Serie 2a
Sistemi Lineari – Metodi Diretti

1 Il metodo di fattorizzazione LU

Data una matrice quadrata A di dimensione $n \times n$, sotto opportune condizioni è possibile fattorizzare la matrice A con il prodotto di due matrici L ed U, dove L è una matrice triangolare inferiore ed U è una matrice triangolare superiore. Tale fattorizzazione permette di risolvere un sistema lineare

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

decomponendolo in due sistemi lineari più semplici

$$L\mathbf{y} = \mathbf{b}, \quad U\mathbf{x} = \mathbf{y}. \tag{1}$$

L'algoritmo della fattorizzazione LU (o fattorizzazione di *Gauss*) è il seguente.
Posto $A^{(1)} = A$ (in componenti $a_{ij}^{(1)} = a_{ij}$, per $i, j = 1, \ldots, n$) si calcoli:

for $k = 1, \ldots, n-1$ **do**
  for $i = k+1, \ldots, n$ **do**
    $l_{ik} = \dfrac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$
    for $j = k+1, \ldots, n$ **do**
      $a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}$
    end for
  end for
end for

Per ogni $k = 1, \ldots, n-1$ la matrice $A^{(k+1)} = (a_{ij}^{(k+1)})$ ha $n-k$ righe e colonne. Al termine di questo processo, gli elementi della matrice triangolare U sono ottenuti :

$$u_{ij} = a_{ij} \quad \text{per } \ i = 1, \ldots, n \ \text{ e } \ j = i, \ldots, n,$$

mentre gli elementi di L sono i coefficienti l_{ik} generati dall'algoritmo. In particolare, gli elementi diagonali di L non sono calcolati, perché per l'unicità della fattorizzazione sono posti uguale ad 1.

I sistemi (1) risultano più agevoli da risolvere perché essendo triangolari, inferiore e superiore, possono essere risolti efficientemente con gli schemi delle sostituzioni in avanti e all'indietro. In particolare il sistema $L\mathbf{y} = \mathbf{b}$ può essere risolto con il seguente algoritmo

$$y_1 = \frac{b_1}{l_{11}}$$
$$y_i = \frac{1}{l_{ii}}\Big(b_i - \sum_{j=1}^{i-1} l_{ij} y_j\Big), \quad i = 2, \ldots, n \quad l_{ii} \neq 0,$$

e in modo analogo $U\mathbf{x} = \mathbf{y}$ con

$$x_n = \frac{y_n}{u_{nn}}$$
$$x_i = \frac{1}{u_{ii}}\Big(y_i - \sum_{j=i+1}^{n} u_{ij}x_j\Big), \quad i = n-1, \ldots, 1 \quad u_{ii} \neq 0.$$

Esercizio 1.1

Si considerino le matrici

$$A = \begin{bmatrix} 50 & 1 & 3 \\ 1 & 6 & 0 \\ 3 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 50 & 1 & 10 \\ 3 & 20 & 1 \\ 10 & 4 & 70 \end{bmatrix}, \quad C = \begin{bmatrix} 7 & 8 & 9 \\ 5 & 4 & 3 \\ 1 & 2 & 6 \end{bmatrix}. \tag{2}$$

1. Si verifichi, utilizzando Matlab® , se le matrici A, B e C soddisfano le condizioni sufficienti ed eventualmente necessarie e sufficienti per l'esistenza della fattorizzazione LU.

2. Scrivere una funzione `[L,U]=lugauss(M)` che, data in ingresso una matrice quadrata $M \in \mathbb{R}^{n\times n}$, determini i fattori L e U tali che $LU = M$.

3. Utilizzare la funzione `lugauss` per fattorizzare le matrici A, B, e C.

4. Scrivere una funzione `fwsub.m` che, dati in ingresso una matrice triangolare inferiore $L \in \mathbb{R}^{n\times n}$ e un vettore $\mathbf{f} \in \mathbb{R}^n$, restituisca in uscita il vettore $\mathbf{x}$ soluzione del sistema $L\mathbf{x} = \mathbf{f}$, calcolata mediante l'algoritmo della sostituzione in avanti (*forward substitution*). L'intestazione della funzione sarà ad esempio: `[x] = fwsub(L,f)`. Analogamente, scrivere la funzione `bksub.m` che implementi l'algoritmo della sostituzione all'indietro (*backward substitution*) per matrici triangolari superiori (U).

5. Supponiamo ora di voler risolvere il sistema $A\mathbf{x} = \mathbf{b}$ con A definita in (2). Si utilizzi come termine noto $\mathbf{b}$ un vettore tale che la soluzione esatta del sistema sia $\mathbf{x}_{ex} = [1, 1, 1]^T$. Si calcoli la soluzione del sistema $A\mathbf{x} = \mathbf{b}$, utilizzando le funzioni `bksub.m` e `fwsub.m`.

