in Pascal-like. Successivamente verranno descritte le condizioni di arresto.

$x_k$	$f(x_k)$
1.4595e+00	2.5172e+01
7.2973e-01	3.6369e+00
3.6486e-01	-2.1155e+01
6.7621e-01	5.6810e-01
6.6806e-01	8.3598e-02
6.6666e-01	-4.6986e-04
6.6667e-01	3.8397e-07

**Tabella 5.19:** Metodo di Dekker-Brent applicato all'equazione  $y = 3x^4 - 11x^3 - 21x^2 + 99x - 54$  con  $x_0 = 0$ ,  $x_1 = 2$  (prime 7 iterazioni)

```

procedure DB(input: x0,f,itmax, out: xdb,fxdb)
  /# SCOPO: calcolo di un'approssimazione dello zero
  /# di una funzione mediante il Metodo di Dekker-Brent
/# SPECIFICHE PARAMETRI:
  var: x0      : reale           {approssimazione iniziale dello zero}
  var: f       : funzione esterna {funzione di cui si cerca lo zero}
  var: itmax   : reale           {numero massimo di iterazioni}
  var: xdb     : reale           {approssimazione dello zero}
  var: fxdb    : reale           {valore di f in xdb}

/# INIZIO ISTRUZIONI:
  y1 := x0
  y0 := y1 + 1
  y_{-1} := y0
  f0 := f(x0)
  f1 := f(x1)
  k := 1

```

Procedura 5.4: Metodo di Dekker-Brent - continua

```

while (condizioni di arresto non verificate)
    if  $y_1 \neq y_{-1}$  then
         $d := f_1(x_1 - x_0)/(f_1 - f_0)$ 
        if  $(d(x_1 - y_1) < 0 \text{ or } |d| > |x_1 - y_1|)$  then
             $d := (x_1 - y_1)/2$ 
        endif
    else
         $d := (x_1 - y_1)/2$ 
    endif
     $x_0 := x_1; f_0 := f_1; x_1 := x_1 - d$ 
     $f_1 := f(x_1); y_{-1} := y_0; y_0 := y_1$ 
    if  $f_0 f_1 < 0$  then
         $y_1 := x_0$ 
    endif
     $k := k + 1$ 
endwhile
 $xdb := y_1;$ 
 $fxdb := f_1;$ 
end DB

```

Procedura 5.4: Metodo di Dekker-Brent - fine

## 5.7 Metodi *one point*. Il Metodo del punto fisso

Nel Metodo di Newton l'approssimazione  $x_{n+1}$  viene calcolata a partire da  $x_n$  come

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad (5.45)$$

ovvero, indicata con  $\Phi(x)$  la funzione di iterazione:

$$\Phi(x) = x - \frac{f(x)}{f'(x)}, \quad (5.46)$$

$x_{n+1}$  è calcolata mediante l'iterazione funzionale:

$$x_{n+1} = \Phi(x_n) \quad (5.47)$$

Metodi del tipo (5.47) sono detti metodi **one-point** perché ciascuna iterazione viene calcolata in funzione solo dell'iterazione precedente.

In generale è possibile, per risolvere un'equazione non lineare  $f(x) = 0$ , costruire metodi iterativi, detti metodi *one-point*, a partire da considerazioni di tipo geometrico (come per il Metodo di Newton) o analitico. Ad esempio, l'osservazione che in generale l'equazione  $f(x) = 0$  può essere equivalentemente scritta come:

$$x = f(x) + x, \quad (5.48)$$

suggerisce il metodo iterativo

$$x_{n+1} = f(x_n) + x_n, \quad (5.49)$$

che è proprio un metodo iterativo *one-point* in cui la funzione di iterazione è data dall'espressione a secondo membro della (5.48).

Più in generale, se  $x = g(x)$  è una equazione equivalente ad  $f(x) = 0$ , un metodo iterativo del tipo (5.47) in cui la funzione  $g$  sia presa come funzione di iterazione, viene detto **del punto fisso**.

Data una funzione di iterazione  $\Phi$  ed un punto iniziale  $x_0$ , è sempre possibile considerare la successione  $x_n$  generata mediante (5.47); si osservi che, se  $\Phi$  è continua e  $x_n \rightarrow x^*$ , allora

$$x^* = \lim x_n = \lim \Phi(x_{n-1}) = \Phi(x^*),$$

cioè  $x^*$  è punto fisso per la funzione  $\Phi$ . Graficamente un punto fisso è una intersezione del grafico della curva di equazione  $y = \Phi(x)$  con la bisettrice del primo e terzo quadrante del piano cartesiano.

Si osservi che un punto fisso della funzione  $\Phi$  definita secondo (5.46) è anche uno zero della funzione  $f$ .

### 5.7.1 Convergenza

Il teorema seguente può essere utilizzato per stabilire condizioni sufficienti alla convergenza del Metodo del punto fisso.

#### Teorema 5.8. [Teorema delle contrazioni]

*Sia  $\Phi$  una funzione continua e derivabile in  $[a, b]$  a valori in  $[a, b]$ . Se esiste un numero  $K < 1$  tale che  $|\Phi'(x)| \leq K$  in  $[a, b]$ , allora*

- la funzione  $\Phi$  ammette un unico punto fisso  $x^*$  in  $[a, b]$ ;
- comunque si scelga  $x_0$  in  $[a, b]$ , la successione generata mediante (5.47) converge a  $x^*$ .

Una stessa equazione può ovviamente essere espressa nella forma  $x = g(x)$  in infiniti modi, dando luogo a diverse funzioni di iterazione per il punto fisso che portano o meno alla convergenza della successione generata mediante (5.47).

