Risoluzione di sistemi di equazioni lineari
Condizionamento
Fattorizzazione LU
Stabilità

Ángeles Martínez Calomardo
amartinez@units.it

Laurea Triennale in Intelligenza Artificiale e Data Analytics
A.A. 2024–2025

Risoluzione di un sistema lineare in MATLAB/OCTAVE

- Il sistema lineare $Ax = b$ si risolve in MATLAB/OCTAVE con il comando `A\b`.
- Se A è una matrice quadrata invertibile generale, l'operatore `\` restituisce la soluzione $x = A^{-1}b$ calcolata con il metodo di eliminazione di Gauss con pivoting (parziale).

Esempio:

```
>> A=[10 -7 0; -3 2 6; 5 -1 5];

>> b=[7; 4; 6];

>> x=A\b
x =
    0
   -1
    1
>> A*x-b
ans =
    0
    0
    0
```

Risoluzione di un sistema lineare in MATLAB/OCTAVE

- Nel caso di termine noto con più colonne, la scrittura $A\backslash B$, con $B = [b_1 b_2 \dots b_m]$ risolve $AX = B$, ovvero gli m sistemi lineari $Ax_i = b_i$, con $X = [x_1 x_2 \dots x_m]$.
- Nel caso di sistemi sovradeterminati (più equazioni che incognite), il comando $\backslash$ calcola automaticamente la soluzione ai minimi quadrati.

Norme matriciali

- Nell'introduzione a MATLAB/OCTAVE abbiamo visto che il comando `norm(v)` calcola la norma del vettore v.
- Analogamente, per calcolare in MATLAB/OCTAVE la norma di una matrice A si usa il comando `norm(A)`.
- Se nessun ulteriore parametro viene specificato tale comando restituisce la norma 2 della matrice ovvero:

$$\|A\|_2 = \sqrt{\rho(A^T A)}$$

dove $\rho(A)$ è il raggio spettrale della matrice A. Altre possibilità sono:

  - norma 1
    $\|A\|_1 = \max_j \sum_{i=1}^n |a_{ij}|$, in MATLAB/OCTAVE `norm(A,1)`;
  - norma infinito
    $\|A\|_\infty = \max_i \sum_{j=1}^n |a_{ij}|$, in MATLAB/OCTAVE `norm(A,inf)`;
  - norma di Frobenius
    $\|A\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^n |a_{ij}|^2}$, in MATLAB/OCTAVE `norm(A,'fro')`;

Norme matriciali

Esempi

```
>> A = [5 -4 2; 1 7 -6; 1 1 9]
A =

    5   -4    2
    1    7   -6
    1    1    9

>> norm(A,1)
 ans =  17

>> norm(A,inf)
 ns =  14

>> norm(A,'fro')
 ans =  14.6287388383278

>> norm(A)
 ans =  12.0560586095913
```

Condizionamento dei sistemi lineari

Consideriamo il sistema di equazioni lineari $Ax = b$, con $A \in \mathbb{R}^{n\times n}$ e $b \in \mathbb{R}^n$.

Sia $\delta x = e = \bar{x} - x$ l'errore sul risultato in seguito ad una perturbazione δb sul termine noto b (per semplicità, assumiamo $\delta A = 0 \in \mathbb{R}^{n\times n}$). Possiamo pensare dunque che il sistema che si risolve sia

$$A(x + \delta x) = b + \delta b \tag{1}$$

Da cui, poichè $Ax = b$ si ha

$$A\delta x = \delta b; \qquad \delta x = A^{-1}\delta b \tag{2}$$

Rispetto ad una qualsiasi norma matriciale indotta da quella vettoriale, seguono le maggiorazioni

$$\|\delta x\| = \|A^{-1}\delta b\| \leq \|A^{-1}\|\,\|\delta b\| \tag{3}$$

$$\|b\| = \|Ax\| \leq \|A\|\,\|x\| \implies \frac{1}{\|x\|} \leq \frac{\|A\|}{\|b\|} \tag{4}$$

Condizionamento di un sistema lineare

Moltiplicando tra di loro la (3) e la (4), si ha infine

$$\frac{\|\delta x\|}{\|x\|} \leq \|A\| \, \|A^{-1}\| \, \frac{\|\delta b\|}{\|b\|}, \tag{5}$$

dove

$$\boxed{\kappa(A) = \|A\| \, \|A^{-1}\|} \tag{6}$$

si chiama *numero di condizionamento* della matrice A.

