

# Capitolo 1

## II MATLAB

### 1.1. Introduzione al MATLAB

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

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

```
>>
```

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

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

```
>> help
```

oppure

```
>> help comando
```

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

```
>> help load
```

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

```
>> demo
```

a questo punto compariranno sullo schermo alcuni menu e basterà scegliere, tramite il mouse, l'argomento del quale si vuole vedere una dimostrazione.

Il Matlab può essere considerato un interprete le cui istruzioni sono del tipo:

```
variabile = espressione
```

oppure

```
variabile
```

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

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

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

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

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

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

## 1.2. Assegnazione di matrici

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

Per esempio l'istruzione

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

asigna alla variabile `A` una matrice di tre righe e tre colonne. Gli elementi di una riga della matrice possono essere separate da virgole o dallo spazio, mentre le diverse righe sono separate da un punto e

virgola. Se alla fine dell'assegnazione viene messo il punto e virgola allora la matrice non viene visualizzata sullo schermo. In generale se vogliamo assegnare ad  $A$  una matrice ad  $m$  righe ed  $n$  colonne la sintassi è la seguente:

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

Per assegnare ad una variabile  $x$  un vettore riga si ha

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

gli elementi possono anche essere separati da una virgola

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

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

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

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

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

mentre in modo analogo si può anche aggiungere una colonna:

```
>> A = [-1 2 3; 0 5 6; -5 4 3];  
>> x = [-7; 0; 9];  
>> A = [ A, x];
```

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

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

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

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

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

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

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

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

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

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

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

e successivamente

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

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

```
Index exceeds matrix dimension
```

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

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

fornisce come risposta

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

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

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

si ha:

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

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

### 1.3. Sottomatrici e notazione ":"

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

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

e

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

sono equivalenti all'assegnazione diretta del vettore **x**:

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

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

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

equivale a scrivere

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

Inoltre è possibile anche l'uso di incrementi negativi:

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

è equivalente a

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

L'istruzione

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

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

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

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

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

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

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

assegna al vettore colonna  $y$  la terza colonna di  $A$ ;

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

assegna al vettore riga  $y$  la seconda riga di  $A$ ;

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

assegna alla matrice  $B$  le prime 4 righe di  $A$ ;

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

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

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

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

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

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

## Operazioni su matrici e vettori

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

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

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

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

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

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

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

si definisce nel seguente modo:

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

ed è la matrice che viene calcolata scrivendo l'istruzione

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

In caso contrario se scriviamo

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

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

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

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

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

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

## 1.4. Costanti predefinite

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

<code>eps</code>	precisione di macchina ( $\simeq 2.2 \cdot 10^{-16}$ )
<code>pi</code>	$\pi$ (cioè 3.14159265358979)
<code>i</code>	unita' immaginaria ( $\sqrt{-1}$ )
<code>j</code>	unita' immaginaria ( $\sqrt{-1}$ )
<code>realmax</code>	il piu' grande numero floating point ( $1.7976e + 308$ )
<code>realmin</code>	il piu' piccolo numero floating point ( $2.2251e - 308$ )
<code>inf</code>	infinito ( $\infty$ )
<code>NaN</code>	Not a Number.

La costante `inf` è ottenuta come risultato di una divisione per zero oppure il calcolo del logaritmo di zero o se il risultato è un overflow (per esempio `2*realmax`). La costante `NaN` invece è ottenuta come risultato di operazioni matematicamente non definite come  $0/0$  oppure  $\infty - \infty$ . Come accade per la maggior parte dei linguaggi di programmazione anche in Matlab è possibile definire variabili il cui nome è una costante predefinita, quindi per esempio è possibile usare la variabile `i` come indice intero.

## 1.5. Operatori relazionali e logici

I seguenti sono gli operatori relazionali

<	minore
>	maggiore
<=	minore o uguale
>=	maggiore o uguale
==	uguale
~=	diverso

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

```
>> 3<5
```

otterremo

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

oppure scrivendo

```
>> 1>5
```

la risposta è

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

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

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

```

    0 1

>> A>B
ans =
    0 1
    1 0

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

```

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

&	AND
	OR
~	NOT

## 1.6. Funzioni scalari

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

<code>sin</code>	seno
<code>cos</code>	coseno
<code>tan</code>	tangente
<code>asin</code>	arcoseno
<code>acos</code>	arcocoseno
<code>atan</code>	arcotangente
<code>sinh</code>	seno iperbolico
<code>cosh</code>	coseno iperbolico
<code>tanh</code>	tangente iperbolica
<code>asinh</code>	arcoseno iperbolico
<code>acosh</code>	arcocoseno iperbolico
<code>atanh</code>	arcotangente iperbolica
<code>exp</code>	esponenziale

<code>log</code>	logaritmo naturale
<code>log10</code>	logaritmo in base 10
<code>sqrt</code>	radice quadrata
<code>abs</code>	valore assoluto
<code>rem</code>	resto della divisione
<code>sign</code>	segno
<code>round</code>	arrotondamento
<code>floor</code>	parte intera inferiore
<code>ceil</code>	parte intera superiore

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

<code>x</code>	<code>round(x)</code>	<code>floor(x)</code>	<code>ceil(x)</code>
3.7	4	3	4
3.1	3	3	4
-4.7	-5	-5	-4
-4.3	-4	-5	-4

Osserviamo come `floor(x)` è sempre più piccolo di `x` mentre `ceil(x)` è maggiore di `x`.

