Esercizi Elaborato (versione 2019-05-24)

Nota bene: l'elaborato dovrà contenere i codici sviluppati, e questi dovranno essere portati alla discussione su una chiavetta USB.

Esercizio 1. Verificare che, per h sufficientemente piccolo,

$$\frac{3}{2}f(x) - 2f(x-h) + \frac{1}{2}f(x-2h) = hf'(x) + O(h^3).$$

Esecizio 2. Quanti sono i numeri di macchina normalizzati della doppia precisione IEEE? Argomentare la risposta.

Esercizio 3. Eseguire il seguente *script* Matlab:

```
format long e
n=75;
u=1e-300;for i=1:n,u=u*2;end,for i=1:n,u=u/2;end,u
u=1e-300;for i=1:n,u=u/2;end,for i=1:n,u=u*2;end,u
```

Spiegare i risultati ottenuti.

Esercizio 4. Eseguire le seguenti istruzioni Matlab:

```
format long e
a=1.11111111111111
b=1.11111111111111
a+b
a-b
```

Spiegare i risultati ottenuti.

Esercizio 5. 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 6. Utilizzare le *function* del precedente esercizio per determinare una approssimazione della radice della funzione

$$f(x) = x - \mathrm{e}^{-x} \cos(x/100),$$

per $tol = 10^{-i}$, $i = 1, 2, \ldots, 12$, partendo da $x_0 = -1$. Per il metodo di bisezione, utilizzare [-1,1], come intervallo di confidenza iniziale. Tabulare i risultati, in modo da confrontare le iterazioni richieste da ciascun metodo. Commentare il relativo costo computazionale.

Esercizio 7. Calcolare la molteplicità della radice nulla della funzione

$$f(x) = x^2 \sin(x^2).$$

Confrontare, quindi, i metodi di Newon, 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 $x_0 = 1$. Tabulare e commentare i risultati ottenuti.

Esercizio 8. 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] = palu(A)
```

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

Esercizio 9. 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 = lusolve(LU,p,b)
```

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

Esercizio 10. Scaricare la function `cremat` al sito:

`http://web.math.unifi.it/users/brugnano/appoggio/cremat.m`

che crea sistemi lineari $n \times n$ la cui soluzione è il vettore $\boldsymbol{x} = \begin{pmatrix} 1 & \dots & n \end{pmatrix}^\top$. Eseguire, quindi, lo *script* Matlab:

```
n = 10;
x = zeros(n,15);
for i = 1:15
    [A,b]  = cremat(n,i);
    [LU,p] = palu(A);
    x(:,i) = lusolve(LU,p,b);
end
```

Confrontare i risultati ottenuti con quelli attesi, e dare una spiegazione esauriente degli stessi.

Esercizio 11. 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 = myqr(A)
```

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

Esercizio 12. 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 = qrsolve(QR,b)
```

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

Esercizio 13. Scaricare la function `cremat1` al sito:

`http://web.math.unifi.it/users/brugnano/appoggio/cremat1.m`

che crea sistemi lineari $m \times n$, con $m \geq n$, la cui soluzione (nel senso dei minimi quadrati) è il vettore $\boldsymbol{x} = \begin{pmatrix} 1 & \dots & n \end{pmatrix}^\top$. Eseguire, quindi, il seguente *script* Matlab per testare le *function* dei precedenti esercizi:

```
for n = 5:10
     xx = [1:n]';
     for m = n:n+10
         [A,b]  = cremat1(m,n);
         QR     = myqr(A);
         x      = qrsolve(QR,b);
         disp([m n norm(x-xx)])
     end
end
```

Esercizio 14. Scrivere un programma che implementi efficientemente il calcolo del polinomio interpolante su un insieme di ascisse distinte.

Esercizio 15. Scrivere un programma che implementi efficientemente il calcolo del polinomio interpolante di Hermite su un insieme di ascisse distinte.

Esercizio 16. Scrivere un programma che implementi efficientemente il calcolo di una spline cubica naturale interpolante su una partizione assegnata.

Esercizio 17 (opzionale). Scrivere un programma che implementi il calcolo di una spline cubica not-a-knot interpolante su una partizione assegnata.

Esercizio 18. Confrontare i codici degli esercizi 14–17 per approssimare la funzione $f(x) = \sin(x)$ sulle ascisse $x_i = i\pi/n$, $i = 0, 1, \ldots, n$, per $n = 1, 2, \ldots, 10$. Graficare l'errore massimo di approssimazione verso n (in semilogy), calcolato su una griglia uniforme di 10001 punti nell'intervallo $[0, \pi]$.

