Esercizi Elaborato di Calcolo Numerico, a.a. 2024-25
2025-05-14

Esercizio 1. Combinare linearmente $y(t+h), y(t), y(t-h), y(t-2h), y(t-3h)$, in modo da ottenere una approssimazione di $y'(t)$ con errore di troncamento $O(h^4)$. Dettagliare il procedimento.

Esecizio 2. La quadrupla precisione IEEE utilizza 128 bit (16 byte) per rappresentare un numero reale, di cui: 1 per il segno della mantissa; 112 per la parte frazionaria della mantissa (per numeri normalizzati); 15 per l'esponente. La rappresentazione è con arrotondamento (evidentemente, la base è 2). Sapendo che lo shift dell'esponente vale 16383, calcolare: la precisione di macchina; il più grande numero di macchina normalizzato; il più piccolo numero di macchina normalizzato positivo. Qual è la rappresentazione del floating di $1/3$? Qual è il corrispondente errore assoluto di rappresentazione?

Esercizio 3. Scrivere una function Matlab che implementi il metodo di Newton. Prevedere valori di default per la tolleranza di arresto e per il numero massimo di iterazioni.

Esercizio 4. Scrivere una function Matlab che implementi il metodo di Newton modificato. Prevedere valori di default per la tolleranza di arresto e per il numero massimo di iterazioni.

Esercizio 5. Scrivere una function Matlab che implementi il metodo di bisezione. Prevedere valori di default per la tolleranza di arresto.

Esercizio 6. Tabulare il costo computazionale dei precedenti metodi, in termini di valutazioni di funzione, per determinare lo zero delle funzioni:

$$f(x) = x - \cos(x), \qquad g(x) = f(x)^3,$$

considerando tolleranze: $10^{-3}, 10^{-6}, 10^{-9}, 10^{-12}$. Utilizzare $x_0 = 0$ per i metodi di Newton, e intervallo iniziale $[0, 1]$ per il metodo di bisezione. Commentare i risultati ottenuti.

Esercizio 7. Scrivere una function Matlab, con sintassi `x = trilow(L,b)`, che risolve un sistema triangolare inferiore $n \times n$, in cui la matrice dei coefficienti `L` è passata come un vettore di lunghezza $n(n+1)/2$.

Esercizio 8. Scrivere una function che implementi efficientemente il metodo di fattorizzazione LU con pivoting parziale. Validare la function su tutti i possibili casi previsti nel codice.

Esercizio 9. Utilizzare la precedente function per risolvere i seguenti sistemi lineari di dimensione n,

$$\begin{pmatrix} 1^0 & 2^0 & \dots & n^0 \\ 1^1 & 2^1 & \dots & n^1 \\ \vdots & \vdots & & \vdots \\ 1^{n-1} & 2^{n-1} & \dots & n^{n-1} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n i^0 \\ \sum_{i=1}^n i^1 \\ \vdots \\ \sum_{i=1}^n i^{n-1} \end{pmatrix},$$

la cui soluzione è $x_i = 1$, $i = 1, \ldots, n$, per $n = 1, \ldots, 20$. Tabulare *convenientemente* l'errore, e spiegare i risultati ottenuti.

Esercizio 10. Scrivere una function che implementi efficientemente il metodo di fattorizzazione $LDL^\top$ per matrici simmetriche e definite positive. Validare la function su tutti i possibili casi previsti nel codice.

Esercizio 11. Utilizzare la precedente function per risolvere i seguenti sistemi lineari di dimensione n,

$$\begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & & \iddots & \vdots \\ \frac{1}{3} & & \frac{1}{n} & & \vdots \\ \vdots & \iddots & & & \frac{1}{2n-2} \\ \frac{1}{n} & \cdots & \cdots & \frac{1}{2n-2} & \frac{1}{2n-1} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} i^{-1} \\ \sum_{i=2}^{n+1} i^{-1} \\ \vdots \\ \sum_{i=n}^{2n-1} i^{-1} \end{pmatrix},$$

la cui soluzione è $x_i = 1$, $i = 1, \ldots, n$, per $n = 1, \ldots, 20$. Tabulare *convenientemente* l'errore, e spiegare i risultati ottenuti.

