1 Lezione del 24-03-25

1.1 Pivoting

L'algoritmo di eliminazione di Gauss che abbiamo definito alla scorsa lezione ha un punto di fallimento nel caso uno degli elementi $a_{ii}^{(i-1)}$ sia $= 0$, o comunque ≈ 0, in quanto vorremmo a quel punto calcolare un moltiplicatore $l_{ji} = \frac{a_{ji}}{a_{ii}} \to$ non ben definito.

In tal caso si può modificare l'algoritmo sfruttando una matrice di permutazione che porti un elemento diverso da zero nella stessa posizione di a_{ii}. Vorremo quindi cercare un indice h tal per cui $a_{hi}^{(i-1)}$ sia di modulo massimo nella sua colonna al di sotto di i, cioè:

$$a_{hi}^{(i-1)} \geq \max_{j=i,\dots,n} |a_{ji}^{(i-1)}|$$

e scambiare la riga i con la riga h. Infatti, se $\det(A) \neq 0$, allora necessariamente esiste un $a_{ji}^{(i-1)} \neq 0$ (altrimenti si ha uno 0 obbligato sulla diagonale, che con la matrice triangolare a blocchi dà $\det(A^{(i-1)}) = 0$). □

Un altra conseguenza di questo approccio è che tutti i moltiplicatori l_{ji} diventeranno ≤ 1. L'algoritmo di Gauss con questa modifica si chiama **eliminazione di Gauss con pivoting parziale** (*parziale* perché ne esistono versioni più sofisticate, che non vedremo).

Osserviamo che ogni scambio di righe equivale a moltiplicare a sinistra per una matrice di permutazione Π_i. Quindi il metodo di Gauss con pivoting può essere rappresentato come:

Algoritmo 1 Eliminazione di Gauss con pivoting parziale

Input: un sistema lineare qualsiasi $Ax = b$
Output: un sistema lineare triangolare superiore $Ux = c$
for $i = 1$ to n **do**
  Trova la matrice Π_i che porta l'elemento di modulo massimo in testa
  $A \leftarrow \Pi_i A$
  for $j = i$ to n **do**
    Calcola il **moltiplicatore** $l_{ji} = \frac{a_{ji}^{(i-1)}}{a_{ii}^{(i-1)}}$
    Aggiungi alla riga j la riga i moltiplicata per l_{ij}
  end for
end for

Vediamo un esempio pratico dell'algoritmo prima di procedere all'implementazione MATLAB. Prendiamo la matrice A e il vettore b:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Nel ridurre la matrice aumentata Ab:

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 1 \\ 4 & 5 & 6 & 2 \\ 7 & 8 & 0 & 3 \end{array}\right)$$

ci accorgiamo che alla prima colonna l'entrata di modulo massimo è 7, di indice 3. Si permutano quindi la prima e la terza riga:

$$\xrightarrow{\Pi_1} \left(\begin{array}{ccc|c} 7 & 8 & 0 & 3 \\ 4 & 5 & 6 & 2 \\ 1 & 2 & 3 & 1 \end{array}\right) \xrightarrow{H_1\Pi_1} \left(\begin{array}{ccc|c} 7 & 8 & 0 & 3 \\ 0 & \frac{3}{7} & 6 & \frac{2}{7} \\ 0 & \frac{6}{7} & 3 & \frac{4}{7} \end{array}\right)$$

nuovamente, l'entrata di modulo massimo è all'indice 3. Si permutano quindi la seconda e la terza riga:

$$\xrightarrow{\Pi_2 H_1\Pi_1} \left(\begin{array}{ccc|c} 7 & 8 & 0 & 3 \\ 0 & \frac{6}{7} & 3 & \frac{4}{7} \\ 0 & \frac{3}{7} & 6 & \frac{2}{7} \end{array}\right) \xrightarrow{H_2\Pi_2 H_1\Pi_1} \left(\begin{array}{ccc|c} 7 & 8 & 0 & 3 \\ 0 & \frac{6}{7} & 3 & \frac{4}{7} \\ 0 & 0 & \frac{9}{2} & 0 \end{array}\right)$$

1.1.1 Implementazione MATLAB del metodo di eliminazione di Gauss con pivoting

Modifichiamo quindi la funzione gauss_decomp() per introdurre il meccanismo di pivoting appena visto:

```
function [A, b] = gauss_decomp(A, b)
    n = height(A);

    if nargin < 2
        b = zeros(n, 1);
    end

    for i = 1:n % i itera sulle diagonali
        % qui fai il pivot
        max_abs = max(abs(A(i:n, i)));
        h = find(abs(A(i:n, i)) == max_abs, 1);
        h = h + i - 1; % max abs si conta da i in poi

        A([i, h], :) = A([h, i], :); % permuta A
        b([i, h]) = b([h, i]); % permuta b

        den = A(i, i);

        for j = (i + 1):n % j itera sulle righe
            mul = A(j, i) / den; % moltiplicatore
            L(j, i) = mul;

            A(j, :) = A(j, :) - A(i, :) * mul;
            b(j) = b(j) - b(i) * mul;
        end
    end
end
```

