Fattorizzazione LU ed eliminazione gaussiana in Matlab

Alvise Sommariva

Università degli Studi di Padova
Dipartimento di Matematica Pura e Applicata

2 giugno 2023

L'ambiente Matlab utilizza varie strategie per risolvere i sistemi lineari. Il comando per fare questa operazione é mldivide (acronimo per *matrix left division*) o alternativamente il backslash.

A tal proposito, digitando "help \" ricaviamo

```
>> help \
 \    Backslash or left matrix divide.
    A\B is the matrix division of A into B, which is roughly the
    same as INV(A)*B , except it is computed in a different way.
    If A is an N-by-N matrix and B is a column vector with N
    components, or a matrix with several such columns, then
    X = A\B is the solution to the equation A*X = B. A warning
    message is printed if A is badly scaled or nearly singular.
    A\EYE(SIZE(A)) produces the inverse of A.
    ...
    See also ldivide, rdivide, mrdivide.

    Reference page for mldivide
    Other functions named mldivide
>>
```

In pratica tale comando serve per poter risolvere sistemi lineari del tipo $Ax = b$.

Il comando mldivide é molto complicato. Matlab distingue tra varie tipologie di matrici e poi determina il metodo piú adatto. Vediamo il seguente esempio

```
>> spparms('spumoni',1);
>> A=gallery('poisson',3); % matrice di tipo Poisson (formulazione
     compatta)
>> A=full(A)  % matrice di tipo Poisson (display componente per
     componente)
ans =
     4    -1     0    -1     0     0     0     0     0
    -1     4    -1     0    -1     0     0     0     0
     0    -1     4     0     0    -1     0     0     0
    -1     0     0     4    -1     0    -1     0     0
     0    -1     0    -1     4    -1     0    -1     0
     0     0    -1     0    -1     4     0     0    -1
     0     0     0    -1     0     0     4    -1     0
     0     0     0     0    -1     0    -1     4    -1
     0     0     0     0     0    -1     0    -1     4
>> b=ones(size(A,1),1); % termine noto
>> x=mldivide(A,b); % risolve Ax=b
sp\: bandwidth = 3+1+3.
sp\: is A diagonal? no.
sp\: is band density (0.647059) > bandden (0.5) to try banded solver? yes
sp\: is LAPACK's banded solver successful? yes
>>
```

Non si capisce molto, ma si intuisce che per questa matrice *sparsa* A, ovvero con molti zeri, Matlab preventivamente ragiona sulla tipologia della matrice e poi sceglie il metodo che viene reputato piú adatto.

Se la matrice é piena, ovvero con poche componenti nulle, dal sito How does the backslash operator work when A is full? si deduce che svolga le seguenti operazioni.

```
 >> x = A \ b;
%  This is pseudo-code for how full \ works:
if size(A,1) == size(A,2)    % A is square
  if isequal(A,tril(A))    % A is lower triangular
    x = A \ b;    % This is a simple forward substitution on b
  elseif isequal(A,triu(A))    % A is upper triangular
    x = A \ b;    % This is a simple backward substitution on b
  else
    if isequal(A,A')    % A is symmetric
      [R,p] = chol(A);
      if (p == 0)    % A is symmetric positive definite
        x = R \ (R' \ b);    % a forward and a backward substitution
        return
      end
    end
    [L,U,P] = lu(A);    % general, square A
    x = U \ (L \ (P*b));    % a forward and a backward substitution
  end
else    % A is rectangular
    [Q,R] = qr(A);
    x = R \ (Q' * b);
end
```

Vediamo un esempio di uso del comando backslash.

```
>> A=[1 3 5; 2 4 5; 1 1 1];
>> % verifichiamo che det(A) non e' nullo
>> % (ovvero la matrice A e' invertibile)
>> det(A)
ans =
    -2
>> % definiamo il termine noto "b"
>> b=[1 1 1]';
>> % calcoliamo la soluzione di A*x=b;
>> % attenzione che e' "\" e non "/".
>> x=A\b
x =
     2
    -2
     1
>> A*x
ans =
     1
     1
     1
>> % Quindi visto che "b" e' il vettore colonna [1 1 1]'
>> % abbiamo che "x" e' la soluzione richiesta.
```

Se é disponibile una fattorizzazione PA=LU, é possibile risolvere il sistema lineare $Ax = b$ in $O(n^2)$ operazioni mediante l'applicazione dei metodi di sostituzione in avanti e all'indietro.