## 1.7. Funzioni vettoriali

Altre funzioni Matlab operano essenzialmente su vettori (riga o colonna), ma possono agire anche su matrici in modo tale da produrre un vettore riga contenente i risultati della loro applicazione a ciascuna colonna. Per ottenere il risultato della loro azione sulle righe basta applicare la stessa funzione alla matrice trasposta  $A'$ . Alcune di queste funzioni sono:

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

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

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

## 1.8. Funzioni di matrici

Le più utili funzioni di matrici sono le seguenti:

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

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

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

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

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

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

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

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

calcola la norma infinito di  $A$  mentre

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

calcola la norma 1 di  $A$ .

## 1.9. Le istruzioni `for`, `while`, `if` e `switch`

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

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

La variabile *var* assume come valore iniziale *val\_0*, viene eseguita la lista di istruzioni che segue, poi è incrementata del valore *step*, vengono rieseguite le istruzioni che seguono e così via, finché il suo valore non supera *val\_1*. Il valore dello *step* può essere negativo, nel qual caso il valore di *val\_0* deve essere logicamente superiore a *val\_1*.

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

```
while espressione logica
    istruzioni
end
```

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

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

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

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

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

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

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

```
switch rem(n,2)
    case 0
        disp('n e' un numero pari')
    case 1
```

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

## 1.10. Istruzioni per gestire il Workspace

Il comando

```
>> who
```

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

```
>> whos
```

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

```
>> clear nome variabile
```

mentre il comando

```
>> clear
```

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

```
>> save
```

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

```
>> save nomefile
```

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

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

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

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

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

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

```
>> load
```

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

```
>> load nomefile
```

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

## 1.11. M-files

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

### Script files

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

```
>> prova
```

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

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

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

## Function files

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

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

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

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

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

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

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

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

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

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

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

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

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

```
return
```

## 1.12. Messaggi di errore, Istruzioni di Input

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

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

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

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

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

l'istruzione `error` causa l'interruzione nell'esecuzione del file. In un M-file è possibile introdurre dati di input in maniera interattiva mediante l'istruzione `input`. Per esempio quando l'istruzione

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

viene eseguita allora il messaggio è visualizzato sullo schermo e il programma rimane in attesa che l'utente digiti un valore da tastiera e tale valore, di qualsiasi tipo esso sia, viene assegnato alla variabile `iter`. Vediamo ora un modo per poter memorizzare in un file di ascii l'output di un M-file oppure di una sequenza di istruzioni Matlab. Infatti

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

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

### 1.13. Formato di output

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

```
format short
```

È il formato utilizzato per default dal programma ed è di tipo fixed point con 4 cifre decimali;

```
format long
```

Tale formato è di tipo fixed point con 14 cifre decimali;

```
format short e
```

Tale formato è la notazione scientifica (esponenziale) con 4 cifre decimali;

```
format long e
```

Tale formato è la notazione scientifica (esponenziale) con 15 cifre decimali. Vediamo per esempio come i numeri  $4/3$  e  $1.2345e - 6$  sono rappresentati nei formati che abbiamo appena descritto e negli altri disponibili:

```
format short          1.3333
format short e       1.3333e+000
format short g       1.3333
format long          1.333333333333333
format long e       1.333333333333333e+000
format long g       1.333333333333333
format rat           4/3
```

```
format short          0.0000
format short e       1.2345e-006
format short g       1.2345e-006
format long          0.00000123450000
format long e       1.234500000000000e-006
format long g       1.2345e-006
format rat           1/810045
```

Oltre ai formati appena visti il comando

```
format compact
```

serve a sopprimere le righe vuote e gli spazi dell'output scrivendo sullo schermo il maggior numero di informazione possibile, in modo appunto compatto.

## 1.14. La grafica con il Matlab

Il Matlab dispone di numerose istruzioni per grafici bidimensionali e tridimensionali e anche di alcune funzioni per la creazione di animazioni. Il comando `plot` serve a disegnare curve nel piano  $xy$ . Infatti se  $x$  e  $y$  sono due vettori di uguale lunghezza allora il comando

```
>> plot(x,y)
```

traccia una curva spezzata che congiunge i punti  $(x(i),y(i))$ . Per esempio

```
>> x=-4:.01:4;  
>> y=sin(x);  
>> plot(x,y)
```

traccia il grafico della funzione seno nell'intervallo  $[-4,4]$ .

