Esercizi Matlab Crediti D

Domenico Sabatini (0324904)
Mihai Alexandru Sandu (0327308)

24 marzo 2025

Indice

1 Esercizi

1.1 Esercizio 1.1

Implementazione dell'algoritmo di valutazione del polinomio d'interpolazione in più punti.

1.2 Esercizio 1.2

Implementazione della formula dei trapezi.

1.3 Esercizio 1.3

Implementazione del metodo di estrapolazione.

1.4 Esercizio 1.4

Implementazione del metodo di Jacobi.

2 Problemi

2.1 Problema 2.1

Interpolazione della funzione $\sqrt{x}$.

(a) Calcolo del vettore degli errori

Sono stati scelti come nodi di interpolazione i punti $x_i = \left\{0, \frac{1}{64}, \frac{4}{64}, \ldots, 1\right\}$ e sono stati calcolati i corrispondenti valori $y_i = \sqrt{x_i}$.

Il polinomio di interpolazione $p(x)$ è stato ottenuto utilizzando la funzione `polyfit` di Matlab. Successivamente è stato valutato in 21 punti $\zeta_i = \frac{i-1}{20}$ per $i = 1, \ldots, 21$, confrontando i risultati con $\sqrt{\zeta_i}$.

Codice Matlab:

```
% Nodi di interpolazione
x_nodes = [0, 1/64, 4/64, 9/64, 16/64, 25/64, 36/64, 49/64, 1];
y_nodes = sqrt(x_nodes);

% Calcolo del polinomio d'interpolazione
p = polyfit(x_nodes, y_nodes, length(x_nodes) - 1);

zeta = (0:20) / 20; % 21 punti nell'intervallo [0,1]
p_zeta = polyval(p, zeta);
sqrt_zeta = sqrt(zeta);
diff_vector = p_zeta - sqrt_zeta;

% stampo il vettore
disp('Vettore differenze:');
for i = 1:length(diff_vector)
    fprintf('%d: %1.5f\n', i, diff_vector(i));
end
```

Il vettore delle differenze mostra quanto il polinomio d'interpolazione si discosta dalla funzione esatta $\sqrt{x}$ in ciascun punto ζ_i.

Output del vettore

```
Vettore differenze (per verifica vedi script Problema21):
1: -0.00000
2:  0.00937
3: -0.01662
4:  0.00627
5:  0.02606
6:  0.00000
7: -0.04680
8: -0.05284
9:  0.01904
10: 0.13666
11: 0.19597
12: 0.07022
13: -0.29867
14: -0.79383
```

```
15: -1.04786
16: -0.46169
17: 1.60012
18: 5.33760
19: 9.64872
20: 10.73148
21: -0.00000
```

(b) Grafico di $\sqrt{x}$ e del polinomio $p(x)$

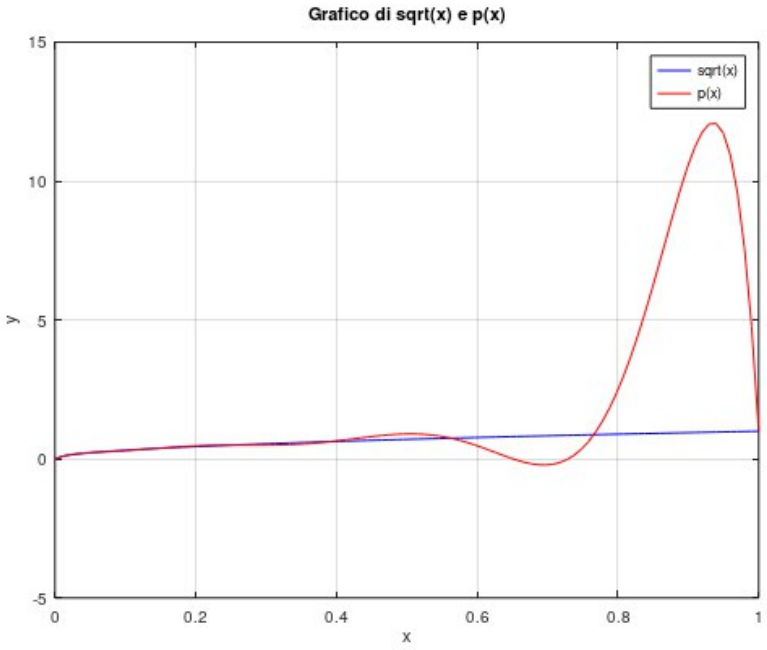

Figura 1: Confronto tra $\sqrt{x}$ (blu) e polinomio d'interpolazione $p(x)$ (rosso).

