Risoluzione di sistemi lineari col metodo di Gauss

4c – maggio 2012

Dato un sistema lineare di m equazioni in m incognite

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n &= b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n &= b_2 \\ & \vdots \\ a_{m,1}x_1 + a_{m,2}x_2 + \cdots + a_{m,n}x_n &= b_m \end{cases}$$

e definita la matrice (m) x (m+1) dei coefficienti e dei termini noti

$$\begin{pmatrix} a_{1,1} & \cdots & a_{1,n} & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ a_{m,1} & \cdots & a_{m,n} & b_m \end{pmatrix}$$

il metodo di Gauss prevede di procedere nel seguente modo:

1. se la prima riga ha il primo elemento nullo, scambiala con una riga che ha il primo elemento non nullo. Se tutte le righe hanno il primo elemento nullo, vai al punto 3.
2. per ogni riga i-esima con primo elemento non nullo, eccetto la prima (i > 1), moltiplica la prima riga per un coefficiente scelto in maniera tale che la somma tra la prima riga e la riga i-esima abbia il primo elemento nullo (quindi coefficiente = $- a_{i1} / a_{11}$). Sostituisci la riga i-esima con la somma appena ricavata.
3. adesso sulla prima colonna tutte le cifre, eccetto forse la prima, sono nulle. A questo punto ritorna al punto 1 considerando la sottomatrice che ottieni cancellando la prima riga e la prima colonna.

In questo modo siamo in grado di ottenere una matrice "triangolare", ovvero con un triangolo di zero in basso a destra.

$$\begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,m-1} & c_{1,m} & d_1 \\ 0 & c_{2,2} & \cdots & c_{2,m-1} & c_{2,m} & d_2 \\ 0 & 0 & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & c_{m-1,m-1} & c_{m-1,m} & d_{m-1} \\ 0 & 0 & \cdots & 0 & c_{m,m} & d_m \end{pmatrix}$$

4. Consideriamo quindi l'ultima riga. Da qui possiamo determinare il valore di $x_m = d_{m,m}/c_{m,m}$. Poniamo quindi e $d_m = d_m/c_m$ e $c_{m,m}=1$. Tale valore di x_m possiamo sostituirlo nelle righe precedenti, fino ad arrivare alla prima. Per esempio per la riga (m-1) porremo $d_{m-1} = d_{m-1} - c_{m-1,m} * x_m$ e $c_{m-1,m}=0$.
5. Ripetiamo l'operazione 4. per la riga (m-1), ovvero determiniamo x_{m-1} e sostituiamo tale valore nelle righe superiori. Così via fino a che si determina il valore di $x_{1,1}$.

Si faccia attenzione a quando un sistema risulta impossibile o indeterminato. Entrami i casi si verificano quando, al punto 4., l'elemento $c_{i,i}$ è uguale a zero. Se anche d_i è uguale a 0, avremo che il sistema è indeterminato; in caso contrario avremo un sistema impossibile.

ESEMPIO

Consideriamo la seguente matrice

$$\begin{pmatrix} 4 & 5 & 5 & 9 \\ 0 & 2 & 2 & 6 \\ 4 & 5 & 9 & 1 \end{pmatrix}$$

Partendo dalla prima riga il punto 1 è soddisfatto. Quindi eseguiamo il punto 2 sulla seconda e sulla terza riga ottenendo:

$$\begin{pmatrix} 4 & 5 & 5 & 9 \\ 0 & 2 & 2 & 6 \\ 0 & 2 & 4 & -8 \end{pmatrix}$$

Ora andiamo a considerare la sottomatrice

$$\begin{pmatrix} 2 & 2 & 6 \\ 2 & 4 & -8 \end{pmatrix}$$

come descritto al punto 3, e ripetiamo le operazioni descritte al punto 1 e 2 su tale sottomatrice. Il risultato sarà dato dalla seguente matrice:

$$\begin{pmatrix} 4 & 5 & 5 & 9 \\ 0 & 2 & 2 & 6 \\ 0 & 0 & 2 & -14 \end{pmatrix}$$

In questo modo abbiamo ottenuto una matrice "triangolare".

