Lezione 03 - Pivoting e Metodi Iterativi

Nelle ultime lezioni abbiamo visto come determinare le matrice L e U con il MEG. Abbiamo anche determinato delle condizioni tale che se la matrice A le completa, possiamo fare la fattorizzazione.

È possibile che una matrice una matrice non rientri nelle 4 condizioni ma sia lo stesso fattorizzabile.

Per garantire il MEG dobbiamo garantire che A non è singolare, questo è utile perché significa che avremo anche una soluzione al sistema.

Prendendo l'esempio di ieri:

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

Il moltiplicatore per il secondo passo sarà impossibile, ma è palese che potremmo scambiare le righe 2 e 3, e il problema si risolverebbe.
Nella prima matrice poi vediamo che la prima sottomatrice non è singolare, invece se cambiamo le righe vediamo che la prima sottomatrice non è singolare, implicando che il sistema è fattorizzabile.

Supponiamo di aver una matrice grande, e durante il MEG un valore è zero, possiamo scambiare una riga. Ma ritornare all'inizio per scambiare la riga e riniziare il MEG da capo non è molto computazionalmente efficiente.

Quello che dobbiamo fare è determinare quando facciamo lo scambio e quali righe scambiamo.

Quando facciamo lo scambio?

Invece di tornare all'inizio e fare lo scambio possiamo fare lo scambio quando affrontiamo il problema, il risultato sarà lo stesso e non dobbiamo buttare via tutti i calcoli che abbiamo che abbiamo già fatto.

Quali righe scambiamo?

Il modo più semplice è di prende la riga sotto, il modo più complicato lo vedremo dopo.

Il scambio delle righe viene chiamato Pivoting.

Pivoting

Chiamo P una matrice di permutazione. Questa matrice sarà usata per scambiare le righe e le colonne nelle matrice e tenere un diario di tutte le righe che abbiamo scambiato.

La matrice P è ortogonale tale che:

$$P^T P = P P^T = I \implies P^T = P^{-1}$$

La matrice P di base è la matrice identità:

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La matrice per scambiare le righe 1 e 3 invece è:

$$P_{13} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Per scambiare la prima e terza riga della matrice allora dobbiamo scambiare la prima e terza riga della matrice identità e poi la applichiamo alla matrice A così:

$$P_{13}A = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a_{31} & a_{32} & a_{33} \\ a_{21} & a_{22} & a_{23} \\ a_{11} & a_{12} & a_{13} \end{bmatrix}$$

Invece per cambiare la prima e terza colonna si scrive la matrice di scambio dopo:

$$AP_{13} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} a_{13} & a_{12} & a_{11} \\ a_{23} & a_{22} & a_{21} \\ a_{33} & a_{32} & a_{31} \end{bmatrix}$$

Ritornando alla matrice grande se vogliamo scambiare la k-esima e (k+1)-esima riga possiamo prende una matrice identità con le righe k e k+1 scambiate.

In matlab quando vogliamo fare la partizione LU di A usiamo la funzione:

$$[L, U, P] = lu(A)$$

Questo estrae le matrici L, U e P. La ragione per cui tiriamo fuori la matrice P è perché se non la esplicitiamo il programma riporterà indietro la matrice $\tilde{L}$ dove $\tilde{L} = P^{-1}L$

Le matrici P sono combinabili, tale che:

$$P = P_n P_{n-1} \dots P_1$$

Creerà P che è la matrice che rappresenta tutti gli scambi che servono per convertire la matrice A da una in cui non funziona il MEG in una in cui il MEG funziona.

Per risolvere i sistemi allora cambiamo la notazione ad esser:

$$\begin{gathered} PAx = Pb \\ L\underbrace{Ux}_{y} = Pb \end{gathered}$$

Il prima sistema che risolviamo è $Ly = Pb$ e il secondo sistema sarà: $Ux = y$. Per un sistema dove non servono cambi P = I

Questo che abbiamo appena visto è un uso minimale del pivoting.

Pivoting per ridurre l'errore

Supponiamo che il l'elemento $a_{kk}^{(k)} = 10^{-15}$.

Tutti i multipli che useremo nel MEG saranno molto sproporzionato rispetto il resto degli elementi nella matrice. La magnificazione causa un aumento degli errori floating point di cui i nostri computer hanno più problemi.

Per risolvere il problema quello che possiamo fare è invece di scambiare questa riga con la riga dopo, possiamo fare uno scan di tutti gli elementi nella stessa colonna nelle righe sotto a quella dove abbiamo trovato questo elemento minuscolo e prendiamo il valore più grande che troviamo. Prendendo il valore più continuamente il valore più grande che riusciamo a trovare ci permette di minimizzare il valore dell'errore che mettiamo nel nostro sistema.

Un esempio:

$$A = \begin{bmatrix} 1 & 1 + 0.5 \times 10^{-15} & 3 \\ 2 & 2 & 20 \\ 3 & 6 & 4 \end{bmatrix}$$

Facendo la differenza di A e la approssimazione con LU troviamo l'errore che in questo caso sarà:

$$A - LU = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

4 è un errore di dimensione immensa anche rispetto all'errore generale che si trova nella matrici.

