▷ Lezione III

Metodi diretti per sistemi lineari

Libro Capitolo 5, sezione 5.1, 5.2, 5.3

Soluzione di (grandi) sistemi lineari

In questa parte vogliamo introdurre schemi e algoritmi numerici per poter risolvere un problema lineare con molte equazioni. Un sistema di equazioni può essere espresso in forma matriciale, come illustrato nell'esempio.

Esempio 3.1

$$\begin{aligned} x_1 + x_2 + 3x_3 &= 1 \\ 2x_1 + 3x_2 + 5x_3 &= 2 \\ 7x_1 + 8x_2 + 9x_3 &= 3 \end{aligned} \xrightarrow{A\boldsymbol{x}=\boldsymbol{b}} \begin{bmatrix} 1 & 1 & 3 \\ 2 & 3 & 5 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

Consideriamo il caso di sistemi di n equazioni in n incognite. Vogliamo quindi trovare $\boldsymbol{x} \in \mathbb{R}^n$ tale che

$$A\boldsymbol{x} = \boldsymbol{b}, \tag{1.1}$$

con $A \in \mathbb{R}^{n \times n}$ matrice e con $\boldsymbol{b} \in \mathbb{R}^n$ vettore termine noto.

Teorema 3.1

Il sistema (1.1) ammette una e una sola soluzione se una delle seguenti condizioni equivalenti è verificata

1. A è invertibile, ovvero che esiste $A^{-1} \in \mathbb{R}^{n \times n}$ tale che
$$AA^{-1} = A^{-1}A = I;$$
2. il rango della matrice (cioè il massimo numero di righe (o colonne) linearmente indipendenti della matrice) A è pieno, ovvero uguale a n;
3. il sistema omogeneo $A\boldsymbol{x} = \boldsymbol{0}$ ammette $\boldsymbol{x} = \boldsymbol{0}$ come unica soluzione.

Una possibilità per risolvere (1.1) è utilizzare la formula di Cramer, ossia calcolare ogni componente x_i come

$$x_i = \frac{\det A_i}{\det A},$$

dove A_i è la matrice ottenuta da A sostituendo $\boldsymbol{b}$ alla i-esima colonna di A. Notiamo che la formula di Cramer richiede un numero di operazioni fra numeri *floating point* (flop) dell'ordine di $3(n+1)!$ per completare la risoluzione. Infatti, il calcolo del determinante è un'operazione molto costosa e richiede $\mathcal{O}(n!)$ operazioni. Se considerassimo un sistema lineare relativamente piccolo per le applicazioni, ad esempio con $n = 1000$ allora il numero di flop richieste per la risoluzione del metodo di Cramer sarebbe dell'ordine di 10^{9141}. Considerando il processore Intel Core i7-9750H che riesce a calcolare $340 \cdot 10^9$ flop al secondo, ci vorrebbe quindi un numero inammissibile di anni per la sua risoluzione, avendo un anno circa $3 \cdot 10^7$ secondi. Allo stesso modo, per lo stesso motivo non è pensabile calcolare la matrice inversa A^{-1} di una matrice di grandi dimensioni, per poi trovare la soluzione come $A^{-1}\boldsymbol{b}$.

È chiaro quindi che questa strada non è percorribile e dobbiamo sviluppare nuove strategie per la risoluzione di (1.1).

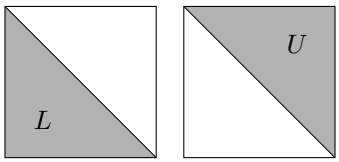
Possiamo classificare i metodi numerici per la risoluzione di un sistema lineare in due macro gruppi: i metodi diretti e i metodi iterativi. Nel primo gruppo troviamo il metodo, ad esempio, di Cramer, il metodo di eliminazione di Gauss, e la fattorizzazione LU. Tali metodi forniscono la soluzione del sistema lineare dopo un numero noto a priori di passi. Nella classe dei metodi iterativi, ad esempio, abbiamo il metodo di Jacobi, il metodo di Gauss-Seidel, i metodi di discesa (gradiente e gradiente coniugato), e molti altri ancora. Utilizzando un metodo iterativo la soluzione viene calcolata in modo iterativo, ovvero data una soluzione di tentativo iniziale $\boldsymbol{x}^0$ viene generata una sequenza $\{\boldsymbol{x}^k\}$ tale che

$$\lim_{k\to\infty} \boldsymbol{x}^k = \boldsymbol{x},$$

dove $\boldsymbol{x}$ è la soluzione del problema (1.1). Discuteremo in seguito in quali casi sia meglio applicare un metodo diretto o un metodo iterativo. Per il momento, concentriamo la nostra attenzione sulla prima classe di metodi.