6. Si calcoli la norma 2 dell'errore relativo $\|\mathbf{x}_{ex} - \mathbf{x}\|_2/\|\mathbf{x}_{ex}\|_2$ e la norma 2 del residuo normalizzato $\|\mathbf{b} - A\mathbf{x}\|_2/\|\mathbf{b}\|_2$ conoscendo la soluzione esatta.

Esercizio 1.2

Si definisce matrice inversa di una matrice quadrata $A \in \mathbb{R}^{n\times n}$, l'unica matrice $X = A^{-1}\mathbb{R}^{n\times n}$ tale che $AX = XA = I$. È possibile determinare i vettori colonna di A^{-1} risolvendo n sistemi lineari del tipo $A\mathbf{v}_i = \mathbf{e}_i$ dove $\mathbf{e}_i$ sono i vettori della base canonica di $\mathbb{R}^n$, ciascuno dei quali ha tutti gli elementi nulli tranne l'elemento i-esimo che è pari a uno. Si ottengono così i vettori $\mathbf{v}_i$ che rappresentano i vettori colonna della matrice A^{-1}, cioè $A^{-1} = [\mathbf{v}_1|\mathbf{v}_2|...|\mathbf{v}_n]$. Risolvere numericamente, utilizzando le funzioni `fwsub.m` e `bksub.m`, i $2n$ sistemi triangolari necessari per ottenere la matrice A^{-1}. Si scelga la matrice A definita nel primo esercizio. Si calcoli l'inversa di A e si confronti il risultato ottenuto con quello fornito dalla funzione Matlab® `inv`.

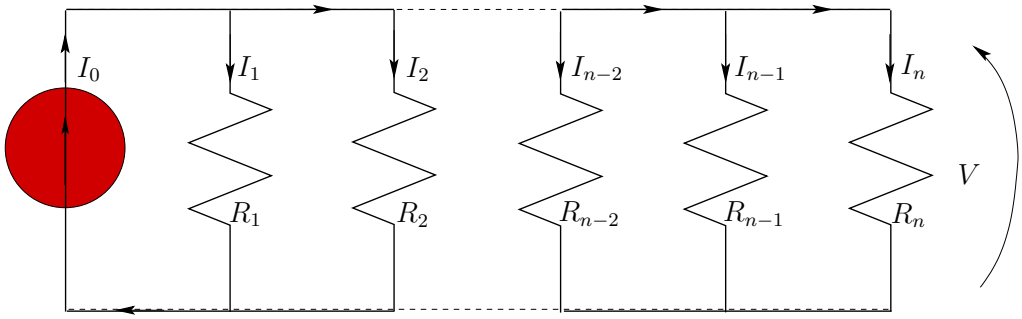


Figura 1: Schema elettrico del problema relativo all' Esercizio 1.3.

Esercizio 1.3

Un generatore di corrente genera una corrente I_0 che fluisce nelle n resistenze elettriche $R_1, R_2, \dots R_n$ in parallelo, come schematizzato in Fig. 1. Nota la corrente I_0, vogliamo calcolare le correnti I_k, con $k = 1, \dots, n$, che fluiscono negli n rami. La tensione V ai capi di ogni resistenza R_k segue la legge di *Ohm*:

$$V = R_k I_k.$$

Per ottenere le n equazioni che ci servono per risolvere il sistema, utilizziamo le leggi di *Kirchhoff* alle maglie e ai nodi dello schema elettrico.

Per la legge di Kirchhoff ai nodi otteniamo l'equazione:

$$I_0 = I_1 + I_2 + \cdots + I_n;$$

per la legge di Kirchhoff alle maglie, otteniamo le rimanenti $n-1$ equazioni

$$R_{k-1} I_{k-1} = R_k I_k \quad con \quad k = 2, .., n.$$

Il calcolo delle correnti I_k conduce a un sistema lineare $A\mathbf{i} = \mathbf{b}$, dove $\mathbf{i} = [I_1 \dots, I_n]^T$ è il vettore delle correnti incognite,

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ R_1 & -R_2 & 0 & 0 & \cdots & 0 \\ 0 & R_2 & -R_3 & 0 & \cdots & 0 \\ \vdots & 0 & \ddots & \ddots & & \vdots \\ \vdots & \vdots & & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & R_{n-1} & -R_n \end{bmatrix},$$

e

$$\mathbf{b} = [I_0, 0, 0, \cdots, 0]^T \in \mathbb{R}^n.$$

1. Si ponga $n = 20, R_k = 1$ con $k = 1, \dots, n$ e $I_0 = 2$ e si assegnino la matrice A e il vettore dei termini noti $\mathbf{b}$.

