Calcolo Numerico, II Esercitazione

CdL Informatica

Esercizi svolti a lezione

1) Consideriamo il seguente sistema lineare, triangolare superiore di dimensione n:

$$Ux = b, \quad \begin{cases} u_{11}x_1 + u_{12}x_2 + \ldots + u_{1n}x_n = b_1 \\ \vdots \\ u_{n-1n-1}x_{n-1} + u_{n-1n}x_n = b_{n-1} \\ u_{nn}x_n = b_n. \end{cases},$$

la cui soluzione $x = (x_1 \; x_2 \; \ldots \; x_n)^T$ si ottiene mediante l'algoritmo di sostituzione all'indietro:

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} u_{ij}x_j}{u_{ii}}, \quad i = n, n-1, \ldots, 1. \tag{1}$$

Scrivere una function `solupper` che, dati in ingresso U e b, esegue i seguenti passi:

- calcola il determinante di U senza usare il comando `det` e restituisce in uscita tale determinante. Se U è singolare, è stampato un messaggio di errore ed interrotta la comunicazione;
- se U è invertibile, viene calcolata e restituita in uscita la soluzione x mediante l'algoritmo definito da (1).

Scrivere uno script che esegue i seguenti passi:

- genera una matrice $U_1 \times \mathbb{R}^{5\times 5}$ random triangolare superiore (vedi l'`help` dei comandi `rand` e `triu`) e un vettore random $b_1 \in \mathbb{R}^{5\times 1}$;
- risolve il sistema $U_1 x = b_1$ usando `solupper` e confronta il risultato ottenuto con quello fornito dall'operatore "backslash" di MATLAB.

2) Analogamente all'esercizio 1), scrivere una function `sollower` per la risoluzione del seguente sistema triangolare inferiore di dimensione n:

$$Lx = b, \quad \begin{cases} l_{11}x_1 = b_1 \\ l_{21}x_1 + l_{22}x_2 = b_2 \\ \vdots \\ l_{n1}x_1 + l_{n2}x_2 + \ldots + l_{nn}x_n = b_n, \end{cases}.$$

sapendo che la componente $i-$esima del vettore soluzione soddisfa

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} l_{ij}x_j}{l_{ii}}, \quad i = 1, \ldots, n.$$

3) Scrivere una function `gauss1` che

- data una matrice quadrata $A \in \mathbb{R}^{n\times n}$, calcola, se possibile, la fattorizzazione LU senza pivoting di A, ovvero
$$A = LU$$
dove L e U sono due matrici triangolari, rispettivamente inferiore e superiore;
- se la fattorizzazione è possibile, la function restituisce i due fattori L e U, altrimenti stampa a video un messaggio di errore.

Scrivere inoltre uno script che

- risolve il sistema $Ax = b$ dove $A \in \mathbb{R}^{3\times 3}$ ed il vettore $b \in \mathbb{R}^3$ sono definiti come segue
$$A = \begin{pmatrix} 10^{-15} & 1 & -2 \\ 1 & 4 & 1 \\ 2 & 5 & 6 \end{pmatrix}, \quad b = A \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix};$$
il sistema viene risolto chiamando la function `gauss1` e risolvendo, in sequenza, i seguenti sistemi lineari triangolari
$$\begin{cases} Ld = b \\ Ux = d \end{cases};$$
- calcola nuovamente la soluzione usando l'operatore `backslash` e confronta le due soluzioni ottenute, calcolando i rispettivi errori relativi e mostrandoli a video.

4) La fattorizzazione LU con pivoting parziale di una matrice $A \in \mathbb{R}^{n\times n}$ si scrive come
$$PA = LU \tag{2}$$
dove L e U due matrici triangolari, rispettivamente inferiore e superiore, mentre P è una matrice che, moltiplicata a sinistra di A, scambia le righe di A secondo la strategia del pivoting parziale. Quest'ultima consiste nello scegliere il pivot $a_{i^*k}^{(k)}$ al passo $k-$esimo nel seguente modo:
$$|a_{i^*k}^{(k)}| = \max_{k\leq i\leq n} |a_{i,k}^{(k)}|.$$