Fattorizzazione LU

Data una matrice $A \in \mathbb{R}^{n\times n}$, supponiamo che esistano due matrici $L \in \mathbb{R}^{n\times n}$ e $U \in \mathbb{R}^{n\times n}$ tali che L è triangolare inferiore e U è triangolare superiore, ovvero



dove quindi gli elementi $l_{ij} = 0$ per $j > i$ e dove $u_{ij} = 0$ per $j < i$. Richiediamo che le due matrici L e U siano tali che possiamo decomporre A nel seguente modo

$$A = LU,$$

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & & & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \dots & & & \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{21} & \dots & u_{n1} \\ 0 & u_{22} & \dots & u_{2n} \\ \dots & & & \\ 0 & 0 & \dots & u_{nn} \end{bmatrix}$$

Abbiamo visto precedentemente che il sistema lineare (1.1) è risolubile nel caso in cui, ad esempio, la matrice $A \in \mathbb{R}^{n\times n}$ risulti invertibile. Ciò equivale ad avere che il $\det A \neq 0$ che implica, per il teorema di Binet,

$$0 \neq \det A = \det(LU) = \det L \det U.$$

La risolubilità del sistema lineare (1.1) richiede che, se A è fattorizzata come il prodotto LU, allora $\det L \neq 0$ e anche che $\det U \neq 0$. Essendo L e U matrici triangolari allora il loro determinante è dato dal prodotto degli elementi sulla diagonale,

$$\det L = l_{11}l_{22}\dots l_{nn} \quad \text{e} \quad \det U = u_{11}u_{22}\dots u_{nn},$$

quindi è necessario che l_{ii} e u_{ii} siano non nulli per ogni $i = 1, \dots, n$.

Perché è utile esprimere A come prodotto di due matrici triangolari? Se la matrice A è fattorizzabile come $A = LU$ allora il sistema (1.1) può essere riscritto come

$$A\boldsymbol{x} = \boldsymbol{b} \quad \xrightarrow{A=LU} \quad LU\boldsymbol{x} = \boldsymbol{b} \quad \xrightarrow{\boldsymbol{y}=U\boldsymbol{x}} \quad \begin{cases} L\boldsymbol{y} = \boldsymbol{b} \\ U\boldsymbol{x} = \boldsymbol{y} \end{cases},$$

quindi al posto di risolvere $A\boldsymbol{x} = \boldsymbol{b}$ si possono risolvere due sistemi in cascata, prima $L\boldsymbol{y} = \boldsymbol{b}$ e successivamente $U\boldsymbol{x} = \boldsymbol{y}$. Risolvere il primo sistema in L è semplice perchè si può utilizzare l'algoritmo di sostituzione in avanti, abbiamo infatti che

$$L\boldsymbol{y} = \boldsymbol{b} \quad \Rightarrow \quad \begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \dots & & & \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix} \quad \Rightarrow \quad \begin{cases} l_{11}y_1 = b_1 \\ l_{21}y_1 + l_{22}y_2 = b_2 \\ \dots \\ l_{n1}y_1 + l_{n2}y_2 + \dots + l_{nn}y_n = b_n \end{cases}$$

Dalla prima equazione possiamo ricavare il valore di y_1, sostituirlo nella seconda equazione e quindi trovare il valore di y_2, e così via. All'ultima equazione abbiamo calcolato il valore di y_i per $i = 1, \dots, n-1$ e possiamo quindi determinare il valore di y_n e concludere la risoluzione del sistema lineare $L\boldsymbol{y} = \boldsymbol{b}$. In formula, l'algoritmo viene scritto come

$$y_1 = \frac{b_1}{l_{11}} \quad y_i = \frac{1}{l_{ii}} \left(b_i - \sum_{j=1}^{i-1} l_{ij}y_j \right) \quad \text{per } i = 2, \dots, n.$$

Avendo quindi calcolato il valore di $\boldsymbol{y}$, possiamo risolvere il problema lineare associato alla matrice U, ovvero $U\boldsymbol{x} = \boldsymbol{y}$. Utilizziamo in questo caso l'algoritmo di sostituzione all'indietro, partendo dall'ultima riga e procedendo verso la prima.

$$U\boldsymbol{x} = \boldsymbol{y} \quad \Rightarrow \quad \begin{bmatrix} u_{11} & u_{21} & \dots & u_{n1} \\ 0 & u_{22} & \dots & u_{2n} \\ \dots & & & \\ 0 & 0 & \dots & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} \quad \Rightarrow \quad \begin{cases} u_{11}x_1 + u_{12}x_2 + \dots + u_{nn}x_n = y_1 \\ \dots \\ u_{n-1,1}x_1 + u_{n-1,2}x_2 = y_{n-1} \\ u_{n1}x_1 = y_n \end{cases}.$$

Dall'ultima equazione possiamo ricavare il valore di x_n, sostituirlo nella penultima equazione e quindi trovare il valore di x_{n-1}, e così via. Alla prima equazione abbiamo calcolato il valore di x_i per $i = 2, \dots, n$ e possiamo quindi determinare il valore di x_1 e concludere la risoluzione del sistema lineare $U\boldsymbol{x} = \boldsymbol{y}$, conseguentemente la soluzione del sistema $A\boldsymbol{x} = \boldsymbol{b}$. In formula, l'algoritmo viene scritto come

$$x_n = \frac{y_n}{u_{nn}} \quad x_i = \frac{1}{u_{ii}} \left(y_i - \sum_{j=i+1}^{n} u_{ij} x_j \right) \quad \text{per } i = n-1, \dots, 1.$$