1.1.2 Fattorizzazione LU con pivoting

Vediamo come ricavare una fattorizzazione LU dal metodo di Gauss modificato con il pivoting. Si ha quindi che la matrice U si evolve come:

$$A \to \Pi_1 A \to H_1\Pi_1 A \to ... \to H_{n-1}\Pi_{n-1}...H_1\Pi_1 A = U$$

mentre per la L dovremo notare che:

$$LU = \Pi A$$

dove la matrice Π rappresenta tutte le permutazioni fatte sulle righe di A.

Si ha quindi che:

- U è la matrice triangolare superiore trovata alla fine del metodo di Gauss con pivoting;
- L è la matrice dei moltiplicatori, a cui però si devono applicare gli scambi delle righe, come segue: se al passo i applico la matrice Π_i, devo applicare lo stesso cambio nelle prime $i-1$ colonne di L, sotto la diagonale.

Vediamo un esempio numerico spieghi il processo di formazione della matrice L e della matrice di permutazione Π. Presa la stessa matrice dell'esempio precedente:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{pmatrix}$$

abbiamo che le permutazioni sono, in sequenza:

$$\Pi_1 : \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \to \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix}, \quad \Pi_2 : \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix} \to \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}$$

o, in forma matriciale:

$$\Pi_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad \Pi_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Calcoliamo quindi L. Π_1 è irrilevante al calcolo di L, quindi la ignoriamo. Vediamo che i primi due moltiplicatori sono $l_{21} = \frac{4}{7}$ e $l_{31} = \frac{1}{7}$, da cui si imposta $L^{(1)}$:

$$L^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{4}{7} & 1 & 0 \\ \frac{1}{7} & 0 & 1 \end{pmatrix}$$

Notiamo quindi che dalla Π_2 dobbiamo scambiare gli elementi sotto la diagonale della prima colonna, quindi:

$$L^{(1)} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{7} & 1 & 0 \\ \frac{4}{7} & 0 & 1 \end{pmatrix}$$

Infine, l'ultimo moltiplicatore $l_{32} = \frac{1}{2}$ non ha ambiguità:

$$L^{(2)} = L = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{7} & 1 & 0 \\ \frac{4}{7} & \frac{1}{2} & 1 \end{pmatrix}$$

Il calcolo di Π deriva invece direttamente studiando la permutazione complessiva data dalle $\Pi_1, ..., \Pi_{n-1}$, in questo caso:

$$\Pi : \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \xrightarrow{\Pi_1} \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix} \xrightarrow{\Pi_2} \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}$$

da cui:

$$\Pi = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Con brevi calcoli si verifica che:

$$LU = \Pi A$$

1.1.3 Implementazione MATLAB completa del metodo di eliminazione di Gauss con pivoting

Vediamo quindi l'implementazione completa, che calcola anche la matrice L e la matrice Π. Notiamo inoltre l'argomento condizionale b, che viene ignorato se non fornito (abbiamo constatato che spesso è così).

```
function [A, b, L, P] = gauss_decomp(A, b)
    n = height(A);

    if nargin < 2
        b = zeros(n, 1);
    end

    L = eye(n); % prepara L
    P = eye(n); % prepara P

    for i = 1:n % i itera sulle diagonali
        % qui fai il pivot
        max_abs = max(abs(A(i:n, i)));
        h = find(abs(A(i:n, i)) == max_abs, 1);
        h = h + i - 1; % max abs si conta da i in poi

        A([i, h], :) = A([h, i], :); % permuta A
        b([i, h]) = b([h, i]); % permuta b
        if i > 1
            L([i, h], 1:(i - 1)) = L([h, i], 1:(i - 1)); % permuta L
        end
        P([i, h], :) = P([h, i], :); % permuta P

        den = A(i, i);

        for j = (i + 1):n % j itera sulle righe
            mul = A(j, i) / den; % moltiplicatore
            L(j, i) = mul;

            A(j, :) = A(j, :) - A(i, :) * mul;
            b(j) = b(j) - b(i) * mul;
        end
    end
end
```

