

Corso di Laurea in Ingegneria Biomedica

Laboratorio di Algebra Lineare Numerica

A.A. 2019/2020 – I Ciclo

Esercitazione 2

Creare una cartella <cognome> in C: dove verranno salvati i file creati nella sessione di lavoro.
Appena entrati in MATLAB posizionarsi in <cognome>.
Risolvere in ambiente MATLAB i seguenti esercizi.

RISOLUZIONE DI SISTEMI LINEARI:

OPERATORE DIVISIONE A SINISTRA

Per la soluzione di sistemi lineari in MATLAB si può usare l'operatore \backslash o divisione a sinistra:

```
>> x = A \ b
```

che calcola la soluzione x del sistema $Ax=b$ con il **metodo di eliminazione di Gauss con pivoting in generale**. Nei casi particolari di sistemi con matrice A triangolare inferiore o superiore risolve con il metodo di sostituzione in avanti o all'indietro rispettivamente.

FATTORIZZAZIONE di matrici $A=LU$, $A=R'R$

La funzione MATLAB che esegue la fattorizzazione LU con pivoting e' $[L,U,P]=lu(A)$, l'output di questa funzione sono le matrici triangolari L ed U ed una matrice di permutazione P tali che $PA=LU$.

La fattorizzazione di Cholesky è $[R,p]=chol(A)$, dove R è matrice triangolare superiore. Il parametro p è zero se A è definita positiva, non zero altrimenti.

FATTORIZZAZIONE di matrici $A=LU$ e RISOLUZIONE $Ly=b$

```
[L,U,P,y] = function fatt_lu([A b])
```

La m-function presa in input una matrice A quadrata di ordine n ed il termine noto vettore colonna b , restituisce in output i fattori L ed U della fattorizzazione di Gauss di A con pivotaggio a perno massimo per colonne, la matrice di permutazione P e il vettore y soluzione di $Ly = b$.

1. Esplorare mediante l'*help* di Matlab i seguenti comandi per la creazione di matrici speciali:

- **zeros** matrice nulla
- **ones** matrice con elementi pari a 1
- **eye** matrice identità
- **diag** matrice diagonale
- **tril** matrice triangolare inferiore
- **triu** matrice triangolare superiore
- **hilb** matrice di Hilbert
- **vander** matrice di Vandermonde
- **rand** matrice di numeri casuali

Calcolare l'indice di condizionamento delle matrici di Hilbert, di Vandermonde e di numeri casuali al variare dell'ordine $n=5,10,15$, utilizzando la built-in function **cond(A,p)** di MATLAB sia in norma 2 (default), che in norma 1 ($p=1$) ed infinito ($p='inf'$).

2. Dopo aver definito nello script **solv.m** i seguenti sistemi lineari nella forma matriciale $Ax=b$

$$\begin{cases} 2x_1 + 4x_2 - 2x_3 - 2 = 0 \\ 4x_1 + 9x_2 - 3x_3 - 8 = 0 \\ -2x_1 - x_2 + 7x_3 - 10 = 0 \end{cases} \quad \begin{cases} x_1 + 4x_2 - x_3 + x_4 = 2 \\ 2x_1 + 7x_2 + x_3 - 2x_4 = 16 \\ x_1 + 4x_2 - x_3 + 2x_4 = -15 \\ 3x_1 - 10x_2 - 2x_3 + 5x_4 = -15 \end{cases} \quad \begin{cases} 12x_1 - 15x_2 = 8 \\ -4x_1 + 5x_2 = 10 \end{cases}$$

stabilire con un algoritmo se i sistemi hanno soluzione e in caso affermativo, risolverli con il metodo della matrice inversa (utilizzando la funzione **inv** di Matlab) e della divisione a sinistra (utilizzando l'operatore **** che calcola la soluzione con il metodo di eliminazione di Gauss: $x = A \backslash b$). Quale dei due metodi risulterà computazionalmente più efficiente?

Sostituire poi la risoluzione diretta di Gauss $x = A \backslash b$ con la risoluzione tramite fattorizzazione LU utilizzando **[L,U,P]=lu(A)**.

3. CONDIZIONAMENTO DEI SISTEMI LINEARI. Nello script file **ex3_hilb.m**, creare le matrici $A=\text{hilb}(n)$, per $n=5,10,20$. Costruire poi un vettore colonna $x_{ex}=\text{ones}(n,1)$, che rappresenta la soluzione esatta, e calcolare il vettore dei termini noti $b=Ax_{ex}$.

a. Per ogni n risolvere i sistemi lineari $Ax=b$ e $Ax=\bar{b}$, dove \bar{b} è una perturbazione del vettore b ($\bar{b}=b+0.01*\text{rand}(n,1)$).

b. Per ogni n stampare l'errore relativo sulla soluzione $\frac{\|x-\bar{x}\|_2}{\|x\|_2}$, l'errore sui dati iniziali

$$\frac{\|b-\bar{b}\|_2}{\|b\|_2} \text{ e il condizionamento della matrice (con } \text{cond}(A)\text{)}.$$

Scrivere uno script simile (**ex3_rand.m**) per ripetere l'esercizio per le matrici di numeri casuali $A=\text{rand}(n)$, $n=5,10,20$.