Il generico passo di sostituzione in avanti richiede $i-1$ prodotti, dati da $l_{ij}y_j$, $i-2$ somme date dalla sommatoria $\sum_{j=1}^{i-1}$, 1 sottrazione e 1 divisione. Il passo richiede quindi $2i-1$ operazioni per calcolare y_i avendo noti gli altri valori. Globalmente il costo computazionale per l'algoritmo di sostituzione in avanti è dato da

$$\sum_{i=1}^{n}(2i-1) = \sum_{i=1}^{n} 2i - \sum_{i=1}^{n} 1 = 2\sum_{i=1}^{n} i - n = 2\frac{n(n+1)}{2} - n = n^2,$$

dove abbiamo sfruttato la formula notevole $\sum_{i=1}^{n} i = n(n+1)/2$.

Il costo computazionale quindi è di n^2 flop, analogamente possiamo calcolarlo per sostituzione all'indietro ed otterremo un risultato analogo. Quindi la risoluzione del sistema lineare, date L e U,

$$\begin{cases} L\boldsymbol{y} = \boldsymbol{b} \\ U\boldsymbol{x} = \boldsymbol{y} \end{cases} .$$

ha un costo computazionale pari a $2n^2$ flop.

Vediamo ora come poter determinare le matrici L e U data la matrice A, ovvero come calcolare la fattorizzazione $A = LU$. Consideriamo un caso semplice, assumiamo $A \in \mathbb{R}^{2\times 2}$ e imponiamo che $A = LU$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix} \quad \Rightarrow \quad \begin{cases} a_{11} = l_{11}u_{11} \\ a_{12} = l_{11}u_{12} \\ a_{21} = l_{21}u_{11} \\ a_{22} = l_{21}u_{12} + l_{22}u_{22} \end{cases}$$

Abbiamo un sistema di 4 equazioni in 6 incognite, ossia un sistema sotto-determinato. Per rendere il sistema determinato assumiamo che $l_{ii} = 1$ per $i = 1, \dots, n$, ovvero

$$L = \begin{bmatrix} 1 & 0 \\ l_{21} & 1 \end{bmatrix} \quad \text{e} \quad U = \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix} \quad \Rightarrow \quad \begin{cases} a_{11} = u_{11} \\ a_{12} = u_{12} \\ a_{21} = l_{21}u_{11} \\ a_{22} = l_{21}u_{12} + l_{22}u_{22}. \end{cases}$$

Otteniamo così un sistema di 4 equazioni in 4 incognite. In generale se $A \in \mathbb{R}^{n\times n}$ allora, fissando $l_{ii} = 1$ per $i = 1, \dots, n$, otterremmo n^2 equazioni in n^2 incognite.

Metodo di eliminazione di Gauss

Per capire come costruire le matrici L e U, partiamo dall'esame di questo esempio.

Esempio 3.2

Si vuole risolvere il seguente sistema lineare dato da

$$\begin{aligned} x_1 + x_2 + 3x_3 &= 1 \\ 2x_1 + 3x_2 + 5x_3 &= 2 \\ 7x_1 + 8x_2 + 9x_3 &= 3 \end{aligned} \quad \xrightarrow{A\boldsymbol{x}=\boldsymbol{b}} \quad \begin{bmatrix} 1 & 1 & 3 \\ 2 & 3 & 5 \\ 7 & 8 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

Una strategia consiste nell'eliminare le incognite cercando di ottenere, con combinazioni lineari delle righe, un sistema triangolare. Poniamo $A^{(1)} = A$ e $\boldsymbol{b}^{(1)} = \boldsymbol{b}$ calcoliamo $A^{(2)}$ sottraendo alla seconda riga di A la prima riga moltiplicata per $a_{21}^{(1)}/a_{11}^{(1)} = 2$ e alla terza riga la prima riga moltiplicata per $a_{31}^{(1)}/a_{11}^{(1)} = 7$. Analogamente per il termine noto $\boldsymbol{b}^{(2)}$, otteniamo un sistema equivalente al sistema originale dato da

$$\begin{bmatrix} 1 & 1 & 3 \\ 0 & 1 & -1 \\ 0 & 1 & -12 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -4 \end{bmatrix}.$$

in cui, nella seconda e terza equazione, l'incognita x_1 non è più presente. Tale operazione si ottiene anche introducendo la matrice M_1 data da

$$M_1 = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -7 & 0 & 1 \end{bmatrix} \quad \rightarrow \quad M_1 \begin{bmatrix} 1 & 1 & 3 \\ 2 & 3 & 5 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 3 \\ 0 & 1 & -1 \\ 0 & 1 & -12 \end{bmatrix}$$

E quindi $A^{(2)} = M_1 A^{(1)}$, analogamente per il vettore $\boldsymbol{b}$.