Il grafico per mostrare il confronto tra la funzione $\sqrt{x}$ e il polinomio $p(x)$ è stato creato usando il seguente codice MatLab:

```
% Creazione di punti per il grafico
x_plot = linspace(0, 1, 100);
% Valutazione del polinomio nei punti x_plot
p_plot = polyval(p, x_plot);
% Valutazione della funzione sqrt(x) nei punti x_plot
sqrt_plot = sqrt(x_plot);

figure;
plot(x_plot, sqrt_plot, 'b-', 'DisplayName', 'sqrt(x)');
hold on;
plot(x_plot, p_plot, 'r-', 'DisplayName', 'p(x)');
xlabel('x');
ylabel('y');
title('Grafico di sqrt(x) e p(x)');
legend;
```

2.2 Problema 2.2

Approssimazione dell'integrale $\int_0^1 e^x\,dx$ *con la formula dei trapezi.*

(a) DA FARE SU CARTA e (b) Determinazione di $n(\varepsilon)$ per vari ε

Il valore esatto dell'integrale è:

$$I = \int_0^1 e^x\,dx = e - 1 \approx 1.718281828459045$$

Abbiamo implementato un algoritmo che, per ogni $\varepsilon \in \{10^{-1}, \ldots, 10^{-10}\}$, determina il numero minimo n tale che l'errore $|I - I_n| \leq \varepsilon$, dove I_n è l'approssimazione dell'integrale con la formula dei trapezi di ordine n.

Codice Matlab dell'algoritmo:

```
function I = formula_trapezi(f, a, b, n)
    x = linspace(a, b, n + 1);
    y = f(x);
    h = (b - a) / n;
    I = h * (sum(y) - 0.5 * (y(1) + y(end)));
end
```

Epsilon	n	I_n	Error
1.0e-01	2	1.753931092464826	3.56493e-02
1.0e-02	5	1.724005619782788	5.72379e-03
1.0e-03	16	1.718841128579994	5.59300e-04
1.0e-04	48	1.718343976513113	6.21481e-05
1.0e-05	151	1.718288108448857	6.27999e-06
1.0e-06	476	1.718282460433049	6.31974e-07
1.0e-07	1506	1.718281891593030	6.31340e-08
1.0e-08	4760	1.718281834778786	6.31974e-09
1.0e-09	15051	1.718281829091139	6.32093e-10
1.0e-10	47595	1.718281828522253	6.32083e-11

Figura 2: Tabella di confronto

(c) Approssimazioni per $n = 2, 4, 8, 16$

Val	I	Error: |I_n - I|
I2	1.753931092464826	3.56493e-02
I4	1.727221904557517	8.94008e-03
I8	1.720518592164302	2.23676e-03
I16	1.718841128579994	5.59300e-04

(d) Interpolazione e calcolo di $p(0)$

Abbiamo considerato il polinomio d'interpolazione $p(x)$ dei valori I_2, I_4, I_8, I_{16} in funzione dei nodi $h_2^2, ..., h_{16}^2$ (dove $h_n = (1-0)/n$), e valutato il polinomio nel punto 0 ($p(0)$).

- $p(0) = 1.718281828460388$
- Errore: $|p(0) - I| = 1.343 \cdot 10^{-12}$

Questo risultato mostra che valutare il polinomio d'interpolazione $p(x)$ come definito sopra nel punto 0 è un'approssimazione molto più accurata di $\int_0^1 e^x \, dx$ rispetto all'utilizzo delle singole formule dei trapezi I_2, I_4, I_8, I_{16}.

2.3 Problema 2.3

Consideriamo la funzione $f(x) = x^2e^{-x}$ *e vogliamo approssimare l'integrale* $\int_0^1 f(x)\,dx$ *con la formula dei trapezi.*

(a) DA FARE SU CARTA PRIMA PARTE Calcolo numerico dell'integrale esatto

Abbiamo calcolato il valore esatto dell'integrale numericamente tramite la funzione `integral` di MATLAB:

$$I \approx 0.160602794142788$$

(b) Calcolo delle approssimazioni $I_5, I_{10}, I_{20}, I_{40}$

Ancora una volta utilizziamo l'algoritmo ricavato nel Problema 2.2 per calcolare la formula dei trapezi:

```
f = @(x) x.^2 .* exp(-x);

function I = formula_trapezi(f, a, b, n)
    x = linspace(a, b, n + 1);
    y = f(x);
    h = (b - a) / n;
    I = h * (sum(y) - 0.5 * (y(1) + y(end)));
end
```

I valori ottenuti per $n = 5, 10, 20, 40$ sono riportati nella tabella del punto (d).

