Proposta esercizi 3
Metodo gaussiano e fattorizzazione LU

Calcolo Numerico (LT Informatica), a.a. 2019-2020
Dipartimento di Scienze Matematiche Fisiche e Informatiche
Università degli Studi di Parma

Docente:
Prof.ssa Chiara Guardasoni

Tutor:
Dott. Ariel S. Boiardi *

1 Matrici elementari e di eliminazione

Dati due vettori non nulli $u, v \in \mathbb{R}^n$ e $\sigma \in \mathbb{R}$ definiamo la *matrice elementare*

$$E(u, v, \sigma) = I - \sigma u v^t \tag{1}$$

Esercizio 1: Scrivere una semplice function `Elem` MATLAB che implementi la costruzione delle matrici elementari come in (1).

Hint. Attenzione che nella trattazione teorica i vettori sono in genere intesi come vettori colonna, nell'implementazione sarebbe meglio controllare la `size`.

Se $\sigma v^t u \neq 1$, la matrice elementare $E(\sigma, u, v)$ è invertibile e l'inversa è sempre una matrice elementare $E(\tau, u, v)$ con

$$\tau = \frac{\sigma}{\sigma v^t u - 1}.$$

Esercizio 2: Non è difficile dimostrare l'asserto sopra. In ogni caso proviamolo con qualche esempio numerico usando la function `Elem`. In particolare provare con u, v vettori pseudo-random di interi e di reali.

Dato un vettore

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^t$$

una matrice elementare $E_k(\mathbf{x}) = E(1, u_{k,\mathbf{x}}, e_k)$ con

$$u_{k,\mathbf{x}} = \frac{1}{x_k} \begin{bmatrix} 0 & \cdots & 0 & x_{k+1} & \cdots & x_n \end{bmatrix}^t$$

*Per dubbi e segnalazioni: `arielsurya.boiardi@studenti.unipr.it`

e e_k è il k-esimo vettore della base canonica è detta *matrice di eliminazione*. Questo nome è giustificato dal fatto che

$$E_k\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_k & 0 & \cdots & 0 \end{bmatrix}^t,$$

cioè elimina tutte le componenti sotto la k-esima.

Esercizio 3: Implementare in una function `Elim` con input un vettore x e un intero k la costruzione delle matrici di eliminazione.

Anche in questo caso non è difficile ricavare un'espressione analoga per la matrice inversa

Esercizio 4: Verificare numericamente che $E_k^{-1} = E(-1, u_{k,\mathbf{x}}, e_k)$ e scrivere una function `invElim` che calcoli esplicitamente l'inversa delle matrici di eliminazione.

2 Il metodo di eliminazione gaussiana

Data una matrice

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

possiamo costruire il metodo di eliminazione gaussiana mediante le matrici di eliminazione costruite nella sezione precedente.

Iniziamo costruendo la prima matrice di eliminazione relativa alla prima colonna di A, E_1. Applicandola ad A, otteniamo una matrice

$$E_1A = \begin{bmatrix} a_{11} & a_{12}^{(2)} & \cdots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix}$$

in cui gli elementi dal secondo i poi della prima colonna sono stati azzerati, e il resto della matrice è stata modificata.

Esercizio 5: Data la matrice

```
A =
     1     1     0     3
     2     1     1     1
    -1     2     3    -1
     3    -1     1     2
```

costruire la prima matrice di eliminazione per la prima colonna e confermare quanto asserito sopra in merito alla struttura della matrice $E_1 A$.

Applicando poi la seconda matrice di eliminazione E_2 otteniamo una matrice in cui gli elementi dal terzo in poi della seconda colonna sono nulli, mentre la prima colonna resta invariata:

$$E_2 E_1 A = \begin{bmatrix} a_{11} & a_{12}^{(2)} & a_{13}^{(3)} & \cdots & a_{1n}^{(3)} \\ 0 & a_{22}^{(2)} & a_{23}^{(3)} & \cdots & a_{2n}^{(3)} \\ 0 & 0 & a_{33}^{(3)} & & a_{3n}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & a_{n3}^{(3)} & \cdots & a_{nn}^{(3)} \end{bmatrix}$$

Esercizio 6: Recuperata la matrice dell'esercizio precedente, confermare numericamente la struttura della matrice $E_2 E_1 A$.