Possiamo quindi calcolare $A^{(3)}$ sottraendo alla terza riga di $A^{(2)}$ la seconda riga moltiplicata per $a_{32}^{(2)}/a_{22}^{(2)} = 1$. Analogamente per il termine noto $\boldsymbol{b}^{(3)}$, otteniamo un sistema equivalente al sistema originale dato da

$$\begin{bmatrix} 1 & 1 & 3 \\ 0 & 1 & -1 \\ 0 & 0 & -11 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -4 \end{bmatrix}$$

Otteniamo quindi un sistema triangolare superiore, che è quindi risolvibile usando l'algoritmo di sostituzione all'indietro. Anche in questo caso possiamo considerare una matrice M_2 tale che

$$M_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \quad \rightarrow \quad M_2 \begin{bmatrix} 1 & 1 & 3 \\ 0 & 1 & -1 \\ 0 & 1 & -12 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 3 \\ 0 & 1 & -1 \\ 0 & 0 & -11 \end{bmatrix}$$

L'esempio ci fornisce una strategia per il caso generale: posso generare una sequenza di matrici $A^{(1)} = A$, $A^{(2)}$, ..., $A^{(n)} = U$ che forniscono in n passaggi la matrice triangolare superiore cercata. In generale possiamo scrivere li Metodo di Eliminazione di Gauss (M.E.G.) come segue.

Algoritmo 3.1 - Eliminazione di Gauss

Data: Data $A^{(1)} = A$;
for $k = 1, \dots, n-1$ **do**
  for $i = k+1, \dots, n$ **do**
    $l_{ik} \leftarrow \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$;
    for $j = k+1, \dots, n$ **do**
      $a_{ij}^{(k+1)} \leftarrow a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}$;
    end
    $b_i^{(k+1)} \leftarrow b_i^{(k)} - l_{ik} b_k^{(k)}$;
  end
end

I valori di l_{ik} sono detti moltiplicatori, mentre i valori di $a_{kk}^{(k)}$ sono detti elementi pivotali (o *pivot*) dati dagli elementi sulla diagonale della matrice $A^{(k)}$. $A^{(k)}$, allo step k dell'algoritmo, è "triangolare" solo fino alla riga k,

$$A^{(k)} = \begin{bmatrix} a_{11}^{(k)} & a_{12}^{(k)} & a_{13}^{(k)} & \dots & \dots & a_{1n}^{(k)} \\ 0 & a_{22}^{(k)} & a_{23}^{(k)} & \dots & \dots & a_{2n}^{(k)} \\ 0 & 0 & a_{33}^{(k)} & \dots & \dots & a_{3n}^{(k)} \\ \dots & & & & & \\ 0 & \dots & 0 & a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ \dots & & & & & \\ 0 & \dots & 0 & a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{bmatrix}.$$

Sappiamo che all'ultimo step del metodo di eliminazione di Gauss la matrice $A^{(n)}$ è triangolare superiore U, mentre è possibile costruire una matrice triangolare inferiore dall'algoritmo stesso considerando i moltiplicatori l_{ik}. Infatti, possiamo definire

$$L = \begin{bmatrix} 1 & 0 & \dots & 0 \\ l_{21} & 1 & \dots & 0 \\ \dots & & & \\ l_{n1} & l_{n2} & \dots & 1 \end{bmatrix}.$$

La matrice L è anche calcolabile nel seguente modo

$$A^{(n)} = M_{n-1} A^{(n-1)} = M_{n-1} \dots M_1 A = U \quad \text{quindi} \quad L = (M_{n-1} \dots M_1)^{-1} = M_1^{-1} \dots M_{n-1}^{-1}$$

Data la particolare struttura della matrice M_i, possiamo caratterizzarne l'inversa che risulta

$$M_i^{-1} = 2I - M_i.$$

Il costo computazionale del metodo di eliminazione di Gauss, che coincide con la creazione delle matrici L e U, è dell'ordine di $\mathcal{O}(\frac{2}{3}n^3)$ flop. Possiamo comparare il costo computazionale, come numero di flop richiesti, tra il metodo eliminazione di Gauss e il metodo di Cramer con la tabella seguente

	MEG	Cramer
$n = 3$	18	27
$n = 5$	80	$\sim 10^3$
$n = 10$	600	$\sim 10^8$
$n = 20$	5000	$\sim 10^{20}$

È chiaro che il metodo di eliminazione di Gauss risulta notevolmente più veloce del metodo di Cramer. Quest'ultimo infatti non viene mai usato nella pratica.