♣ **Esempio 5.27.** Si supponga di risolvere l'equazione:

$$x^3 - 3 = 0$$

che ovviamente ammette l'unica soluzione:

$$\bar{x} = \sqrt[3]{3}.$$

Il Metodo del punto fisso può essere utilizzato con

$$g(x) = 3/x^2,$$

essendo l'equazione di partenza equivalente a:

$$x = 3/x^2$$

Si ottiene, quindi, la formula iterativa

$$x_{n+1} = \frac{3}{x_n^2}$$

che, scegliendo come punto iniziale  $x_1 = 1$ , genera una successione che non converge a  $\bar{x}$  (addirittura non regolare). Si noti che essendo  $g'(\bar{x}) = |-2| > 1$ , non esiste alcun intervallo contenente la radice in cui valgono le ipotesi del teorema precedente. ♣

♣ **Esempio 5.28.** Si supponga di voler risolvere l'equazione dell'esempio precedente

$$x^3 - 3 = 0$$

Tale equazione ammette anche una formulazione equivalente:

$$x = \sqrt[3]{3/x}$$

a partire dalla quale è possibile definire la formula iterativa

$$x_{n+1} = \sqrt[3]{3/x_n}$$

Si osservi che per tale formulazione si verifica  $g'(\bar{x}) = |-1/2| < 1$ . Quindi è possibile trovare intervalli, di ampiezza sufficientemente piccola contenenti la radice  $\bar{x}$ , tali che le ipotesi del teorema precedente sono verificate. ♣

♣ **Esempio 5.29.** Volendo risolvere l'equazione di Keplero (5.3), si osservi che questa è equivalente all'equazione:

$$x = E \sin x + M$$

e quindi:

$$x_{n+1} = E \sin x_n + M \quad (5.50)$$

Essendo  $|g'(x)| = |E \cos x| \leq E$  e per ipotesi  $E < 1$ , si ha, in virtù del teorema precedente, che la successione generata mediante la (5.50) converge alla soluzione dell'equazione di Keplero, qualunque sia la scelta del punto iniziale. ♣

Per quanto riguarda il raggio e la velocità di convergenza, supposta la validità delle ipotesi del Teorema 5.8, si osserva che per l'errore di troncamento, dal Teorema del valor medio, risulta:

$$e_{n+1} = x_{n+1} - \bar{x} = g(x_{n+1}) - g(\bar{x}) = g'(\xi_n)(x_{n+1} - \bar{x})$$

con  $\xi_n \in \text{int}(x_{n+1} - \bar{x})$  e, quindi,

$$\frac{e_{n+1}}{e_n} = g'(\xi_n)$$

Se  $g'$  è continua in  $\bar{x}$  e  $|g'(\bar{x})| < 1$ , il Metodo del punto fisso ha convergenza lineare con raggio di convergenza  $|g'(\bar{x})|$ .

Nell'esempio 5.28, se si sceglie  $g(x) = \sqrt{3/x}$  si ottiene che il raggio di convergenza è  $1/2$ .

## 5.8 Criteri di arresto per i metodi iterativi

I metodi numerici per la risoluzione di una equazione non lineare  $f(x) = 0$  generano una successione  $\{x_n\}$  di approssimazioni di una soluzione  $x^*$ . Un aspetto fondamentale legato all'utilizzo efficace di tali metodi è la scelta di un criterio opportuno in base al quale decidere quando arrestare il processo iterativo. A tal proposito possiamo affermare che un metodo iterativo è caratterizzato da due aspetti:

- 1) un criterio per la scelta delle approssimazioni numeriche delle soluzioni, o degli intervalli che la contengono;
- 2) un criterio di arresto che consente di decidere quando arrestare il processo iterativo.

In generale, un criterio di arresto è ritenuto soddisfacente se il suo utilizzo conduce ad un risultato sufficientemente accurato, o meglio, consente di ottenere una accuratezza prefissata. In particolare, in questo contesto, le condizioni da utilizzare per stabilire quando terminare il processo iterativo, devono in generale:

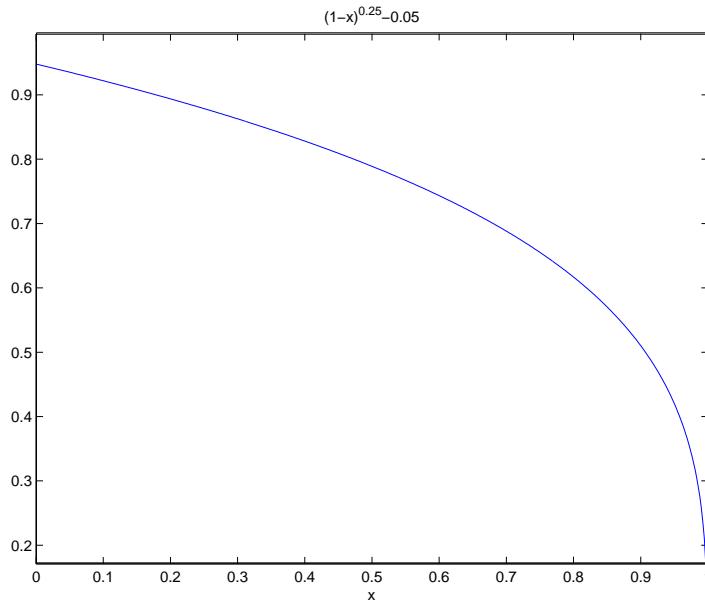


Figura 5.25: Grafico di  $f(x) = \sqrt[4]{1-x} - 0.05$  in  $[0, 1]$

1. verificare se il valore  $f(x)$  è "sufficientemente" vicino a zero. A tal fine, si può utilizzare una condizione del tipo:

$$|f(x_k)| < \epsilon_f \quad (5.51)$$

con  $\epsilon_f$  valore reale positivo prefissato. Quindi, si richiede di arrestare il procedimento quando il valore (assoluto) che la funzione assume nel valore  $x_k$  è minore di una quantità, detta **tolleranza**, la quale rappresenta l'accuratezza richiesta sulla soluzione dell'equazione.

