Calcolo Numerico: Laboratorio

Gianna Del Corso <gianna.delcorso@unipi.it>
Federico Poloni <federico.poloni@unipi.it>

28 Ottobre 2021

1 Fattorizzazione LU

Ricordate come funziona il passo k-esimo di fattorizzazione LU:

$$E^{(k)} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & * & 1 & 0 & 0 \\ 0 & 0 & * & 0 & 1 & 0 \\ 0 & 0 & * & 0 & 0 & 1 \end{pmatrix}, \quad A^{(k)} = \begin{pmatrix} * & * & * & * & * & * \\ 0 & * & * & * & * & * \\ 0 & 0 & A_{33}^{(k)} & * & * & * \\ 0 & 0 & A_{43}^{(k)} & * & * & * \\ 0 & 0 & A_{53}^{(k)} & * & * & * \\ 0 & 0 & A_{63}^{(k)} & * & * & * \end{pmatrix}$$

dovete moltiplicare $A^{(k)}$ per una matrice $E^{(k)}$ della forma indicata in modo che gli elementi in posizione $(k+1,k),(k+2,k),\dots,(n,k)$ diventino zero. La matrice L è il prodotto delle matrici $(E^{(k)})^{-1}$, che non va veramente calcolato ma, grazie alle proprietá delle matrici elementari di Gauss, si può determinare senza sforzi ulteriori con la relazione

$$L_{ij} = \frac{A_{ij}^{(j)}}{A_{jj}^{(j)}}, \quad \forall i \geq j.$$

Esercizio 1. Scrivete una `function [L,U]=my_lu(A)` che calcoli la fattorizzazione LU di una matrice A.

```
octave:47> M=4*eye(5)+ones(5,5)
M =

  5  1  1  1  1
  1  5  1  1  1
  1  1  5  1  1
  1  1  1  5  1
  1  1  1  1  5
```

```
octave:48> [L,U]=my_lu(M)
octave:49> L*U-M
ans =

 Columns 1 through 3:

   0.00000000000000e+00 0.00000000000000e+00 0.00000000000000e+00
   0.00000000000000e+00 0.00000000000000e+00 0.00000000000000e+00
   0.00000000000000e+00 0.00000000000000e+00 -8.88178419700125e-16
   0.00000000000000e+00 0.00000000000000e+00 2.22044604925031e-16
   0.00000000000000e+00 0.00000000000000e+00 2.22044604925031e-16

 Columns 4 and 5:

   0.00000000000000e+00 0.00000000000000e+00
   0.00000000000000e+00 0.00000000000000e+00
   0.00000000000000e+00 0.00000000000000e+00
   0.00000000000000e+00 2.22044604925031e-16
   2.22044604925031e-16 0.00000000000000e+00
```

Perché qui sopra non avete degli zeri esatti?

Esercizio 2. Scrivete una `function x=sys_solve(A,b)` che risolva un sistema lineare generico utilizzando la fattorizzazione LU, `inf_solve` e `sup_solve`.

Potete utilizzare le funzioni della volta scorsa per risolvere sistemi triangolari, oppure copiate e incollate le seguenti[1]

```
function x=inf_solve(L,b)
%risolve un sistema con L triangolare inferiore
  s=size(L);
  n=s(1);
  x=b; %x "vettore di accumulatori"
  for i=1:n
    x(i)=x(i)/L(i,i);
    x(i+1:n)=x(i+1:n) - L(i+1:n,i)*x(i);
  endfor
endfunction
```

```
function x=sup_solve(U,b)
%risolve un sistema con U triangolare superiore
  s=size(U);
  n=s(1);
  x=b; %x "vettore di accumulatori"
  for i=n:-1:1
```

[1] In alternativa scaricate `inf_solve.m` e `sup_solve.m` dalla pagina del corso tra le soluzioni della terza lezione di laboratorio

```
    x(i)=x(i)/U(i,i);
    x(1:i-1)=x(1:i-1) - U(1:i-1,i)*x(i);
  endfor
endfunction
```