L'istruzione `plot` ammette un parametro opzionale di tipo stringa (racchiuso tra apici) per definire il tipo e il colore del grafico. Infatti è possibile scegliere tra 4 tipi di linee, 5 di punti e 8 colori base. In particolare

'-'	linea continua
'--'	linea tratteggiata
'-.'	linea tratteggiata e a punti
':'	linea a punti
'+'	piu'
'o'	cerchio
'x'	croce
'.'	punto
'*'	asterisco
'y'	colore giallo
'r'	colore rosso
'c'	colore ciano
'm'	colore magenta
'g'	colore verde
'w'	colore bianco
'b'	colore blu
'k'	colore nero

Volendo tracciare per esempio un grafico con linea a puntini e di colore verde L'istruzione è:

```
>> plot(x,y,':g')
```

L'istruzione `plot` consente di tracciare più grafici contemporaneamente. Per esempio

```
>> x=-pi:pi/500:pi;  
>> y=sin(x);  
>> y1=sin(2*x);  
>> y2=sin(3*x);  
>> plot(x,y,'r',x,y1,'+g',x,y2,'--b')
```

traccia tre grafici nella stessa figura, il primo a tratto continuo (tratto di default) rosso, il secondo verde con i punti segnati dal carattere '+' e il terzo tratteggiato e di colore blu. Vediamo ora le altre più importanti istruzioni grafiche:

```
>> title(stringa)
```

serve a dare un titolo al grafico che viene visualizzato al centro nella parte superiore della figura;

```
>> xlabel(stringa)
```

stampa una stringa al di sotto dell'asse delle ascisse;

```
>> ylabel(stringa)
```

stampa una stringa a destra dell'asse delle ordinate (orientata verso l'alto). Per inserire un testo in una qualsiasi parte del grafico esiste il comando

```
>> text(x,y,'testo')
```

che posiziona la stringa di caratteri *testo* nel punto di coordinate  $(x, y)$  ( $x$  e  $y$  non devono essere vettori). Di tale comando ne esiste anche una versione che utilizza il mouse:

```
>> gtext('testo')
```

posiziona il testo nel punto selezionato all'interno del grafico schiacciando il pulsante sinistro del mouse.

Il grafico tracciato con il comando `plot` è scalato automaticamente, questo vuol dire che le coordinate della finestra grafica sono calcolate dal programma, tuttavia l'istruzione

```
>> axis([x_min x_max y_min y_max])
```

consente di ridefinire gli assi, e quindi le dimensioni della finestra del grafico corrente.

Una volta tracciato il grafico per poterlo stampare è necessario che venga memorizzato in un file in formato postscript. L'istruzione che consente ciò è

```
>> print -dps nome
```

in questo caso il grafico tracciato in precedenza viene memorizzato nel file *nome.ps* che può essere successivamente stampato utilizzando il comando di stampa del sistema operativo che si sta utilizzando.

A volte può essere utile, una volta tracciato un grafico, ingrandire alcune parti dello stesso. Questo può essere fatto utilizzando il comando `zoom`. Per attivare tale caratteristica è sufficiente il comando

```
>> zoom on
```

mentre per disattivarlo bisogna scrivere:

```
>> zoom off
```

Il funzionamento di tale istruzione è molto semplice. Una volta attivato lo zoom per ingrandire un'area del grafico è sufficiente portare il puntatore del mouse in tale area e cliccare con il tasto sinistro dello stesso. Tale operazione può essere ripetuta alcune volte (non si può ottenere l'ingrandimento un numero molto grande di volte). Per rimpicciolire il grafico bisogna cliccare con il tasto destro.

Le istruzioni grafiche del Matlab permettono di tracciare curve in tre dimensioni, superfici, di creare delle animazioni e così via. Per approfondire le istruzioni che consentono queste operazioni può essere fatto richiedendo l'`help` per le istruzioni `plot3`, `mesh` e `movie`.

# Capitolo 2

## Sistemi lineari

### 2.1. Introduzione al Calcolo Numerico

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

### 2.2. Elementi di Algebra Lineare

Sia  $\mathbb{R}$  l'insieme dei numeri reali. Generalmente si indica con  $\mathbb{R}^{m \times n}$  l'insieme delle *matrici* ad elementi reali aventi  $m$  righe ed  $n$  colonne. Quindi una matrice è una *tabella a doppia entrata* di numeri reali. Per

esempio:

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

con  $a_{ij} \in \mathbb{R}$  ed  $m, n \in \mathbb{N}$ . I numeri interi  $m$  ed  $n$  si dicono *dimensioni della matrice*, ovvero  $A$  si dice matrice di *dimensioni*  $m \times n$  o di *ordine*  $m \times n$ . Se  $m = n$  allora la matrice  $A$  si dice *quadrata* di dimensione  $n$  o di ordine  $n$  altrimenti si dice *rettangolare*. Se  $i$  e  $j$  sono numeri interi con  $1 \leq i \leq m$  e  $1 \leq j \leq n$  allora l'elemento della matrice  $A$  di dimensione  $m \times n$  che si trova in posizione  $(i, j)$  viene indicato con  $a_{ij}$ . Gli elementi  $a_{ij}$  di una matrice quadrata  $A$  di ordine  $n$  tali che  $i = j$  sono detti *elementi principali* o *diagonali* e formano la cosiddetta diagonale principale di  $A$ .