Il comando `lu` calcola la corrispettiva fattorizzazione di una matrice A con n righe e colonne, ottenuta mediante eliminazione di Gauss con pivoting.

La chiamata

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

determina le matrici

- $L = (l_{i,j}) \in \mathbb{R}^{n \times n}$ triangolare inferiore, ovvero tale che $l_{i,j} = 0$ se $j > i$ con $l_{i,i} = 1$,
- $U = (u_{i,j}) \in \mathbb{R}^{n \times n}$ triangolare superiore, ovvero tale che $u_{i,j} = 0$ se $j < i$,
- $P = (p_{i,j}) \in \mathbb{R}^{n \times n}$ di permutazione, ovvero con esclusivamente un valore non nullo e pari a 1 per ogni riga e colonna,

cosicchè $PA = LU$.

Vediamo un primo esempio. Digitiamo questo codice sul nostro ambiente di lavoro.

```
>> A=[4 -2 -1 0; -2 4 1 0.5; -1 1 4 1; 0 0.5 1 4];
>> [L,U,P]=lu(A);
>> L % L triang. inf. con elementi diagonali uguali a 1
L =
    1.0000         0         0         0
   -0.5000    1.0000         0         0
   -0.2500    0.1667    1.0000         0
         0    0.1667    0.2500    1.0000
>> U % %  triangolare sup.
U =
    4.0000   -2.0000   -1.0000         0
         0    3.0000    0.5000    0.5000
         0         0    3.6667    0.9167
         0         0         0    3.6875
>> % P di permutazione
>> P
P =
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
>> % P e' la matrice identica (nessuno "scambio di righe")
>> norm(P*A-L*U)
ans =
     0
>> % Numericamente P*A=L*U, ovvero P*A-L*U=0, ovvero norm(P*A-L*U)=0.
```

Vediamo un secondo esempio. Digitiamo il codice sul nostro ambiente di lavoro.

```
>> A=[ 1    78    33; 62    10    65; 90    23    88];
>> A
A =
     1    78    33
    62    10    65
    90    23    88
>> [L,U,P]=lu(A);
>> L % L triang. inf. con elementi diagonali uguali a 1
L =
    1.0000         0         0
    0.0111    1.0000         0
    0.6889   -0.0752    1.0000
>> U % U triangolare sup.
U =
   90.0000   23.0000   88.0000
         0   77.7444   32.0222
         0         0    6.7851
>> % P di permutazione
>> P
P =
     0     0     1
     1     0     0
     0     1     0
>> P*A
ans =
    90    23    88
     1    78    33
    62    10    65
>> % P*A e' ottenuta scambiando righe di A (1 -> 2, 2 -> 3, 3 -> 1).
>> norm(P*A-L*U)
ans =
     0
>> % Numericamente P*A=L*U, ovvero P*A-L*U=0, ovvero norm(P*A-L*U)=0
```

Supponiamo sia

- $A \in \mathbb{R}^{n\times n}$, con $\det(A) \neq 0$,
- $b \in \mathbb{R}^n$,

In tali ipotesi, esiste un unico $x^* \in \mathbb{R}^n$ che risolva il sistema lineare $Ax = b$.
Sotto queste ipotesi si può provare che se $PA = LU$, allora necessariamente

- $\det(P) = \pm 1$, $P^{-1} = P^T$,
- $\det(L) = 1$
- $\det(U) \neq 0$.

