1 Lezione del 28-03-25

Riprendiamo il discorso sull'errore inerente dei sistemi lineari.

Avevamo preso delle perturbazioni sulle matrici A e b (dovute a vari effetti reali, quali errori di arrotondamento, di misura, ecc...) nella forma:

$$(A + \delta A)(x + \delta x) = (b + \delta b)$$

e volevamo capire quanto può essere grande l'errore relativo $\frac{|\delta x|}{|x|}$, da:

$$\frac{\text{sol. perturbata} - \text{sol. reale}}{\text{sol. reale}} = \frac{|x + \delta x - x|}{|x|} = \frac{|\delta x|}{|x|}$$

Nel caso di $\delta A = 0$, abbiamo visto di poter maggiorare tale quantità come:

$$\frac{|\delta x|}{|x|} \leq \mu(A) \cdot \frac{|\delta b|}{|b|}$$

con $\mu(A) = |A| \cdot |A^{-1}|$ *numero di condizionamento* (definizione 9.1).

Riguardo a $\mu(A)$, si ha che è ≥ 1, cioè chiaramente non si può ridurre l'errore oltre la perfezione, e se $\mu(A) \approx 10^k$, k è il numero di cifre significative che si *perdono* nel risultato $x + \delta x$.

Possiamo quindi enunciare il seguente teorema:

Teorema 1.1: Condizionamento in δA

Se $|\delta A| \cdot |A^{-1}| < 1$ allora si ha:

$$\frac{|\delta x|}{|x|} \leq \frac{\mu(A)}{1 - \mu(A) \cdot \frac{|\delta A|}{|A|}} \cdot \left(\frac{|\delta A|}{|A|} + \frac{|\delta b|}{|b|} \right)$$

dove osserviamo che se $\delta A = 0$ si otiene la stessa diseguaglianza che abbiamo visto prima.

1.0.1 Stima di μ

In genere è abbastanza costoso calcolare il numero di condizionamento, in quanto bisogna calcolare un inversa e quindi la sua norma. Quello che si può fare è cercarne una stima.

Ad esempio, se A è hermitiana ($A = A^H$) e si considera la norma euclidea $|\cdot|_2$, si ha che:

$$|A|_2 \cdot |A^{-1}|_2 = \rho(A) \cdot \rho(A^{-1}) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$$

cioè prendiamo il rapporto fra l'autovalore più grande e l'autovalore più piccolo di A, per cui si possono usare i metodi per gli autovalori (che vedremo verso la fine del corso).

1.0.2 Stime a posteriori

Supponiamo di aver calcolato $\tilde{x} \in \mathbb{C}^n$ con un qualunque metodo di approssimazione di x, prese A e b per buone. Per valutare se $\tilde{x}$ è una *buona* approssimazione basta guardare al **vettore residuo**, cioè:

$$r = b - A\tilde{x}$$

Potremmo chiederci se, con $|r|$ piccolo, si hanno anche $|x - \tilde{x}|$ piccoli. Sottraiamo allora $Ax = b$ da r:

$$A(x - \tilde{x}) = r \implies x - \tilde{x} = A^{-1}r$$

e quindi vale la diseguaglianza:

$$|x - \tilde{x}| \leq |A^{-1}||r|$$

Usando:

$$|x| \geq \frac{|b|}{|A|}$$

si otterrà allora che:

$$\frac{|x - \tilde{x}|}{|x|} \leq \frac{|A||A^{-1}||r|}{|b|} = \mu(A) \cdot \frac{|r|}{|b|}$$

Allora, in problemi ben condizionati, avremo che $\frac{|x-\tilde{x}|}{|x|}$ errore relativo e $|r|$ sono comparabili, mentre in problemi con condizionamento $\mu(A) >> 1$ si potrebbe avere:

$$\frac{\frac{|x-\tilde{x}|}{|x|}}{|r|} \approx \mu(A)$$

1.1 Tecniche per l'approssimazione delle inverse

Ogni volta che c'è bisogno di risolvere una forma del tipo:

$$Ax = b \implies x = A^{-1}b$$

il metodo naive sarebbe quello di calcolare A^{-1} e moltiplicare per b. Vediamo se si può fare di meglio.