Procedendo in questo modo fino alla penultima colonna (l'ultima non va toccata), se tutto va liscio, in $n-1$ passi otteniamo una matrice triangolare superiore U

$$E_{n-1} \cdots E_2 E_1 A = U \tag{2}$$

Abbiamo quindi realizzato l'algoritmo di eliminazione gaussiana attraverso una successione di matrici di eliminazione.

Osserviamo subito che:

- Ogni passo dell'algoritmo è realizzato da una matrice di eliminazione la cui costruzione richiede di dividere per l'elemento diagonale corrispondente $a_{kk}^{(k)}$ (l'apice indica il passo, i pedici la posizione nella matrice); tali elementi sono spesso chiamati *pivot*. Qualora si incontri un pivot nullo l'algoritmo non funziona e si rende necessario introdurre delle permutazioni.
- Abbiamo visto che le matrici di eliminazione sono invertibili, (2) risulta quindi equivalente a
$$A = E_1^{-1} E_2^{-1} \cdots E_{n-1}^{-1} U \tag{3}$$

Esercizio 7: Completare l'algoritmo di eliminazione gaussiana sulla matrice A dei precedenti esercizi e

- Confermare la struttura triangolare inferiore dell'ultima matrice ottenuta,
- Costruire il prodotto indicato in (3) delle inverse delle matrici di eliminazione utilizzate utilizzando la function `invElim`. Che struttura ha la matrice $E_1^{-1} E_2^{-1} \cdots E_{n-1}^{-1}$?

Hint. Un modo efficace per calcolare i prodotti che ci interessano senza dover salvare tutte le matrici di eliminazione e le rispettive inverse è di inizializzare due matrici, ad esempio `U=A` ed `L=eye`. Ad ogni passo caoliamo `E` la matrice di eliminazione che ci serve e `H` la sua inversa e aggiorniamo con i prodotti a destra e a sinistra come da eq. (2) e (3)

```
U = E * U
L = L * H
```

3 Fattorizzazione LU

Esercizio 8: Organizzare i passaggi della sezione precedente in una function `myLU` che prendendo in input una matrice `A` ne restituisce la fattorizzazione in `[L,U]` triangolari inferiore e superiore rispettivamente. Cosa succede sulla matrice

```
B =
          1     1     0     3
          2     1    -1     1
         -1     2     3    -1
          3    -1    -1     2
```

e perché?

Hint. Il debugger di MATLAB permette di interrompere l'esecuzione in punti particolari del codice. Per capire meglio cosa sta succedendo so può fermare l'esecuzione e vedere i risultati intermedi (ad esempio dopo ogni passo dell'algoritmo di eliminazione...)

Se al passo $k < n - 1$ dell'algoritmo di eliminazione troviamo un pivot nullo $a_{kk}^{(k)}$ la costruzione della matrice di eliminazione E_k incontra una divisione per zero e l'algoritmo non funziona più

Esercizio 9: Se non lo avete già fatto, nella function `myLU` aggiungete un controllo che ferma l'esecuzione con un messaggio di errore utile nel caso venga trovato un pivot nullo.

3.1 Fattorizzazione LU con permutazioni

Supponiamo di essere arrivati al passo k mediante $k-1$ eliminazioni $E_1, \dots, E_{k-1}$ alla matrice

$$E_{k-1}\cdots E_1 A = \begin{bmatrix} a_{11} & a_{12}^{(2)} & \cdots & a_{1k-1}^{(k)} & a_{1k}^{(k)} & a_{1k+1}^{(k)} & \cdots & a_{1n}^{(k)} \\ 0 & a_{22}^{(2)} & & \vdots & \vdots & a_{2k+1}^{(k)} & \cdots & a_{2n}^{(k)} \\ 0 & 0 & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & & a_{k-1k-1}^{(k)} & a_{k-1k}^{(k)} & a_{k-1k+1}^{(k)} & & \vdots \\ 0 & 0 & \cdots & 0 & 0 & a_{kk+1}^{(k)} & \cdots & a_{kn}^{(k)} \\ \vdots & \vdots & & \vdots & a_{k+1k}^{(k)} & a_{k+1k+1}^{(k)} & & \vdots \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & a_{nk}^{(k)} & a_{nk+1}^{(k)} & \cdots & a_{nn}^{(k)} \end{bmatrix}.$$

Se la matrice non è singolare (che è l'unico caso in cui possiamo essere interessati ad una fattorizzazione...) la k-esima colonna ha un elemento non nullo sotto la diagonale, che potremo spostare sulla diagonale mediante una opportuna matrice di permutazione P_{k-1} costruita con il metodo spiegato nella precedente esercitazione.