Esso è sempre ≥ 1, in quanto si ha:

$$1 = \|I\| = \|AA^{-1}\| \leq \|A\| \, \|A^{-1}\| = \kappa(A)$$

Malcondizionamento

Considerando che

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

si ha

$$A\bar{x} = b+\delta b \qquad \Longrightarrow \qquad \delta b = A\bar{x} - b = -r$$

Vediamo dunque che la norma dell'errore relativo e quella del residuo relativo sono legate mediante la relazione:

$$\frac{\|x-\bar{x}\|}{\|x\|} \leq K(A)\frac{\|r\|}{\|b\|}.$$

- Per valori molto grandi di $\kappa(A)$, diciamo per $\kappa(A) > 10^3$, l'errore relativo sulla soluzione può essere molto grande anche se è piccolo l'errore relativo sui dati.
- Vale a dire che *a residuo piccolo può non corrispondere un errore piccolo.* In questi casi si parla di *malcondizionamento* del sistema o della matrice.
- Numeri di condizionamento diversi si hanno in corrispondenza a scelte diverse della norma matriciale.

Numero di condizionamento in MATLAB/OCTAVE

Il numero di condizionamento di una matrice definito come $\|A\|_2 \cdot \|A^{-1}\|_2$ si calcola in MATLAB/OCTAVE con il comando `cond`.

```
>> A = [4 -1 2; 1 3 1; 0 -3 5]
A =

    4   -1    2
    1    3    1
    0   -3    5
>> cond(A)
ans =  2.4249
```

Specificando come secondo parametro del comando `cond` uno tra i seguenti valori: 1, inf, 'fro', si ottiene il numero di condizionamento in norma 1, infinito e di Frobenius, rispettivamente.

```
>> cond(A,1)
ans =  3.7183
>> cond(A,inf)
ans =  3.1549
>> cond(A,'fro')
ans =  3.8379
```

Esempi di matrici malcondizionate

Matrici di Hilbert

- Una classe di matrici malcondizionate è fornita dalle matrici di Hilbert di ordine n i cui elementi sono $h_{ij} = \dfrac{1}{i+j-1}$.
- In MATLAB/OCTAVE tali matrici si creano tramite il comando `hilb(n)`.

```
>> H = hilb(3)
H =

    1.00000   0.50000   0.33333
    0.50000   0.33333   0.25000
    0.33333   0.25000   0.20000

>> cond(H)
ans =  524.06
>> H1 = hilb(10);
>> cond(H1)
ans =  1.6025e+13
```

- Più grande è il condizionamento, meno accurata potrebbe essere la soluzione del sistema lineare.

Esempi di matrici malcondizionate

Matrici di Hilbert

Il seguente script (*scripthilb.m*) crea le matrici di Hilbert di ordine n, per $n = 3, \ldots, 15$ e ne calcola il numero di condizionamento.

```
for  n=3:15
     H=hilb(n);
     k=cond(H);
     fprintf('\n \t  [N ]   %3d          [COND K(H)]   %6.4e  ',n,  k)
end
fprintf('\n')
```

I numeri di condizionamento calcolati sono:

```
    [N]     3          [COND]   5.2406e+02
    [N]     4          [COND]   1.5514e+04
    [N]     5          [COND]   4.7661e+05
    [N]     6          [COND]   1.4951e+07
    [N]     7          [COND]   4.7537e+08
    [N]     8          [COND]   1.5258e+10
    [N]     9          [COND]   4.9315e+11
    [N]    10          [COND]   1.6025e+13
    [N]    11          [COND]   5.2260e+14
    [N]    12          [COND]   1.6776e+16
    [N]    13          [COND]   1.7590e+18
    [N]    14          [COND]   3.0821e+17
    [N]    15          [COND]   4.4333e+17
>>
```

Esempi di matrici malcondizionate

Esercizio

Si generi la matrice di Hilbert H di ordine 12 e si risolva il sistema lineare $Hx = b$, in cui b corrisponde alla soluzione vera $x = [1, \cdots 1]^T$.
Si visualizzino il numero di condizionamento di H e l'errore e il residuo relativo.