(c) Calcolo di $p(0)$ tramite interpolazione su h^2

Abbiamo considerato il polinomio d'interpolazione $p(x)$ dei valori $I_5, I_{10}, I_{20}, I_{40}$ in funzione dei nodi $h_5^2, ..., h_{40}^2$ (dove $h_n = (1-0)/n$), e valutato il polinomio nel punto 0 ($p(0)$).

$$p(0) = 0.160602794143036$$

(d) Tabella dei risultati

n	I_n	Errore |I_n - I|
5	0.161816576820683	1.21378e-03
10	0.160908578632096	3.05784e-04
20	0.160679386811339	7.65927e-05
40	0.160621951474857	1.91573e-05
p(0)	0.160602794142805	1.61815e-14

Figura 3: Approssimazioni dell'integrale con la formula dei trapezi e interpolazione.

(e) DA FARE SU CARTA Determinazione di n tale che $|I_n - I| \leq |p(0) - I|$

2.4 Problema 2.4

Si consideri il sistema lineare $Ax = b$, con

$$A = \begin{bmatrix} 5 & 1 & 2 \\ -1 & 7 & 1 \\ 0 & 1 & -3 \end{bmatrix}, \quad b = \begin{bmatrix} 13 \\ 16 \\ -7 \end{bmatrix}$$

(a) Soluzione esatta

La soluzione esatta del sistema lineare è stata calcolata in Matlab tramite il seguente codice:

```
x_exact = A \ b;
fprintf('(a) Soluzione esatta:\n');
disp(x_exact);
```

La soluzione esatta restituita è il vettore colonna $x = [1.0000, 2.0000, 3.0000]^T$

(b) Metodo di Jacobi – prime 10 iterazioni

Si è applicato il metodo di Jacobi con vettore iniziale $x^{(0)} = [0, 0, 0]^T$. Le prime 10 iterazioni sono state salvate in una matrice $S \in \mathbb{R}^{3\times12}$, dove: - le colonne da 1 a 11 sono $x^{(0)}, x^{(1)}, \ldots, x^{(10)}$, - l'ultima colonna è la soluzione esatta x.

Il codice Matlab implementa la formula iterativa:

$$x^{(k+1)} = Px^{(k)} + q, \quad \text{con } P = MatriceIterazione = D^{-1}(D - A),\ q = D^{-1}b$$

```
(b) Matrice S contenente x^(0), ..., x^(10), x esatta:
         0    2.6000    1.2095    0.8971    0.9536    1.0038    1.0055    1.0006    0.9995    0.9999    1.0000    1.0000
         0    2.2857    2.3238    2.0163    1.9699    1.9926    2.0020    2.0011    2.0000    1.9999    2.0000    2.0000
         0    2.3333    3.0952    3.1079    3.0054    2.9900    2.9975    3.0007    3.0004    3.0000    3.0000    3.0000
```

Figura 4: Matrice 3x12

(c) Convergenza al variare di ε

Si è analizzata la convergenza del metodo di Jacobi per vari epsilon $\varepsilon \in \{10^{-1}, 10^{-2}, \ldots, 10^{-10}\}$. Per ciascuna soglia sono stati registrati:

- il numero minimo di iterazioni K_ε affinché $\|x^{(k)} - x^{(k-1)}\|_\infty \leq \varepsilon$,
- la soluzione approssimata x_ε,
- l'errore rispetto alla soluzione esatta: $\|x - x_\varepsilon\|_\infty$.

Di seguito la tabella risultante in output:

```
 epsilon = 1.0e-01, K = 5, soluzione approssimata:
    1.0038
    1.9926
    2.9900

errore norma infinito = 1.00e-02

 epsilon = 1.0e-02, K = 6, soluzione approssimata:
    1.0055
    2.0020
    2.9975

errore norma infinito = 5.50e-03

 epsilon = 1.0e-03, K = 9, soluzione approssimata:
    0.9999
    1.9999
    3.0000

errore norma infinito = 1.50e-04

 epsilon = 1.0e-04, K = 11, soluzione approssimata:
    1.0000
    2.0000
    3.0000

errore norma infinito = 2.08e-05


 epsilon = 1.0e-05, K = 13, soluzione approssimata:
    1.0000
    2.0000
    3.0000

errore norma infinito = 2.08e-06

 epsilon = 1.0e-06, K = 15, soluzione approssimata:
    1.0000
    2.0000
    3.0000

errore norma infinito = 1.62e-07

 epsilon = 1.0e-07, K = 17, soluzione approssimata:
    1.0000
    2.0000
    3.0000

errore norma infinito = 1.45e-08

 epsilon = 1.0e-08, K = 19, soluzione approssimata:
    1.0000
    2.0000
    3.0000

errore norma infinito = 1.82e-09
```

```
 epsilon = 1.0e-09, K = 21, soluzione approssimata:
    1.0000
    2.0000
    3.0000

errore norma infinito = 2.57e-10

 epsilon = 1.0e-10, K = 23, soluzione approssimata:
    1.0000
    2.0000
    3.0000

errore norma infinito = 3.25e-11
```