La matrice $P_{k-1}E_{k-1}\cdots E_1$ ha quindi un pivot non nulla sulla k-esima colonna, possiamo pertanto costruire una opportuna matrice di eliminazione che azzeri gli elementi sotto la diagonale.

Esercizio 10: Eseguire sulla matrice `B` dell'esercizio 8 uno per volta i passi dell'algoritmo di riduzione: quando trovate il pivot nullo costruite una matrice di permutazione che porti sulla diagonale un elemento non nullo (da sotto la diagonale!).

Hint. Ricordiamo che la matrice che scambia gli elementi di indici i e j in in vettore colonna (di lunghezza maggiore di i e j) mediante moltiplicazione a sinistra si ottiene semplicemente scambiando le righe i e j della matrice identità.

Anche in questo caso l'applicazione delle matrici di permutazione non tocca le colonne precedenti: la permutazione riguarda solo elementi sotto la diagonale che nelle colonne precedenti sono tutti nulli...

Inserendo opportunamente le matrici di permutazione otteniamo una successione di trasformazioni

$$E_{n-1}P_{n-1}\cdots E_2P_2E_1P_1A = U$$

possiamo poi raccogliere tutte le matrici di permutazione e, invertendo tutte le eliminazioni, ottenere una fattorizzazione LU analoga alla (3)

$$\underbrace{P_{n-1}\cdots P_2P_1}_{P}A = \underbrace{P_1E_1^{-1}P_1P_2E_2^{-1}P_2\cdots P_{n-1}E_{n-1}^{-1}P_{n-1}}_{L}U. \tag{4}$$

Siccome è meglio non avere sulla diagonale elementi piccoli, una correzione della fattorizzazione in LU è quella del pivoting, in cui non solo si applica una permutazione per togliere i pivot nulli ma anche per spostare sulla diagonali l'elemento di modulo massimo. Una volta capito come funziona il caso per pivot nulli non è difficile generalizzare...

Esercizio 11: Implementare in una function `myLUP` la fattorizzazione LU con pivoting spiegata sopra e restituisce in output tre matrici `L,U,P`. Successivamente

- Verificare che la function funziona con le matrici `A` e `B` degli esercizi precedenti, ad esempio calcolando

```
>> [L,U, P] = myLUP(A)
>> L * U - P * A
```

- Verificare che, nonostante le permutazioni, le matrici ottenute hanno struttura triangolare inferiore e superiore rispettivamente.

Hint. Osserviamo che la matrice P può essere pensata come precodizionatore.

Negli esempi precedenti sembra che fa la fattorizzazione con permutazione sia meno precisa di quella senza (vi siete accorti?); questo però è abbastanza naturale quando si lavora con sistemi piccoli

Esercizio 12: Provare ad eseguire il seguente codice

```
>> A = rand(500);
>> [L,U] = myLU(A);
>> max(max(L*U - A))

ans =

        -------

>> [L,U,P] = myLUP(A);
>> max(max(L*U - P*A))

ans =

        -------
```

Se `myLU` dà errore provate con un'altra matrice pseudo-random e sperate in bene. Commentare il risultato.

Esercizio 13: Ripetere l'esercizio precedente con una matrice di Hilbert, commentare il risultato.

4 Altri algoritmi per la fattorizzazione LU

L'algoritmo di eliminazione gaussiana non è il più efficiente per il calcolo della fattorizzazione LU. Un algoritmo con la stessa applicabilità del metodo gaussiano senza permutazioni ma decisamente più efficiente è quello di Doolittle che è stato illustrato a lezione

$l_{ii} = 1;$
for $k = 0, \ldots, n$ **do**
  for $i = k, \ldots, n$ **do**
    $u_{ki} = a_{ki} - \sum_{p=1}^{k-1} l_{kp} u_{pi}$
  end
  for $i = k+1, \ldots, n$ **do**
    $l_{ik} = \frac{1}{u_{kk}} \left[a_{ik} - \sum_{p=1}^{k-1} l_{ip} u_{pk} \right]$
  end
end

Algorithm 1: Algoritmo Doolittle

Esercizio 14: Implementare l'algoritmo di Doolittle in una function MATLAB `DooLU` o altro nome che la differenzi dalle precedenti e da quelle di libreria. Confrontare quindi i tempi di esecuzione della function `DooLU` e `myLU` su matrici di varie dimensioni (random, Hilbert,...).

Hint. Date un occhio all'`help` per `tic` e `toc`.