Casi Speciali per la fattorizzazione

Ci sono due tipi di casi speciali di matrici per cui esistono algoritmi più veloci per risolverle.

Matrici Tridiagonali

Queste matrici sono matrici che hanno elementi non nulla sulla diagonale principale e le diagonali sopra e sotto alla diagonale principale.

La matrici LU terranno la forma uguale dove la matrice L sarà una bidiagonale inferiore e avrà 1 sulla diagonale principale e elementi non nulla sulla diagonale sotto, invece la U sarà una bidiagonale superiore con elementi non nulla sia sulla diagonale sopra che la diagonale principale stessa.

L'algoritmo che risolve questi sistema velocemento è l'algoritmo di Thomas e ha un costo di O(n)

Matrici Simmetriche definite e positive

Anche in questo caso le matrici LU prendono la forma della matrice originale, questo implica LU saranno simmetriche rispetto l'un l'altra e allora $A = R^T R$

L'algoritmo per risolvere questo sistema si chiama Fattorizzazione di Cholesky e un costo di O($\frac{n^3}{3}$) che è meta del costo normale dato che le matrici sono uguali. Tutti gli elementi delle due matrici sono positivi non uguali a 1.

Metodi Iterativi

I metodi iterativi funzionano ponendo il risultato ricavato dall'ultimo calcolo nella stessa equazione per raffinarlo.

Generalmente possiamo pensare dei metodi iterativi come un black box in cui mettiamo un guess iniziale sul risultato finale, alla fine del calcolo mettiamo il risultato del calcolo come il nuovo input, più iterazioni facciamo più idealmente siamo vicini al risultato vero.

Possiamo creare un insieme di tutte le iterazioni approssimate come:

$$x^{(k)}$$

Si spera che con $k \to \infty$ $x^{(k)} \to x$
Possiamo scrivere questo come:

$$\lim_{k \to \infty} x^{(k)} = x \to convergenza$$

Possiamo anche scrivere l'errore come:

$$e^{(k)} = x - x^{(k)}$$

Per noi che guardiamo vettori, l'errore non sarà un numero ma sarà un vettore che ci dice l'errore di approssimazione elemento per elemento.

Possiamo allora anche che per i metodi iterativi:

$$\lim_{k \to \infty} e^{(k)} = \underline{0}$$

Criteri d'arresto

Generalmente useremo 2 criteri d'arresto per limitare il numero di iterazioni che facciamo e per non creare un programma che non si ferma mai.

Il primo criterio è quello della accuratezza, cioè $|e^{(k)}| \leq 10^{-9}$

A volte non sappiamo l'errore quindi quello che facciamo è prendere una stimatore/proxy S che sappiamo sarà più grande dell'errore vero, quindi quando lo stimatore è minore dell'errore voluto possiamo determinare l'errore vero come sotto all'errore voluto:

$$|e^{(k)}| \leq S \leq 10^{-9}$$

Il secondo criterio è quello di numero massimo di iterazioni

Come abbiamo visto l'errore ha equazione $|x - x^{(k)}|$. Questo si può usare se sappiamo l'obbiettivo x. Se non lo sappiamo è un problema, quindi usiamo lo stimare, che è basato su $x_{(k)}$ avere una stima dell'errore, questo lo prendiamo tale che sarà sempre maggiore dell'errore vero quindi quando sappiamo che lo stimatore è minore dell'limite dell'errore sappiamo che anche l'errore vero lo sarà e possiamo fermare il programma.

Un tipo di stimare è il risultate che sarà:

$$\frac{|b - Ax^{(k)}|}{|b|}$$

Metodi Iterativi per risolvere sistemi di equazioni lineari

Per il sistema $Ax = b$ prendiamo arbitrariamente che il black box abbia equazione:

$$x^{(k+1)} = Bx^{(k)} + g \; ; k \geq 0$$

Dove $B \in \mathbb{R}^{n \times n}$ è una matrice di iterazione
E $g \in \mathbb{R}^n$

Dobbiamo prendere B e g tali che $x = Bx + g$ questo criterio è detto relazione di consistenza. La consistenze è si vede quando rimpiazziamo $x^{(k)}$ come x il sistema sta in piedi.

Cioè per $x^{(k)} \to x$ troviamo che $x^{(k)} = Bx^{(k)} + g$ vale.
Con la consistenza verifichiamo che il metodo per risolvere il sistema, cioè B e g, sono sensate e risolvono il problema. Quando un matematico considera una soluzione ad un sistema deve assicurarsi che mantenga la consistenza.

B è legata ad A e g è legata ad A e b.

Possiamo manipolare la equazione che abbiamo preso facendo:

$$Ix = Bx + g$$
$$g = (I - B)x = (I - B)A^{-1}b$$

Dopo aver richiesto la consistenza dobbiamo mettere le condizione per garantire la convergenza.

Presa l'equazione iniziale, la sottraiamo dalla condizione di consistenza:

$$x - x^{(k+1)} = Bx - g - Bx^{(k)} - g$$
$$e^{(k+1)} = B(\underbrace{x - x^{(k)}}_{e^{(k)}})$$

C'è una correlazione tra errori successivi. Seguiamo qui per trovare le condizioni per la convergenza.