Esercizi Elaborato (versione 2021-05-17)

Esercizio 1. Verificare che, per h sufficientemente piccolo,

$$\frac{3f(x) - 4f(x-h) + f(x-2h)}{2h} = f'(x) + O(h^2).$$

Esecizio 2. Calcolare, motivandone i passaggi, la precisione di macchina della doppia precisione dello standard IEEE. Confrontare questa quantità con quanto ritornato dalla variabile `eps` di Matlab, commentando a riguardo.

Esercizio 3. Eseguire il seguente *script* Matlab:

```
format long e
a = 1e10
b = 0.1
a+b-a
```

Spiegare i risultati ottenuti.

Esercizio 4. Scrivere una *function* Matlab, `seno(x)` che, avendo in ingresso un numero `x` nell'intervallo $[-\pi, \pi]$, ne calcoli la funzione seno, utilizzando solo operazioni algebriche elementari, con la massima precisione possibile. Confrontare la *function* con `sin(x)` di Matlab.

Esercizi 5-6. Scrivere *function* Matlab distinte che implementino efficientemente i seguenti metodi per la ricerca degli zeri di una funzione:

- metodo di bisezione;
- metodo di Newton;
- metodo delle secanti;
- metodo delle corde.

Detta x_i l'approssimazione al passo i-esimo, utilizzare come criterio di arresto

$$|x_{i+1} - x_i| \leq tol \cdot (1 + |x_i|),$$

essendo tol una opportuna tolleranza specificata in ingresso.

Esercizio 7. Utilizzare le *function* del precedente esercizio per determinare una approssimazione della radice della funzione

$$f(x) = x - \cos(x),$$

per $tol = 10^{-3}, 10^{-6}, 10^{-9}, 10^{-12}$, partendo da $x_0 = 0$. Per il metodo di bisezione, utilizzare [0,1], come intervallo di confidenza iniziale, mentre per il metodo delle secanti utilizzare le approssimazioni iniziali $x_0 = 0$ e $x_1 = 1$. Tabulare i risultati, in modo da confrontare le iterazioni richieste da ciascun metodo. Commentare il relativo costo computazionale.

Esercizio 8. Calcolare la molteplicità della radice $x = 1$ della funzione

$$f(x) = (x-1)^3 e^{(x-1)}.$$

Confrontare, quindi, i metodi di Newton, Newton modificato, e di Aitken, per approssimarla per gli stessi valori di *tol* del precedente esercizio (ed utilizzando il medesimo criterio di arresto), partendo da $x_0 = 0$. Tabulare e commentare i risultati ottenuti.

Esercizio 9. Scrivere una *function* Matlab che, data in ingresso una matrice A, restituisca una matrice, LU, che contenga l'informazione sui suoi fattori L ed U, ed un vettore $\boldsymbol{p}$ contenente la relativa permutazione, della fattorizzazione LU con *pivoting* parziale di A:

```
function [LU,p] = plu(A)
```

Curare particolarmente la scrittura e l'efficienza della *function*.

Esercizio 10. Scrivere una *function* Matlab che, data in ingresso la matrice LU ed il vettore $\boldsymbol{p}$ creati dalla *function* del precedente esercizio, ed il termine noto del sistema lineare $A\boldsymbol{x} = \boldsymbol{b}$, ne calcoli la soluzione:

```
function x = mialu(LU,p,b)
```

Curare particolarmente la scrittura e l'efficienza della *function*, e validarla su due esempi generati casualmente, di cui sia nota la soluzione.

Esercizio 11. Scrivere una *function* Matlab,

```
function x = mialdl(A,b)
```

che, dati in ingresso una matrice `A` sdp ed il vettore `b`, calcoli la soluzione del corrispondente sistema lineare. Validare la *function* su due esempi generati casualmente, di cui sia nota la soluzione.

Esercizio 12. Siano dati i due sistemi lineari equivalenti:

$$A\boldsymbol{x} = \boldsymbol{b}, \qquad (A^\top A)\boldsymbol{x} = (A^\top \boldsymbol{b}),$$

con

```
A =
    0.5635   -0.4651   -0.2978
   -0.4651    0.5746   -1.1147
   -0.2978   -1.1147    9.8619

 b =
   -1.2601
   -2.6600
   27.0585
```

la cui soluzione esatta è $\boldsymbol{x} = (1,\, 2,\, 3)^\top$. Essendo la matrice A nonsingolare, dimostrare che $A^\top A$ è sdp. Risolvere, quindi, il primo sistema lineare con le functions `plu` e `mialu`, ed il secondo con la function `mialdl`. Calcolare la norma dell'errore nei due casi, e spiegare i risultati ottenuti.

Esercizio 13. Scrivere una *function* Matlab che, data in ingresso una matrice $A \in \mathbb{R}^{m\times n}$, con $m \geq n = \text{rank}(A)$, restituisca una matrice, QR, che contenga l'informazione sui fattori Q ed R della fattorizzazione QR di A:

```
function QR = qrfat(A)
```

Curare particolarmente la scrittura e l'efficienza della *function*.