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**Esempio 2.2.1** Siano  $A$  e  $B$  le seguenti matrici:

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

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

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

In definitiva

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

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

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

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

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

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

e

$$B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

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

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

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

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

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

$$AI_n = I_nA = A.$$

**Definizione 2.2.1** Una matrice che si ottiene da  $I_n$  scambiando alcune righe (o colonne) viene detta matrice di permutazione.

**Esempio 2.2.2** Sia  $P$  la seguente matrice di permutazione:

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

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

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

e calcoliamo il prodotto  $PA$ :

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

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

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

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

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

**Definizione 2.2.2** Se  $A \in \mathbb{R}^{n \times n}$  è una matrice di ordine  $n$ , si definisce determinante di  $A$  il numero

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

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

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

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

Il determinante è pure uguale a

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

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

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

allora

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

Il determinante ha le seguenti proprietà:

1. Se  $A$  è una matrice triangolare o diagonale allora

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

2.  $\det I = 1$ ;
3.  $\det A^T = \det A$ ;
4.  $\det AB = \det A \det B$  (Regola di Binet);

5. se  $\alpha \in \mathbb{R}$  allora  $\det \alpha A = \alpha^n \det A$ .
6.  $\det A = 0$  se una riga (o una colonna) è combinazione lineare di due (o più) righe (o colonne) di  $A$ .
7. Se  $A$  è una matrice triangolare a blocchi

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

con  $B$  e  $D$  matrici quadrate, allora

$$\det A = \det B \det D. \quad (2.1)$$

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

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

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

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

## Vettori

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

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

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

**Esempio 2.2.3** *Sia*

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

e sia  $\mathbf{x}$  il vettore

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

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

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

Tra vettori sono consentite le seguenti operazioni:

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

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

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

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

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

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

$$A = \mathbf{xy}^T$$

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

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

**Esempio 2.2.4** *Siano  $\mathbf{x}$  e  $\mathbf{y}$  i seguenti vettori:*

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

e

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

*Calcoliamo prima il prodotto interno:*

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

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

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

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

*Tale prodotto non gode della proprietà commutativa, infatti:*

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

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

## Norme Vettoriali

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

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

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

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

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

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

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

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

## Norme su Matrici

**Definizione 2.2.3** Una funzione  $\|\cdot\| : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$  tale che per ogni matrice  $A \in \mathbb{R}^{n \times n}$ ,  $\|A\|$  soddisfa:

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

si dice norma di matrice.

**Definizione 2.2.4** Si dice che una norma di matrice è compatibile con una norma di vettore se per ogni matrice  $A$  e per ogni vettore  $\mathbf{x}$  risulta

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

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

$$\|A\| = \sup_{\mathbf{x} \neq 0} \frac{\|A\mathbf{x}\|}{\|\mathbf{x}\|}. \quad (2.2)$$

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

La (2.2) può anche scriversi:

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

anzi è possibile dimostrare che

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

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

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

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

## 2.3. Metodi diretti per sistemi lineari

Ci poniamo il seguente problema:

*Siano assegnati  $n^2 + n$  numeri reali, che indichiamo rispettivamente con  $a_{ij}$  e  $b_i$ ,  $i, j = 1, \dots, n$ . Determinare, se esistono,  $n$  numeri reali*

$x_i$ , con  $i = 1, \dots, n$ , tali che:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= b_2 \\ \vdots & \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n &= b_n. \end{aligned} \tag{2.3}$$

Le (2.3) definiscono un *sistema di  $n$  equazioni algebriche lineari* nelle  $n$  incognite  $x_1, x_2, \dots, x_n$ . Se definiamo la matrice  $A = [a_{ij}]$  con  $i, j = 1, \dots, n$ , ed i vettori  $\mathbf{x}^T = (x_1, x_2, \dots, x_n)$  e  $\mathbf{b}^T = (b_1, b_2, \dots, b_n)$ , cosicchè le equazioni (2.3) assumono la forma:

$$\mathbf{Ax} = \mathbf{b}. \tag{2.4}$$

Il vettore  $\mathbf{x}$  viene detto *vettore soluzione*,  $A$  è detta *matrice dei coefficienti* e  $\mathbf{b}$  *vettore dei termini noti*. Nel seguito assumeremo che il determinante di  $A$  sia diverso da 0. Un metodo universalmente noto per risolvere il problema (2.4) è l'applicazione della cosiddetta *Regola di Cramer* la quale fornisce:

$$x_i = \frac{\det A_i}{\det A} \quad i = 1, \dots, n, \tag{2.5}$$

dove  $A_i$  è la matrice ottenuta da  $A$  sostituendo la sua  $i$ -esima colonna con il termine noto  $\mathbf{b}$ . Dalla (2.5) è evidente che per ottenere una componente della soluzione è necessario il calcolo di  $n + 1$  determinanti di ordine  $n$ . Si può facilmente dedurre che il numero di operazioni necessarie per il calcolo del determinante di una matrice di ordine  $n$  è circa  $n!$ , quindi questa strada non permette di poter determinare velocemente la soluzione del nostro sistema.

