Metodi Computazionali della Fisica

Matrici e sistemi di equazioni lineari

Problema da risolvere

I **sistemi di equazioni lineari** si trovano spesso in Fisica poichè la "**linearizzazione**" è un'assunzione o approssimazione molto comune nella descrizione dei processi fisici.

Supponiamo di studiare un sistema fisico, descrivibile con N equazioni **lineari**, accoppiate, in N i**ncognite**:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N = b_2 \\ \quad . \qquad . \qquad\qquad . \qquad . \\ \quad . \qquad . \qquad\qquad . \qquad . \\ \quad . \qquad . \qquad\qquad . \qquad . \\ a_{N1}x_1 + a_{N2}x_2 + \ldots + a_{NN}x_N = b_N \end{cases}$$

Problema da risolvere

dove $\{a_{ij}\}$ e $\{b_i\}$ sono parametri **noti** e $\{x_i\}$ sono le **incognite**.

E' conveniente scrivere le nostre equazioni lineari nella **forma matriciale**:

$$\begin{pmatrix} a_{11}\ a_{12}\ \ldots a_{1N} \\ a_{21}\ a_{22}\ \ldots a_{2N} \\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot \\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot \\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot \\ a_{N1}\ a_{N2}\ \ldots a_{NN} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_N \end{pmatrix}$$

Problema da risolvere

o, in termini compatti, $AX = B \qquad (1)$

essendo ***A*** una **matrice** (**nota**) (*N*x*N*), ***B*** un **vettore** (**noto**) di lunghezza *N* e ***X*** un **vettore** (**incognito**) di lunghezza *N*.

N.B. qui ci limitiamo a descrivere metodi "**diretti**", appropriati per trattare matrici "**dense**" (in cui la maggior parte degli elementi sono $\neq 0$), di dimensioni relativamente **piccole** ($N \leq 100$); spesso, nelle applicazioni pratiche con matrici molto **grandi**, si sfrutta il fatto che quasi sempre sono "**sparse**" (molti degli elementi sono *0*) e si utilizzano opportune routines di **librerie** matematiche.

Chiaramente la (1) sarebbe **risolta** conoscendo la **matrice inversa** $\boldsymbol{A^{-1}}$, infatti:

$$X = A^{-1}B$$

Librerie matematiche

Alcune importanti librerie matematiche per uso **scientifico** sono le seguenti (vedi, ad esempio, http://www.gnu.org/software/gsl/) :

- **NETLIB** a WWW metalibrary of free math libraries
- **IMSL** International Mathematical and Statistical libraries
- **ESSL** Engineering and Scientific Subroutine Library (IBM)
- **DXML** Advanced Mathematical Library (DEC)
- **NAG** Numerical Algorithms Group (UK Labs)
- **SLATEC** Comprehensive Mathematical and Statistical package
- **LAPACK** Linear Algebra Package
- **CERN** European Center for Nuclear Research
- **BLAS** Basic Linear Algebra Subprograms

Problema da risolvere

D'altra parte sappiamo che : $A^{-1} = \dfrac{adj(A)}{\det(A)}$

dove la matrice "**aggiunta**" $adj(A) \equiv C^T$

C^T essendo la "**trasposta**" di C (scambio di righe con colonne) e con:

$C_{ij} = (-1)^{i+j} (cofA)_{ij}$ dove il "**cofattore**" $(cofA)_{ij}$ è dato

calcolando il **determinante** della matrice *(N-1)*x*(N-1)*, ottenuta dalla A eliminando l'i-esima riga e la j-esima colonna;

ad esempio, se:

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 4 & 2 & 2 \\ 2 & 4 & 1 \end{pmatrix}$$

Problema da risolvere

allora : $(cofA)_{11} = -6 \quad (cofA)_{12} = 0 \quad (cofA)_{13} = 12$

$(cofA)_{21} = -2 \quad (cofA)_{22} = -1 \quad (cofA)_{23} = 0$

$(cofA)_{31} = 2 \quad (cofA)_{32} = -2 \quad (cofA)_{33} = -6$

e $det(A) = 1 \cdot (-6) - 2 \cdot 0 + 1 \cdot 12 = 6$, allora:

$$C = \begin{pmatrix} -6 & 0 & 12 \\ 2 & -1 & 0 \\ 2 & 2 & -6 \end{pmatrix} \Rightarrow C^T = \begin{pmatrix} -6 & 2 & 2 \\ 0 & -1 & 2 \\ 12 & 0 & -6 \end{pmatrix} \Rightarrow$$

$$A^{-1} = \frac{C^T}{\det(A)} = \begin{pmatrix} -1 & 1/3 & 1/3 \\ 0 & -1/6 & 1/3 \\ 2 & 0 & -1 \end{pmatrix}$$