Esercizio 3 (facoltativo). Capite come funziona questa versione di `inf_solve` e `sup_solve`. Notate che i calcoli che esegue non sono identici alla versione della scorsa lezione! Ricordate che la formula è

$$x_i = \frac{b_i - \sum_{j=0}^{i-1} L_{ij} x_j}{L_{ii}}, \quad x = 1 \dots n.$$

Esercizio 4. Si considerino le matrici $L, M \in \mathbb{R}^{n \times n}$ definite nel seguente modo

$$l_{ij} = \begin{cases} 2 & i = j \\ -1 & i = j+1 \\ 0 & \text{altrimenti} \end{cases} \qquad m_{ij} = \begin{cases} 1 & i = j \\ -2 & i = j+1 \\ 0 & \text{altrmenti} \end{cases}$$

a) Si dimostri che le due matrici L ed M sono invertibili,

b) Si calcoli $\mu_\infty(L), \mu_\infty(M)$.

c) Si scriva una funzione Matlab (di complessità lineare) `y=inf_bisolve(A, b)` che, presa una matrice bidiagonale inferiore ed un vettore colonna `b`, resituisca il vettore `y` tale che $A\mathbf{y} = \mathbf{b}$.

d) Per $n = 10^3$ sia `x=rand(n,1)`, `b_1=L*x` e `b_2=M*x`. Utilizzando la funzione scritta si calcolino le soluzioni dei sistemi $L\mathbf{y_1} = \mathbf{b_1}$ e $M\mathbf{y_2} = \mathbf{b_2}$, che in aritmetica esatta produrrebbero le soluzioni $\mathbf{y_1} = \mathbf{x}$ e $\mathbf{y_2} = \mathbf{x}$.

Si riportino gli errori relativi in norma infinito, cioè

$$\frac{\|\mathbf{y_i} - \mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty}, \quad i = 1, 2.$$

Si motivi il risultato alla luce del teorema sul condizionamento dei sistemi lineari.

e) Si ripeta lo stesso ragionameto ma scegliendo `x=ones(n,1)`. Cosa si ottiene in questo caso? Motivate la risposta.

2 Esperimenti numerici

Esercizio 5. Testare i seguenti metodi di soluzione di un sistema lineare $Ax = b$:

- La funzione `sys_solve` appena scritta
- Il comando di Matlab `x=inv(A)*b`, che calcola la matrice inversa e la moltiplica per b.

- Il comando di Matlab `x=A\b`: il comando `\` (backslash) serve proprio per risolvere sistemi lineari, ed è basato sulla fattorizzazione LU con pivoting parziale (ad ogni passo scelgo l'emento della colonna da azzerare più grande in modulo), più stabile della semplice fattorizzazione LU.

Per testarli, utilizzate le seguenti matrici:

- La matrice `M1=9*eye(10)+ones(10)`, che è a predominanza dominante.
- Una matrice generata da `M2=rand(10)`, con elementi casuali — può essere abbastanza mal condizionata! Potete controllare il condizionamento con il comando `cond(M2)`.
- La matrice data da `M3=M1;M3(9,1:9)=0`, che ha una riga quasi tutta di zeri che rende la sottomatrice principale 9×9 singolare (e quindi non ammette fattorizzazione LU).
- La matrice data da `M4=M2;M4(9,1:9)=sum(M4(1:8,1:9))`:

  ```
  * * * * * * * * * *
  * * * * * * * * * *
  * * * * * * * * * *
  * * * * * * * * * *
  * * * * * * * * * *
  * * * * * * * * * *
  * * * * * * * * * *
  * * * * * * * * * *
  * * * * * * * * * *
  * * * * * * * * * *
  ```

  gli elementi in rosso sono ognuno la somma degli otto elementi che stanno direttamente sopra di esso; quindi la sottomatrice principale 9×9 è singolare (ma lavorando con i numeri floating point...).
- La matrice data da `M5=M2;M5(10,1:10)=sum(M5(1:9,1:10))`: l'ultima riga è la somma delle 9 precedenti, quindi la matrice è singolare (ma lavorando con i numeri floating point...).
- La matrice di Hilbert `hilb(10)`. Riuscite a spiegare il numero di cifre significative ottenute in base al suo numero di condizionamento, che potete calcolare con `cond(hilb(10))`?