Esercizio 14. Scrivere una *function* Matlab che, data in ingresso la matrice QR creata dalla *function* del precedente esercizio, ed il termine noto del sistema lineare $A\boldsymbol{x} = \boldsymbol{b}$, ne calcoli la soluzione nel senso dei minimi quadrati:

```
function x = miaqr(QR,b)
```

Curare particolarmente la scrittura e l'efficienza della *function*. Validare le *function* `qrfat` e `miaqr` confrontandole con due esempi, generati casualmente, e risolti con l'operatore Matlab \

Esercizio 15. Utilizzare le *function* `qrfat` e `miaqr` per risolvere, nel senso dei minimi quadrati, il sistema lineare sovradeterminato definito dai seguenti dati:

```
A =
     1     3     2
     3     5     4
     5     7     6
     3     6     4
     1     4     2
b =
    14
    26
    38
    29
    17
```

Esercizio 16-17. Date le $n+1$ ascisse distinte

$$a \leq x_0 < x_1 < \ldots, x_n \leq b,$$

siano definiti i corrispondenti polinomi di base di Lagrange:

$$L_{in}(x) = \prod_{j=0,\, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}, \qquad i = 0, \ldots, n,$$

tali che per ogni $i, j = 0, \ldots, n$, $L_{in}(x_j) = \delta_{ij}$, dove

$$\delta_{ij} = \begin{cases} 1, & se\ i = j, \\ 0, & se\ i \neq j, \end{cases}$$

è il *delta di Kronecker*. Dimostrare che, definiti i polinomi di grado $2n+1$

$$\Phi_{in}(x) = L_{in}^2(x)\left[1 - 2(x - x_i)L'_{in}(x_i)\right], \qquad \Psi_{in}(x) = (x - x_i)L_{in}^2(x), \qquad i = 0, \ldots, n,$$

essi soddisfano le seguenti proprietà, per ogni $i, j = 0, \ldots, n$:

- $\Phi_{in}(x_j) = \delta_{ij}, \qquad \Psi_{in}(x_j) = 0,$
- $\Phi'_{in}(x_j) = 0, \qquad \Psi'_{in}(x_j) = \delta_{ij}.$

Esercizio 18. Con riferimento ai polinomi introdotti nel precedente esercizio, dimostrare che il polinomio interpolante di Hermite, tale che

$$p(x_i) = f(x_i) \equiv f_i, \qquad p'(x_i) = f'(x_i) \equiv f'_i, \qquad i = 0, \ldots, n,$$

può essere scritto, in *forma di Lagrange*, come:

$$p(x) = \sum_{i=0}^{n} \left[f_i \Phi_{in}(x) + f'_i \Psi_{in}(x) \right].$$

Esercizio 19. Scrivere due function Matlab che implementano la forma di Newton e quella di Lagrange del polinomio interpolante una funzione.

Esercizio 20. Scrivere due function Matlab che implementano la forma di Newton e quella di Lagrange del polinomio di Hermite interpolante una funzione.

Esercizio 21. Scrivere una function Matlab che calcoli le ascisse di Chebyshev per un generico polinomio interpolante di grado n su un intervallo $[a, b]$ assegnato.

Esercizio 22. Scrivere una function Matlab, `spline0`, che abbia la stessa sintassi della function `spline` di Matlab, e che calcoli la spline cubica naturale interpolante una funzione.

Esercizio 23-24. Data la funzione di Runge,

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

tabulare l'errore di approssimazione, $\|f - q\|$, approssimato numericamente valutando il massimo di $|f(x) - q(x)|$ su 10001 punti equispaziati nell'intervallo $[-5, 5]$, quando:

- $q(x)$ è il polinomio interpolante $f(x)$ su 31 ascisse equidistanti;
- $q(x)$ è il polinomio interpolante $f(x)$ su 31 ascisse di Chebyshev;
- $q(x)$ è il polinomio di Hermite interpolante $f(x)$ su 15 ascisse equidistanti;
- $q(x)$ è il polinomio di Hermite interpolante $f(x)$ su 15 ascisse di Chebyshev;
- $q(x)$ è la spline cubica naturale interpolante $f(x)$ su 31 ascisse equidistanti;
- $q(x)$ è la spline cubica not-a-knot interpolante $f(x)$ su 31 ascisse equidistanti.

Utilizzare sia la forma di Newton che quella di Lagrange, per valutare i polinomi interpolanti. Graficare i risultati ottenuti in ciascun caso.

Esercizio 25. Tabulare l'errore di approssimazione (stimandolo in modo analogo a quanto fatto nel precedente esercizio) nell'approssimare le funzioni

$$\sin(x), \qquad \cos(x), \qquad x \in [0, 2\pi],$$

con le spline cubiche naturale e not-a-knot, su $n+1$ ascisse equidistanti nell'intervallo $[0, 2\pi]$, per $n = 10, 20, 30, \ldots, 100$. Graficare l'errore nei vari casi e spiegare quanto osservato.

Esercizio 26. Scrivere una function Matlab che calcoli i coefficienti della formula di Newton-Cotes di grado n assegnato. Tabulare, in forma razionale, i coefficienti delle formule di Newton-Cotes di grado $n = 1, \ldots, 7, 9$.

Esercizio 27. Scrivere una function Matlab che utilizzi una qualunque delle formule viste nel precedente esercizio come formule composite, suddividendo l'intervallo di integrazione in due parti uguali. Fornire in uscita una stima dell'errore commesso.

Esercizio 28. Utilizzare la function del precedente esercizio per calcolare le approssimazioni dell'integrale

$$\int_0^{\pi} \mathrm{e}^{\sin(x)} \mathrm{d}x,$$

con ciascuna delle formule (composite) di grado $n = 1, \ldots, 7, 9$, riportando la stima dell'errore ottenuta. Tabulare convenientemente i risultati.

Esercizio 29. Scrivere una function Matlab che implementi efficientemente il metodo di Simpson adattivo per approssimare un dato integrale con una precisione *tol* specificata.

Esercizio 30. Utilizzare la function del precedente esercizio per calcolare le approssimazioni del seguente integrale,

$$\int_{-0.5}^{1.095} \cos(x^{20}) \mathrm{d}x,$$

con tolleranze $tol = 10^{-i}$, $i = 2, \ldots, 10$. Costruire una tabella in cui si riportino le tolleranze utilizzate, le stime ottenute, ed il numero di valutazioni di funzione effettuate.