Problema da risolvere

Un tale calcolo richiede di valutare N^2 **determinanti** di matrici *(N-1)*x*(N-1);* se, per la generica matrice *A* *N*x*N*, il determinante è calcolato usando la formula standard:

$$\det A = \sum_P (-)^P A_{1P1} A_{2P2} \ldots\ldots A_{NPN} \qquad (2)$$

dove *P* indica una delle *N!* permutazioni delle *N* colonne, allora sono necessarie *N!* operazioni per calcolare (2), quindi, per *N=20*, servono $2 \cdot 10^{18}$ moltiplicazioni, allora se un computer fa 10^8 moltiplicazioni al secondo sarebbero necessari circa **600 anni** !

Non è il metodo adatto a meno che *N* non sia molto ***piccolo***.

Uno dei più semplici metodi **pratici** per valutare A^{-1} è costituito dal "**Gauss-Jordan elimination method**".

"Gauss-Jordan elimination"

L'**idea fondamentale** è quella di considerare una classe di **operazioni elementari** sulle **righe** della matrice *A*; queste includono:

- **moltiplicare** una particolare riga per una **costante**;
- **scambiare** due righe;
- **aggiungere** un **multiplo** di una riga ad un'altra.

ognuna di queste 3 operazioni può essere realizzata moltiplicando **"da sinistra"** la matrice *A* per una matrice *T*; ad esempio, se $N=3$, allora le matrici:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/2 & 0 & 1 \end{pmatrix}$$

"Gauss-Jordan elimination"

realizzano, rispettivamente, le seguenti operazioni:

- **moltiplicano** la terza riga per 2;
- **scambiano** la prima con la seconda riga;
- **sottraggono** *1/2* della prima riga dalla terza.

La **strategia** del metodo di "**eliminazione Gauss-Jordan**" è quella di trovare una **sequenza di operazioni** T=.....$T_3T_2T_1$, tale che, applicata ad A, la "**riduce**" alla matrice **unitaria**:

$$TA = (.....T_3T_2T_1)A = I$$

allora, evidentemente $T=A^{-1}$ è la matrice inversa **richiesta**.

"Gauss-Jordan elimination"

Consideriamo ancora la matrice *A* dell'esempio precedente e la matrice unitaria *I* (con *N=3*):

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 4 & 2 & 2 \\ 2 & 4 & 1 \end{pmatrix} \quad , \quad I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

e applichiamo la seguente procedura in **3 passi**:

"Gauss-Jordan elimination"

I) cerchiamo di **azzerare** tutti gli elementi, **tranne il primo**, nella **prima colonna** di *A*:

- **sottraiamo** 4 volte la prima riga dalla seconda:

$$TA = \begin{pmatrix} 1 & 2 & 1 \\ 0 & -6 & -2 \\ 2 & 4 & 1 \end{pmatrix} \quad , \quad TI = \begin{pmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- **sottraiamo** 2 volte la prima riga dalla terza:

$$TA = \begin{pmatrix} 1 & 2 & 1 \\ 0 & -6 & -2 \\ 0 & 0 & -1 \end{pmatrix} \quad , \quad TI = \begin{pmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ -2 & 0 & 1 \end{pmatrix}$$

"Gauss-Jordan elimination"

II) cerchiamo di **azzerare** tutti gli elementi, **tranne il secondo**, nella **seconda colonna** di *A*:

- **sommiamo** 1/3 della seconda riga alla prima:

$$TA = \begin{pmatrix} 1 & 0 & 1/3 \\ 0 & -6 & -2 \\ 0 & 0 & -1 \end{pmatrix} \quad , \quad TI = \begin{pmatrix} -1/3 & 1/3 & 0 \\ -4 & 1 & 0 \\ -2 & 0 & 1 \end{pmatrix}$$