Ponete `v=transpose(1:10)` (vettore conentente i numeri da 1 a 10 in ordine); per ognuna di queste matrici, calcolate `b`i`=M`i`*v` (per $i = 1, \dots, 6$), e andate a risolvere il sistema `M`i`*x=b`i. La soluzione esatta di questo sistema è v; di quanto si discostano le soluzioni calcolate?

Esercizio 6 (facoltativo). Guardate la fattorizzazione LU di M4. `U(9,9)` è molto piccolo; perché? `U(10,10)` è molto grande; perché?

Esercizio 7. Prendete

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Quanto vale la matrice `A=L*U`? La matrice A soddisfa le ipotesi sufficienti del teorema visto a lezione per l'esistenza della fattorizzazione LU? Che risultato ritorna la vostra routine `my_lu(A)` su questa matrice?

2.1 Calcolare il determinante

Una delle applicazioni immediate delle fattorizzazione LU è il calcolo del determinante. Dalle proprietà di L nella fattorizzaziona $A = LU$ segue immediatamente che $\det(L) = 1$ (perchè?). Questo ci permette di calcolare

$$\det(A) = \det(U).$$

Ricordando che U è triangolare superiore concludiamo che il determinante è il prodotto degli elementi diagonali (e quindi molto facile da calcolare!).

Esercizio 8. Realizzare una funzione `d = mydet2(A)` che calcoli il determinante di una matrice A utilizzando la sua fattorizzazione LU.

3 Risoluzione di sistemi lineari strutturati

Utilizzando il comando `triu(ones(n))` si può generare una matrice triangolare superiore con tutte le componenti uguali a 1 che denotiamo con A_n.

Esercizio 9. Scrivete una `function x=sys_solve_ones(b)` che preso $\mathbf{b} \in \mathbb{R}^n$, calcola la soluzione del sistema $A_n\mathbf{x} = \mathbf{b}$, con $A_n =$ `triu(ones(n))`. La funzione deve avere costo computazionale $O(n)$.

Si consideri il sistema tridiagonale

$$A = \begin{bmatrix} \alpha_1 & \gamma_1 & & 0 & \\ \beta_1 & \alpha_2 & \gamma_2 & & \\ & \beta_2 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \gamma_{n-1} \\ 0 & & & \beta_{n-1} & \alpha_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_n \end{bmatrix}$$

Per questo tipo di sistemi la soluzione può essere ottenuta con $O(n)$ operazioni anzichè le $O(n^3)$ richieste dal metodo di Gauss per matrici dense. Infatti posto

$$a_1 = \alpha_1$$

$$b_i = \beta_i / a_i$$
$$a_{i+1} = \alpha_{i+1} - b_i\gamma_i \quad i = 1, \ldots, n-1$$

se tutti gli $a_i \neq 0$ vale

$$LU = \begin{bmatrix} 1 & & & & 0 \\ b_1 & 1 & & & \\ & b_2 & 1 & & \\ & & \ddots & \ddots & \\ 0 & & & b_{n-1} & 1 \end{bmatrix} \begin{bmatrix} a_1 & c_1 & & 0 & \\ & a_2 & c_2 & & \\ & & a_3 & \ddots & \\ & & & \ddots & c_{n-1} \\ 0 & & & & a_n \end{bmatrix}$$

Esercizio 10. Si scriva una funzione di costo lineare `function x=tri_solve(A, y)` per risolve un sistema lineate con A matrice tridiagonale.

Utilizzando i comandi `tic, toc` si confrontino i tempi di esecuzione ottenuti utilizzando la funzione appena scritta e la funzione `sys_solve(A, y)` scritta la scorsa volta per matrici dense. Quale delle due funzini risulta più veloce?

Esercizio 11. Sia $A \in \mathbb{R}^{n\times n} = (a_{i,j})$ definita da

$$a_{i,j} = \begin{cases} \beta \text{ se } i = j; i \neq n \\ 1 \text{ se } j = n, 1 \leq i \leq n; \\ \alpha \text{ se } i = n, j = 1; \\ 0 \text{ altrimenti.} \end{cases}$$

Per $n = 4$ si ottiene

$$A = \begin{bmatrix} \beta & 0 & 0 & 1 \\ 0 & \beta & 0 & 1 \\ 0 & 0 & \beta & 1 \\ \alpha & 0 & 0 & 1 \end{bmatrix}.$$

1. Si determini $s > 0$ tale che A è invertibile $\forall \alpha, \beta$ con $|\beta| > s$ e $|\alpha| < s$.
2. Si determini per quali valori di α e β la matrice A ammette fattorizzazione LU.
3. Per tali valori si determini la fattorizzazione LU.
4. Si determini per quali valori di α e β la matrice risulta singolare.
5. Si scriva una funzione Matlab che, presi in ingresso α, β ed il vettore $\mathbf{b}$ implementi un metodo a costo lineare in termini di operazioni aritmetiche ed occupazione di memoria per la risoluzione del sistema lineare $A\mathbf{x} = \mathbf{b}$.