Scrivere una function `gauss2` che, data una matrice `A` di dimensioni `nxn`, calcola la fattorizzazione di Gauss con pivoting parziale. In particolare, la function restituisce i fattori L, U ottenuti mediante (2) e un vettore p contenente gli indici di riga scambiati; quest'ultimo viene inizializzato come $p = (1, 2, \ldots, N)^T$ e, ogni volta che la riga $i-$esima di A viene scambiata con la sua riga $j-$esima, viene aggiornato scambiando i con j.
Scrivere uno script che risolve il sistema `Ax=b` dove `A=[-5 8 -7; 12 -5 -3; 1 10 14]` e `b=A*ones(3,1)`. A tal fine, lo script chiama la function `gauss2` per i calcolo dei fattori L, U, p, e risolve in sequenza i sistemi triangolari
$$\begin{cases} Ld = Pb \\ Ux = d \end{cases} \tag{3}$$
Inoltre lo script stampa a video se `gauss2` ha effettuato o meno il pivoting delle righe di A.

5) Si scriva la function `hilbert` che costruisce la matrice $A = (A_{ij})$ quadrata di dimensione n, detta di Hilbert, tale che

$$A_{ij} = \frac{1}{i+j-1}, \quad i, j = 1, \ldots, n$$

Eseguire la function usando $n = 4, 6, 8, 10$ e verificare che il condizionamento di A aumenta con la dimensione (per il condizionamento usare il comando `cond(A,1)`, oppure `cond(A,2)`, `cond(A,inf)`).

6) È dato il sistema lineare

$$Ax = b.$$

Usando il comando `xc = A\b`, si calcola la soluzione numerica del sistema. Il vettore residuo associato ha la forma `r=b-A*xc`. Dalla teoria del condizionamento, si ricordi che

$$\frac{\|x - xc\|}{\|x\|} \leq k(A)\frac{\|r\|}{\|b\|}. \tag{4}$$

Scrivere una function che, dati `A` e `b`, calcola `xc` e restituisce: `xc`, il numero di condizionamento di `A`, la norma del vettore residuo `r` e la limitazione superiore in (4), ovvero $d = k(A)\frac{\|r\|}{\|b\|}$.
Eseguire la function usando come `A` la matrice di Hilbert introdotta nell'esercizio 5 e come termine noto il vettore colonna `b=A*ones(n,1)` al variare di `n=4,6,8,10`.
Esaminare ed interpretare i risultati della function e l'errore relativo fra la soluzione calcolata e quella esatta.

Esercizi suggeriti per casa

1) Scrivere una function che, preso in ingresso un vettore x di N componenti, costruisca la matrice di Vandermonde di ordine N:

$$V = \begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 & \dots & x_1^{N-1} \\ 1 & x_2 & x_2^2 & x_2^3 & \dots & x_2^{N-1} \\ \vdots & & & & & \vdots \\ 1 & x_N & x_N^2 & x_N^3 & \dots & x_N^{N-1} \end{pmatrix}.$$

Eseguire la function per $N = 3, 4, \dots, 10$ e x vettore di N punti equispaziati nell'intervallo $[1, 2]$. Per ogni N, calcolare il numero di condizionamento di V (vedi `cond(V,1)`, `cond(V,2)`, `cond(V,inf)`). Che cosa si può dire del condizionamento di V all'aumentare di N?

2) Scrivere una function `invlower` che calcola l'inversa di una matrice triangolare inferiore L di dimensioni $N \times N$ risolvendo il sistema

$$LX = I_N$$

dove I_N è la matrice identità di ordine N. Nella command window, creare una matrice A a valori casuali in $(0, 1)$, estrarne la parte triangolare inferiore (vedi `tril(A)`) salvandola in L ed eseguire `invlower` su L. Confrontare il risultato con quello ottenuto con il comando `inv(L)`.

3) Scrivere una function che, data una matrice A di dimensioni $N \times N$, calcola l'inversa X di A, risolvendo il sistema

$$AX = I_N$$

dove I_N è la matrice identità di ordine N. Eseguire la function sulla matrice

$$A = \begin{pmatrix} 2 & 1 & 0 & 0 & 0 \\ 1 & 4 & 2 & 0 & 0 \\ 0 & 2 & 6 & 3 & 0 \\ 0 & 0 & 3 & 8 & 4 \\ 0 & 0 & 0 & 4 & 10 \end{pmatrix}.$$

Confrontare il risultato con quello ottenuto con il comando `inv(A)`.