CREDITI DI TIPO D MATLAB

Attività svolta da Christian Sfeir (0284535) e Matteo Cipolletta (0306676)

PROBLEMI

Problema 2.1. Si consideri la funzione $\sqrt{x}$.

(a) Sia $p(x)$ il polinomio d'interpolazione di $\sqrt{x}$ sui nodi

$$x_0 = 0, \quad x_1 = \frac{1}{64}, \quad x_2 = \frac{4}{64}, \quad x_3 = \frac{9}{64}, \quad x_4 = \frac{16}{64}, \quad x_5 = \frac{25}{64}, \quad x_6 = \frac{36}{64}, \quad x_7 = \frac{49}{64}, \quad x_8 = 1.$$

Calcolare il vettore (colonna)

$$\left[p(\zeta_1) - \sqrt{\zeta_1} \qquad p(\zeta_2) - \sqrt{\zeta_2} \qquad \cdots \qquad p(\zeta_{21}) - \sqrt{\zeta_{21}} \right]^T$$

dove $\zeta_i = \frac{i-1}{20}$ per $i = 1, \ldots, 21$, e osservare in che modo varia la differenza $p(\zeta_i) - \sqrt{\zeta_i}$ al variare di i da 1 a 21.

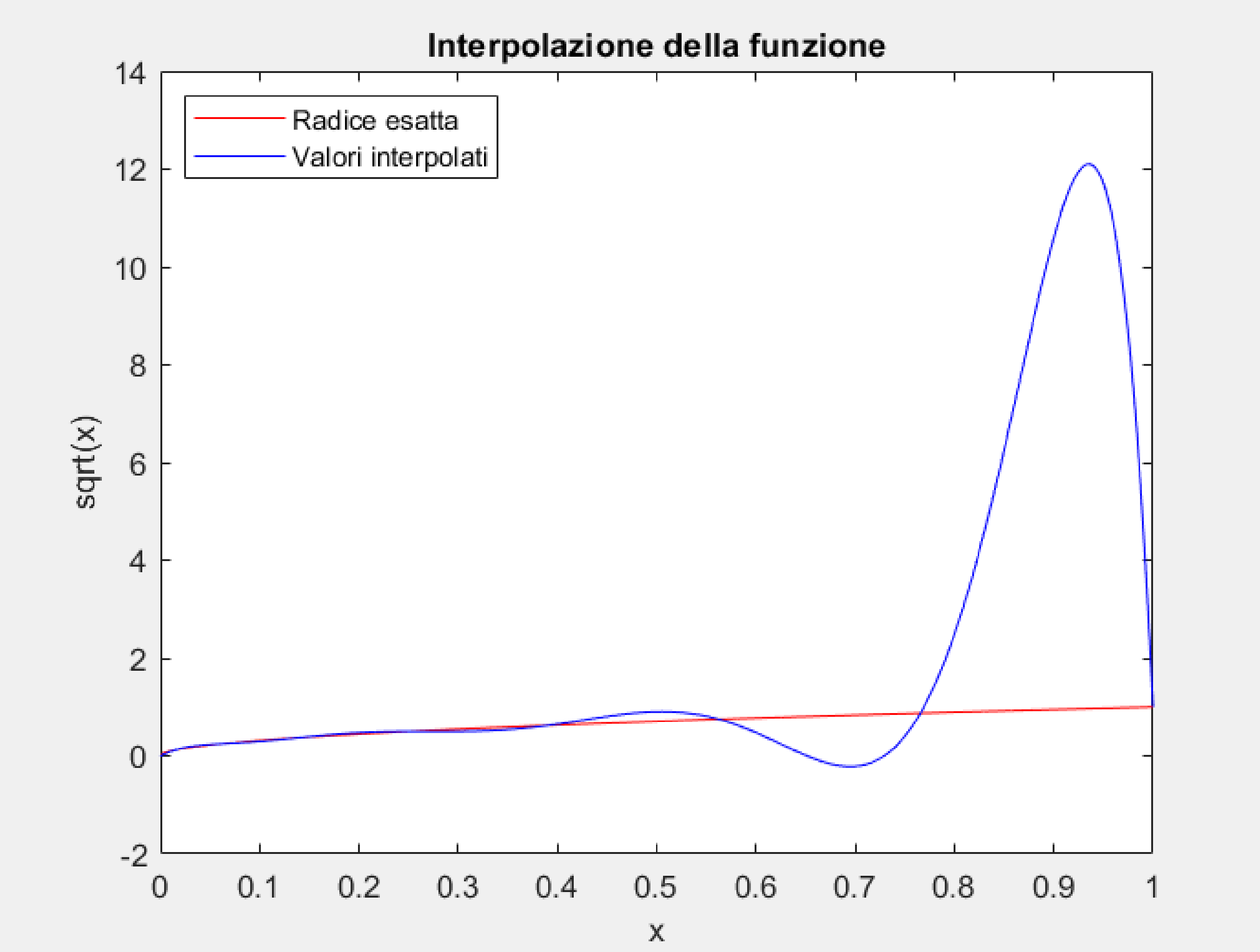
(b) Tracciare il grafico di $\sqrt{x}$ e di $p(x)$ sull'intervallo $[0, 1]$, ponendo i due grafici su un'unica figura e inserendo una legenda che ci dica qual è la funzione $\sqrt{x}$ e qual è il polinomio $p(x)$.