♣ **Esempio 5.30.** Consideriamo l'equazione:

$$f(x) = \sqrt[4]{1-x} - 0.05 = 0$$

Essa ha uno zero in  $x^* = 0.99999375$ . Applicando il Metodo di Bisezione dopo 7 iterazioni si ha:

$$x_7 = 0.9922, \quad f(x_7) = 0.2473$$

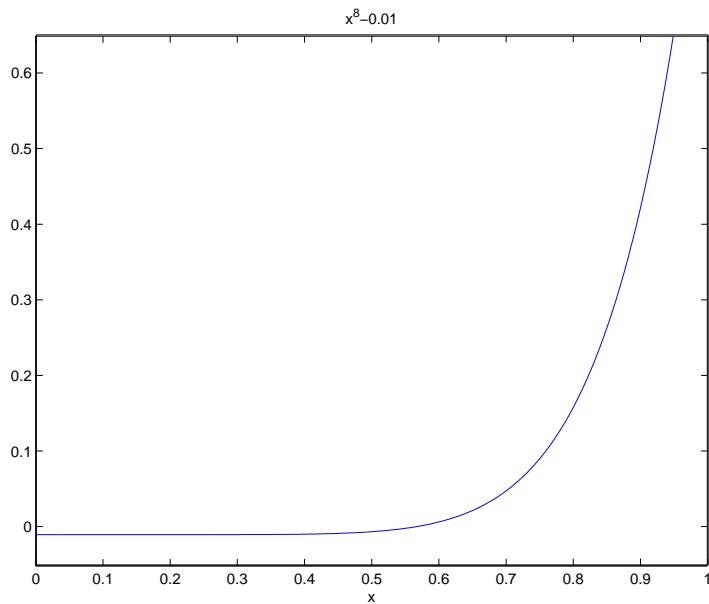
ovvero l'errore relativo su  $x^*$ :

$$E_r = \frac{|x_7 - x^*|}{|x^*|} = \frac{0.9999 - 0.9922}{|0.9999|} = 0.0076$$

Dunque l'errore che si commette sull'approssimazione della radice è di 0.7% mentre:

$$E_a = |0 - f(x_7)| = 0.2473$$

ovvero l'errore che si commette sull'approssimazione di  $f(x^*) = 0$  è di circa il 24%. ♣

Figura 5.26: Grafico di  $f(x) = x^8 - 0.01$  in  $[0, 1]$ 

2. verificare se  $x_k$  è "sufficientemente" vicino allo zero  $x^*$ .

♣ **Esempio 5.31.** Consideriamo l'equazione:

$$f(x) = x^8 - 0.01 = 0$$

Essa ha uno zero in  $x^* = 0.562341\dots$ . Arrestando il Metodo di Bisezione al passo  $k$ , per cui:

$$|f(x_k)| < 0.01$$

dopo un'iterazione abbiamo:

$$x_1 = 0.5$$

e

$$E_a = |0 - f(x_1)| = 0.0061 < 0.01;$$

considerando l'errore relativo su  $x^*$  si ottiene:

$$E_r = \frac{|x_1 - x^*|}{|x^*|} = \frac{0.562341 - 0.5}{|0.562341|} = 0.119$$

Dunque l'errore che si commette sull'approssimazione della radice è dello 11%. ♣

In generale, una condizione per l'arresto del metodo iterativo è richiedere che l'errore assoluto,  $|x_k - x^*|$ , sia minore di una quantità  $\epsilon_x$ :

$$|x_{k+1} - x^*| \leq \epsilon_x. \quad (5.52)$$

Tuttavia, l'utilizzo della (5.52) come criterio di arresto presuppone la conoscenza della soluzione dell'equazione, che non è nota a priori.

Nel caso del Metodo di Bisezione, si è già osservato che una stima dell'errore è fornita dalla semiampiezza dell'intervallo corrente. Quindi, un criterio di arresto "effettivamente utilizzabile", perché basato su quantità calcolabili dall'algoritmo, è il seguente:

$$(b_k - a_k)/2 \leq \epsilon_x. \quad (5.53)$$

Se, in particolare, si desidera stimare l'errore relativo di troncamento analitico, e quindi controllare l'accuratezza della approssimazione in termini di cifre significative, basta osservare che dalla (5.14) segue che:

$$\frac{|x_{k+1} - x^*|}{|x^*|} \leq \frac{b_k - a_k}{2|a_k|},$$

ottenendo così il criterio di arresto basato sulla distanza relativa tra gli estremi dell'intervallo corrente:

$$\frac{b_k - a_k}{2|a_k|} \leq \epsilon_x. \quad (5.54)$$

♣ **Esempio 5.32.** Riprendiamo l'equazione della Figura 5.26:

$$f(x) = x^8 - 0.01 = 0$$

La funzione ha uno zero in  $x^* = 0.562341\dots$ . Assegniamo, come condizione di arresto, il verificarsi di una delle eventualità descritte nei punti 1. e 2., ovvero che al generico passo  $k$ :

1.  $|f(x_k)| < \epsilon_f = 10^{-3}$ ;
2.  $\frac{b_k - a_k}{2|a_k|} \leq \epsilon_x = 10^{-4}$ .

Dai valori della Tabella 5.20, si osserva che dopo 10 iterazioni del Metodo di Bisezione,

1.  $|f(x_{10})| = |f(0.560546)| \approx 2.52 \times 10^{-4} < \epsilon_f = 10^{-3}$ ;
2.  $\frac{b_{10} - a_{10}}{2|a_{10}|} = \frac{0.5625 - 0.56054}{2(0.56054)} \approx 10^{-3} > \epsilon_x = 10^{-4}$ .

$k$	$a_k$	$b_k$	$x_k$	$f(x_k)$
1	0	1	0.5	-0.60937
2	0.5	1	0.75	$0.9 \times 10^{-1}$
3	0.5	0.75	0.625	0.0132
4	0.5	0.625	0.5625	$0.259 \times 10^{-4}$
5	0.5	0.5625	0.5312	$-0.366 \times 10^{-2}$
...	...	...	...	...
8	0.5546	0.5625	0.5546	$-0.52 \times 10^{-3}$
9	0.5585	0.5625	0.5585	$-0.25 \times 10^{-3}$
<b>10</b>	<b>0.56054</b>	<b>0.5625</b>	<b>0.560546</b>	$-0.11 \times 10^{-3}$

**Tabella 5.20:** 10 iterazioni del Metodo di Bisezione applicato alla funzione in Fig. 5.26

per cui il Metodo di Bisezione si arresta, in questo caso, per il verificarsi della condizione 1. ♣

♣ **Esempio 5.33.** Riprendiamo l'equazione della Figura 5.25:

$$f(x) = \sqrt[4]{(1-x)} - 0.05 = 0$$

La funzione ha uno zero in  $x^* = 0.9999\dots$ . Assegnamo, come condizione di arresto, il verificarsi di una delle eventualità descritte nei punti 1. e 2., ovvero che al generico passo  $k$ :

1.  $|f(x_k)| < \epsilon_f = 10^{-3}$ ;
2.  $\frac{b_k - a_k}{2|a_k|} \leq \epsilon_x = 10^{-3}$ .

