Metodi Numerici per il Calcolo

Esercitazione 8: $A\mathbf{x} = \mathbf{b}$ e Fattorizzazione LU

A.A.2023/24

Scaricare dalla pagina web del corso l'archivio matlab_mnc2324_8.zip e scompattarlo nella propria home directory. Verrà creata una cartella con lo stesso nome contenente alcuni semplici script e function Matlab/Octave. Si svolga la seguente esercitazione che ha come obiettivo sperimentare la fattorizzazione LU di una matrice e la soluzione di sistemi lineari.

A. Soluzione di Sistemi Lineari

Nella cartella sono presenti alcune function che se richiamate restituiscono una matrice $n \times n$ non singolare che può essere utilizzata come matrice test per un sistema lineare $A\mathbf{x} = \mathbf{b}$. Come metodologia di lavoro si proceda definendo il vettore $\mathbf{x} \in \mathbb{R}^n$ (per esempio $\mathbf{x} = (1, 1, \ldots, 1)^T$) e si determini $\mathbf{b}$ affinché la soluzione del sistema sia il vettore $\mathbf{x}$, così da conoscerne la soluzione esatta.

1. **Function `lu` di Matlab/Octave**
   Completare lo script `main_linsys.m` che definito un sistema lineare $A\mathbf{x} = \mathbf{b}$, lo risolve nei due seguenti modi:

   - utilizzando l'operatore "left-division" di Matlab/Octave;
   - utilizzando la function di Matlab/Octave `lu` che implementa la fattorizzazione di Gauss con scambio delle righe e perno massimo. Più precisamente siamo interessati alla chiamata:

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

     (vedere `help lu`); quindi si usino le function `lsolve.m` e `usolve.m` presenti nella cartella per risolvere i sistemi

     $$L\mathbf{y} = P\mathbf{b}$$
     $$U\mathbf{x} = \mathbf{y}.$$

2. **Sulla stabilità della fattorizzazione LU**
   Si completi lo script `main_lufact.m` in modo che richiami la function `lu` di Matlab come nell'esercizio A.1 (fattorizzazione LU di una matrice con scambio delle righe e perno massimo) e verifichi che:

   $$\max |\ell_{i,j}| \leq 1, \qquad \max |u_{i,j}| \leq 2^{n-1} \max |a_{i,j}|$$

   dove $\ell_{i,j}$ e $u_{i,j}$ sono gli elementi delle matrici L e U determinate. Si stampi una tabella con i seguenti valori per le matrici di esempio `mat_k, k=2,3,4,5` di dimensioni $n = 5, 10, 50$

mat_k	$n \times n$	$\max \lvert \ell_{i,j} \rvert$	$\max \lvert u_{i,j} \rvert$	$2^{n-1} \max \lvert a_{i,j} \rvert$

Si realizzi uno script `main_qrfact.m` che richiami la built-in function Matlab `qr`, del tutto simile allo script `main_lufact.m`, per confrontare i valori degli elementi delle matrici Q ed R ottenuti nella fattorizzazione QR; si verifichi inoltre che

$$\max |q_{i,j}| \leq 1, \qquad \max |r_{i,j}| \leq \sqrt{n} \max |a_{i,j}|.$$

3. **Esempio sulla Stabilità della fattorizzazione LU**
Le function `LUsimple.m` e `LUsimple_solve.m` implementano la fattorizzazione LU senza scambio di righe e soluzione dei due sistemi triangolare inferiore e superiore. Si completi lo script `main_stab_lufact.m` per risolve il seguente sistema lineare $A\mathbf{x} = \mathbf{b}$ con

$$A = \begin{pmatrix} 10^{-20} & 1 \\ 1 & 1 \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

utilizzando sia la fattorizzazione con scambio delle righe e perno massimo (function `lu` di Matlab) che la fattorizzazione senza scambio di righe (function `LUsimple`). Si confrontino le soluzioni ottenute nei due casi.

4. **Esempio sul condizionamento di un sistema lineare**
Si consideri la matrice H di Hilbert $n \times n$ (function Matlab `hilb`, vedi l'`help`) Si calcoli $\mathbf{b}$ in modo che $H\mathbf{x} = \mathbf{b}$, dove $\mathbf{x} = (1, 1, \ldots, 1)^T$. Si aggiunga a $\mathbf{b}$ una perturbazione

$$\delta \mathbf{b}_p = 10^{-p} \texttt{rand}(n, 1).$$

Si risolva il sistema $H\tilde{\mathbf{x}}_p = \mathbf{b} + \delta \mathbf{b}_p$, per $p = 1, \ldots, 5$ e si stampi per ogni valore di p la seguente quantità:

$$K_p = \frac{\|\mathbf{x} - \tilde{\mathbf{x}}_p\|}{\|\mathbf{x}\|} \frac{\|\mathbf{b}\|}{\|\delta \mathbf{b}_p\|}.$$

I valori K_p così ottenuti sono il valore effettivo sperimentale del numero di condizione della matrice H. Verificare che per ogni p sia

$$K_p \leq \texttt{cond}(H).$$

Lo script `main_hilb.m` implementa già quanto detto; eseguire per differenti dimensioni $n \times n$ e analizzare i risultati.

B. Ricostruzione di curve per interpolazione

1. Lo script `sbezcurv2d_de.m` carica una curva di Bézier ed usa la function `gc_pol_de2d` della libreria `anmglib_4.0` per elevarne il grado. Vengono disegnati i punti di controllo della curva iniziale e di quella con grado elevato di 1. Si elevi ulteriormente il grado della curva e si osservi la relazione fra la vecchia poligonale di controllo e la nuova.

2. Il file `sppbez_draw.p` è uno script Matlab in 'proprietary mode', ossia si può eseguire in ambiente Matlab, ma non si può accedere al sorgente. Eseguendolo verrà presentato un semplice disegno/immagine che si vuole replicare con una curva di Bézier a tratti, ottenuta come join di curve di Bézier di interpolazione. Si colori quindi la curva ricostruita per replicare, come richiesto, il disegno/immagine. Lo script si chiami `sppbez_interp2d.m`.
   (Sugg. Per campionare punti del bordo dell'immagine ci si avvicini ad un punto con il mouse; Matlab mostrerà le cordinate di quel punto. Dopo avere acquisito alcuni punti del bordo si proceda alla loro interpolazione con curve di Bézier; si proceda poi al join di tali curve di Bézier per ottenere un'unica curva di Bézier a tratti che definisca il bordo).