Esercizio 19. Calcolare (numericamente) la costante di Lebesgue per i polinomi interpolanti di grado $n = 2, 4, 6, \ldots, 40$, sia sulle ascisse equidistanti che su quelle di Chebyshev (utilizzare 10001 punti equispaziati per valutare la funzione di Lebesgue). Graficare convenientemente i risultati ottenuti. Spiegare, quindi, i risultati ottenuti approssimando la funzione

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

utilizzando le ascisse equidistanti e di Chebyshev precedentemente menzionate (tabulare il massimo errore valutato su una gliglia 10001 punti equidistanti nell'intervallo $[-5, 5]$).

Esercizio 20. Con riferimento al precedente esercizio, tabulare il massimo errore di approssimazione (calcolato come sopra indicato), sia utilizzando le ascisse equidistanti che quelle di Chebyshev su menzionate, relativo alla spline cubica naturale interpolante $f(x)$ su tali ascisse.

Esercizio 21. Uno strumento di misura ha una accuratezza di 10^{-6} (in opportune unità di misura). I dati misurati nelle posizioni x_i sono dati da y_i, come descritto dalla seguente tabella:

i	x_i	y_i
0	0.010	1.003626
1	0.098	1.025686
2	0.127	1.029512
3	0.278	1.029130
4	0.547	0.994781
5	0.632	0.990156
6	0.815	1.016687
7	0.906	1.057382
8	0.913	1.061462
9	0.958	1.091263
10	0.965	1.096476

. Calcolare il grado minimo, ed i relativi coefficienti, del polinomio che meglio approssima i precedenti dati nel senso dei minimi quadrati con una adeguata accuratezza. Graficare convenientemente i risultati ottenuti.

Esercizio 22. Scrivere due functions che implementino efficientemente le formule adattattive dei trapezi e di Simpson.

Esercizio 23. Sapendo che

$$I(f) = \int_0^{\text{atan}(30)} (1 + \tan^2(x))\mathrm{d}x = 30,$$

tabulare il numero dei punti richiesti dalle formule adattative dei trapezi e di Simpson per approssimare $I(f)$, utilizzate con tolleranze

$$tol = 10^{-i}, \qquad i = 2, \ldots, 8,$$

assieme ai relativi errori.

Esercizio 24. Scrivere una function che implementi efficientemente il metodo delle potenze.

Esercizio 25. Sia data la matrice di *Toeplitz* simmetrica

$$A_N = \begin{pmatrix} 4 & -1 & & -1 & & \\ -1 & \ddots & \ddots & & \ddots & \\ & \ddots & \ddots & \ddots & & -1 \\ -1 & & \ddots & \ddots & \ddots & \\ & \ddots & & \ddots & \ddots & -1 \\ & & -1 & & -1 & 4 \end{pmatrix} \in \mathbb{R}^{N\times N}, \qquad N \geq 10, \tag{1}$$

in cui le extra-diagonali più esterne sono le none. Partendo dal vettore $\boldsymbol{u}_0 = (1, \ldots, 1)^\top \in \mathbb{R}^N$, applicare il metodo delle potenze con tolleranza $tol = 10^{-10}$ per $N = 10 : 10 : 500$, utilizzando la function del precedente esercizio. Graficare il valore dell'autovalore dominante, e del numero di iterazioni necessarie per soddisfare il criterio di arresto, rispetto ad N. Utilizzare la function `spdiags` di Matlab per creare la matrice e memorizzarla come matrice sparsa.

Esercizio 26. Scrivere una function che implementi efficientemente un metodo iterativo, per risolvere un sistema lineare, definito da un generico splitting della matrice dei coefficienti.

Esercizio 27. Scrivere le function ausiliarie, per la function del precedente esercizio, che implementano i metodi iterativi di Jacobi e Gauss-Seidel.

Esercizio 28. Con riferimento alla matrice A_N definita in (1), risolvere il sistema lineare

$$A_N \boldsymbol{x} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \in \mathbb{R}^N,$$

con i metodi di Jacobi e Gauss-Seidel, per $N = 10 : 10 : 500$, partendo dalla approssimazione nulla della soluzione, ed imponendo che la norma del residuo sia minore di 10^{-8}. Utilizzare, a tal fine, la function dell'esercizio 26, scrivendo function ausiliarie *ad hoc* (vedi esercizio 27) che sfruttino convenientemente la struttura di sparsità (nota) della matrice A_N. Graficare il numero delle iterazioni richieste dai due metodi iterativi, rispetto ad N, per soddisfare il criterio di arresto prefissato.

Nota bene: alla discussione dell'elaborato portare anche il calcolatore (o una pennina usb) in modo da potere, eventualmente, testare i codici sviluppati.