- **moltiplichiamo** la seconda riga per (-1/6):

$$TA = \begin{pmatrix} 1 & 0 & 1/3 \\ 0 & 1 & 1/3 \\ 0 & 0 & -1 \end{pmatrix} \quad , \quad TI = \begin{pmatrix} -1/3 & 1/3 & 0 \\ 2/3 & -1/6 & 0 \\ -2 & 0 & 1 \end{pmatrix}$$

"Gauss-Jordan elimination"

III) cerchiamo di **azzerare** tutti gli elementi, **tranne il terzo**, nella **terza colonna** di *A*:

- **sommiamo** 1/3 della terza riga alla prima ed alla seconda:

$$TA = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \quad , \quad TI = \begin{pmatrix} -1 & 1/3 & 1/3 \\ 0 & -1/6 & 1/3 \\ -2 & 0 & 1 \end{pmatrix}$$

- **moltiplichiamo** la terza riga per (-1):

$$TA = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad , \quad TI = \begin{pmatrix} -1 & 1/3 & 1/3 \\ 0 & -1/6 & 1/3 \\ 2 & 0 & -1 \end{pmatrix}$$

matrice unitaria !

"Gauss-Jordan elimination"

allora $TI=T$ è la matrice **inversa** richiesta.

Questo algoritmo può essere facilmente **generalizzato** ad una matrice (NxN) e si può dimostrare che, per N grande, richiede dell'ordine di **N^3 operazioni** (moltiplicazioni ed addizioni), quindi, a meno che N non sia troppo grande, è un algortimo **utilizzabile** in pratica.

Osservazioni sull'uso pratico:

- in qualche punto della procedura può darsi che il termine **diagonale** nella colonna su cui si sta lavorando diventi 0, allora, per evitare che la matrice diventi **singolare** (determinante=0), è necessario lo **scambio** di 2 righe o colonne per far sì che questo elemento **"pivot"** sia $\neq 0$;

"Gauss-Jordan elimination"

- possono anche sorgere problemi associati all'**arrotondamento** numerico se ci sono elementi della matrice che **differiscono molto** in grandezza; allora è spesso utile "**riscalare**" le file o le colonne in modo che tutti gli elementi siano **dello stesso ordine** di grandezza ("**equilibratura**");
- vari casi **speciali** (ad esempio quando *A* è **simmetrica**) possono portare ad una **riduzione** dello sforzo numerico;
- se interessa calcolare **solo il determinante** di *A* è sufficiente effettuare **solo** le trasformazioni delle righe (le quali hanno un effetto **semplice** e **calcolabile** sul determinante*) che riducono *TA* ad una forma "**lower diagonal**" o "**upper diagonal**" (tutti gli elementi sono **nulli sopra** o **sotto** la diagonale, rispettivamente), come all'inizio della fase **II)** descritta precedentemente;

"Gauss-Jordan elimination"

allora il determinante si calcola banalmente facendo il **prodotto** degli elementi diagonali (*det(A)=6* nel nostro caso):

$$TA = \begin{pmatrix} 1 & 0 & 1/3 \\ 0 & -6 & -2 \\ 0 & 0 & -1 \end{pmatrix}$$

*:dalla (2) si può facilmente dimostrare che:

- **scambiare** due righe di una matrice **cambia il segno** del determinante;
- **sommare** un multiplo di una riga ad un'altra lascia il determinante **inalterato**;
- **moltiplicare per una costante** una riga **moltiplica** il determinante per la **stessa** costante.

Programma in C++ per risolvere AX=B

Gauss.cpp

```
#include <iostream>
#include <cmath>
#define N 100
int main() {
using namespace std;
/* soluzione dei sistemi di equazioni con il metodo di eliminazione delle incognite di Gauss */
double A[N][N], b[N], x[N];
double C, S;
int n, i, j, k;
/* inserisce i dati iniziali */
cout << " Inserisci il numero di equazioni (<100): “ << endl;
cin >> d >> n ;
cout << " Inserisci ora i coefficienti del sistema
e i termini noti“ << endl;
for (i = 0; i < n; i++) {
for (j = 0; j < n; j++) {

cout << "A["<<i<<","<< j<<"] = " << endl;
cin >> A[i][j];
}
cout << "b["<<i<<"] = " << endl;
cin >>b[i];
}
```