Preso $n = 1$, la forma sarà quella di una semplice equazione lineare:

$$ax = b. \implies x = \frac{b}{a}$$

cioè basta fare una sola divisione, mentre l'approccio naive risulterebbe nel:

1. Calcolare il reciproco a^{-1};
2. Moltiplicare il reciproco per $b \implies x = a^{-1} \cdot b$.

che chiaramente risulta in più passaggi, e quindi più approssimazioni intermedie e in definitiva maggiore errore.

Nel caso matriciale il metodo di divisione equivale a fare una divisione matrice-vettore (che in MATLAB si effettua come `A \ b`), di complessità $O(\frac{2}{3}n^3)$. Di contro, il metodo naive (che in MATLAB si effettua come `inv(A) * b`) avrà complessità intorno ad $O(\frac{8}{3}n^3)$, in quanto calcola la fattorizzazione LU e risolve 2n sistemi triangolari.

Inoltre, l'approccio naive è anche meno accurato, in quanto se il numero di condizionamento $\mu(A)$ è alto, l'errore di approssimazione nei passaggi intermedi potrebbe accumularsi molto di più che rispetto all'approccio della semplice divisione matrice-vettore (tanto che la stessa documentazione di MATLAB suggerisce di evitarlo).

1.2 Metodi iterativi per i sistemi lineari

Veniamo quindi a trattare i metodi iterativi per la risoluzione dei sistemi lineari.

L'idea è quella di approssimare la soluzione di un sistema lineare $Ax = b$ generando una successione di vettori $\{x^{(k)}\}_{k\in\mathbb{N}}$ tali che:

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

La motivazione è chiaramente quella di eludere l'alta complessità di $\sim O(n^3)$ che ha la risoluzione con metodi diretti. Altra motivazione potrebbe essere quella di non conoscere direttamente A, ma solo l'applicazione:

$$v \to A \cdot v$$

(si pensi, anche se rappresenta un caso *non* lineare, ai metodi di discesa a gradiente che ottimizzano funzioni non immediatamente calcolabili o anche solo esprimibili).

Abbiamo quindi che un buon metodo iterativo dovrà:

1. Costare meno di $O(n^3)$ ad ogni passaggio (altrimenti sarebbe inutile rispetto ai metodi diretti), e quindi richiedere solo prodotti di matrici con vettori, o risoluzione di sistemi lineari favorevoli (triangolari, diagonali, ecc...);
2. Data una certa **accuratezza** posta come obiettivo, impiegare un numero ragionevole di iterazioni per raggiungerla.

1.2.1 Metodi di punto fisso

L'idea è quella di partire da $Ax - b = 0$ e di riscrivere come un'equazione equivalente in forma:

$$x = Hx + c, \quad H \in \mathbb{C}^{n\times m},\ c \in \mathbb{C}^n$$

Facciamo alcuni esempi:

1. Si sceglie una matrice $G \in \mathbb{C}^{n\times n}$ invertibile e si considera:

   $$x = x \cdot G(Ax - b) = (I - GA)x + Gb$$

   cioè:

   $$H = I - GA, \quad c = Gb$$

2. Si scompone A come:

   $$A = A_1 + A_2$$

   per cui:

   $$Ax = b \ \Leftrightarrow\ (A_1 + A_2)c = b \ \Leftrightarrow\ A_1x = -A_2x + b \ \Leftrightarrow\ x = -A_1^{-1}A_2x + A_1^{-1}b$$

   cioè ancora:

   $$H = -A_1^{-1}A_2, \quad c = A_1^{-1}b$$

Una volta trovata un'equazione di punto fisso $x = Hx + c$, quindi, si considera il seguente metodo iterativo:

$$\begin{cases} x^{(0)} \text{ dato} \\ x^{(k+1)} = Hx^{(k)} + c, \quad k = 1, 2, 3, ... \end{cases}$$

partendo da $x^{(0)}$ come dato a priori (sarà la soluzione che vorremo raffinare).

Vediamo che infatti:

Definizione 1.1: Matrice di iterazione

La matrice H di un equazione di punto fisso $x = Hx + c$ viene detta matrice di iterazione.