Esercizio 12. Scrivere una function Matlab che implementi efficientemente la fattorizzazione QR di una matrice di rango massimo per risolvere un sistema lineare sovradeterminato $A\boldsymbol{x} = \boldsymbol{b}$ nel senso dei minimi quadrati. Ritornare, in uscita, la norma del relativo residuo, $\|\boldsymbol{r}\|_2^2$. Validare la function su tutti i possibili casi previsti nel codice.

Esercizio 13. Modificare la precedente function in modo che minimizzi, indicato al solito $\boldsymbol{r} := A\boldsymbol{x} - \boldsymbol{b} \in \mathbb{R}^m$ il residuo del sistema lineare sovradeterminato assegnato, invece che $\|\boldsymbol{r}\|_2^2 = \sum_{i=1}^{m} r_i^2$, la seguente norma pesata:

$$\|\boldsymbol{r}\|_\omega^2 = \sum_{i=1}^{m} \omega_i r_i^2, \tag{1}$$

dove $\omega_1, \ldots, \omega_m$ sono pesi positivi assegnati.

Esercizio 14. Siano assegnati:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 3 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}, \qquad \boldsymbol{b} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix}.$$

Risolvere il sistema $A\boldsymbol{x} = \boldsymbol{b}$ nel senso dei minimi quadrati, calcolando la soluzione e $\|\boldsymbol{r}\|_2^2$. Assegnati i pesi $\omega_i = 5 - i$, $i = 1, \ldots, 4$, calcolare la soluzione (e la relativa norma pesata), ottenuta minimizzando (1).

Esercizio 15. Scrivere una function che implementi efficientemente il metodo di Newton per sistemi di equazioni nonlineari.

Esercizio 16. Utilizzare la precedente function per calcolare il punto stazionario della funzione

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top Q\boldsymbol{x} - \sum_{i=1}^{3}\sin(x_i), \qquad Q = \begin{pmatrix} 4 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 4 \end{pmatrix},$$

a partire dal punto iniziale $\boldsymbol{x}_0 = \mathbf{0} \in \mathbb{R}^3$.

Esercizio 17. Scrivere una function Matlab,

```
y = lagrange( xi, fi, x )
```

in cui (`xi`,`fi`) contengono i dati del problema e `x` è un vettore di punti a cui corrispondono le valutazioni del polinomio interpolante in forma di Lagrange restituite nel vettore `y`.

Esercizio 18. Scrivere una function Matlab,

```
y = newton( xi, fi, x )
```

che ha la stessa sintassi e funzione della function del precedente esercizio, ma con l'utilizzo della forma di Newton del polinomio interpolante.

Esercizio 19. Scrivere una function Matlab,

```
[y,y1] = hermite( xi, fi, fi1, x )
```

in cui (`xi`,`fi`,`fi1`) contengono i dati del problema e `x` è un vettore di punti a cui corrispondono le valutazioni del polinomio interpolante di Hermite, e della sua derivata prima, rispettivamente restituite nei vettori `y` e `y1`.

Esercizio 20. Scrivere una function Matlab,

```
x = chebyshev( a, b, n )
```

che restituisca le ascisse di Chebyshev per il polinomio interpolante di grado `n` sull'intervallo [`a`,`b`].