1.1.4 Determinante con pivoting

Possiamo sfruttare la matrice Π per il calcolo del determinante. Si ha infatti dal teorema di Binet-Cauchy (4.1) che:

$$\det(\Pi)\det(A) = \det(\Pi A) = \det(LU) = \det(L)\det(U)$$

e quindi:

$$\det(A) = \det(\Pi)^{-1} \det(L) \det(U)$$

dove $\det(L) = 1$ (triangolare inferiore con diagonale di 1). Si nota poi che $\det(\Pi)^{-1}$ è $(-1)^s$ è il numero di pivot che effettuiamo. A questo punto $\det(U)$ è semplicemente il prodotto degli elementi sulla diagonale (triangolare superiore), cioè:

$$\prod_{i=1}^{n} a_{ii}^{(i-1)} = \prod_{i=1}^{n} u_{ii}$$

e quindi:

$$\det(A) = (-1)^s \prod_{i=1}^{n} a_{ii}^{(i-1)} = (-1)^s \prod_{i=1}^{n} u_{ii}$$

Si può quindi usare il metodo di Gauss, ancora una volta, per il calcolo del determinante di una matrice, con costo pari al costo dell'eliminazione di Gauss ($O(\frac{2}{3}n^3)$), molto meglio dello sviluppo di Laplace! ($O(n!)$).

1.1.5 Implementazione MATLAB del metodo di Gauss per il determinante

In MATLAB si può calcolare il prodotto delle diagonali come `prod(diag(A))` e il segno di una permutazione come `det(P)` (anche se sicuramente esistono approcci più efficienti). Si può quindi realizzare uno script simile al seguente per il calcolo del determinante sfuttando `gauss_decomp()` con permutazioni:

```
function d = gauss_det(A)
    function s = perm_sign(P)
        s = det(P); % ci sono modi piu' efficienti, vale l'esempio
    end

    [U, ~, ~, P] = gauss_decomp(A);
    d = prod(diag(U)) * perm_sign(P);
end
```

1.2 Condizionamento di un sistema lineare

Date $A \in \mathbb{C}^{n \times n}$ e $b \in \mathbb{C}^n$, supponiamo di voler trovare $Ax = b$ ma a causa di errori nei dati o errori di arrotondamento troviamo (con un qualunque metodo) un vettore perturbato $x + \delta x \in \mathbb{C}^n$ che risolve un sistema lineare di per sé perturbato:

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

con δA e δB perturbazioni "piccole" della matrice A e del vettore b, quindi $A + \delta A \in \mathbb{C}^{n \times n}$ e $b + \delta b \in \mathbb{C}^n$

La domanda è, se δA e δb sono piccole, posso concludere che anche δx è relativamente piccolo? Si scopre che la risposta a questa domanda è generalmente no. Prendiamo ad esempio il sistema 2×2:

$$\begin{pmatrix} 1 & -1 \\ 1 & 1.000001 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

da cui x esatto è $\begin{pmatrix} 1 & 0 \end{pmatrix}$. Perturbando b a $\begin{pmatrix} 0.999999 & 1 \end{pmatrix}$, si ha x perturbato a $\begin{pmatrix} -10^{-6} & -1 \end{pmatrix}$, che è chiaramente un cambiamento drastico. Viene da sé che agendo sulla matrice A potremo ottenere effetti anche più drammatici.

Riprendiamo quindi la definizione di errore relativo:

$$\epsilon = \frac{|\delta x|}{|x|}$$

assumendo $\delta A = 0$ con $\det(A) \neq 0$, come nell'esempio precedente, e quindi perturbazioni solo del termine noto, si ha:

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

visto che $Ax = b$. Passando alle norme, si ha che:

$$|\delta x| \leq |A^{-1}| \cdot |\delta b|$$

e inoltre:

$$|Ax| = |b| \implies |A||x| \geq |b| \implies |x| \geq \frac{|b|}{|A|}$$

quindi:

$$\frac{|\delta x|}{|x|} \leq \frac{|A^{-1}| \cdot |\delta b| \cdot |A|}{|b|} = \frac{|\delta b|}{|b|} \cdot |A| \cdot |A^{-1}|$$

dove ci interessa il valore $|A| \cdot |A^{-1}|$, l'unico che non dipende dall'errore assoluto δb.

Definizione 1.1: Numero di condizionamento

Chiamiamo numero di condizionamento di una matrice A, data una certa norma $|\cdot|$, il valore:

$$\mu(A) = |A| \cdot |A^{-1}|$$

Abbiamo quindi che se $\mu(A) >> 1$, allora l'errore relativo può essere molto più grande dell'errore relativo dei dati e il problema si dice *mal condizionato*.