Quindi, posto $Pb = c$, visto che P ínvertibile, abbiamo

$$Ax = b \Leftrightarrow PAx = Pb \Leftrightarrow LUx = Pb \Leftrightarrow LUx = c.$$

Di conseguenza

- posto $y = Ux$, y è soluzione del sistema triangolare inferiore $Ly = c$;
- calcolato y, la soluzione x del sistema $Ax = b$ è pure soluzione del sistema triangolare superiore $Ux = y$;

Quindi per risolvere $Ax = b$, nota la fattorizzazione `PA=LU`:

1. si valuta `c=P*b`;
2. si risolve `L*y=c`;
3. si risolve `U*x=y`.

In generale per risolvere il sistema lineare, utilizzando backslash solo per risolvere convenientemente i sistemi triangolari, possiamo introdurre la seguente funzione metodo_LU per risolvere i sistemi lineari (che scarichiamo nell'ambiente di lavoro).

```
function x=metodo_LU(A,b)
[L,U,P]=lu(A); % fattorizzazione PA=LU
c=P*b;
y=L\c; % sistema triangolare inferiore
x=U\y; % sistema triangolare superiore
```

Vogliamo paragonarlo con il comando di backslash su alcune matrici di speciale interesse introdotte nella gallery di Matlab.

```
>> help gallery
  gallery Higham test matrices.
   [out1,out2,...] = gallery(matname, param1, param2, ...)
   takes matname, a string that is the name of a matrix family, and
   the family's input parameters.
     ...
   chebvand    Vandermonde-like matrix for the Chebyshev polynomials.
     ...
   minij       Symmetric positive definite matrix MIN(i,j).
   moler       Moler matrix -- symmetric positive definite.
     ...
   poisson     Block tridiagonal matrix from Poisson's equation ...
     ...
   tridiag     Tridiagonal matrix (sparse).
     ...
>>
```

Implementiamo il seguente confronto tra metodo_LU e backslash. Scarichiamo il file test_metodo_LU nella directory di lavoro.

```
function test_metodo_LU
warning off; % non scrive "warnings"
for n=5:5:20
    A=gallery('chebvand',n);b=rand(n,1); condA=condest(A);

    % risoluzione con "metodo_LU"
    tic; x_LU=metodo_LU(A,b);  t_metodo_LU=toc; % tempo impiegato

    % risoluzione con "backslash"
    tic; x_backslash=A\b; t_backslash=toc; % tempo impiegato

    % errore norma 2 soluzione
    err=norm(x_LU-x_backslash)/norm(x_backslash);

    % errore relativo soluzione backslash
    residuo_backslash=norm(b-A*x_backslash)/norm(b);

    % errore fattorizzazione LU
    [L,U,P]=lu(A); errLU=norm(P*A-L*U);

    % stampe
    fprintf('\n n: %3.0f cond: %1.3e err LU: %1.3e',n,condA,errLU);
    fprintf('\n err. LU vs backs.: %1.3e res. relativo: %1.3e',...
        err,residuo_backslash);
    fprintf('\n  tempo impiegato: LU: %1.3e backslash: %1.3e \n',...
        t_metodo_LU,t_backslash);
end
fprintf('\n')
```

Il codice

- mediante la gallery di Matlab, definisce una particolare matrice di Vandermonde di ordine `n` e la assegna ad `A` e di seguito un vettore `b` random, calcolando infine un approssimazione del numero di condizionamento `condA` della matrice `A`;
- calcola un approssimazione `x_LU` soluzione del sistema $Ax = b$ con la routine `metodo_LU` e "stima" il tempo impiegato per determinare l'approssimazione;
- calcola un approssimazione `x_backslash` soluzione del sistema $Ax = b$ con il comando `backslash` e "stima" il tempo impiegato per determinare l'approssimazione;
- valuta l'errore relativo `err` in norma 2, tra le due approssimazioni;
- valuta il *residuo relativo*

$$\texttt{residuo\_backslash} = \frac{\|b - A * \texttt{x\_backslash}\|_2}{\|b\|_2}$$

  che è un indicatore di quanto `x_backslash` è soluzione del sistema lineare;
- calcola quanto accurata è la fattorizzazione LU mediante `errLU=norm(P*A-L*U)`;
- stampa alcuni risultati di rilievo.