Scrivere uno script simile (**ex3_A.m**) per ripetere l'esercizio per la matrice

$$A = \begin{bmatrix} 10 & 1 & 4 & 0 \\ 1 & 10 & 5 & -1 \\ 4 & 5 & 10 & 7 \\ 0 & -1 & 7 & 9 \end{bmatrix}$$

4. Il determinante di una matrice triangolare è uguale al prodotto dei suoi elementi diagonali. Utilizza questo fatto per sviluppare una function MATLAB (**detA=calcDet(A)**) che calcoli il determinante di una matrice A arbitraria di dimensione n utilizzando la fattorizzazione LU.

5. Realizzare una function MATLAB **UTriSol.m** che risolva il sistema lineare con matrice dei coefficienti triangolare superiore A e vettore dei termini noti b ($Ax=b$) mediante il metodo di sostituzione all'indietro:

function x = UTriSol(A,b)

.....

6. Costruire una funzione MATLAB dal nome *LU_solve_gen.m* per il calcolo della soluzione di una generale equazione matriciale $AX=B$, con X,B matrici, che utilizza la fattorizzazione LU.

Richiamarla in uno script *ex6.m* per il calcolo dell'inversa A^{-1} delle seguenti matrici

$$\begin{bmatrix} 3 & 5 & 7 \\ 2 & 3 & 4 \\ 5 & 9 & 11 \end{bmatrix}, \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}$$

Confrontare i risultati con la built-in function MATLAB `inv(A)`.

7. Utilizzando le function *UtriSol(A,b)* e *fatt_lu([A b])* calcolare la soluzione dei sistemi seguenti

a) $A = \begin{bmatrix} 3 & 5 & 7 \\ 2 & 3 & 4 \\ 5 & 9 & 11 \end{bmatrix}$,

b) $v = [3 \ 1 \ -2 \ 3 \ 4 \ 0 \ 0 \ 0]$, $A = \text{gallery('circul',v)}$

c) $A = \text{hilb}(12) + \text{eye}(12)$

Con termine noto dato da $b = A * \text{ones}(n,1)$, con n dimensione matrice A.

Calcolare l'indice di condizionamento della matrice e l'errore relativo sulla soluzione.

Per il caso c): perturbare l'elemento $b(1)$ del termine noto del 2% e risolvere nuovamente il sistema lineare. Calcolare l'errore relativo sulla soluzione e confrontarlo con l'errore relativo sul termine noto.

8. Realizzare l'algoritmo di Thomas $[x]=\text{thomas}(b,a,c,d)$, (variante del metodo di Gauss per matrici tridiagonali), con b,c vettori sotto e sopra-diagonali, a vettore diagonale principale e d termine noto.

Sperimentare il metodo `tridiag()` per la risoluzione del sistema lineare con matrici nella forma $b = c = -\text{ones}(n-1,1)$, $a = 4 * \text{ones}(n,1)$, $d(1)=3$, $d(2:n-1)=2$, $d(n)=3$, al variare di $n=100,1000,1500$.

Confrontare i tempi di esecuzione (si faccia uso delle funzioni di Matlab `tic` e `toc` per calcolare il tempo) tra la risoluzione tramite `tridiag()` e `fatt_lu([A d]) + UtriSol()`. La matrice piena A si ottiene da:

$$A = \text{diag}(b,-1) + \text{diag}(a) + \text{diag}(c,1);$$

9. Costruire una matrice B simmetrica di ordine $n=3000$, nel seguente modo:

$$A = \text{rand}(3000);$$

$$B = A' * A;$$

Verificare che B sia definita positiva facendo uso dell'algoritmo di fattorizzazione di Cholesky fornito dalla built in function $R = \text{chol}(B)$.

In caso positivo, risolvere il sistema lineare $B x = y$, con y termine noto scelto in maniera tale che la soluzione del sistema lineare sia il vettore unitario,

- a) sfruttando la fattorizzazione di Cholesky ($\text{chol}(\mathbf{B})$) e operatori \backslash per la risoluzione del sistema triang. inferiore e superiore.
- b) sfruttando la fattorizzazione di Gauss ($[\mathbf{L}, \mathbf{U}, \mathbf{P}] = \text{lu}(\mathbf{B})$) e l'operatore \backslash come nell'esercizio precedente.
- Confrontare i tempi dei due metodi a) e b) per verificare che b) ha un costo computazionale circa doppio rispetto ad a).

10. Scrivere una function per la soluzione di un sistema lineare con matrice ortogonale

function [x]=RisolviOrtagonale(A,b)

dove A è una matrice ortogonale di ordine n e b è il termine noto. Realizzare uno script *ex10.m* che permetta all'utente di risolvere il sistema $Ax=b$ con

$A = [1, 0, 0, 0; 0, \cos(\theta), -\sin(\theta), 0; 0, \sin(\theta), \cos(\theta), 0; 0, 0, 0, 1]$
(θ scelto dall'utente in radianti)

e il termine noto b scelto in maniera tale che la soluzione sia il vettore unitario.