Dai valori della Tabella 5.21, si osserva che dopo 10 iterazioni del Metodo di Bisezione,

1.  $|f(x_{10})| = |f(0.9902)| \approx 1.26 \times 10^{-1} > \epsilon_f = 10^{-3}$ ;
2.  $\frac{b_{10} - a_{10}}{2|a_{10}|} = \frac{1 - 0.9902}{2(0.9902)} \approx 4.88 \times 10^{-4} < \epsilon_x = 10^{-3}$ .

per cui il Metodo di Bisezione si arresta, in questo caso, per il verificarsi della condizione 2. ♣

Per il Metodo di Newton invece è possibile stimare la quantità  $|x_n - x^*|$  a partire da considerazioni di tipo analitico. Si osservi, infatti, che

$$x^* - x_n = -\frac{f(x_n)}{f'(\xi_n)} \quad \xi_n \in [x_n, x^*].$$

$k$	$a_k$	$b_k$	$x_k$	$f(x_k)$
1	0	1	0.5	$7.9 \times 10^{-1}$
2	0.5	1	0.75	$6.5 \times 10^{-1}$
3	0.75	1	0.875	$5.4 \times 10^{-1}$
4	0.875	1	0.937	$4.5 \times 10^{-1}$
5	0.937	1	0.968	$3.7 \times 10^{-1}$
...	...	...	...	...
8	0.998	1	0.992	$2.0 \times 10^{-1}$
9	0.992	1	0.996	$1.6 \times 10^{-1}$
10	<b>0.9902</b>	<b>1</b>	<b>0.9902</b>	$1.26 \times 10^{-1}$

**Tabella 5.21:** 10 iterazioni del Metodo di Bisezione applicato alla funzione in Fig. 5.25

Se assumiamo  $f'(x_n) \simeq f'(\xi_n)$  (assunzione ragionevole se la successione è convergente e  $x_n$  sia in un intorno sufficientemente piccolo della soluzione) allora si ha:

$$x^* - x_n \simeq -\frac{f(x_n)}{f'(x_n)} = x_{n+1} - x_n \quad (5.55)$$

Una stima dell'errore relativo è invece data da

$$\frac{x^* - x_n}{x_{n+1}} \simeq \frac{x_{n+1} - x_n}{x_n}$$

♣ **Esempio 5.34.** Applichiamo la stima calcolabile dell'errore (5.55) per il Metodo di Newton applicato alla risoluzione dell'equazione:

$$x^2 - 10^{-6} = 0$$

la Tabella 5.22 riporta il confronto tra l'errore (noto in quanto tale è la soluzione) e stima dell'errore stesso. ♣

Infine, le condizioni da utilizzare per stabilire quando terminare il processo iterativo, devono

3. considerare un controllo sul numero di iterazioni eseguite; ciò può essere fatto ponendo un limite massimo di iterazioni sul numero da eseguire:

$$k > ITMAX \quad (5.56)$$

n	$f(x_n)$	$ x_n - x^* $	$ x_n - x_{n+1} $
1	.250D+00	.499D+00	.500D+00
2	.625D-01	.249D+00	.250D+00
3	.156D-01	.124D+00	.125D+00
4	.391D-02	.615D-01	.625D-01
5	.976D-03	.303D-01	.312D-01
6	.244D-03	.146D-01	.156D-01
7	.607D-04	.686D-02	.779D-02
8	.149D-04	.299D-02	.386D-02
9	.350D-05	.112D-02	.187D-02
10	.680D-06	.296D-03	.825D-03

**Tabella 5.22: Metodo di Newton applicato alla risoluzione dell'equazione  $x^2 - 10^{-6} = 0$  con  $x_0 = 1$ . ( $x^* = 0.001$ )**

La convergenza di un metodo iterativo ad una soluzione di  $f(x) = 0$  deve essere controllata utilizzando in modo combinato la condizione al punto 1) ed una di quelle al punto 2).

È importante una corretta interpretazione e gestione delle condizioni di arresto. Ad esempio il verificarsi della condizione (5.56) può essere sintomo sia di non convergenza sia di convergenza lenta; discernere fra i due casi è di importanza fondamentale, ma non facile.

Si osserva infine che le condizioni di arresto ai punti 1. e 2. potrebbero essere, in generale, non soddisfacenti, per ogni metodo iterativo. Una condizione di arresto valida deve infatti consentire di gestire eventualità differenti:

- (a) *Raggiungimento della soluzione* Nella iterata corrente,  $x_n$ , il valore della funzione è piccolo. Ciò, ad esempio, può essere verificato con una condizione del tipo

$$|f(x_n)| < \epsilon_f \quad (5.57)$$

con tolleranza prefissata,  $\epsilon_f$ , sul valore della funzione. Si noti che una condizione *assoluta* del tipo (5.57) potrebbe essere fuorviante essendo dipendente dall'ordine di grandezza della funzione.

Una condizione *relativa*, più soddisfacente della (5.57), in quanto indipendente dall'ordine di grandezza della funzione di cui si vogliono determinare le radici è la seguente:

$$|f(x_n)| < \epsilon_f \|f\| \quad (5.58)$$

dove  $\|\cdot\|$  è una qualsiasi norma; si osservi che una condizione di questo tipo, in generale più valida della (5.57), è

$$|f(x_n)| < \epsilon_f |f(x_0)| \quad (5.59)$$

Il vantaggio di (5.59) rispetto a (5.58) è quello di non richiedere il calcolo di alcuna norma; tuttavia, (5.59) può essere non praticabile in alcuni casi (ad es. quando  $x_0$  è già una buona stima della soluzione).