Risoluzione esercizio

Riportiamo di seguito lo script (*solhilb.m*) che risolve l'esercizio.

```
n = 12;
xvera = ones(n,1);
H = hilb(n);
kappa = cond(H);
fprintf('\n \t [COND K_2(H)]          %3.3e ',kappa);
b = H* xvera;
x = H\b;
err_rel = norm(x-xvera)/norm(xvera);
fprintf('\n \t [ERRORE RELATIVO]      %3.3e',err_rel);
res_rel = norm(b-H*x)/norm(b);
fprintf('\n \t [RESIDUO RELATIVO]     %3.3e',res_rel);
fprintf('\n ');
```

Commenti all'esercizio

L'esecuzione dello script fornisce i seguenti risultati

```
>> solhilb

    [COND K_2(H)]          1.680e+16
    [ERRORE RELATIVO]      1.641e-01
    [RESIDUO RELATIVO]     1.166e-16
```

Si osserva che:

- il residuo relativo è circa la precisione di macchina mentre l'errore relativo è $1.64 \cdot 10^{-1}$, maggiore di quindici ordini di grandezza rispetto al residuo.
- ciò non è sorprendente se si considera che il numero di condizionamento è molto grande.
- MATLAB/OCTAVE produce il messaggio di warning per indicare che la matrice è molto malcondizionata e quindi i risultati potrebbero non essere attendibili.

Esempi di matrici malcondizionate

Matrice di Vandermonde

Un altro esempio di matrice malcondizionata è la matrice di Vandermonde di ordine n, definita a partire da un vettore $x = x_1, \cdots, x_n$ come $V_{ij} = x_i^{j-1}$.

In MATLAB/OCTAVE tale matrice si crea con il comando `vander(x)`.
Il seguente script *vandercond.m*, dato il vettore $x = 1 : 1/(n-1) : 2$, crea le matrici `vander(x)` per $n = 3, \ldots, 10$ e ne calcola il numero di condizionamento. Inoltre, realizza un grafico semilogaritmico del condizionamento in funzione della dimensione della matrice.

```
vcond=zeros(10,1);
for  n=3:10
     x=1:1/(n-1):2;
     A=vander(x)
     c=cond(A)
     vcond(n)=c;
pause;
end
semilogy(3:10,vcond(3:10),'r-');
```