## 2.4. Sistemi triangolari

Prima di affrontare la soluzione algoritmica di un sistema lineare vediamo qualche particolare sistema che può essere agevolmente risolto.

Assumiamo che il sistema da risolvere abbia la seguente forma:

$$\begin{array}{cccccc}
 a_{11}x_1 & +a_{12}x_2 & \dots & +a_{1i}x_i & \dots & +a_{1n}x_n & = b_1 \\
 & a_{22}x_2 & \dots & +a_{2i}x_i & \dots & +a_{2n}x_n & = b_2 \\
 & & \ddots & \vdots & & \vdots & \vdots \\
 & & & a_{ii}x_i & \dots & +a_{in}x_n & = b_i \\
 & & & & \ddots & \vdots & \vdots \\
 & & & & & a_{nn}x_n & = b_n
 \end{array} \quad (2.6)$$

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

$$\left\{ \begin{array}{l} x_n = \frac{b_n}{a_{nn}} \\ \\ x_i = \frac{b_i - \sum_{j=i+1}^n a_{ij}x_j}{a_{ii}} \quad i = n-1, \dots, 1. \end{array} \right. \quad (2.7)$$

Il metodo (2.7) prende il nome di *metodo di sostituzione all'indietro*, poichè il vettore  $\mathbf{x}$  viene calcolato partendo dall'ultima componente. Anche per il seguente sistema il vettore soluzione è calcolabile in modo analogo.

$$\begin{array}{cccccc}
 a_{11}x_1 & & & & & = b_1 \\
 a_{21}x_1 & +a_{22}x_2 & & & & = b_2 \\
 \vdots & \vdots & \ddots & & & \vdots \\
 a_{i1}x_1 & +a_{i2}x_2 & \dots & +a_{ii}x_i & & = b_i \\
 \vdots & \vdots & & & \ddots & \vdots \\
 a_{n1}x_1 & +a_{n2}x_2 & \dots & +a_{ni}x_i & \dots & +a_{nn}x_n = b_n
 \end{array} \quad (2.8)$$

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

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

## 2.5. Metodo di Eliminazione di Gauss

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

$$\begin{aligned} 2x_1 + x_2 + x_3 &= -1 \\ 6x_1 + 2x_2 + x_3 &= 1 \\ 4x_1 - 2x_2 + x_3 &= 2 \end{aligned}$$

La soluzione di un sistema lineare non cambia se un'equazione viene sostituita dalla combinazione lineare di due (o più) equazioni dello stesso sistema. L'idea alla base del metodo di Gauss è quella di ottenere un sistema lineare con matrice dei coefficienti triangolare superiore effettuando opportune combinazioni lineari tra le equazioni. Poniamo

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

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

equazione. Poniamo

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

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

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

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

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

Poniamo

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

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

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

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

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

Al secondo passo il sistema lineare è diventato:

$$\begin{array}{rclcrcl} 2x_1 & +x_2 & +x_3 & = & -1 & \\ & -x_2 & -2x_3 & = & 4 & \\ & -4x_2 & -x_3 & = & 4 & \end{array}$$

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

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

Cerchiamo ora di azzerare gli elementi sottodiagonali della seconda colonna. Lasciamo inalterata le prime due equazioni del sistema. Poniamo

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

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

$$-4x_2 - 8x_3 = 16$$

La nuova terza equazione sarà la differenza tra la terza equazione e la seconda appena modificata

$$\begin{array}{rcl} -4x_2 & -x_3 & = 4 \\ 4x_2 & +8x_3 & = -16 \\ \hline & 7x_3 & = -12 \quad [\text{Nuova terza equazione}]. \end{array}$$

Abbiamo ottenuto un sistema triangolare superiore:

$$\begin{array}{rcl} 2x_1 & +x_2 & +x_3 & = -1 \\ & -x_2 & -2x_3 & = 4 \\ & & 7x_3 & = -12. \end{array}$$

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

$$A^{(3)} = \begin{pmatrix} 2 & 1 & 1 \\ 0 & -1 & -2 \\ 0 & 0 & 7 \end{pmatrix}, \quad \mathbf{b}^{(3)} = \begin{pmatrix} -1 \\ 4 \\ -12 \end{pmatrix}.$$

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

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

Fissiamo quindi una riga  $i$ ,  $i \geq 2$ , e calcoliamo gli elementi  $a_{ij}^{(2)}$ :

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

dove

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

e

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

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

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

Fissiamo quindi una riga  $i$ ,  $i \geq 3$ , e calcoliamo gli elementi  $a_{ij}^{(3)}$ :

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

dove

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

e

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

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

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} a_{kj}^{(k)}, \quad i, j = k + 1, \dots, n \quad (2.9)$$

e

$$b_i^{(k+1)} = b_i^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} b_k^{(k)}, \quad i = k + 1, \dots, n. \quad (2.10)$$

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

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

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

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

con

$$l_{i1} = -\frac{a_{i1}^{(1)}}{a_{11}^{(1)}} \quad i = 2, \dots, n. \quad (2.11)$$

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

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

avendo indicato con  $\mathbf{l}_1$  il vettore i cui elementi sono definiti da (2.11), con  $\mathbf{a}_1^T$  gli elementi della prima riga di  $A^{(1)}$ , escluso il primo, con  $\hat{\mathbf{a}}_1^T$  gli elementi della prima colonna di  $A^{(1)}$ , escluso il primo, e con  $I_{n-1}$  la matrice identità di ordine  $n - 1$ . Calcoliamo ora il prodotto tra queste due matrici:

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

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

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

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

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

quindi  $L^{(1)}A^{(1)} = A^{(2)}$ . In modo simile si può verificare che  $\mathbf{b}^{(2)} = L^{(1)}\mathbf{b}^{(1)}$ .