- ( $\beta$ ) La successione  $x_n$  si è numericamente arrestata ovvero si verifica una condizione del tipo:

$$|x_n - x_{n-1}| < \epsilon_x \quad (5.60)$$

con  $\epsilon_x$  tolleranza prefissata. Il criterio (5.60), essendo assoluto (indipendente cioè dall'ordine di grandezza delle quantità in gioco), è in genere sconsigliabile. Più affidabile è, ad esempio, il seguente:

$$|x_n - x_{n-1}| < \epsilon |x_{n-1}| \quad (5.61)$$

È possibile considerare un criterio ulteriore per stabilire la convergenza della successione  $\{x_n\}$ , dipendente dal sistema aritmetico floating point a precisione finita utilizzato. Infatti si potrebbe verificare che:

$$x_{n-1} \oplus (x_n \ominus x_{n-1}) \equiv x_{n-1} \quad (5.62)$$

che esprime il fatto che la quantità  $(x_n - x_{n-1})$  nel sistema aritmetico non dà contributo alla somma con  $x_{n-1}$ . La (5.62), però, è una condizione troppo forte e quindi non utilizzabile in pratica.

Siamo ora in grado di modificare gli algoritmi dei metodi di Bisezione, Newton e Dekker-Brent. Di seguito, riportiamo l'algoritmo di Bisezione con il criterio di arresto descritto e l'algoritmo di Newton. Per quest'ultimo si osservi come non si sia tenuto conto di problemi legati alla precisione della macchina; si è comunque prevista l'eventualità che si verifichi  $f'(x_n) \simeq 0$ . Inoltre, contrariamente a quanto avviene con il Metodo di Bisezione, il controllo sul massimo numero di iterazioni eseguite è qui utilizzato anche per verificare la convergenza o meno del metodo.

```

procedure fxbis(input:  $a,b,f,\epsilon_f,MAXIT$ , out:  $c,fc$ )
  /# SCOPO: calcolo di un'approssimazione dello zero
  /# di una funzione mediante il Metodo di Bisezione
  /# SPECIFICHE PARAMETRI:
    var:  $a$  : reale {primo estremo dell'intervallo di ricerca}
    var:  $b$  : reale {secondo estremo dell'intervallo di ricerca}
    var:  $f$  : funzione esterna {funzione di cui si cerca lo zero}
    var:  $\epsilon_f$  : reale {accuratezza richiesta}
    var:  $MAXIT$  : reale {numero massimo di iterazioni}
    var:  $c$  : reale {approssimazione dello zero}
    var:  $fc$  : reale {valore di  $f$  in  $c$ }
  /# INIZIO ISTRUZIONI:
     $fa := f(a);$ 
     $fb := f(b);$ 
     $k := 0;$ 
    repeat
      /# calcolo del punto medio e del valore della funzione
       $c := (a + b)/2;$ 
       $fc := f(c);$ 
      if ( $fc * fa \leq 0$ ) then {test sul segno di  $f$ }
         $b := c;$ 
         $fb := fc;$ 
      else

```

Procedura 5.5: Algoritmo di Bisezione (seconda versione) - continua

```
a := c;  
fa := fc;  
endif  
k := k + 1;  
until(( $(b - a)/2 \leq \epsilon_{mac}a$ ) or ( $fc < \epsilon_f$ )) or  $k > MAXIT$ ) {criterio di arresto}  
end fxbis
```

Procedura 5.5: Algoritmo di Bisezione (seconda versione) - fine

```

procedure newton(input:  $x_0, f, f', ftol, xtol, itmax$ , out:  $x_{new}, f_{xnew}$ )
  /* SCOPO: calcolo di un'approssimazione dello zero
  /* di una funzione mediante il Metodo di Newton
  /* SPECIFICHE PARAMETRI:
    var:  $x_0$  : reale {approssimazione iniziale dello zero}
    var:  $f$  : funzione esterna {funzione di cui si cerca lo zero}
    var:  $f'$  : funzione esterna {derivata di  $f$ }
    var:  $ftol$  : funzione esterna {accuratezza valutazione}
    var:  $xtol$  : reale {accuratezza soluzione}
    var:  $itmax$  : reale {numero massimo di iterazioni}
    var:  $x_{new}$  : reale {approssimazione dello zero}
    var:  $f_{xnew}$  : reale {valore di  $f$  in  $x_{new}$ }
  /* INIZIO ISTRUZIONI:
     $k := 0$ 
     $oldx := x_0$ 
  /* generazione della successione delle approssimazioni
  repeat
    if  $f'(oldx) = 0$  then
       $rapx := 0$ 
    else
       $rapx := f(oldx)/f'(oldx)$ 
    endif

```

Procedura 5.6: Algoritmo per il calcolo degli zeri di una funzione con il Metodo di Newton  
 - continua

```
xnew := oldx - rapx
err := |rapx|
relerr := err/|xnew|
oldx := xnew
k := k + 1
until ( (|fxnew| ≥ ftol) and (relerr ≥ xtol) and (k < itmax) )
end newton
```

Procedura 5.6: Algoritmo per il calcolo degli zeri di una funzione con il Metodo di Newton  
- fine

```

procedure DB(input:  $x_0, f, itmax$ , out:  $x_{db}, f_{x_{db}}$ )
  /* SCOPO: calcolo di un'approssimazione dello zero
  /* di una funzione mediante il Metodo di Dekker-Brent
  /* SPECIFICHE PARAMETRI:
    var:  $x_0$  : reale {approssimazione iniziale dello zero}
    var:  $f$  : funzione esterna {funzione di cui si cerca lo zero}
    var:  $itmax$  : reale {numero massimo di iterazioni}
    var:  $x_{db}$  : reale {approssimazione dello zero}
    var:  $f_{x_{db}}$  : reale {valore di  $f$  in  $x_{db}$ }
  /* INIZIO ISTRUZIONI:
     $y_1 := x_0$ 
     $y_0 := y_1 + 1$ 
     $y_{-1} := y_0$ 
     $f_0 := f(x_0)$ 
     $f_1 := f(x_1)$ 
     $k := 1$ 
    while ( $|x_1 - y_1| > \epsilon_x |x_1|$  and  $|f_1| > \epsilon_f$  and  $k < itmax$ )
      if  $y_1 \neq y_{-1}$  then
         $d := f_1(x_1 - x_0)/(f_1 - f_0)$ 
        if ( $d(x_1 - y_1) < 0$  or  $|d| > |x_1 - y_1|$ ) then
           $d := (x_1 - y_1)/2$ 
        endif
      else
         $d := (x_1 - y_1)/2$ 
      endif