Programma in C++ per risolvere AX=B

```
/* triangolarizza la matrice */
for (i = 0; i < n; i++) {
/* divide l'i-esima equazione per l'elemento diagonale C */
C = A[i][i];
for (j = i; j < n; j++) {
A[i][j] /= C;
}
b[i] /= C;
/* sottrae l'equazione normalizzata dalle altre */
for (k = i + 1; k < n; k++) {
C = A[k][i];
for (j = i; j < n; j++) {
A[k][j] -= A[i][j] * C;
}
b[k] -= C * b[i];
}
}
/* risolve */
for (k = n - 1; k >= 0; k--) {
S = 0.;
for (i = k + 1; i < n; i++) {
S += A[k][i] * x[i];
}
x[k] = b[k] - S;
}
```

Programma in C++ per risolvere AX=B

```
/* stampa il risultato */
 for (i = 0; i < n; i++) {

cout << "x["<<i<<"] = " << x[i] << endl;

 }
}
```

Applicazioni:

- col programma precedente si può **invertire** la matrice dell'esempio :

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 4 & 2 & 2 \\ 2 & 4 & 1 \end{pmatrix}$$

- si può risolvere $AX=B$ considerando una matrice relativamente **grande**, ad esempio la matrice di **Hilbert** (*100*x*100*), con:

$$a_{ij} = \frac{1}{i+j-1} \quad , \quad b_i = \frac{1}{i}$$

la cui **soluzione** è : $x_i = \delta_{i1}$

Applicazioni:

matrice di **Hilbert**:

$$A = \begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 & . & . & 1/100 \\ 1/2 & 1/3 & 1/4 & 1/5 & . & . & 1/101 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 1/100 & 1/101 & & . & & . & 1/199 \end{pmatrix}$$

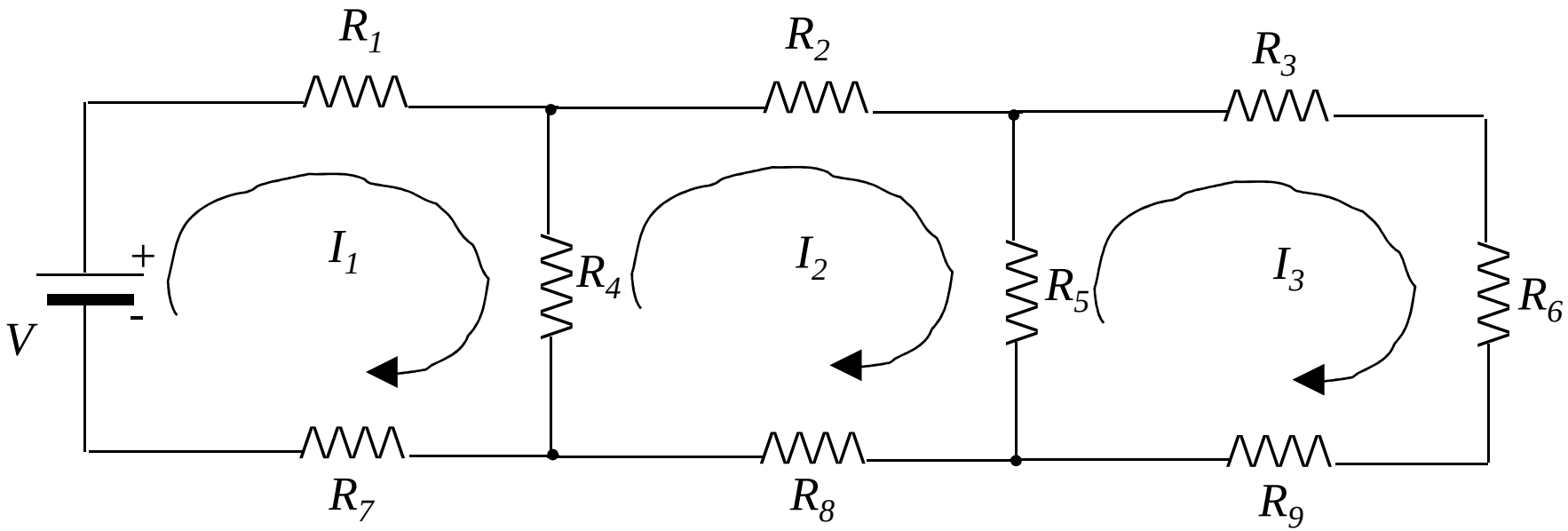
con :

$$B = \begin{pmatrix} 1 \\ 1/2 \\ 1/3 \\ . \\ . \\ 1/100 \end{pmatrix} \quad \Rightarrow \quad X = \begin{pmatrix} 1 \\ 0 \\ 0 \\ . \\ . \\ 0 \end{pmatrix}$$