2. Si calcoli la fattorizzazione LU con pivoting della matrice A, mediante la funzione Matlab® `lu`. Verificare che la tecnica del pivoting non è stata usata in questo caso. (SUGGERIMENTO: verificare la sintassi del comando `lu` tramite l'uso del comando `help`.)

3. Risolvere numericamente, utilizzando le funzioni `fwsub.m` e `bksub.m` il sistema $A\mathbf{i} = \mathbf{b}$.

4. Si calcoli la norma 2 dell'errore relativo $\|\mathbf{i}_{ex} - \mathbf{i}\|_2/\|\mathbf{i}_{ex}\|_2$ e la norma 2 del residuo normalizzato $\|\mathbf{b} - A\mathbf{i}\|_2/\|\mathbf{b}\|_2$ sapendo che la soluzione esatta è il vettore $i_{ex,\,k} = \dfrac{I_0}{n}$, $k = 1, \ldots, n$. Si commenti il risultato ottenuto partendo dal numero di condizionamento della matrice A. (Si utilizzino i comandi `norm` e `cond`).

5. Si ponga $R_1 = 10^3$ e si calcoli la nuova distribuzione delle correnti. Si calcoli il numero di condizionamento di A e si commenti il risultato ottenuto.

2 Il numero di condizionamento di una matrice

Il numero di condizionamento di una matrice A, cond(A), è un numero che indica quanto sia "difficile" risolvere il sistema lineare $A\boldsymbol{x} = \boldsymbol{b}$; "difficile" indica sia la sensibilità alla propagazione di errori numerici (per cui la soluzione numerica di un sistema lineare la cui matrice sia mal condizionata sarà, in generale, poco precisa), che l'impiego di risorse computazionali.

Un tipico esempio di matrice mal condizionata è la matrice di Hilbert (vedi laboratorio 3). Per calcolare il condizionamento di una matrice in norma 2 si usa il comando `cond`, che per una matrice di Hilbert di dimensione 1000×1000 restituisce:

```
>> cond(hilb(1000))
ans =
   5.5647e+20
```

mentre, ad esempio, per una matrice di numeri casuali:

```
>> cond(rand(1000))
ans =
   8.0236e+04
```

Nota: un esempio di matrice mal condizionata forse ancor più di quella di Hilbert si ottiene in Matlab® tramite il comando:
`» gallery(3)` [1]:

```
ans =
  -149   -50  -154
   537   180   546
   -27    -9   -25
>> cond(gallery(3))
ans =
   2.7585e+05
```

[1] per maggiori informazioni sul comando `gallery` usare `help gallery`

mentre:

```
>> cond(hilb(3))
ans =
  524.0568
```

Esercizio 2.1

Siano `A=hilb(1000)` e `B=rand(1000)`.

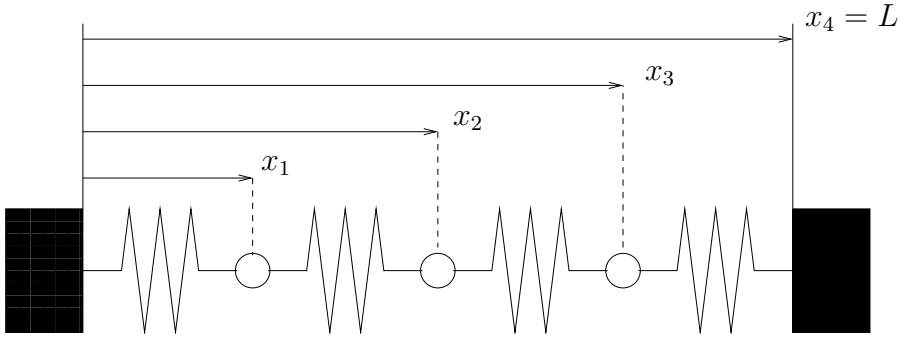
1. Costruire il vettore $\boldsymbol{b}$ in modo che il sistema $A\boldsymbol{x} = \boldsymbol{b}$ sia risolto da `x=ones(1000,1)`, e il vettore $\boldsymbol{c}$ in modo che il sistema $B\boldsymbol{y} = \boldsymbol{c}$ sia risolto da `y=ones(1000,1)`;
2. Calcolare i vettori `x` e `y` come soluzione dei sistemi lineari $A\boldsymbol{x} = \boldsymbol{b}$, $B\boldsymbol{y} = \boldsymbol{c}$ utilizzando il comando \ di Matlab® e Octave. Per ulteriori informazioni `help mldivide`;
3. Calcolare gli errori relativi rapportandoli al numero di condizionamento delle corrispondenti matrici. Quanto influenza il numero di condizionamento di A ?

