Sistemi lineari e matrice inversa

A cura di Martina Dorigo, Luigi Maninchedda, Antonio Stefani

Realizzato nell'ambito del ***Progetto Archimede***
con la supervisione dei prof Fabbro Valentina, Rizzo Gabriele
I.S.I.S.S. M.Casagrande, Pieve di Soligo, gennaio/aprile 2015

Sommario

Durante questa esperienza abbiamo usato le matrici per risolvere sistemi lineari. Inoltre abbiamo usato il linguaggio di programmazione c++ per scrivere un programma che risolva sistemi linrari con soluzione unica.
Ma ora la vera domanda...come abbiamo fatto?

In primis abbiamo definito la tipologia di sistemi lineari di cui ci vogliamo occupare. Il sistema lineare in forma base è il seguente.

$$\begin{cases} a_{11}x_{11} + a_{12}x_{12} + \ldots + a_{1n}x_{1n} = b_1 \\ a_{,1}x_{21} + a_{22}x_{22} + \ldots + a_{2n}x_{2n} = b_2 \\ \vdots \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \\ a_{n1}x_{n1} + a_{n2}x_{n2} + \ldots + a_{nn}x_{,n} = b_n \end{cases}$$

A questo sistema abbiamo associato una matrice A che è quadrata in quanto il numero di equazioni e di incognite è uguale.

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{pmatrix}$$

Dal momento che ci occupiamo solo di sistemi lineari la cui soluzione esiste ed è unica, supponiamo che la matrice A sia invertibile, cioè abbia determinante non nullo.
Associamo al sistema anche il vettore di termini noti $\mathbf{b}$ e il vettore soluzione $\mathbf{x}$.

$$\mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Il sistema si può quindi riscrivere nella forma $\mathbf{Ax} = \mathbf{b}$, la cui soluzione è data da $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ dove $\mathbf{A}^{-1}$ è la matrice inversa di $\mathbf{A}$.

A questo punto abbiamo trovato un algoritmo in $c++$ che dà come output la soluzione $\mathbf{x}$ del sistema lineare della tipologia considerata dando in input i coefficienti del sistema (cioè la matrice $\mathbf{A}$) e i termini noti del sistema (cioè il vettore $\mathbf{b}$).
Descriviamo brevemente cosa fa questo algoritmo prima di riportarne il codice.
Innanzitutto inseriamo i coefficienti delle incognite a_{ij}. E i termini noti delle equazioni? Questi andranno a formare un vettore monodimensionale dinamico (un insieme di valori allocati in memoria dinamicamente in base alla grandezza principale della matrice) che ci servirà in seguito.

Per prima cosa l'algoritmo deve costruire una seconda matrice chiamata matrice identità, cioè con i termini nella diagonale tutti pari a 1 e tutti gli altri termini pari a 0.
Ciò che deve fare l'algoritmo adesso è la creazione di una matrice ausiliaria di dimensioni $2n \times 2n$.
Dopo aver riempito questa matrice con i valori della matrice $\mathbf{A}$ nel blocco in alto a destra, con la matrice identità nel blocco in basso a sinistra e con tutti zeri altrove, si trasforma la matrice ausiliara in una matrice triangolare utilizzando il metodo di eliminazione di Gauss. Ma perché questa trasformazione? Innanzitutto perché una matrice triangolare è molto più semplice da gestire e in secondo luogo perché così si può ottenere una matrice a blocchi in cui un blocco è rappresentato proprio dalla matrice inversa $\mathbf{A}^{-1}$.
Ora per trovare i valori delle incognite e risolvere quindi il sistema, bisogna moltiplicare la matrice inversa e la matrice dei termini noti così da ottenere il vettore soluzione finale.

Ecco riportato di seguito il programma così che voi lettori potreste, volendo, provarlo:

Programma	**Spiegazione**
`int main() {`	
`int agg;`	
`double **A`	Allocazione del vettore bidimensionale dinamico
`int n;`	Grandezza del vettore bidimensionale dinamico
`cout « ″Creazione matrice n x n - Inserire il valore di n:  ″;`	
`cin » n;`	
`cout « endl;`	
`A = new double*[n];`	Creazione del vettore, di ampiezza n*n
`for (int i = 0; i<n; i++) A[i] = new double[n];`	Assegnamento delle locazioni di memoria
`double **I;`	Allocazione del secondo vettore bidimensionale
`I = new double [n];`	Reazione del vettore di ampiezza n*n
`for (int i = 0; i<n; i++) I[i] = new double[n];`	Assegnamento delle locazioni di memoria
`double *Y;`	Allocazione del vettore dinamico dei termini noti
`Y = new double [n];`	Creazione del vettore di ampiezza n
`for (int i = 0; i<n; i++)`	
`for (int j = 0; i<n; j++)`	
`if (i == j) I[i][j] = 1;`	Assegnazione dei valori 1 alla diagonale della matrice identità
`else I[i][j] = 0;`	Assegnazione dei valori 0 al resto degli elementi della matrice
`cout « ********** INSERIMENTO MATRICE **********;`	Inserimento degli elementi via tastiera
`for (int i = 0; i<n; i++)`	
`for (int j = 0; j<n; j++)`	
`cout « Elem  « i « ,  « j « :  ;`	
`cin » A[i][j];`	inserimento valori della matrice
`cout « ********************************************;`	
`cout « ********** INSERIMENTO TERMINI NOTI **********;`	Matrice termini noti (monodimensionale)
`for (int i = 0; i < n; i++){`	
`cout « Elem  « i « :  ;`	
`cin » Y[i];`	Inserimento valori della matrice dei termini noti
`}`	
`cout « ********************************************;`	

```
cout « ***** MATRICE A ******;                                   Stampa della matrice A
for (int i = 0; i<n; i++)
for (int j = 0; j<n; j++)
cout « A[i][j];
cout « **********************;

cout « **** TERMINI NOTI ****;                                   Stampa della matrice dei temini noti
for (int j = 0; j<n; j++)
cout « Y[j];
cout « **********************;

// *** Costruzione della matrice [A|I] ***//
double **B;                                                     Creazione di un vettore ausialiario al calcolo
B = new double*[n];                                             Allocazione della memoria
for (int i = 0; i<n; i++) B[i] = new double[2 * n];             Creazione della matrice di grandezza 2n*2n

for (int i = 0; i<n; i++)
for (int j = 0; j<n; j++)
B[i][j] = A[i][j];                                              Popolazione matrice ausialiaria, in alto a dx

int k = 0;
for (int i = 0; i<n; i++) {
for (int j = n; j<2 * n; j++, k++)
B[i][j] = I[i][k];                                              Popolazione matrice ausialiaria, in basso a sx
k = 0;
}                                                               Creazione matrice triangolare

// *** Eliminazione sotto la diagonale principale *** //
double *tmp; tmp = new double[2 * n];
for (int j = 0; j<n - 1; j++)
for (int i = j + 1; i<n; i++)
if (B[i][j] != 0) {
double mol = B[i][j] / B[j][j];
for (int k = 0; k<2 * n; k++) tmp[k] = mol*B[j][k];
for (int k = 0; k<2 * n; k++) B[i][k] -= tmp[k];
}

// *** Eliminazione sopra la diagonale principale *** //
for (int j = n - 1; j>0; j-)
for (int i = j - 1; i >= 0; i-)
if (B[i][j] != 0) 4{
double mol = B[i][j] / B[j][j];
for (int k = 0; k<2 * n; k++) tmp[k] = mol*B[j][k];
for (int k = 0; k<2 * n; k++) B[i][k] -= tmp[k];
}

// *** Ultimo step per ottenere la matrice a blocchi [I|A] inversa *** //
for (int i = 0; i<n; i++)
if (B[i][i] != 1) {
double mol = B[i][i];
for (int k = 0; k<2 * n; k++)
B[i][k] = B[i][k] / mol;
}
```

```
// *** Copia dell'inversa ottenuta *** //
double** Inv;
Inv = new double*[n];
for (int i = 0; i<n; i++) Inv[i] = new double[n];
k = 0;
for (int i = 0; i<n; i++) {
for (int j = n; j<2 * n; j++, k++)
Inv[i][k] = B[i][j];
k = 0;
}

double *soluz;
soluz = new double[n];
double add = 0;
for (int i = 0; i < n; i++){
for (int j = 0; j < n; j++){
add = add + (Inv[i][j]*Y[j]);
soluz[i] = soluz[n] + add;
}
add = 0;
}

cout « ** Stampa soluzione **;
for (int i = 0; i < n; i++)
cout « soluz[i] « ;



cout « **************************;
// *** Deallocazione della memoria *** //
for (int i = 0; i<n; i++)  delete I[i]; delete A[i];
delete I;
delete A;
delete Y;
delete soluz;
for (int i = 0; i<n; i++) delete B[i];
delete B;

system("PAUSE");
}
```

Esempio Facciamo ora un esempio pratico.

Abbiamo un sistema a tre equazioni e tre incognite

$$\begin{cases} x + y - z = 2 \\ x - y + 3z = 0 \\ 2x + y - z = 1 \end{cases}$$

Inseriamo la matrice $\mathbf{A}$ inserendo i coefficenti delle incognite

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & -1 \\ 1 & -1 & 3 \\ 2 & 1 & -1 \end{pmatrix}$$

Inseriamo anche il vettore mondimensionale dinamico dei termini noti $\mathbf{b}$

$$\mathbf{b} = \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix}$$

L'algoritmo calcola quindi la matrice inversa $\mathbf{A}^{-1}$

$$\mathbf{A}^{-1} = \begin{pmatrix} -1 & 0 & 1 \\ \frac{7}{2} & \frac{1}{2} & -2 \\ \frac{3}{2} & \frac{1}{2} & -1 \end{pmatrix}$$

e il vettore soluzione $\mathbf{x}$ (le cui componenti corrispondono rispettivamente alle incognite x, y e z del sistema).

$$\mathbf{x} = \begin{pmatrix} -1 \\ 5 \\ 2 \end{pmatrix}$$

Sitografia L'algoritmo è stato scritto prendendo come spunto da uno trovato sul seguente sito web:

`http://www.iprogrammatori.it/forum-programmazione/cplusplus/matrice-inversa-t6991.html`