Un altro fattore importante per valutare un algoritmo è la sua occupazione di memoria fisica una volta implementato in un computer. Partendo dalla matrice A della forma

$$A^{(0)} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & & & & \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix}$$

Dato che al passo k ho analizzato la riga k-esima della matrice A, e calcolato i termini di L e U fino alla riga k, è possibile memorizzare la matrice all'iterazione $A^{(k)}$ come segue

$$A^{(k)} = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & & \dots & a_{1n}^{(1)} \\ l_{21} & a_{22}^{(2)} & a_{23}^{(2)} & & \dots & a_{2n}^{(2)} \\ l_{31} & l_{32} & a_{33}^{(3)} & & \dots & a_{3n}^{(3)} \\ \dots & & & & & \\ & & & a_{kk}^{(k)} & \dots & a_{kn}^{(k)} \\ \dots & & & a_{nk}^{(k)} & \dots & a_{nn}^{(k)} \end{bmatrix} \xrightarrow{\text{per } k=n} \begin{bmatrix} u_{11} & u_{12} & u_{13} & & \dots & u_{1n} \\ l_{21} & u_{22} & u_{23} & & \dots & u_{2n} \\ l_{31} & l_{32} & u_{33} & & \dots & u_{3n} \\ \dots & & & & & \\ l_{n1} & l_{n2} & l_{n3} & \dots & l_{n,n-1} & u_{nn} \end{bmatrix}$$

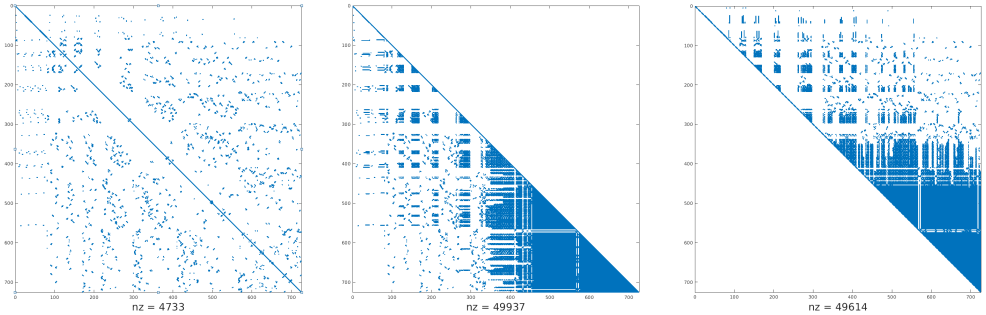
dove la prima riga è stata calcolata dalla prima iterazione del metodo di eliminazione di Gauss, e così anche le altre righe che vengono aggiornate di volta in volta fino alla riga $k-1$.

Se la matrice A è sparsa, ovvero se il numero di elementi non nulli è molto minore della dimensione della matrice stessa, il metodo di eliminazione di Gauss può generare il fenomeno del fill-in, ossia le matrici L e U possono essere piene e quindi molto più costose.

Esempio 3.3

Consideriamo una matrice $A \in \mathbb{R}^{n \times n}$ con $n = 725$ qui sotto riportata. Osserviamo che A, la matrice a sinistra, contiene circa 5×10^5 elementi, di cui solo 4733 non nulli. Tuttavia, con la fattorizzazione LU troviamo che sia L sia U, in centro e a

destra, contengono ben 5×10^4 elementi non nulli.



Approfondimento 3.3

Cosa si intende per *matrice sparsa*? Spesso le matrici che risultano dalla discretizzazione di problemi quali le equazioni alle derivate parziali hanno grandi dimensioni, ma pochi elementi non nulli. Per essere più precisi, nel caso di matrici $A \in \mathbb{R}^{n \times n}$, se il numero di elementi non nulli n_{nz} è dell'ordine di n si parla di matrice sparsa. In questo caso è possibile memorizzare la matrice in modo da risparmiare memoria. Infatti, è inutile memorizzare gli elementi nulli. Un'alternativa è memorizzare tre vettori di lunghezza n_{nz} che contengono, per ogni elemento non nullo, il suo indice di riga i, di colonna j, e il suo valore.

Esistenza della fattorizzazione LU

In questa sezione, ci chiediamo sotto quali condizioni esiste la fattorizzazione LU. Partiamo dal seguente esempio.

Esempio 3.4

Sia data la seguente matrice

$$A = \begin{bmatrix} 1 & 1 & 3 \\ 2 & 2 & 2 \\ 3 & 6 & 4 \end{bmatrix}$$

Notiamo che A è invertibile, infatti $\det(A) = 12$. Applichiamo il metodo di eliminazione di Gauss, impostando $A^{(1)} = A$ otteniamo

$$A^{(2)} = \begin{bmatrix} 1 & 1 & 3 \\ 0 & 0 & -4 \\ 0 & 3 & -5 \end{bmatrix} \quad \rightarrow \quad A^{(3)} = \begin{bmatrix} 1 & 1 & 3 \\ 0 & 0 & -4 \\ ? & ? & ? \end{bmatrix}$$

tuttavia nel calcolo di $A^{(3)}$ se sottraiamo alla terza riga di $A^{(2)}$ la seconda riga moltiplicata per il fattore $a_{32}^{(2)}/a_{22}^{(2)} = 3/0$ ottengo un'operazione non valida. In questo caso il metodo il metodo di eliminazione di Gauss si interrompe e non può calcolare la soluzione.