Esempi di matrici malcondizionate

Matrice di Vandermonde

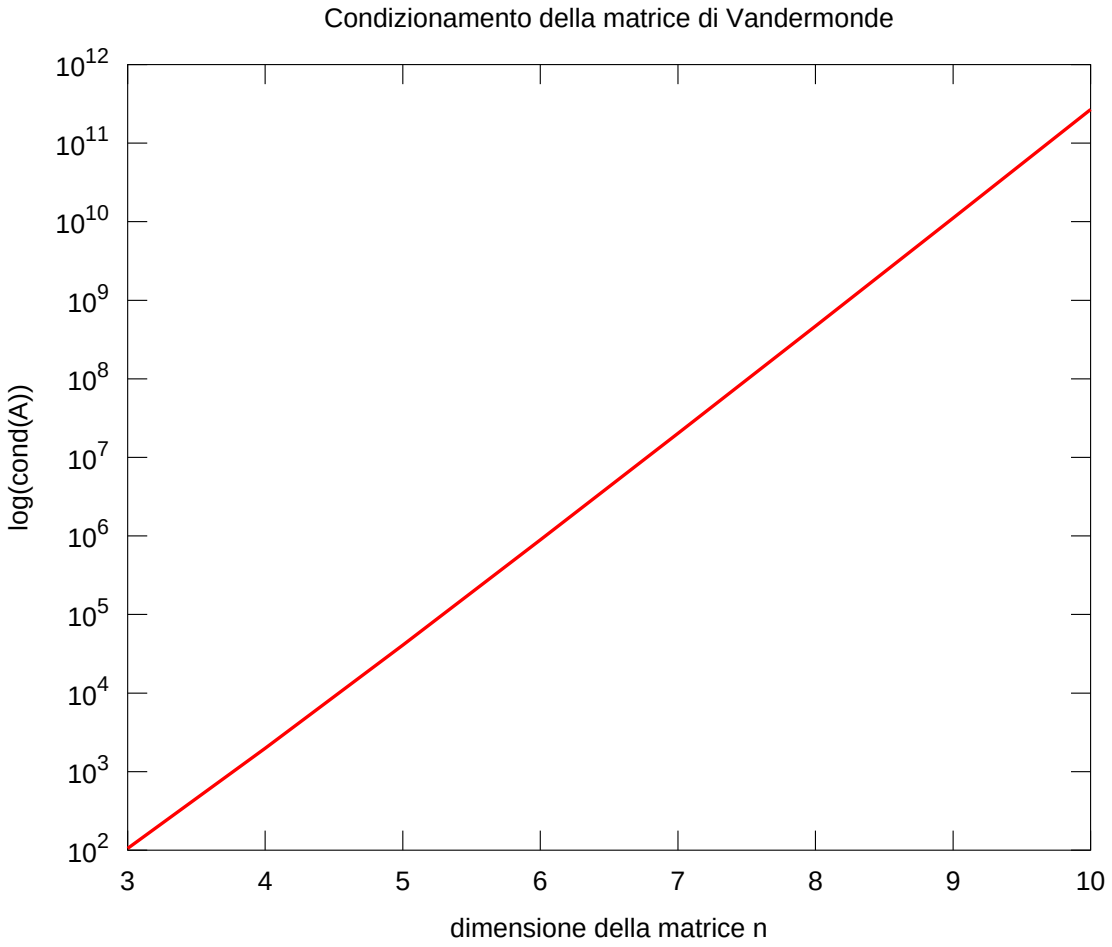

Figura: Grafico in scala semilogaritmica del condizionamento della matrice di Vandermonde al crescere della dimensione n.

Esercizio proposto

Data la matrice

$$A = \begin{pmatrix} 15 & 6 & 8 & 11 \\ 6 & 6 & 5 & 3 \\ 8 & 5 & 7 & 6 \\ 11 & 3 & 6 & 9 \end{pmatrix}$$

1. Si risolvano i sistemi $A_1 x = b_1$, $A_2 x = b_2$, $A_3 x = b_3$ dove $A_i = A^i, i = 1,2,3$ e $b_i = A_i * ones(4,1)$.
2. Si visualizzi il vettore soluzione di ogni sistema con 14 cifre decimali significative.
3. Si spieghino i risultati ottenuti.

Suggerimento: si calcolino, con il comando `eig` gli autovalori di A_i. Si ricordi che per una matrice simmetrica definita positiva (si verifichi che A lo è) $K(A) = \frac{\lambda_{\max}}{\lambda_{\min}}$ e che se $\lambda \in \sigma(A)$ allora $\lambda^p \in \sigma(A^p)$.

Fattorizzazione LU

In MATLAB il comando `lu` calcola la fattorizzazione LU di A, ottenuta mediante eliminazione di Gauss con pivoting parziale per righe.

La sintassi è:

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

con P matrice di permutazione tale che $PA = LU$.

Se il comando `lu` viene chiamato

```
[L,U]=lu(A)
```

allora U è la matrice triangolare superiore ottenuta dal MEG e la matrice L è in realtà $P^{-1}L$.

Dalla fattorizzazione LU alla soluzione del sistema $Ax = b$

Data la matrice A se la sua fattorizzazione LU è stata ottenuta mediante eliminazione di Gauss con pivoting si ha:

$$PA = LU \Longrightarrow PAx = L\underbrace{Ux}_{y} = Pb \Longleftrightarrow \begin{cases} Ly = Pb \\ Ux = y \end{cases}$$

Da un sistema siamo passati a dover risolvere due sistemi, ma più semplici perchè triangolari:

1. $Ly = Pb$ si risolve mediante *sostituzione in avanti*
2. $Ux = y$ si risolve mediante *sostituzione all'indietro*