Per avere una verifica della validità dei metodi di punto fisso, enunciamo il seguente teorema:

Teorema 1.2: Validità dei metodi di punto fisso

Il metodo di punto fisso dato da:

$$\begin{cases} x^{(0)} \text{ dato} \\ x^{(n+1)} = Hx^{(k)} + c, \quad k = 1, 2, 3, ... \end{cases}$$

converge $\forall x^{(0)} \in \mathbb{C}^n$ se e solo se $\rho(H) < 1$.

Questo si dimostra prendendo l'equazione di punto fisso $x = Hx + c$, soddisfatta da x soluzione esatta, per cui:

$$x^{(k+1)} - x = Hx^{(k)} + c - (Hx + c) = H(x^{(k)} - x)$$

Possiamo chiamare $e^{(k)} = x^{(k)} - x$, cioè l'errore al passo k, e quindi l'errore al passo $k + 1$ sarà:

$$e^{(k+1)} = He^{(k)} = H^k e^{(0)}$$

Basterà allora prendere il limite:

$$\lim_{k \to +\infty} e^{(k+1)} = \lim_{k \to +\infty} H^k e^{(0)}$$

Perchè questo tenda a 0, basterà imporre $\rho(H) < 1$, in quanto in tal caso:

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

qualsiasi sia l'errore iniziale $e^{(0)}$ (e quindi la scelta di soluzione iniziale $x^{(0)}$) □

In particolare, possiamo ricordare che se $|H| < 1 \implies \rho(H) < 1$, quindi può risultare verificare questa condizione, che è sì più stretta ma anche più facile da verificare. Ricordiamo che non vale assolutamente il contrario, cioè $|H| > 1 \not\implies \rho(H) > 1$. uff uff

Di contro, vale anche la condizione $|\det(H)| \geq 1 \implies \rho(H) > 1$, che come prima non si inverte in $|\det(H)| < 1 \not\implies \rho(H) < 1$.

1.2.2 Velocità di convergenza

Guardando alla dimostrazione del teorema 10.2, possiamo osservare che il raggio spettrale $\rho(H)$ ci dà un informazione anche riguardo alla **velocità di convergenza** del metodo di punto fisso.

Infatti, si avrà che vale:

$$\frac{|e^{(k)}|}{|e^{(0)}|} \leq |H^k|$$

Dall'algebra lineare, si ha che quando k è abbastanza grande, $H^k \approx \rho(H)^k$, almeno per norme indotte, in quanto:

$$\lim_{k \to +\infty} \sqrt[k]{|H^k|} = \rho(H)$$

Questo ci dice che se si hanno 2 metodi di punto fisso con matrici di iterazione H_1 e H_2 tali che:

$$\rho(H_1) < \rho(H_2) < 1$$

allora il primo metodo converge più velocemente del secondo, è dall'ulteriore ipotesi < 1, entrambi convergono.

Inoltre, si può stimare il numero di iterazioni k necessarie a raggiungere un certo valore dell'errore:

$$\frac{|e^{(k)}|}{|e^{(0)}|} = \frac{|x^{(k)} - x|}{|e^{(0)}|} \leq \delta$$

infatti, in tal caso basterà imporre:

$$\rho(H)^k \leq \delta \implies k \geq \frac{\log(\delta)}{\log(\rho(H))}$$

1.2.3 Criteri di stop

Molto spesso non è pratico decidere un errore e fare le k iterazioni che dovrebbero portare a tale errore (in quanto non è scontato conoscere $\rho(H)$), ma bensì si preferisce definire un **criterio di stop** per la terminazione dell'algoritmo raggiunte determinate condizoni.

Queste condizioni possono essere:

1. Si può restringere il residuo:
$$\frac{|r^{(k)}|}{|b|} < \delta$$
2. Si può restringere direttamente la variazione di errore:
$$|x^{k+1} - x^k| < \delta$$

1.3 Metodi di Jacobi e Gauss-Seidel

Vediamo i metodi più famosi di questo tipo, detti metodi di **Jacobi** e **Gauss-Seidel**.