Si dice che una matrice quadrata è singolare se ha il proprio determinante uguale a zero. Tuttavia, abbiamo visto che non tutte le matrici non singolari ammettono la fattorizzazione LU, e dobbiamo definire criteri aggiuntivi.

Definiamo le sotto-matrici principali A_i di A come le matrici ottenute estraendo da A le sole prime i-righe e colonne:

$$A_1 = [a_{11}] \quad A_2 = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad A_3 = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad \dots$$

Teorema 3.2 - Esistenza e unicità della fattorizzazione LU

Data una matrice $A \in \mathbb{R}^{n\times n}$ non singolare, la sua fattorizzazione LU esiste ed è unica se e solo se le sotto-matrici principali A_i di A di ordine $i = 1, \dots, n-1$ sono non singolari.

Il teorema fornisce una condizione *necessaria* e *sufficiente*, esistono anche delle condizioni *sufficienti* che garantiscono la fattorizzazione

1. se A è una matrice a dominanza diagonale stretta per righe o per colonne, ovvero se

$$\text{per righe} \quad |a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}| \quad i = 1, \dots, n$$

$$\text{per colonne} \quad |a_{jj}| > \sum_{j=1, j\neq i}^{n} |a_{ij}| \quad j = 1, \dots, n.$$

   Nel primo caso se il termine diagonale della riga i-esima è maggiore in modulo della somma di tutti gli elementi extra-diagonali della stessa riga, in modulo. Analogamente, ma per colonne, la seconda condizione.

2. se A è una matrice simmetrica definita positiva (SPD), ovvero se

$$\text{(simmetria)} \quad a_{ij} = a_{ji} \quad \forall i, j = 1, \dots, n$$

$$\text{(definita positività)} \quad \lambda_i > 0$$

   dove i λ_i sono gli autovalori di A. Una condizione equivalente al richiedere che gli autovalori siano tutti strettamente positivi è richiedere che

$$\boldsymbol{x}^\top A\boldsymbol{x} \geq 0 \quad \forall \boldsymbol{x} \quad \text{e} \quad \boldsymbol{x}^\top A\boldsymbol{x} = 0 \iff \boldsymbol{x} = \boldsymbol{0}.$$

Esempio 3.5

Consideriamo la seguente matrice

$$A = \begin{bmatrix} 4 & -1 & 2 \\ -1 & 3 & 0 \\ 1 & 2 & -5 \end{bmatrix}$$

essa risulta a dominanza diagonale stretta per righe, ma non per colonne a causa della seconda colonna.

Tecnica del pivoting

Il Teorema 1.3 richiede condizioni aggiuntive alla sola invertibilità della matrice A per garantire l'esistenza della fattorizzazione LU. Questo significa che per alcuni casi può capitare che la matrice A sia invertibile, e quindi possibile la soluzione di un sistema lineare in A, ma la fattorizzazione LU non esista. Come possiamo risolvere il problema? Consideriamo il seguente esempio.

Esempio 3.6

Sia data la matrice A a cui si vuole applicare il metodo di eliminazione di Gauss per il calcolo della fattorizzazione LU, data da

$$A = A^{(1)} = \begin{bmatrix} 1 & 1 & 3 \\ 2 & 2 & 2 \\ 3 & 6 & 4 \end{bmatrix} \quad \rightarrow \quad A^{(2)} = \begin{bmatrix} 1 & 1 & 3 \\ 0 & 0 & -4 \\ 0 & 3 & -5 \end{bmatrix}$$

risulta quindi impossibile calcolare la matrice $A^{(3)}$, tuttavia se scambiamo la seconda e la terza riga della matrice A otteniamo una nuova matrice $\widetilde{A}$

$$\widetilde{A} = \widetilde{A}^{(1)} = \begin{bmatrix} 1 & 1 & 3 \\ 3 & 6 & 4 \\ 2 & 2 & 2 \end{bmatrix} \quad \rightarrow \quad \widetilde{A}^{(2)} = \begin{bmatrix} 1 & 1 & 3 \\ 0 & 3 & -5 \\ 0 & 0 & 1 \end{bmatrix} = \widetilde{A}^{(3)}.$$

In questo caso siamo quindi in grado di calcolare la fattorizzazione LU. Notiamo che se definiamo la matrice P, qui di seguito riportata, quando moltiplicata a sinistra ha l'effetto di permutare la terza e la seconda riga

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \rightarrow \quad PA = \widetilde{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 3 \\ 2 & 2 & 2 \\ 3 & 6 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 3 \\ 3 & 6 & 4 \\ 2 & 2 & 2 \end{bmatrix}$$

L'approccio utilizzato nell'esempio è nota come tecnica del pivoting, che consiste nel permutare opportunamente le righe della matrice di partenza e del termine noto in modo da evitare elementi pivotali nulli. Questa operazione di permutazione è descritta da una matrice di pivoting P, o matrice di permutazione, tale per cui

$$PA = LU \quad \xrightarrow{\text{il che implica la soluzione del problema}} \quad PA\boldsymbol{x} = P\boldsymbol{b}. \tag{1.2}$$

P ha elementi uguali a 0 o 1 ed è uguale alla matrice identità nel caso in cui non avvenga nessuna permutazione. L'algoritmo può essere quindi riassunto nel seguente.