Eliminazione di Gauss e Fattorizzazione LU

Sia $Ax = b$ con $A \in \mathbb{R}^{n\times n}$ regolare

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}.$$

Supponiamo che ad ogni passo k del processo di eliminazione, con $1 \le k \le n-1$, sia $a_{kk}^{(k-1)} \neq 0$ (elemento *pivot*). Poniamo $A^{(1)} = A$.
Al primo passo (per $k = 1$): Si azzerano tutti gli elementi della 1^a colonna eccetto il primo.
Per azzerare a_{21} si trasforma la 2^a riga di A nel seguente modo:

$$R_2 \leftarrow R_2 - (\ell_{21} R_1), \quad (2^a \text{ riga} - \text{moltiplicatore} \times 1^a \text{ riga}).$$

dove $\ell_{21} = \frac{a_{21}}{a_{11}}$.
Analogamente si possono azzerare tutti gli altri elementi della 1^a colonna:

$$\boxed{\ell_{i1} = \frac{a_{i1}}{a_{11}}} \quad \Longrightarrow \quad R_i \leftarrow R_i - (\ell_{i1} R_1), \quad 2 \le i \le n$$

Eliminazione di Gauss e Fattorizzazione LU

Quindi, dopo il primo passo, la matrice $A^{(1)}$ è stata trasformata, mediante *trasformazioni elementari* (di Gauss) nella matrice:

$$A^{(2)} = \begin{pmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \dots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \dots & \dots & \dots \\ 0 & a_{n2}^{(2)} & \dots & a_{nn}^{(2)} \end{pmatrix}$$

Per $k = 2$, secondo passo. Escludendo la prima riga (rimasta inalterata) e la prima colonna di $A^{(2)}$, riapplichiamo il procedimento alla sua sottomatrice $n-1 \times n-1$.

Proseguendo così, dopo $n-1$ passi si ottiene la matrice *triangolare superiore* $A^{(n)} = U$.
Gli elementi l_{ij} formeranno la matrice triangolare inferiore L, con $l_{ii} = 1, i = 1, \dots, n$, tale che $A = LU$.

Algoritmo di eliminazione di Gauss

L'algoritmo del metodo di eliminazione di Gauss si può schematizzare come segue

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

Esercizio

A partire dallo pseudocodice dell'Algoritmo precedente si scriva una `function` MATLAB/OCTAVE *:* `function[L,U] = lugauss(A)` *che restituisce i due fattori triangolari* L *ed* U*.*

Test della function `lugauss`

Esercizio

Si scriva uno script che definita la matrice

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

e il termine noto $b = (3\ 4\ 3)^T$, *risolva il sistema* $Ax = b$ *(soluzione vera* $x = (1\ 1\ 1)^T$*), richiamando la function* `lugauss` *e risolvendo i due sistemi triangolari usando il comando* \ *di* MATLAB/OCTAVE .

Stabilità dell'algoritmo di eliminazione di Gauss

Pivoting

- La fattorizzazione LU non sempre è accurata a causa degli errori di arrotondamento: il fatto che al passo k l'elemento diagonale $a_{kk}^{(k-1)}$ (pivot) sia *piccolo*, pur non impedendo la conclusione del calcolo della fattorizzazione, **come succederebbe nel caso si incontrasse un pivot nullo**, può comunque comportare gravi perdite di accuratezza.

- L'uso della tecnica del pivoting migliora di molto l'accuratezza della fattorizzazione LU.

- Pivoting parziale: Scegliere come elemento *pivot* il massimo in modulo tra gli elementi nella sottocolonna k.

  Questo garantisce che tutti i moltiplicatori siano in modulo ≤ 1 e impedisce di conseguenza la crescita eccessiva degli elementi nella matrice U nel caso generale.

Stabilità dell'algoritmo di eliminazione di Gauss

Esercizio proposto

Si risolva il sistema lineare $Ax = b$ dove, fissato $\varepsilon = 10^{-14}$,

$$A = \begin{pmatrix} 1 & 1 & -3 \\ 2 & 2-\varepsilon & 4 \\ 1 & 9 & 4 \end{pmatrix}$$

e il termine noto $b = A\bar{x}$ si ottiene dopo aver imposto la soluzione vera pari a $\bar{x} = (1\ 1\ 1)^T$.

Una volta risolto il sistema e ricavata la soluzione approssimata x, si calcoli il residuo relativo $\dfrac{\|b - Ax\|}{\|b\|}$ e l'errore relativo $\dfrac{\|x - \bar{x}\|}{\|\bar{x}\|}$ commentando i risultati ottenuti.

Si ripetano gli stessi passi utilizzando poi la fattorizzazione LU con pivot fornita dalla function MATLAB/OCTAVE `[L,U,P] = lu(A)`.