Siamo passati dal sistema  $A^{(1)}\mathbf{x} = \mathbf{b}^{(1)}$  al sistema equivalente ad esso  $A^{(2)}\mathbf{x} = \mathbf{b}^{(2)}$ . Al passo successivo definiamo la matrice

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

con

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

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

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

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

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

e

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

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

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

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

1° passo:

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

2° passo:

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

⋮

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

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

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

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

e gli elementi  $a_{ij}^{(k)}$  e  $b_i^{(k)}$  sono dati dalle relazioni (2.9) e (2.10).

Nell'eseguire il metodo di Gauss si è fatta l'implicita ipotesi (vedi formule (2.9) e (2.10)) che i cosiddetti *elementi pivotali*  $a_{kk}^{(k)}$  siano non nulli per  $k = 1, 2, \dots, n-1$ . In vero questa non è un'ipotesi limitante in quanto la non singolarità di  $A$  permette, con un opportuno scambio di righe in  $A^{(k)}$ , di ricondursi a questo caso. Infatti scambiare due righe in  $A^{(k)}$  significa sostanzialmente scambiare due equazioni nel sistema  $A^{(k)}\mathbf{x} = \mathbf{b}^{(k)}$  e ciò non altera la natura del sistema stesso.

Consideriamo la matrice  $A^{(k)}$  e supponiamo  $a_{kk}^{(k)} = 0$ . Se  $a_{ik}^{(k)} = 0$  per  $i = k+1, \dots, n$ , allora  $A^{(k)}$  ha la seguente struttura:

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

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

$$A^{(k)} = \begin{pmatrix} T_{k-1} & * \\ 0 & \hat{T}_{n-k+1} \end{pmatrix}$$

segue che  $\det A^{(k)} = 0$  e di conseguenza dovrebbe essere anche  $\det A = 0$  e questo contrasta con l'ipotesi fatta. Quindi possiamo concludere che se  $a_{kk}^{(k)} = 0$  e  $\det A \neq 0$  deve necessariamente esistere un elemento  $a_{ik}^{(k)} \neq 0$ , con  $i \in \{k+1, k+2, \dots, n\}$ .

## 2.6. Equivalenza tra Fattorizzazione LU e Metodo di Gauss

Vogliamo ora provare che il metodo di eliminazione di Gauss senza scambio di righe equivale a fattorizzare la matrice  $A$  dei coefficienti nel prodotto di una matrice triangolare inferiore, che denotiamo con  $L$  avente elementi diagonali tutti uguali a 1, e una matrice triangolare superiore  $U$ :

$$A = LU.$$

A tal fine risultano utili i seguenti risultati.

**Lemma 2.6.1** *L'inversa della  $k$ -esima matrice di Gauss*

$$L^{(k)} = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & l_{k+1,k} & 1 & \\ & & \vdots & & \ddots \\ & & l_{nk} & & & 1 \end{pmatrix}$$

è data da

$$(L^{(k)})^{-1} = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -l_{k+1,k} & 1 & \\ & & \vdots & & \ddots \\ & & -l_{nk} & & & 1 \end{pmatrix}.$$



Gauss senza scambio di righe equivale a fattorizzare  $A$  nella forma  $A = LU$  dove  $L$  è triangolare inferiore con elementi diagonali uguali a 1 mentre  $U$  è triangolare superiore. Infatti applicando il metodo di Gauss senza scambio di righe abbiamo:

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

ovvero

$$A = (L^{(1)})^{-1} \dots (L^{(n-1)})^{-1} A^{(n)}$$

dove  $A^{(n)}$  è, per costruzione, triangolare superiore. Ora, a causa della struttura di  $L^{(i)}$  e in virtù dei Lemma 2.6.1 e 2.6.2, segue:

$$(L^{(1)})^{-1} \dots (L^{(n-1)})^{-1} = \begin{pmatrix} 1 & & & 0 \\ -l_{21} & 1 & & \\ \vdots & \ddots & \ddots & \\ -l_{n1} & \dots & -l_{n,n-1} & 1 \end{pmatrix}.$$