3 Algoritmo di Thomas

Spesso in applicazioni concrete ci si trova a dover risolvere sistemi lineari la cui matrice è tridiagonale, cioè del tipo:

$$A = \begin{bmatrix} a_1 & c_1 & & & & \\ e_1 & a_2 & c_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & e_{n-2} & a_{n-1} & c_{n-1} & \\ & & & e_{n-1} & a_n & \end{bmatrix}.$$

Ad esempio, consideriamo un sistema fisico formato da N molle ideali, disposte in fila su un piano orizzontale e agganciate l'una all'altra tramite degli anelli; la prima e l'ultima molla sono ancorate ad una parete. La distanza fra le due pareti è L, e tutte le molle hanno la stessa costante di rigidezza K. Vogliamo calcolare qual è la configurazione di equilibrio delle molle.



Per descrivere il sistema, utilizziamo $N+1$ variabili x_i tali che:

- $x_0 = 0$;
- x_i è la posizione dell' i-esimo anello;
- $x_N = L$.

Le equazioni che dobbiamo scrivere sono il bilancio delle forze per ciascun anello; indicando con $\boldsymbol{F}_i$ la forza esercitata dall'i-esima molla, il bilancio delle forze si scrive come $\boldsymbol{F}_1 + \boldsymbol{F}_2 = 0$ per il primo anello, $\boldsymbol{F}_2 + \boldsymbol{F}_3 = 0$ per il secondo anello, ed in generale:

$$\boldsymbol{F}_i + \boldsymbol{F}_{i+1} = 0$$

per l'i-esimo anello, con $i = 1, .., N-1$. La forza esercitata da una molla è proporzionale al suo allungamento Δl rispetto alla sua lunghezza in una configurazione a riposo, $|\boldsymbol{F}| = K\Delta l$. Utilizzando le variabili x_i, e considerando nulla la lunghezza a riposo, l'equazione di equilibrio per l'i-esimo anello è:

$$-K(x_i - x_{i-1}) + K(x_{i+1} - x_i) = 0 \Rightarrow Kx_{i-1} - 2Kx_i + Kx_{i+1} = 0.$$

Ricordando che $x_0 = 0$, $x_N = L$, possiamo scrivere il sistema complessivo come:

$$K \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ x_{N-2} \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ -KL \end{bmatrix}$$

Per risolvere sistemi tridiagonali come quello appena presentato, un algoritmo molto efficiente è l'algoritmo di Thomas. Questo algoritmo:

1. sfrutta la struttura tridiagonale della matrice per calcolare in modo rapido la fattorizzazione LU della matrice A. Le matrici L, U che si ottengono risultano bidiagonali;
2. utilizza tali informazioni sulla struttura di L, U per risolvere efficientemente i due sistemi $L\mathbf{y} = \mathbf{b}$ e $U\mathbf{x} = \mathbf{y}$.

In particolare, se $A \in \mathbb{R}^{n \times n}$ è della forma:

$$A = \begin{bmatrix} a_1 & c_1 & & & \\ e_1 & a_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & e_{n-2} & a_{n-1} & c_{n-1} \\ & & & e_{n-1} & a_n \end{bmatrix},$$

allora le matrici L ed U sono date da:

$$L = \begin{bmatrix} 1 & & & & \\ \delta_1 & 1 & & & \\ & \ddots & \ddots & & \\ & & \delta_{n-2} & 1 & \\ & & & \delta_{n-1} & 1 \end{bmatrix}, \qquad U = \begin{bmatrix} \alpha_1 & c_1 & & & \\ & \alpha_2 & c_2 & & \\ & & \ddots & \ddots & \\ & & & \alpha_{n-1} & c_{n-1} \\ & & & & \alpha_n \end{bmatrix},$$

con

$$\alpha_1 = a_1, \qquad \delta_{i-1} = \frac{e_{i-1}}{\alpha_{i-1}}, \qquad \alpha_i = a_i - \delta_{i-1}c_{i-1}, \qquad i = 2, \ldots, n.$$

Quindi possiamo risolvere in sequenza i due sistemi bidiagonali tramite le relazioni:

$$(L\mathbf{y} = \mathbf{b}) \qquad y_1 = b_1, \qquad y_i = b_i - \delta_{i-1}y_{i-1}, \qquad i = 2, \ldots, n$$

$$(U\mathbf{x} = \mathbf{y}) \qquad x_n = \frac{y_n}{\alpha_n}, \qquad x_i = \frac{y_i - c_i x_{i+1}}{\alpha_i}, \qquad i = n-1, \ldots, 1.$$