Se lanciamo test_metodo_LU nella directory di lavoro, otteniamo quali risultati

```
>> test_metodo_LU

 n:   5 cond: 1.493e+03 err LU: 3.165e-16
 err. LU vs backs.: 0.000e+00 res. relativo: 4.509e-15
  tempo impiegato: LU: 1.709e-04 backslash: 7.191e-05

 n:  10 cond: 4.846e+07 err LU: 7.295e-16
 err. LU vs backs.: 0.000e+00 res. relativo: 1.886e-10
  tempo impiegato: LU: 1.999e-04 backslash: 7.516e-05

 n:  15 cond: 1.586e+12 err LU: 1.303e-15
 err. LU vs backs.: 0.000e+00 res. relativo: 5.375e-08
  tempo impiegato: LU: 1.512e-04 backslash: 6.478e-05

 n:  20 cond: 5.301e+16 err LU: 1.590e-15
 err. LU vs backs.: 0.000e+00 res. relativo: 1.486e-02
  tempo impiegato: LU: 2.225e-04 backslash: 2.802e-04

>>
```

che mostrano che

- i tempi di calcolo per risolvere problemi di piccola dimensione siano meno di millesimi di secondo;
- la routine metodo_LU e backslash calcolano la stessa approssimazione(!);
- purtroppo il condizionamento delle matrici non permette di calcolare accuratamente la soluzione (nonostante lo sia la fattorizzazione PA=LU).

Partendo dalla demo precedente definiamo la routine test_metodo_LU2 per cui:

- utilizziamo il ciclo for: for n=200:200:1000 invece di for n=5:5:20;
- sostituiamo A=gallery('minij',n); a A=gallery('chebvand',n).

Scarichiamo tale file test_metodo_LU2.m

La gallery relativa a minij definisce la matrice $A = (a_{i,j})$ simmetrica e definita positiva tale che $a_{i,j} = \min(i, j)$. Lanciando tale routine, ricaviamo:

```
>> test_metodo_LU2

 n: 200 cond: 8.040e+04 err LU: 0.000e+00
 err. LU vs backs.: 5.819e-16 res. relativo: 3.890e-14
  tempo impiegato: LU: 3.922e-03 backslash: 1.944e-03

 n: 400 cond: 3.208e+05 err LU: 0.000e+00
 err. LU vs backs.: 3.851e-16 res. relativo: 1.313e-13
  tempo impiegato: LU: 7.169e-03 backslash: 1.860e-03

 n: 600 cond: 7.212e+05 err LU: 0.000e+00
 err. LU vs backs.: 5.199e-16 res. relativo: 3.332e-13
  tempo impiegato: LU: 1.275e-02 backslash: 4.595e-03

 n: 800 cond: 1.282e+06 err LU: 0.000e+00
 err. LU vs backs.: 5.058e-16 res. relativo: 5.925e-13
  tempo impiegato: LU: 2.992e-02 backslash: 1.200e-02

 n: 1000 cond: 2.002e+06 err LU: 0.000e+00
 err. LU vs backs.: 7.805e-16 res. relativo: 7.416e-13
  tempo impiegato: LU: 5.314e-02 backslash: 2.064e-02

>>
```

Dall'esperimento numerico su `minij` si vede che:

- Le matrici questa volta non sono *piccole*, ma comunque il tempo di calcolo e' ancora nell'ordine dei centesimi/millesimi di secondo;
- I condizionamenti delle matrici non sono piccoli, ma molto inferiori a quelli visti nell'esempio precedente e i risultati ottenuti sono relativamente accurati.
- Le fattorizzazioni LU sono molto accurate.
- Il metodo LU e il `backslash` di Matlab non forniscono gli stessi risultati ma sono comunque entrambi *simili*, peró questa volta il `backslash` sembra più rapido.
- Nonostante le matrici abbiano comunque condizionamenti rilevanti, le soluzioni sono calcolate *relativamente bene*.