Posto  $L = (L^{(1)})^{-1} \dots (L^{(n-1)})^{-1}$  e  $U = A^{(n)}$  segue

$$A = LU.$$

## 2.7. Esistenza e Unicità della Fattorizzazione LU

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

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

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

allora

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

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

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

**Teorema 2.7.1** *Sia  $A$  una matrice quadrata di ordine  $n$  non singolare. Sia  $A_k$  la sottomatrice principale di testa di ordine  $k$  di  $A$  e sia  $\det A_k \neq 0$  per  $k = 1, 2, \dots, n$ , cioè siano i minori principali di testa non nulli. Allora esistono un'unica matrice  $L$  triangolare inferiore con elementi diagonali uguali a 1 ed un'unica  $U$  triangolare superiore tali che:  $A = LU$ .*

## 2.8. Strategie di Pivoting nel Metodo di Eliminazione di Gauss

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

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

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

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

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

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

In alternativa alla strategia di pivoting parziale si può effettuare la strategia di *Pivoting totale* che costa un po' di più ma che rende più stabile il metodo di Gauss. La strategia di pivoting totale è la seguente:

1. determinare gli indici  $r, s$  tali che

$$|a_{rs}^{(k)}| = \max_{k \leq i, j \leq n} |a_{ij}^{(k)}|;$$

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

## 2.9. Fattorizzazione di Matrici Non Singolari

Abbiamo visto che se  $A$ , matrice quadrata di ordine  $n$ , è non singolare e tutti i suoi minori principali di testa fino all'ordine  $n - 1$  sono non nulli allora  $A$  ammette un'unica fattorizzazione  $LU$ . Se  $A$  è solo non singolare non è detto che ammetta fattorizzazione  $LU$ . In particolare si può dimostrare che è sempre possibile trovare una *matrice di permutazione*  $P$  (vedere definizione 2.2.1) tale che  $PA = LU$ .

Infatti vale il seguente risultato.

**Teorema 2.9.1** *Se  $A$  è una matrice quadrata non singolare allora esistono una matrice di permutazione  $P$ , una matrice  $L$  triangolare inferiore con elementi uguali a 1 sulla diagonale principale ed una matrice  $U$  triangolare superiore tali che:*

$$PA = LU.$$

## 2.10. Classi di Matrici che non hanno bisogno di Pivoting

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

**Definizione 2.10.1** *A si dice a predominanza diagonale per colonne se:*

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

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

**Definizione 2.10.2** *Una matrice quadrata  $A$ , simmetrica, di ordine  $n$  si dice definita positiva se e solo se:*

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

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

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

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

## 2.11. Fattorizzazione Diretta

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

$$A = LU$$

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

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

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

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

e poi quello triangolare superiore

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

Imponiamo quindi che la matrice  $A$  ammetta fattorizzazione  $LU$ :

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

Deve essere

$$a_{ij} = \sum_{k=1}^{\min(i,j)} l_{ik}u_{kj} \quad i, j = 1, \dots, n. \quad (2.12)$$

Uguagliando la parte triangolare superiore di  $A$  abbiamo

$$a_{ij} = \sum_{k=1}^i l_{ik}u_{kj} \quad j \geq i \quad (2.13)$$

ovvero

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

Imponendo  $l_{ii} = 1$  risulta

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj} \quad j \geq i \quad (2.14)$$

e ovviamente  $u_{1j} = a_{1j}$ , per  $j = 1, \dots, n$ . Uguagliamo ora la parte triangolare strettamente inferiore di  $A$ . Sempre da (2.12) risulta:

$$a_{ij} = \sum_{k=1}^j l_{ik}u_{kj} \quad i > j \quad (2.15)$$

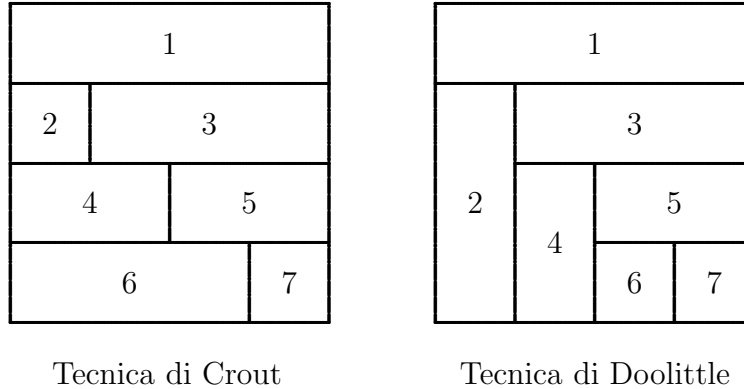


Figura 2.1: Tecniche di calcolo diretto della fattorizzazione di una matrice

ovvero

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

da cui

$$l_{ij} = \frac{1}{u_{jj}} \left( a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj} \right) \quad i \geq j. \quad (2.16)$$

Si osservi che le formule (2.14) e (2.16) vanno implementate secondo uno degli schemi riportati in Figura 2.1.