Esempio fisico: circuiti elettrici e legge di Kirchoff

La (**seconda**) legge di **Kirchoff** (o legge "**delle maglie**") stabilisce che la somma algebrica delle differenze di potenziale rilevate su un circuito **chiuso**, in un giro completo è **nulla**.

Se consideriamo il circuito seguente, con una **batteria** di d.d.p. V e 9 **resistenze**, R_1, R_2,…, R_9 :



Esempio fisico: circuiti elettrici e legge di Kirchoff

allora otteniamo (considerando le **3 maglie** evidenziate in figura e la **prima** legge di Kirchoff per ridurre il numero di correnti incognite) il seguente sistema di equazioni lineari:

$$\begin{cases} V - I_1 R_1 - (I_1 - I_2) R_4 - I_1 R_7 = 0 \\ (I_1 - I_2) R_4 - I_2 R_2 - (I_2 - I_3) R_5 - I_2 R_8 = 0 \\ (I_2 - I_3) R_5 - I_3 R_3 - I_3 R_6 - I_3 R_9 = 0 \end{cases}$$

che si può riscrivere come:

$$\begin{cases} (R_1 + R_4 + R_7) I_1 - R_4 I_2 + 0 \cdot I_3 = V \\ - R_4 I_1 + (R_2 + R_4 + R_5 + R_8) I_2 - R_5 I_3 = 0 \\ 0 \cdot I_1 - R_5 I_2 + (R_3 + R_5 + R_6 + R_9) I_3 = 0 \end{cases}$$

dove I_1, I_2, I_3 sono le incognite.

Esempio fisico: circuiti elettrici e legge di Kirchoff

Allora, trasformando in forma **matriciale**, il vettore $B=(V,0,0)$, il vettore $X=(I_1, I_2, I_3)$ e la matrice A è:

$$A=\begin{pmatrix} (R_1+R_4+R_7) & -R_4 & 0 \\ -R_4 & (R_2+R_4+R_5+R_8) & -R_5 \\ 0 & -R_5 & (R_3+R_5+R_6+R_9) \end{pmatrix}$$

e se supponiamo che $V=36$ *volt*,

$$R_1=R_2=R_3=R_4=R_7=R_8=R_9=10\ \Omega$$

$$R_5=R_6=20\ \Omega$$

$$\Rightarrow A=\begin{pmatrix} 30 & -10 & 0 \\ -10 & 50 & -20 \\ 0 & -20 & 60 \end{pmatrix}$$

matrice **simmetrica** $(a_{ij}=a_{ji})$

Esempio fisico: circuiti elettrici e legge di Kirchoff

Allora, applicando il **programma** precedente alla matrice:

$$A = \begin{pmatrix} 30 & -10 & 0 \\ -10 & 50 & -20 \\ 0 & -20 & 60 \end{pmatrix}$$

si trova facilmente che la **soluzione** (in Ampere) è:

$$\begin{cases} I_1 = 1.3 \\ I_2 = 0.3 \\ I_3 = 0.1 \end{cases}$$

Esercizio 4a: circuiti elettrici e legge di Kirchoff

Usare il **programma precedente** per **calcolare** le correnti $i_1, i_2, i_3, i_4, i_5, i_6$ passanti attraverso le ***6*** resistenze ($R_1=R_2=R_3=R_4=R_5=R_6=1\Omega$) del seguente circuito elettrico (relativamente complicato) , con le d.d.p. $\varepsilon_1=\varepsilon_2=1.5\ V$, $\varepsilon_3=3\ V$; il problema si può risolvere anche "*a mano*" (analiticamente) ma la procedura è piuttosto **noiosa** (… e la possibilità di commettere **errori** elevata !) :