Proseguiamo ora come descritto ai punti 4. e 5. Partendo ore dall'ultima riga m=3, vediamo che possiamo ricavare x_3 = -14/2=-7. Quindi possiamo sostituire questo valore a tutte le righe precedenti, ovvero la seconda riga diverrà

$$(0 \quad 2 \quad 0 \quad 6\text{-}2*x_3)$$

Discorso analogo per le precedenti righe. La matrice diverrà quindi

$$\begin{pmatrix} 4 & 5 & 0 & 44 \\ 0 & 2 & 0 & 20 \\ 0 & 0 & 1 & -7 \end{pmatrix}$$

Partendo ora dalla penultima riga ripetiamo le ultime operazioni (ai punti 4. e 5.) per la riga (m-1)=2, ricavando che x_2 = 20/2=10. Dopo la sostituzione di tale valore nelle righe superiori, otterremo la matrice:

$$\begin{pmatrix} 4 & 0 & 0 & -6 \\ 0 & 1 & 0 & 10 \\ 0 & 0 & 1 & -7 \end{pmatrix}$$

Ripetiamo la stessa procedura per la riga (m-2)=1 da cui ricaviamo

$$\begin{pmatrix} 1 & 0 & 0 & -3.5 \\ 0 & 1 & 0 & 10 \\ 0 & 0 & 1 & -7 \end{pmatrix}$$

da cui si "legge" banalmente che x_1=-3.5, x_2=10 e x_3=-7.

SUGGERIMENTI PER SCRIVERE IL PROGRAMMA

- Dichiarare una costante `N` (che definisce la dimensione della matrice) ed una matrice `double M[N][N+1]`.
- Definire le funzioni
  - `void leggi(double M[N][N+1])` che legge la matrice da tastiera.
  - `void stampa(double M[N][N+1])` che stampa a video la matrice.
  - `void randm(double M[N][N+1])` che inizializza la matrice random.
  - `void scambia(double M[N][N+1],int q, int p)` che scambia le righe q e p della matrice.
  - `void somma(double M[N][N+1], int q, int p, double fact)` che calcola (riga q + fact * riga p) e mette il risultato nella riga q.
  - `int cercariga(double M[N][N+1], int p)` che cerca la riga >=p con coefficiente p-esimo!=0 (attenzione! Tale riga non sempre esiste. In tale caso bisognerà saltare direttamente al punto 3.).
  - `void gauss(double M[N][N+1])` che, servendosi delle funzioni cercariga, scambia e somma, riduce M ad una matrice triangolare come descritto ai punti 1, 2 e 3.
  - `void solve(double M[N][N+1])` che, data in input una matrice triangolare, esegue le operazioni descritte ai punti 4. e 5. Attenzione a considerare i casi in cui il sistema è indeterminato o impossibile!
- In questo modo il programma principale sarà costituito semplicemente

```
randm(M);   // inizializzazione random
stampa(M);  // stampa matrice di partenza
gauss(M);   // calcola la matrice triangolare
stampa(M);  // stampa la matrice triangolare
solve(M);   // risolve la matrice triangolare
stampa(M);  // stampa la soluzione
```

- NB: per motivi di arrotondamento i controlli sui valori nulli non vanno fatti semplicemente con una condizione del tipo if(M[i][j]==0.) . Per problemi di arrotondamento potrebbe verificarsi che un dato valore M[i][j] sia "quasi" 0 e non esattamente nullo come dovrebbe essere; si riuscirebbe quindi ad ottenere soluzione determinata anche per un sistema impossibile o indeterminato. Se non viene implementato un sistema più corretto ed articolato per prevenire questo genere di errori, anziché usare la condizione if(M[i][j]==0.) avrà più senso usare una cosa del tipo if(fabs(M[i][j])<1E-12.).