Esercizio (risolto)

Si scriva uno script `test_metodo_LU3` *che*

1. *ponga* `n=100;`
2. *definisca, mediante un opportuno ciclo-for nella variabile delle righe e di seguito uno nella variabile delle colonne, contenente una opportuna istruzione condizionale, la matrice* $A = (A(s,t)) \in \mathbb{R}^{n\times n}$ *che ha per componenti*

$$A(s,t) = \begin{cases} s/t, & 1 \le s \le t \le n \\ t/s, & 1 \le t \le s \le n \end{cases}$$

3. *definisca, mediante un opportuno ciclo-for, il termine noto* $b = (b(i))_i \in \mathbb{R}^{n\times 1}$ *(vettore colonna)*

$$b_i = \begin{cases} -1, & i \text{ dispari} \\ 1, & i \text{ pari} \end{cases}$$

   Si osservi che per sapere se un numero k *é pari si digita* `rem(k,2)`*. Se il risultato é* 0 *allora il numero é pari, altrimenti é dispari.*

4. *Si risolva* $Ax = b$ *con la routine* `metodo_LU`*.*
5. *Si ponga* `err=norm(b-A*x)/norm(b)` *e si stampi il valore del cosidetto residuo relativo* `err` *in notazione esponenziale con una cifra prima della virgola e una cifra dopo la virgola.*

La soluzione a quanto richiesto é lo script test_metodo_LU3 definito da:

```
function test_metodo_LU3
n=100;
% definizione matrice A
for s=1:n
    for t=1:n
        if s <= t
            A(s,t)=s/t;
        else
            A(t,s)=t/s;
        end
    end
end
% definizione termine noto
for k=1:n
        if rem(k,2) == 0 %indice pari
            b(k,1)=+1;
        else
            b(k,1)=-1;
        end
end
% soluzione sistema lineare
x=metodo_LU(A,b);
% valutazione residuo relativo
err=norm(b-A*x)/norm(b);
fprintf('\n \t residuo relativo: %1.1e \n',err);
```

Dal codice otteniamo

```
>> test_metodo_LU3
   residuo relativo: 4.3e-16
>>
```

Il risultato puó variare con processore e versione di Matlab (basta sia molto piccolo!).

Esercizio (1)

Il seguente pseudocodice

function [L,A]=fattorizzazione_LU(A)

for *k=1,...,n-1* **do**
  $l_{k,k} = 1$ **for** *i=k+1,...,n* **do**
    $l_{i,k} = \frac{a_{i,k}}{a_{k,k}}$
    for *j=k,...,n* **do**
      $a_{i,j} = a_{i,j} - l_{i,k} a_{k,j}$
    end
  end
end
$l_{n,n} = 1$

- *si noti che i cicli for hanno l'indice "*`k`*" che varia da* 1 *a* $n-1$ *e "*`i`*" che varia da* $k+1$ *a* n *(nel programmare il codice fare bene attenzione agli indici!),*
- *calcola la fattorizzazione LU di una matrice* $A = (a_{i,j})$ *data in input,*
- *in output offre le matrici triangolari* $L = (l_{i,j})$ *e* $U = (u_{i,j})$ *che corrisponde alla matrice A alla fine del processo (ovvero la cosidetta* fattorizzazione LU sul posto*).*

Lo si implementi in Matlab mediante la function `fattorizzazione_LU` *ricordando che "n" è il numero di righe e colonne di A.*

Esercizio (2)

Si scriva uno script `demo_eliminazione_gaussiana` *che definita la matrice*

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

e il termine noto

$$b = \begin{pmatrix} 3 \\ 4 \\ 3 \end{pmatrix}$$

calcoli la soluzione del sistema lineare $Ax = b$*, risolvendo i due sistemi triangolari citati in precedenza (con P uguale alla matrice identica), mediante il comando* $\backslash$ *di Matlab.*

La soluzione corretta è

$$x = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$