```

Procedura 5.7: Metodo di Dekker-Brent - continua

```
x0 := x1; f0 := f1; x1 := x1 - d  
f1 := f(x1); y-1 := y0; y0 := y1  
if f0f1 < 0 then  
    y1 := x0  
endif  
    k := k + 1  
endwhile  
xdb := y1;  
fxdb := f1;  
end DB
```

Procedura 5.7: Metodo di Dekker-Brent - fine

## 5.9 Esercizi sulla risoluzione numerica di equazioni non lineari

### 5.9.1 Metodo di bisezione - Esercizi

**Esercizio 1** Sia  $c_n = \frac{1}{2}(a_n + b_n)$ ,  $r = \lim_{n \rightarrow \infty} c_n$  e  $e_n = r - c_n$ . Denotiamo con  $[a_n, b_n]$  la successione di intervalli che derivano dal metodo di bisezione applicato ad una funzione continua  $f$ .

- (a) Dimostrare che  $|e_n| \leq 2^{-n}(b_1 - a_1)$ .
- (b) Dimostrare che  $e_n = \mathcal{O}(2^{-n})$  per  $n \rightarrow \infty$ .
- (c) È vero che  $|e_0| \geq |e_1| \geq \dots$ ? Spiegare perché.
- (d) Dimostrare che  $|c_n - c_{n+1}| = 2^{-n-2}(b_0 - a_0)$ .
- (e) Dimostrare che, per ogni  $n, m$ ,  $a_m \leq b_n$ .
- (f) Dimostrare che  $r$  è l'unico elemento dell'insieme  $\bigcup_{n=0}^{\infty} [a_n, b_n]$ .
- (g) Dimostrare che, per ogni  $n$ ,  $[a_{n+1}, b_{n+1}] \subset [a_n, b_n]$ .

**Esercizio 2** Nel metodo di bisezione il generico intervallo  $[a_{n-1}, b_{n-1}]$  è diviso a metà ed una delle due parti è scelta come nuovo intervallo. Definiamo  $d_n = 0$  se  $[a_n, b_n]$  è la metà sinistra di  $[a_{n-1}, b_{n-1}]$ , altrimenti  $d_n = 1$ . Esprimere la radice determinata dall'algoritmo, in termini della sequenza  $d_1, d_2, \dots$ . (**Suggerimento:** Considerare il caso in cui  $[a_0, b_0] = [0, 1]$ , quindi pensare alla rappresentazione binaria della radice)

**Esercizio 3** Esiste  $\lim_{n \rightarrow \infty} |r - c_{n+1}| / |r - c_n|$ ? Dimostrare.

**Esercizio 4** Trovare una formula che coinvolga  $b_0 - a_0$  e  $\epsilon$  per il numero di passi del metodo di bisezione, necessari per garantire  $|r - c_n| < \epsilon$ .

**Esercizio 5** Trovare una radice positiva dell'equazione

$$x^2 - 4x\sin(x) + (2\sin(x))^2 = 0$$

con due cifre significative corrette. Utilizzare un sistema aritmetico a precisione finita, caratterizzato da  $t = 11$ .

**Esercizio 6** Consideriamo il metodo di bisezione applicato all'intervallo  $[1.5, 3.5]$ .

- (a) Qual è l'ampiezza dell'intervallo dopo  $n$  passi?
- (b) Qual è la massima distanza possibile tra la radice  $r$  ed il punto medio di tale intervallo?

**Esercizio 7** Applicando il metodo di bisezione ad una funzione continua si genera una successione di intervalli  $[a_0, b_0], [a_1, b_1], \dots$ . Sia  $r = \lim_{n \rightarrow \infty} a_n$ . Quali di queste affermazioni può essere falsa?

- (a)  $a_0 \leq a_1 \leq a_2 \leq \dots$
- (b)  $|r - 2^{-1}(a_n + b_n)| \leq 2^{-n}(b_0 - a_0) \quad (n \geq 0)$
- (c)  $|r - 2^{-1}(a_{n+1} + b_{n+1})| \leq |r - 2^{-1}(a_n + b_n)| \quad (n \geq 0)$
- (d)  $[a_{n+1}, b_{n+1}] \subset [a_n, b_n] \quad (n \geq 0)$
- (e)  $|r - a_n| = \mathcal{O}(2^{-n})$  per  $n \rightarrow \infty$
- (f)  $|r - c_n| < |r - c_{n-1}| \quad (n \geq 1)$

**Esercizio 8** Determinare una formula che coinvolga  $a_0, b_0, \epsilon$  che fornisca il numero di passi necessario per ottenere un'approssimazione della radice con un errore relativo minore o uguale a  $\epsilon$ . Assumere che  $a_0 > 0$ .

**Esercizio 9** Se utilizziamo il metodo di bisezione sull'intervallo  $[128, 129]$ , su un calcolatore che è caratterizzato da un epsilon macchina dell'ordine di  $10^{-6}$ , possiamo calcolare la radice con un errore assoluto minore di  $10^{-6}$ ?

Rispondere alla stessa domanda, riferendosi all'errore relativo.

**Esercizio 10** Provare che il punto  $c$  calcolato nel metodo di bisezione può essere pensato come il punto di intersezione della retta per i punti  $(a, \text{sign}(f(a))), (b, \text{sign}(f(b)))$ , con l'asse delle ascisse.

### 5.9.2 Metodo di bisezione - Computer problems

**Problema 1** Scrivere e testare un sottoprogramma o una procedura che implementi l'algoritmo di bisezione. Utilizzare le seguenti funzioni:

- (a)  $x^{-1} - \tan(x)$  in  $[0, \pi/2]$
- (b)  $x^{-1} - 2^x$  in  $[0, 1]$
- (c)  $2^{-1} + e^x + 2\cos(x) - 6$  in  $[1, 3]$
- (d)  $(x^3 + 4x^2 + 3x + 5)/(2x^3 - 9x^2 + 18x - 2)$  in  $[0, 4]$

**Problema 2** Trovare una radice di  $f(x) = x - \tan(x)$  in  $[1, 2]$ .

**Problema 3** Trovare una radice di

$$x^8 - 36x^7 + 546x^6 - 4536x^5 + 22449x^4 - 67284x^3 + 118124x^2 - 109584x + 40320 = 0,$$

nell'intervallo  $[5.5, 6.5]$ . Cambiare  $-36$  con  $-36.001$  e ripetere l'esecuzione dello svolgimento.

