Esercizi Elaborato (versione 2020-04-29)

Esercizio 1. Verificare che, per h sufficientemente piccolo,

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

Esecizio 2. Eseguire il seguente *script* Matlab, e spiegare cosa calcola.

```
u = 1; while 1, if 1+u==1, break, end, u = u/2; end, u
```

Esercizio 3. Eseguire il seguente *script* Matlab:

```
a = 1e20; b = 100; a-a+b
a = 1e20; b = 100; a+b-a
```

Spiegare i risultati ottenuti.

Esercizio 4. Scrivere una *function* Matlab, `radn( x, n )` che, avendo in ingresso un numero positivo `x` ed un intero `n`, ne calcoli la radice n-esima con la massima precisione possibile.

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

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

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 7. Calcolare la molteplicità della radice nulla della funzione

$$f(x) = x^2 \cdot \tan(x).$$

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 $x_0 = 1$. 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/linsis.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;
xref = (1:10)';
for i = 1:10
    [A,b]  = linsis(n,i);
    [LU,p] = palu(A);
    x = lusolve(LU,p,b);
    disp(norm(x-xref))
end
```

Tabulare in modo efficace, e spiegare in modo esauriente, i risultati ottenuti.

Esercizio 11. Scrivere una *function* Matlab che, data in ingresso una matrice $A \in \mathbb{R}^{m\times n}$, con $m \geq n = \mathrm{rank}(A)$, 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. Utilizzare le *function* scritte negli esercizi 11 e 12 per risolvere, nel senso dei minimi quadrati, il sistema lineare sovradeterminato definito dai seguenti dati:

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

```
26
29
38
```

Esercizio 14. Le seguenti istruzioni,

```
A = rot90(vander(1:10)); A = A(:,1:8); x = (1:8)'; b = A*x;
```

creano una matrice $A \in \mathbb{R}^{10\times 8}$ di rango massimo ed un vettore $\boldsymbol{b} \in \mathbb{R}^8$, che definisce un sistema lineare la cui soluzione è data dal vettore

$$\boldsymbol{x} = (\,1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8\,)^\top .$$

Spiegare quindi qual è il significato delle espressioni Matlab:

```
A\b,        (A'*A)\(A'*b)
```

Spiegare i risultati ottenuti.

Esercizio 15. Approssimare la funzione $f(x) = \cos(\pi x^2/2)$ con i polinomi interpolanti rispettivamente costruiti con $n+1$ ascisse equidistanti e con $n+1$ascisse di Chebyshev sull'intervallo $[-1,1]$. Graficare (in formato `semilogy`) il massimo errore di interpolazione, per $n = 1,2,\ldots,40$. Commentare i risultati ottenuti.

Esercizio 16. Approssimare la funzione $f(x) = \cos(\pi x^2/2)$ con i polinomi interpolanti di Hermite rispettivamente costruiti con $n+1$ ascisse equidistanti e con $n+1$ascisse di Chebyshev sull'intervallo $[-1,1]$. Graficare (in formato `semilogy`) il massimo errore di interpolazione, per $n = 1,2,\ldots,20$. Commentare i risultati ottenuti.

Esercizio 17. Utilizzando la function `spline0` vista in esercitazione, costruire una function Matlab, `splinenat`, avente la stessa sintassi dell function `spline`, che calcoli la spline cubica naturale interpolante una funzione.

Esercizio 18. Utilizzare la function `splinenat` su definita per approssimare la funzione $f(x) = \cos(\pi x^2/2)$ rispettivamente su $n+1$ ascisse equidistanti e su $n+1$ascisse di Chebyshev sull'intervallo $[-1,1]$. Graficare (in formato `semilogy`) il massimo errore di interpolazione, per $n = 4,\ldots,100$.

Esercizio 19. Utilizzare la function `spline` di Matlab per approssimare la funzione $f(x) = \cos(\pi x^2/2)$ rispettivamente su $n+1$ ascisse equidistanti e su $n+1$ascisse di Chebyshev sull'intervallo $[-1, 1]$. Graficare (in formato `semilogy`) il massimo errore di interpolazione, per $n = 4, \ldots, 100$. Compararli con quelli dell'esercizio precedente.

Esercizio 20. Sia assegnata la seguente perturbazione della funzione $f(x) = \cos(\pi x^2/2)$:

$$\tilde{f}(\mathtt{x}) = f(\mathtt{x}) + 10^{-3}\,\mathtt{rand(size(x))},$$

in cui `rand` è la function built-in di Matlab. Calcolare polinomio di approssimazione ai minimi quadrati di grado m, $p(x)$, sui dati $(x_i, \tilde{f}(x_i))$, $i = 0, \ldots, n$, con

$$x_i = -1 + 2i/n, \qquad n = 10^4.$$

Graficare (in formato `semilogy`) l'errore di approssimazione $\|f - p\|$, relativo all'intervallo $[-1, 1]$, rispetto ad m, per $m = 1, 2, \ldots, 20$. Commentare i risultati ottenuti.

Esercizio 21. Costruire una function Matlab che, dato in input n, restituisca i pesi della quadratura della formula di Newton-Cotes di grado n. Tabulare, quindi, i pesi delle formule di grado 1, 2,. . . , 7 (come numeri razionali).

Esercizio 22. Utilizzare la function del precedente esercizio per graficare, in formato `semilogy`, il rapporto κ_n/κ rispetto ad n, essendo κ il numero di condizionamento di un integrale definito, e κ_n quello della formula di Newton-Cotes utilizzata di grado n per approssimarlo. Riportare i risultati per $n = 1, \ldots, 50$.

Esercizio 23. Tabulare le approssimazioni dell'integrale

$$I(f) = \int_{-1}^{1.1} \tan(x)\mathrm{d}x \equiv \log\frac{\cos(1)}{\cos(1.1)},$$

ottenute mediate le formule di Newton-Cotes di grado n, $n = 1, \ldots, 9$. Tabulare anche il relativo errore (in notazione scientifica con 3 cifre significative).

Esercizio 24. Confrontare le formula composite dei trapezi e di Simpson per approssimare l'integrale del precedente esercizio, per valori crescenti del numero dei sottointervalli dell'intervallo di integrazione. Commentare i risultati ottenuti, in termini di costo computazionale.

Esercizio 25. Confrontare le formule adattive dei trapezi e di Simpson, con tolleranze $tol = 10^{-2}, 10^{-3}, \ldots, 10^{-6}$, per approssimare l'integrale definito:

$$I(f) = \int_{-1}^{1} \frac{1}{1 + 10^2 x^2} \mathrm{d}x.$$

Commentare i risultati ottenuti, in termini di costo computazionale.

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