L'idea è di scomporre la matrice A come:

$$A = D - E - F$$

con D diagonale, E l'opposta della triangolare inferiore a diagonale nulla e F l'opposta della triangolare superiore a diagonale nulla. scrivile Varrà quindi:

$$Ax = b \Leftrightarrow (D - E - F)x = b$$

1.3.1 Metodo di Jacobi

Il metodo di Jacobi consiste nel riscrivere quanto trovato come:

$$Dx = (E+F)x + b \implies x = D^{-1}(E+F)x + D^{-1}b$$

cioè trovare un equazione di punto fisso con:

$$H = D^{-1}(E+F), \quad c = D^{-1}b$$

e quindi applicare l'algoritmo:

$$\begin{cases} x^{(0)} \text{ dato} \\ x^{(k+1)} = D^{-1}(E+F)x^{(k)} + D^{-1}b = D^{-1}((E+F)x^{(k)} + b) \end{cases}$$

Osserviamo che ad ogni iterazione si calcola un prodotto matrice-vettore e si risolve un sistema diagonale ($O(n)$), in quanto la D si inverte facilmente (è diagonale): scrivile

$$H_J = D^{-1}(E+F) = ...$$

per cui:

$$(H_j)_{ij} = \begin{cases} -\frac{a_{ij}}{a_{ii}}, & i \neq j \\ 0 & i = j \end{cases}$$

Le matrici che descriveranno il passo di Jacobi saranno allora:

$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{i \neq j}^{n} a_{ij} x_j^{(k)} \right)$$

osservando che per calcolare $x_i^{(k+1)}$ servano *tutte* le componenti di $x^{(k)}$, cioè in codice bisogna mantenere due vettori, in quanto non si può sovrascrivere $x^{(k)}$ prima di aver finito di calcolare $x^{(k+1)}$.

metti implementazione

1.3.2 Metodo di Gauss-Seidel

Il metodo di Gauss-Sidel consiste nel riscrivere quanto trovato come:

$$(D-E)x = Fx + b \implies x = (D-E)^{-1}Fx + (D-E)^{-1}b$$

cioè trovare un equazione di punto fisso con:

$$H = (D-E)^{-1}F, \quad c = (D-E)^{-1}b$$

e quindi applicare l'algoritmo:

$$\begin{cases} x^{(0)} \text{ dato} \\ x^{(k+1)} = (D-E)^{-1}Fx^{(k)} + (D-E)^{-1}b \end{cases}$$

Avremo che la matrice di iterazione è:

$$H_{GS} = (D-E)^{-1}F$$

Osserviamo quindi che non si forma H_{GS}, ma ad ogni iterazione si calcola il prodotto matrice-vettore:

$$x^{(k)} \to Fx^{(k)}$$

e si risolve:

$$(D - E)y = Fx^{(k)}$$

Stesso discorso per c, che si trova risolvendo una sola volta il sistema lineare:

$$(D - E)c = b$$

Osserviamo poi che H_{GS} ha come prima colonna il vettore di zeri in quanto F ha anch'essa la prima colonna al vettore di zeri. scrivile! (lui prende F = [0 | f2 ... f_n])

implementa

Vediamo quindi che, preso:

$$x^{(k+1)} = (D - E)^{-1}Fx^{(k)} + (D - E)^{-1}b$$

moltiplicando per $D^{-1}(D - E)$ si ha.

$$x^{(k+1)} = D^{-1}Ex^{(k+1)} + D^{-1}Fx^{(k)} + D^{-1}b$$

Esplicitare una dipendenza di $x^{(k+1)}$ da se stessa potrebbe sembrare poco intuitivo, ma è invece conveniente in quanto ogni elemento di $x^{(k+1)}$ dipende dai soli elementi precedenti, cioè si ha:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), \quad i = 1, ..., n$$

In questo modo si hanno due vantaggi:

1. Non occorre risolvere sistemi triangolari;
2. Per calcolare $x_i^{(k+1)}$ non si ha bisogno di x_h^k per $h < i$, e quindi si possono sovrascrivere le entrate di $x^{(k)}$, e mantenere un unico vettore.

Un'ultima osservazione è che sia Jacobi che Gauss-Seidel hanno come condizione che $a_{ii} \neq 0$, $\forall i$, in quanto altrimenti D diagonale non sarebbe invertibile. Se questa condizione non è soddisfatta, si possono fare delle trasformazioni "indolori" sul sistema per ritrovare la diagonale non nulla (applicare matrici di permutazione e applicare il metodo sul sistema permutato, per trovare poi una soluzione che al limite andrà *de*-permutata).

Ad esempio, si può pensare di applicare Jacobi a:

$$\Pi Ax = \Pi b$$

per trovare una qualche permutazione di x. quale?