### 5.9.3 Metodo di Newton - Esercizi

**Esercizio 11** Trovare il più piccolo punto iniziale positivo per il quale il metodo di Newton diverge se applicato alla funzione  $f(x) = \tan^{-1}(x)$ .

**Esercizio 12** Per calcolare l'inverso di un numero  $R$  senza effettuare la divisione, si può cercare la radice  $r = 1/R$  di  $f(x) = x^{-1} - R$ . Scrivere un breve algoritmo per trovare  $1/R$  utilizzando il metodo di Newton, evitando l'uso di divisioni ed esponenziali.

**Esercizio 13** Se applichiamo il metodo di Newton a  $f(x) = x^2 - 1$  con  $x_0 = 10^{10}$ , quanti passi sono necessari per ottenere una radice con un'accuratezza di  $10^{-8}$ ? (Risolvere analiticamente e non sperimentalmente)

**Esercizio 14** Supponiamo che  $r$  sia uno zero di molteplicità doppia di  $f$ . Allora,  $f(r) = f'(r) = 0 \neq f''(r)$ . Mostrare che se  $f''$  è continua, allora il metodo di Newton avrà  $e_{n+1} \approx 1/2 \cdot e_n$ , cioè convergenza lineare.

**Esercizio 15** Consideriamo una variazione del metodo di Newton in cui una sola deriva-ta deve essere calcolata:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)}.$$

Determinare  $C$  e  $s$  tali che

$$e_{n+1} = Ce_n^s.$$

**Esercizio 16** Provare che l'iterazione del metodo di Newton diverge per le seguenti funzioni, per qualsiasi punto (reale) iniziale

- (a)  $f(x) = x^2 + 1$
- (b)  $f(x) = 7x^4 + 3x^2 + \pi$

**Esercizio 17** Trovare le condizioni su  $\alpha$  in modo da garantire che l'iterazione

$$x_{n+1} = x_n - \alpha f(x_n)$$

converga linearmente allo zero di  $f$  se il punto iniziale è vicino a zero.

**Esercizio 18** Dimostrare che se  $r$  è uno zero di molteplicità  $k$  della funzione  $f$ , allora si ottiene una convergenza quadratica sostituendo al metodo di Newton l'iterazione seguente

$$x_{n+1} = x_n - k \frac{f(x_n)}{f'(x_n)}.$$

### 5.9.4 Metodo di Newton - Computer problems

**Problema 4** Scrivere un programma che risolva l'equazione  $x = \tan(x)$  con il metodo di Newton. Trovare le radici più vicine a 4.5 e 7.7.

**Problema 5** Trovare il minimo positivo della funzione

$$f(x) = x^{-2} \tan(x)$$

calcolando lo zero di  $f'$  con il metodo di Newton.

**Problema 6** Risolvere l'equazione

$$x^3 + 3x = 5x^2 + 7$$

con il metodo di Newton. Eseguire 10 passi partendo da  $x_0 = 5$ .

**Problema 7** L'equazione

$$2x^4 + 24x^3 + 61x^2 - 16x + 1 = 0$$

ha due radici vicino a 0.1. Determinarle usando il metodo di Newton.

**Problema 8** Perturbare il termine costante della funzione

$$f(x) = e^x - 1.5 - \operatorname{arctg}(x)$$

e vedere come varia la radice negativa.

**Problema 9** Scrivere e testare un programma che valuti le prime 10 radici dell'equazione

$$\tan(x) = x.$$

**Problema 10** Scrivere le prime cinque iterazioni del metodo di Newton applicato, nel campo complesso, alla funzione

$$f(z) = 1 + z^2 + e^z,$$

partendo da  $z_0 = -1 + 4i$ .

### 5.9.5 Metodo delle Secanti - Computer problems

**Problema 11** Scrivere un programma che implementi il metodo delle secanti, ricevendo, in input, due valori iniziali. Testarlo sulle funzioni seguenti:

- (a)  $\sin(x/2) - 1$

- (b)  $e^x - \tan(x)$   
(c)  $x^3 - 12x^2 + 3x + 1$

**Problema 12** Scrivere e testare un raffinamento del metodo delle secanti che utilizzi l'approssimazione di  $f'$

$$f'(x) \approx \frac{k^2 f(x+h) - h^2 f(x+k) + (h^2 - k^2) f(x)}{(k-h)kh}$$

nella formula di Newton. Osserviamo che sono necessari tre valori iniziali.

**Problema 13** Risolvere le equazioni seguenti senza utilizzare la derivata:

- (a)  $x^{20} - 1$ , nell'intervallo  $[0, 10]$   
(b)  $\tan(x) - 30x$ , nell'intervallo  $[1, 1.57]$   
(c)  $x^2 - (1-x)^{10}$ , nell'intervallo  $[0, 1]$   
(d)  $x^3 + 10^{-4}$ , nell'intervallo  $[-0.75, 0.5]$   
(e)  $x^{19} + 10^{-4}$ , nell'intervallo  $[-0.75, 0.5]$   
(f)  $x^5$ , nell'intervallo  $[-1, 10]$   
(g)  $x^9$ , nell'intervallo  $[-1, 10]$   
(h)  $xe^{-x^2}$ , nell'intervallo  $[-1, 4]$

**Problema 14** Testare il programma che implementa il metodo delle secanti sulla funzione

$$f(x) = x^3 - \sinh(x) + 4x^2 + 6x + 9,$$

utilizzando come punti iniziali, dapprima 7 e 8 e poi 3 e 10. Commentare i risultati ottenuti.