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

## 2.12. Condizionamento di sistemi lineari

Uno dei più importanti aspetti legati all'uso di algoritmi numerici per la soluzione di problemi matematici è l'affidabilità dei risultati ottenuti. I metodi descritti in questo capitolo funzionerebbero sempre "bene" (cioè fornirebbero risultati numericamente affidabili) se non si dovesse aver a che fare con una serie di problemi legati alla struttura dell'elaboratore

che stiamo utilizzando. Un primo problema che ci troviamo ad affrontare è il modo con cui i numeri reali sono rappresentati nella memoria di un elaboratore. Giusto per rendersi conto delle difficoltà che si incontrano va osservato che i numeri reali sono infiniti mentre la memoria di un calcolatore ha una capacità finita. Una seconda osservazione consiste nel fatto che un numero reale ammette molteplici rappresentazioni. Per esempio il numero 12.47 può essere scritto in diversi modi

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

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

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

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

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

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

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

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

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

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

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

prende il nome di *rappresentazione in base  $\beta$  di  $x$* . Il numero  $p$  viene detto *esponente* (o *caratteristica*), i valori  $d_i$  sono le cifre della rappresentazione, mentre  $0.d_1d_2 \dots d_k \dots$  si dice *mantissa*. Il numero  $x$  viene normalmente rappresentato con la cosiddetta *notazione posizionale*  $x = \text{segno}(x)(.d_1d_2d_3 \dots) \times \beta^p$ , che viene detta *normalizzata*. In alcuni casi è ammessa una rappresentazione in notazione posizionale tale che  $d_1 = 0$ ,

che viene detta *denormalizzata*. Quindi un qualunque numero reale  $x \neq 0$  può essere rappresentato con *infinite cifre* nella mantissa. Inoltre come è noto l'insieme dei numeri reali ha cardinalità infinita. Poichè un elaboratore è dotato di *memoria finita* non è possibile memorizzare:

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

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

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

$$\mathcal{F}(\beta, t, m, M) = \{ x \in \mathbb{R} : x = \pm \beta^p \times 0.d_1 d_2 \dots d_t \text{ con } 0 \leq d_i \leq \beta - 1, \\ d_1 \neq 0, -m \leq p \leq M \} \cup \{0\}.$$

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

Osserviamo che un elaboratore che abbia le seguenti caratteristiche:

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

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

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

$$x = \text{segno}(x) \beta^p \times 0.d_1 d_2 \dots d_t$$

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

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

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

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

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

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

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

dove

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

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

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

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

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

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

**Teorema 2.12.1** *Sia  $x = \pm\beta^p 0.d_1d_2\dots d_t\dots$  tale che la sua rappresentazione macchina non dia luogo a fenomeni di underflow o overflow, allora risulta:*

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

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

**Teorema 2.12.2** *Sia  $x = \pm\beta^p 0.d_1d_2\dots d_t\dots$ ,  $x \neq 0$ , se  $\tilde{x}$  è la sua rappresentazione di macchina cioè  $\tilde{x} \in \mathcal{F}(\beta, t, m, M)$ , allora*

$$\begin{aligned} \left| \frac{\tilde{x} - x}{x} \right| &< u \\ \left| \frac{\tilde{x} - x}{\tilde{x}} \right| &< u \end{aligned}$$

dove

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

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

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

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

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

Per poter realizzare la naturale ed importante *Proprietà di chiusura* di una operazione in un certo insieme risulta importante definire delle *operazioni di macchina* che permettano appunto di realizzare tale proprietà. Un requisito essenziale che si richiede nel costruire un'aritmetica di macchina è il seguente.

Indicata con  $\cdot$  una delle quattro operazioni aritmetiche  $+$ ,  $-$ ,  $\times$ ,  $\div$  e con  $\odot$  la corrispondente operazione di macchina dev'essere:

$$x \odot y = (x \cdot y)(1 + \varepsilon), \quad |\varepsilon| < u \quad (2.17)$$

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

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

e

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

soddisfano la (2.17) e dunque danno luogo ad operazioni di macchina. Le quattro operazioni così definite danno luogo alla *aritmetica di macchina* o *aritmetica finita*.

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

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

dall'elaboratore quando calcola  $x - y$ .

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

Una maggiorazione per l'errore relativo è la seguente

$$\begin{aligned}
 \frac{|(fl(x) \ominus fl(y)) - (x - y)|}{|x - y|} &\leq |\varepsilon| + \frac{|x|}{|x - y|} (|\varepsilon_x| + |\varepsilon||\varepsilon_x|) + \\
 &+ \frac{|y|}{|x - y|} (|\varepsilon_y| + |\varepsilon||\varepsilon_y|).
 \end{aligned} \tag{2.18}$$

Se  $x$  e  $y$  hanno segno opposto risulta

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

e dalla (2.18) segue la maggiorazione

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

dove  $u$  è la precisione di macchina ed  $O(u^2)$  indica la maggiorazione per i cosiddetti termini quadratici dell'errore (cioè i prodotti del tipo  $\varepsilon\varepsilon_x$ , infatti tali quantità sono maggiorabili singolarmente in modulo dalla precisione di macchina ed il loro prodotto è una quantità trascurabile). Se  $x$  e  $y$  hanno lo stesso segno allora l'errore relativo può essere molto grande quanto più  $x$  e  $y$  sono vicini. Questo fenomeno prende il nome di *cancellazione di cifre significative*.

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

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

si risolve il sistema "perturbato"

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

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

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

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