1 Esercizi Capitolo 4

Esercizio 4.1

Scrivere una function Matlab che implementi il calcolo del polinomio interpolante di grado n in forma di Lagrange. La forma della funcion deve essere del tipo: `y = lagrange( xi, fi, x )`

Esercizio 4.2

Scrivere una function Matlab che implementi il calcolo del polinomio interpolante di grado n in forma di Newton. La forma della funcion deve essere del tipo: `y = newton( xi, fi, x )`

Esercizio 4.3

Scrivere una function Matlab che implementi il calcolo del polinomio interpolante di Hermite. La forma della funcion deve essere del tipo:
`y = hermite( xi, fi, f1i, x )`

Esercizio 4.4

Utilizzare le functions degli esercizi precedenti per disegnare l'approssimazione della funzione $\sin x$ nell'intervallo $[0, 2\pi]$, utilizzando le ascisse di interpolazione $x_i = i\pi$, $i = 0, 1, 2$.

Esercizio 4.5

Scrivere una function Matlab che implementi la *spline* cubica interpolante (naturale o *not-a-knot*, come specificato in ingresso) delle coppie di dati assegnate. La forma della funcion deve essere del tipo:
`y = spline3( xi, fi, x, tipo )`

Esercizio 4.6

Scrivere una function Matlab che implementi il calcolo delle ascisse di Chebyshev per il polinomio interpolante di grado n, su un generico intervallo $[a, b]$. La function deve essere del tipo: `xi = ceby( n, a, b )`.

Esercizio 4.7

Utilizzare le function degli Esercizi 4.1 e 4.6 per graficare l'approssimazione della funzione di Runge sull'intervallo $[-6, 6]$, per $n = 2, 4, 6, , \ldots, 40$. Stimare, numericamente, l'errore commesso in funzione del grado n del polinomio interpolante.

Esercizio 4.8

Relativamente al precedente esercizio, stimare numericamente, la crescita della costante di Lebesgue.

Esercizio 4.9

Utilizzare la function dell'Esercizio 4.1 per approssimare la funzione di Runge, sull'intervallo $[-6, 6]$, su una partizione uniforme di $n + 1$ ascisse, $n = 2, 4, 6 \ldots, 40$. Stimare le corrispondenti costati di Lebesgue.

Esercizio 4.10

Stimare, nel senso dei minimi quadrati, posizione, velocità iniziale, ed accelerazione, relativo ad un moto rettilineo uniformemente accelerato per cui sono note le seguenti misurazioni delle coppie (tempo,spazio): $(1, 2.9)$, $(1, 3.1)$, $(2, 6.9)$, $(2, 7.1)$, $(3, 12.9)$, $(3, 13.1)$, $(4, 20.9)$, $(4, 21.1)$, $(5, 30.9)$, $(5, 31.1)$.

2 Esercizi Capitolo 5

Esercizio 5.1

Scrivere una function Matlab che implementi la formula composita dei trapezi su $n + 1$ ascisse equidistanti nell'intervallo $[a, b]$, relativamente alla funzione implementata da `fun(x)`. La function deve essere del tipo:
`If = trapcomp( n, a, b, fun )`

Esercizio 5.2

Scrivere una function Matlab che implementi la formula composita di Simpson su $2n + 1$ ascisse equidistanti nell'intervallo $[a, b]$, relativamente alla fun-

zione implementata da `fun(x)`. La function deve essere del tipo:
`If = simpcomp( n, a, b, fun )`

Esercizio 5.3

Scrivere una function Matlab che implementi la formula composita dei trapezi adattativa nell'intervallo $[a, b]$, relativamente alla funzione implementata da `fun(x)`, e con tolleranza `tol`. La function deve essere del tipo:
`If = trapad( a, b, fun, tol )`

Esercizio 5.4

Scrivere una function Matlab che implementi la formula composita di Simpson adattativa nell'intervallo $[a, b]$, relativamente alla funzione implementata da `fun(x)`, e con tolleranza `tol`. La function deve essere del tipo:
`If = simpad( a, b, fun, tol )`

Esercizio 5.5

Calcolare quante valutazioni di funzione sono necessarie per ottenere una approssimazione di

$$\mathcal{I}(f) = \int_0^1 \exp(-10^6 x)\,\mathrm{d}x,$$

che vale 10^{-6}, in doppia precisione IEEE, con una tolleranza 10^{-9}, utilizzando le functions dei precedenti esercizi. Argomentare quantitativamente la risposta.

3 Esercizi capitolo 6

Esercizio 6.1

Scrivere una function Matlab che generi la matrice *sparsa* $n \times n$, con $n > 10$,

$$A_n = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}, \quad \text{con} \quad a_{ij} = \begin{cases} 4, & \text{se} \quad i = j, \\ -1, & \text{se} \quad i = j \pm 1, \\ -1, & \text{se} \quad i = j \pm 10. \end{cases} \tag{1}$$

Utilizzare, a questo fine, la function Matlab `spdiags`.

Esercizio 6.2

Utilizzare il metodo delle potenze per calcolarne l'autovalore dominante della matrice A_n del precedente esercizio, con una approssimazione $tol = 10^{-5}$, partendo da un vettore con elementi costanti. Riempire, quindi, la seguente tabella:

n	numero di iterazioni effettuate	stima autovalore
100		
200		
$\vdots$		
1000		

Esercizio 6.3

Utilizzare il metodo di Jacobi per risolvere il sistema lineare

$$A_n \mathbf{x} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, \qquad (2)$$

dove A_n è la matrice definita in (1), con tolleranza $tol = 10^{-5}$, e partendo dal vettore nullo. Graficare il numero di iterazioni necessarie, rispetto alla dimensione n del problema, con n che varia da 100 a 1000 (con passo 20).

Esercizio 6.4

Ripetere una procedura analoga a quella del precedente esercizio utilizzando il medodo di Gauss-Seidel.

Esercizio 6.5

Con riferimento al sistema lineare (2), con $n = 1000$, graficare la norma dei residui, rispetto all'indice di iterazione, gererati dai metodi di Jacobi e Gauss-Seidel. Utilizzare il formato `semilogy` per realizzare il grafico, corredandolo di opportune *label*.

Nota bene.

Inserire, nell'elaborato, i codici utilizzati per svolgere gli Esercizi 6.2–6.5.