Esercizio 3.1

Scrivere una `function` che implementi il metodo di Thomas per risolvere un generico sistema lineare tridiagonale ed utilizzarlo per risolvere il problema delle molle. L'intestazione della `function` dovrà essere:

```
function [L,U,x] = thomas(A,b)
```

Si assumano come parametri $N = 20$ (e quindi le dimensioni della matrice A sono $n \times n$, $n = 19$), $L = 20$ m, $K = 100$ N/m. Come va modificato il sistema lineare se a ciascun anello è applicata una forza esterna $\boldsymbol{F}_{i,ext}$?

4 Confronto delle prestazioni di diversi algoritmi

È interessante verificare i tempi di calcolo richiesti da diversi algoritmi per calcolare la soluzione dello stesso sistema lineare. In questo caso confronteremo sul sistema precedente tre algoritmi: LU, Thomas, Cholesky.

L'algoritmo di Cholesky si applica a matrici simmetriche e definite positive; la fattorizzazione che si ottiene è del tipo $A = H^T H$. I vantaggi di questa fattorizzazione sono almeno due: il fatto che in questo caso $U \equiv L^T$ rende l'algoritmo più veloce (la seconda matrice è "gratis"); inoltre l'algoritmo è stabile rispetto alla propagazione degli errori di arrotondamento. La fattorizzazione di Cholesky si ottiene in Matlab® e Octave tramite il comando `chol(A)`. Nel nostro caso Cholesky non è immediatamente applicabile, perchè si può dimostrare che A è definita negativa (e quindi ha tutti gli autovalori minori di 0). È pero sufficiente riscrivere il sistema $A\boldsymbol{x} = \boldsymbol{b}$ come $-A\boldsymbol{x} = -\boldsymbol{b}$, cioè $\tilde{A}\boldsymbol{x} = \tilde{\boldsymbol{b}}$, con $\tilde{A} = -A$ e $\tilde{\boldsymbol{b}} = -\boldsymbol{b}$, e applicare Cholesky a questa formulazione.

Esercizio 4.1

Confrontare i tempi di calcolo dei metodi \, LU, Thomas, Cholesky sul sistema lineare associato al sistema delle molle, facendo variare il valore del parametro n che definisce la dimensione della matrice A da $n = 200$ a $n = 2000$, con passo 200. Per quale valore di n le differenze sono apprezzabili?

5 Fill-in

Se la matrice del sistema lineare è sparsa (e non strutturata), $A \in \mathbb{R}^{n\times n}$ ha un numero di elementi non nulli dell'ordine di n. In tal caso durante il processo di fattorizzazione si possono generare un gran numero di elementi non nulli in corrispondenza di elementi nulli della matrice di partenza. Questo fenomeno, noto con il nome di riempimento o *fill-in*, è particolarmente gravoso da un punto di vista computazionale perché non consente di memorizzare la fattorizzazione di una matrice sparsa nella stessa area di memoria necessaria per memorizzare la matrice stessa. La tecnica di pivotazione totale può essere convenientemente applicata per prevenire e/o contenere il fenomeno del fill-in.

Esercizio 5.1

Utilizzare opportunamente i comandi Matlab `ones` e `diag` per costruire una matrice $A \in \mathbb{R}^{n\times n}$ con $n = 20$ tale che, $a_{1,i} = 1$ e $a_{i,1} = 1$ per $i = 1, \ldots, n$, $a_{ii} = 4$ per $i = 2, \ldots, n$ ed infine $a_{i,i+1} = -1$ e $a_{i+1,i} = -1$ per $i = 2, \ldots, n-1$. Tutti gli altri coefficienti di A sono nulli. Tale matrice si dice *sparsa* poiché il numero degli elementi non-nulli di A è $\mathcal{O}(n) \ll n^2$, dunque molto elevato[2]. Per visualizzare la posizione degli elementi non nulli di A si utilizzi il comando `spy`. Calcolare inoltre la fattorizzazione LU di A. Utilizzare il comando `spy` per determinare se le matrici L ed U siano o meno sparse. A vostro parere, le caratteristiche di sparsità dei fattori L e U sono vantaggiose o svantaggiose?

[2] In tal caso è vantaggioso memorizzare solo gli elementi non-nulli di A. In Matlab ciò viene fatto attraverso il comando `sparse`, ad esempio `B=sparse(A)`.