4 Facoltativo: eliminazione di Gauss "senza L"

Si possono riorganizzare i calcoli in modo che il calcolo di L venga effettuato implicitamente; questo probabilmente è equivalente a come avete visto l'eliminazione di Gauss lo scorso anno ad algebra lineare. Lavoriamo sulle equazioni anziché sulle matrici:

$$\begin{array}{llll} a_{11}x_1 & +a_{12}x_2 & +a_{13}x_3 & = b_1. \\ a_{21}x_1 & +a_{22}x_2 & +a_{23}x_3 & = b_2. \\ a_{31}x_1 & +a_{32}x_2 & +a_{33}x_3 & = b_3. \end{array}$$

Cerchiamo di eliminare x_1 dalla seconda e terza equazione:

$$\begin{array}{llll} a_{11}x_1 & +a_{12}x_2 & +a_{13}x_3 & = b_1. \\ a_{21}x_1 & +a_{22}x_2 & +a_{23}x_3 - \frac{a_{21}}{a_{11}}\left(a_{11}x_1 + a_{12}x_2 + a_{13}x_3\right) & = b_2 - \frac{a_{21}}{a_{11}}b_1. \\ a_{31}x_1 & +a_{32}x_2 & +a_{33}x_3 - \frac{a_{31}}{a_{11}}\left(a_{11}x_1 + a_{12}x_2 + a_{13}x_3\right) & = b_3 - \frac{a_{31}}{a_{11}}b_1. \end{array}$$

Quindi reiteriamo, fino a trasformare A in una matrice triangolare.

Prendiamo come riferimento per l'analisi il primo passo. Come vengono modificati quindi la matrice dei coefficienti (A) e il termine noto (b) che stiamo tenendo in memoria? Sulla matrice A facciamo le stesse operazioni che abbiamo fatto la scorsa lezione nel calcolo della fattorizzazione LU. Come abbiamo visto, la quantità che dobbiamo sottrarre ad A è una matrice di rango 1:

`A(2:3,1:3) = A(2:3,1:3) -` $\begin{bmatrix} \frac{a_{21}}{a_{11}} \\ \frac{a_{31}}{a_{11}} \end{bmatrix} * \begin{bmatrix} a_{11} & a_{12} & a_{13} \end{bmatrix}$

In più, dobbiamo effettuare un calcolo simile anche su b:

`b(2:3) = b(2:3) -` $\begin{bmatrix} \frac{a_{21}}{a_{11}} \\ \frac{a_{31}}{a_{11}} \end{bmatrix} * b_1$

Possiamo anche interpretare i calcoli effettuati in termini di prodotti di matrici: nel calcolo della fattorizzazione LU ottenevamo L^{-1} come prodotto di matrici parziali $L_n L_{n-1} \cdots L_1$, ognuna della forma

$$\begin{bmatrix} I & & \\ & 1 & \\ & v & I \end{bmatrix},$$

ora quello che facciamo è effettuare subito il prodotto con la matrice parziale

$$\begin{bmatrix} 1 & & \\ -\frac{a_{21}}{a_{11}} & 1 & \\ -\frac{a_{31}}{a_{11}} & 0 & 1 \end{bmatrix}$$

in modo da trasformare il sistema nel sistema equivalente (che ha la stessa soluzione)

$$L_1 A x = L_1 b,$$

Al termine degli n passi, abbiamo trasformato il sistema in un sistema equivalente ma con matrice dei coefficienti triangolare, e questo lo sappiamo risolvere.

Esercizio 12 (facoltativo). Scrivere una `function x=sys_solve2(A,b)` che risolva un sistema lineare con il metodo qui sopra.