Esercizio 21. Graficare l'errore massimo (calcolato come il massimo di 10001 punti equispaziati nell'intervallo) per i polinomi interpolanti degli esercizi 17–19, relative alla funzione di Runge,

$$f(x) = (1 + x^2)^{-1}, \qquad x \in [-5, 5],$$

utilizzando `n` + 1 ascisse equispaziate, con `n` $= 5, 10, 15, \ldots, 100$, per i polinomi interpolanti in 17-18, e `n` $= 5, 10, 15, \ldots, 50$, per il polinomio di Hermite. Per quest'ultimo polinomio, graficare anche l'errore della derivata prima. Utilizzare il modo più appropriato per graficare i risultati. Commentare i risultati ottenuti.

Esercizio 22. Ripetere l'esercizio precedente, utilizzando le ascisse di Chebyshev in luogo di quelle equidistanti.

Esercizio 23. Scrivere una function Matlab,

```
x = tridia( b, a, c, y )
```

che risolve un sistema tridiagonale, in cui `b, a, c` sono i vettori contenenti gli elementi sulle 3 diagonali, e `y` è un vettore contenente il termine noto.

Esercizio 24. Utilizzando la function del precedente esercizio 23, scrivere la function Matlab,

`y = spline3( xi, fi, x, type )`

in cui (`xi`, `fi`) sono i dati di interpolazione e `x` è un vettore (riga o colonna), contenente i punti in cui valutare la spline cubica interpolante, restituiti nel vettore `y`, che avrà le stesse dimensioni di `x`. Infine, `type` è un flag che vale 1 (valore di default) per la spline cubica not-a-knot, o 0, per quella naturale.

Esercizio 25. Utilizzare la function `spline3` per approssimare la funzione di Runge, come nell'esercizio 21, utilizzando `n` + 1 ascisse equidistanti, con `n` = $100, 200, \ldots, 1000$. Detta $h = 10/\texttt{n}$ la distanza tra due punti consecutivi, graficare, in `loglog` l'errore massimo, calcolato come nell'esercizio 21, rispetto ad h. Con che potenza di h decresce, asintoticamente, l'errore?

Esercizio 26. Ripetere il precedente esercizio considerando, invece dell'intervallo $[-5, 5]$, l'intervallo $[-1, 9]$. Commentare i risultati ottenuti, rispetto a quelli del precedente esercizio.

Esercizio 27. La function `polyfit` calcola il polinomio interpolante ai minimi quadrati. Creare dei dati esatti e perturbati, relativi alla funzione di Runge, con le seguenti istruzioni:

```
x  = linspace(-5,5,1001);
y  = 1./(1+x.^2);
rng(0)
r  = ( rand(1,1001)-rand(1,1001) )/5;
yr = y + r;
plot(x,yr,'r.',x,y,'b','LineWidth',2)
xlabel('x'), ylabel('y')
shg
```

Calcolare il massimo errore tra il valore restituito dal polinomio di approssimazione ai minimi quadrati di grado n, costruito sulle coppie di dati (`x`, `yr`), rispetto al valore esatto contenuto nel vettore `y`. Graficare l'errore rispetto a n, per $n = 1, 2, \ldots, 40$, in formato `semilogy`. Commentare i risultati ottenuti.

Esercizio 28. Costruire una function Matlab, `ci = ncotes( n )` che calcoli i coefficienti della formula di Newton-Cotes di grado `n`. Costruire una tabella con tutti i coefficienti delle formule ammissibili per ragioni di condizionamento. I coefficienti della tabella devono essere riportati come numeri razionali.

Esercizio 29. Costruire una function Matlab, `Ikn = ncotescomp( fx, a, b, k, n )` che calcoli l'approssimazione dell'integrale della funzione `fx` sull'intervallo [`a`,`b`], utilizzando la formula di Newton-Cotes di grado `k` su `n` + 1 ascisse equidistanti.

Esercizio 30. Utilizzare la function del precedente esercizio per approssimare

$$\int_0^{10} \frac{1}{1+x^2}\mathrm{d}x \;\equiv\; \arctan(10),$$

con le formule composite dei trapezi e di Simpson. Tabulare l'errore per `n` = $10, 20, 30, 40, 50$.