Algoritmo 3.2 - Eliminazione di Gauss con pivoting

Data: Data $A^{(1)} = A$ e $P = I$;
for $k = 1, \dots, n-1$ **do**
  $f \leftarrow |a_{fk}^{(k)}| = \max_{r=k,\dots,n} |a_{rk}^{(k)}|$;
  scambio la riga k con la riga f in A, in P e in $\boldsymbol{b}$;
  for $i = k+1, \dots, n$ **do**
    $l_{ik} \leftarrow \dfrac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$;
    for $j = k+1, \dots, n$ **do**
      $a_{ij}^{(k+1)} \leftarrow a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}$;
    end
    $b_i^{(k+1)} \leftarrow b_i^{(k)} - l_{ik} b_k^{(k)}$;
  end
end

Una volta calcolata la tripletta (L, U, P), la soluzione del problema $A\boldsymbol{x} = \boldsymbol{b}$ segue i seguenti passi

$$A\boldsymbol{x} = \boldsymbol{b} \xrightarrow{\text{moltiplico a sinistra per } P} PA\boldsymbol{x} = P\boldsymbol{b} \xrightarrow{\text{sostituisco } PA=LU} LU\boldsymbol{y} = P\boldsymbol{b}$$

pongo quindi come fatto in precedenza $\boldsymbol{y} = U\boldsymbol{x}$ ed ottengo

$$\begin{cases} L\boldsymbol{y} = P\boldsymbol{b} \\ U\boldsymbol{x} = \boldsymbol{y} \end{cases}$$

Il costo computazionale del metodo di eliminazione di Gauss con pivoting rimane del tutto analogo al metodo senza pivoting. Il calcolo di (L, U, P) è dell'ordine di $\frac{2}{3}n^3$ flop mentre la soluzione con gli algoritmo di sostituzione in avanti e indietro ha un costo computazionale complessivo dell'ordine di $2n^2$. Utilizzando la tecnica del pivoting è possibile ottenere una fattorizzazione LU con pivoting P, eventualmente $P = I$ in alcuni casi, *per ogni A invertibile*.

In Python, è possibile calcolare la fattorizzazione LU con pivoting utilizzando il seguente comando

```
P, L, U = scipy.linalg.lu(A)
```

Notiamo che Python restituisce la seguente decomposizione $A = PLU$, quindi nel nostro approccio dobbiamo usare la trasposta `P.T` dato che $PP^\top = I$. Nel caso in cui `P` è la matrice identità allora non c'è stato alcun pivoting, altrimenti è stato effettuato un pivoting per il calcolo della fattorizzazione. Per la risoluzione quindi del problema utilizzo l'algoritmo di sostituzione in avanti e indietro già discusso.

Stabilità del metodo di eliminazione di Gauss

In questa sezione vogliamo studiare la stabilità del metodo di eliminazione di Gauss. Il concetto di *stabilità* è trasversale in analisi numerica e riguarda la

dipendenza della soluzione dalla perturbazione dei dati del problema (o del problema stesso). Nel caso dei sistemi lineari, vorremmo risolvere il cosiddetto sistema *originale*, ma, a causa di errori di arrotondamento, risolviamo invece un sistema *perturbato*,

$$A\boldsymbol{x} = \boldsymbol{b} \quad \xrightarrow{\text{in realtà risolvo}} \quad (A + \delta A)(\boldsymbol{x} + \delta \boldsymbol{x}) = \boldsymbol{b} + \delta \boldsymbol{b},$$

dove δA è una matrice che perturba la matrice originale A, analogamente $\delta \boldsymbol{b}$ perturba il termine noto originale $\boldsymbol{b}$ e $\boldsymbol{x}+\delta \boldsymbol{x}$ è la soluzione del problema perturbato data dalla soluzione del problema originale e da un contributo $\delta \boldsymbol{x}$ che rappresenta la perturbazione.