### 5.9.6 Metodo del punto fisso - Esercizi

**Esercizio 19** In astronomia, l'equazione di Keplero risulta

$$x = y - \epsilon \cdot \sin(y),$$

con  $0 < \epsilon < 1$ . Mostrare che, per ogni  $x \in [0, \pi]$  c'è un  $y$  che soddisfa l'equazione. Interpretare tale risultato in termini di punto fisso.

**Esercizio 20** Consideriamo una funzione di iterazione del tipo

$$F(x) = x + f(x)g(x),$$

con  $f(r) = 0$ ,  $f'(r) \neq 0$ . Trovare le condizioni su  $g$  affinché il metodo del punto fisso abbia una convergenza cubica.

**Esercizio 21** Avendo a disposizione una calcolatrice (o, equivalentemente, utilizzando un sistema aritmetico a precisione finita, caratterizzato da  $t = 11$ ), che numero si ottiene, in corrispondenza di  $\pi/4$ , premendo ripetutamente il bottone della funzione coseno? Motivare il risultato.

**Esercizio 22** Se si utilizza il metodo di Newton per cercare il punto fisso della funzione  $F$ , cioè per risolvere l'equazione  $F(x) - x = 0$ , che formula iterativa si ottiene?

**Esercizio 23** Sia  $p$  un numero positivo. Dire qual è il valore della seguente espressione:

$$x = \sqrt{p + \sqrt{p + \sqrt{p + \dots}}}$$

Notare che tale espressione può essere interpretata come

$$x = \lim_{n \rightarrow \infty} x_n,$$

con  $x_1 = \sqrt{p}$ ,  $x_2 = \sqrt{p + \sqrt{p}}$  e così via.

(**Suggerimento:** osservare che  $x_{n+1} = \sqrt{p + x_n}$ )

**Esercizio 24** Sia  $p > 1$ . Dire qual è il valore della seguente espressione:

$$x = \frac{1}{p + \frac{1}{p + \frac{1}{p + \dots}}}$$

Utilizzare l'idea proposta nell'esercizio precedente. Dimostrare che la sequenza converge utilizzando il Teorema delle contrazioni.

**Esercizio 25** In base ai risultati ottenuti nell'esercizio precedente, trovare le radici dell'equazione  $x^2 + px + q = 0$ , sviluppando un metodo iterativo.

**Esercizio 26** Molti processi iterativi non sono espressi nella semplice formulazione  $x_{n+1} = F(x_n)$ , con  $F : \mathbb{R} \rightarrow \mathbb{R}$ . Per esempio si potrebbe avere una funzione  $F : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ . Mostrare che il metodo di bisezione e quello delle secanti sono di questo tipo. In entrambi i casi definire  $F$  esplicitamente.

**Esercizio 27** Se il metodo della iterazione funzionale è applicato alla funzione

$$F(x) = 2 + (x - 2)^4,$$

partendo da  $x_0 = 2.5$ , che ordine di convergenza si ottiene? Trovare l'intervallo a cui deve appartenere  $x_0$  affinché il metodo converga. Notare che 2 è un punto fisso.

**Esercizio 28** Mostrarre che le seguenti funzioni sono contrazioni nei domini dati, ma che esse non hanno punti fissi.

(a)  $F(x) = 3 - x^2$ , su  $[-1/4, 1/4]$

(b)  $F(x) = -x/2$ , su  $[2, -1] \cup [1, 2]$

Perché tale risultato non contraddice il Teorema delle contrazioni?

**Esercizio 29** Provare che se  $f$  è una funzione continua nell'intervallo  $[a, b]$  e risulta  $a \leq f(a)$  e  $b \leq f(b)$ , allora  $f$  ha un punto fisso nell'intervallo  $[a, b]$ . Osserviamo che non si suppone che  $a \leq f(x) \leq b$  per ogni  $x \in [a, b]$ .

**Esercizio 30** Per trovare uno zero della funzione  $f$  si può pensare di cercare il punto fisso di

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

Per trovare un punto fisso di  $F$  si può applicare il metodo di Newton all'equazione  $F(x) - x = 0$ . Facendo ciò che formula iterativa si ottiene?

**Esercizio 31** Se il metodo della iterazione funzionale è applicato alla funzione

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

a partire da  $x_0 = 7$ , la sequenza ottenuta converge? Se così, qual è il limite? Motivare la risposta rigorosamente.

**Esercizio 32** Provare o confutare:

Se  $F : \mathbb{R} \rightarrow [a, b]$  è una contrazione in  $[a, b]$ , allora  $F$  ha un unico punto fisso che può essere ottenuto con il metodo dell'iterazione funzionale, partendo da un qualsiasi punto iniziale.

**Esercizio 33** Fornire esempi di funzioni che non hanno punti fissi ma che soddisfino a tali proprietà:

- (a)  $f : [0, 1] \rightarrow [0, 1]$
- (b)  $f : (0, 1) \rightarrow (0, 1)$  continua
- (c)  $f : A \rightarrow A$  continua, con  $A = [0, 1] \cup [2, 3]$
- (d)  $f : \mathbb{R} \rightarrow \mathbb{R}$  continua

**Esercizio 34** Verificare che

$$f(x) = 2 + x - \arctg(x)$$

è tale che  $|f'(x)| < 1$ . Provare, inoltre, che  $f$  non ha punti fissi e dire perché ciò non contraddice il Teorema delle contrazioni.

**Esercizio 35** Scrivere due diverse procedure di punto fisso per trovare lo zero di

$$f(x) = 2x^2 + 6e^{-x} - 4.$$

**Esercizio 36** Se il metodo dell'iterazione funzionale è applicato a

$$F(x) = x^2 + x - 2$$

e produce una successione convergente di numeri positivi, qual è il suo limite e qual è il valore di  $x_0$ ?

**Esercizio 37** Consideriamo una funzione del tipo  $F(x) = x - f(x)f'(x)$ , con  $f(r) = 0$  e  $f'(r) \neq 0$ . Trovare le condizioni su  $f$  affinché il metodo dell'iterazione funzionale converga al più cubicamente a  $r$  se  $x_0$  appartiene ad un suo intorno.

**Esercizio 38** Se, erroneamente, il metodo di Newton è riportato come

$$x_{n+1} = f(x_n)/f'(x_n),$$

viene trovato uno zero di  $f$ ? Qual è l'ordine di convergenza?