Nell'ipotesi semplificativa in cui l'errore commesso interessi solo il termine noto ($\delta A = 0$) l'errore relativo che viene commesso nella risoluzione del problema con il metodo di eliminazione di Gauss soddisfa questa disuguaglianza:

$$\frac{1}{\text{cond}(A)} \frac{\|\delta \boldsymbol{b}\|}{\|\boldsymbol{b}\|} \leq \frac{\|\delta \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq \text{cond}(A) \frac{\|\delta \boldsymbol{b}\|}{\|\boldsymbol{b}\|}, \tag{1.3}$$

dove $\text{cond}(A)$ è il numero di condizionamento della matrice A che, per matrici simmetriche e definite positive, è calcolabile come

$$\text{cond}(A) = \frac{\lambda_{\text{max}}(A)}{\lambda_{\text{min}}(A)}.$$

Diciamo che A è ben condizionata se $\text{cond}(A) \approx 1$, mentre A è mal condizionata se $\text{cond}(A) \gg 1$ (notiamo che, per definizione, $\text{cond}(A)$ non può mai essere minore di 1!). In quest'ultimo caso, da (1.3), possiamo concludere che per piccole perturbazioni del termine noto, ovvero per $\|\delta \boldsymbol{b}\|$ piccolo, si possono avere grandi errori sulla soluzione, ossia $\|\delta \boldsymbol{x}\|$ grande. Il condizionamento di una matrice è quindi un elemento fondamentale per poter valutare l'accuratezza della soluzione numerica ottenuta.

In generale il risultato di prima può essere scritto anche in termini della perturbazione su A ed è il seguente

$$\frac{\|\delta \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq \frac{\text{cond}(A)}{1 - \text{cond}(A)\, \|\delta A\| / \|A\|} \left(\frac{\|\delta \boldsymbol{b}\|}{\|\boldsymbol{b}\|} + \frac{\|\delta A\|}{\|A\|} \right).$$

Approfondimento 3.6

Per matrici generiche il numero di condizionamento è definito come

$$\text{cond}(A) = \|A\| \left\|A^{-1}\right\|$$

Ma cosa si intende per norma di una matrice? Se $\|\boldsymbol{v}\|$ indica la norma di un vettore la corrispondente norma di matrice si definisce come

$$\|A\| = \sup_{\boldsymbol{v} \in \mathbb{R}^n, \boldsymbol{v} \neq \boldsymbol{0}} \frac{\|A\boldsymbol{v}\|}{\|\boldsymbol{v}\|}.$$

Possiamo dimostrare una parte della disuguaglianza 1.3 con i seguenti passaggi. Partendo dall'osservazione che $A\delta\boldsymbol{x} = \delta\boldsymbol{b}$, e quindi $\delta\boldsymbol{x} = A^{-1}\delta\boldsymbol{b}$ passiamo alle norme e normalizziamo rispetto a $\|\boldsymbol{x}\|$:

$$\|\delta\boldsymbol{x}\| = \left\|A^{-1}\delta\boldsymbol{b}\right\| \leq \left\|A^{-1}\right\| \|\delta\boldsymbol{b}\|$$
$$\frac{\|\delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq \left\|A^{-1}\right\| \frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{x}\|}$$

Osserviamo che inoltre che $\|A\|\,\|\boldsymbol{x}\| \geq \|\boldsymbol{b}\|$, quindi $\|\boldsymbol{x}\| \geq \dfrac{\|\boldsymbol{b}\|}{\|A\|}$. Sostituendo nel risultato precedente troviamo

$$\frac{\|\delta\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq \|A\|\left\|A^{-1}\right\| \frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}$$

dove riconosciamo la definizione di condizionamento di A.

Altri metodi diretti

Nel caso in cui la matrice A è simmetrica e definita positiva, possiamo utilizzare la fattorizzazione di Cholesky che per questa classe di matrici esiste ed è unica. L'algoritmo fornisce

$$A \quad \xrightarrow{\text{fattorizzazione di Cholesky}} \quad A = R^\top R,$$

dove R è una matrice triangolare superiore. Il costo computazionale per calcolare R è dell'ordine di $\frac{1}{3}n^3$, la metà rispetto alla fattorizzazione LU. Inoltre, è possibile memorizzare solamente una matrice triangolare e non due, riducendo quindi anche lo spazio macchina utilizzato.

Nel caso in cui la matrice A è tridiagonale, ovvero se $a_{ij} = 0$ per $|i-j| > 1$, l'algoritmo di Thomas fornisce una soluzione con un costo computazionale proporzionale a n. Risulta quindi molto conveniente per questa classe di problemi. La fattorizzazione, in questo caso, si calcola semplicemente come:

$$L = \begin{bmatrix} 1 & & & 0 \\ \beta_2 & 1 & & \\ & \ddots & \ddots & \\ 0 & & \beta_n & 1 \end{bmatrix} \qquad U = \begin{bmatrix} \alpha_1 & a_{12} & & 0 \\ & \alpha_2 & & \\ & \ddots & \ddots & a_{n-1,n} \\ 0 & & & \alpha_n \end{bmatrix}$$

dove gli elementi α_i e β_i si possono trovare con semplici calcoli, ovvero impostato $\alpha_1 = a_1$ allora per $i = 2, \ldots, n$

$$\beta_i = \frac{b_i}{\alpha_{i-1}} \quad \text{e} \quad \alpha_i = a_i - \beta_i c_{i-1}$$