SOLUZIONE:

a)

```
R =

  Columns 1 through 9

         0    0.0094   -0.0166    0.0063    0.0261   -0.0000   -0.0468   -0.0528    0.0190

  Columns 10 through 18

    0.1367    0.1960    0.0702   -0.2987   -0.7938   -1.0479   -0.4617    1.6001    5.3376

  Columns 19 through 21

    9.6487   10.7315   -0.0000
```

b)



Problema 2.2. Si consideri la funzione

$$f(x) = \mathrm{e}^x.$$

Per ogni intero $n \geq 1$ indichiamo con I_n la formula dei trapezi di ordine n per approssimare

$$I = \int_0^1 f(x)\mathrm{d}x = 1.7182818284590...$$

(a) Per ogni fissato $\varepsilon > 0$ determinare un $n = n(\varepsilon)$ tale che $|I - I_n| \leq \varepsilon$.

(b) Costruire una tabella che riporti vicino ad ogni $\varepsilon \in \{10^{-1}, 10^{-2}, \ldots, 10^{-10}\}$:
- il numero $n(\varepsilon)$;
- il valore I_n per $n = n(\varepsilon)$;
- il valore esatto I (in modo da confrontarlo con I_n);
- l'errore $|I - I_n|$ (che deve essere $\leq \varepsilon$).

(c) Calcolare le approssimazioni di I ottenute con le formule dei trapezi I_2, I_4, I_8, I_{16} e confrontarle con il valore esatto I.

(d) Sia $p(x)$ il polinomio d'interpolazione dei valori I_2, I_4, I_8, I_{16} sui nodi h_2^2, h_4^2, h_8^2, h_{16}^2, dove $h_2 = \frac{1}{2}$, $h_4 = \frac{1}{4}$, $h_8 = \frac{1}{8}$, $h_{16} = \frac{1}{16}$ sono i passi di discretizzazione relativi alle formule dei trapezi I_2, I_4, I_8, I_{16} rispettivamente. Calcolare $p(0)$ e confrontare I_2, I_4, I_8, I_{16}, $p(0)$ con il valore esatto I. Che cosa si nota?

SOLUZIONE:

a)

A) SIA $f(x) = e^x$. PER IL TEOREMA SUL RESTO DELLA FORMULA DEI TRAPEZI:

$$\left| \int_0^1 e^x dx - I_m \right| = \left| -\frac{f''(\eta)}{12} \cdot \left(\frac{1}{m}\right)^2 \right| = \frac{|f''(\eta)|}{12m^2} \qquad (\eta \in [0,1])$$

CALCOLIAMO $f''(x)$:

$f'(x) = e^x$

$f''(x) = e^x$

$\forall x \in [0,1] : |f''(x)| = |e^x| = e^x \leq e^1 = e$

DUNQUE, $\left| \int_0^1 e^x dx - I_m \right| = \frac{|f''(\eta)|}{12m^2} \leq \frac{e}{12m^2}$

IMPONGO $\frac{e}{12m^2} \leq \varepsilon \iff m \geq \sqrt{\frac{e}{12\varepsilon}} \Rightarrow m(\varepsilon) = \left\lceil \sqrt{\frac{e}{12\varepsilon}} \right\rceil$

DUNQUE, SE PRENDO $m \geq m(\varepsilon)$ ALLORA SONO SICURO CHE $\left| \int_0^1 e^x dx - I_m \right| \leq \varepsilon$

b)

Epsilon	n(epsilon)	I_n	Errore
1.0e-01	2	1.7539310925	3.5649264006e-02
1.0e-02	5	1.7240056198	5.7237913237e-03
1.0e-03	16	1.7188411286	5.5930012095e-04
1.0e-04	48	1.7183439765	6.2148054069e-05
1.0e-05	151	1.7182881084	6.2799898122e-06
1.0e-06	476	1.7182824604	6.3197400291e-07
1.0e-07	1506	1.7182818916	6.3133985595e-08
1.0e-08	4760	1.7182818348	6.3197409528e-09
1.0e-09	15051	1.7182818291	6.3209260048e-10
1.0e-10	47595	1.7182818285	6.3191452071e-11

c - d)

```
Valore esatto I: 1.7182818285
Valori calcolati con i trapezi:
I2 = 1.7539310925, I4 = 1.7272219046, I8 = 1.7205185922, I16 = 1.7188411286
Valore di p(0): 1.7182818285

Confronto:
Errore |I2 - I| = 3.5649264006e-02
Errore |I4 - I| = 8.9400760985e-03
Errore |I8 - I| = 2.2367637053e-03
Errore |I16 - I| = 5.5930012095e-04
Errore |p(0) - I| = 1.3438139490e-12
```

Si nota che p(0) è un'approssimazione di I molto più accurata delle singole formule dei trapezi I_2, I_4, I_8 e I_{16}.

Problema 2.3. Consideriamo la funzione $f(x) = x^2 e^{-x}$ e indichiamo con I_n la formula dei trapezi di ordine n per approssimare $I = \int_0^1 f(x)dx$.

(a) Calcolare I prima manualmente e poi con la funzione simbolica `int` di MATLAB.

(b) Calcolare $I_5, I_{10}, I_{20}, I_{40}$.

(c) Calcolare $p(0)$, dove $p(x)$ è il polinomio d'interpolazione dei dati (h_0^2, I_5), (h_1^2, I_{10}), (h_2^2, I_{20}), (h_3^2, I_{40}) e h_0, h_1, h_2, h_3 sono i passi di discretizzazione delle formule dei trapezi $I_5, I_{10}, I_{20}, I_{40}$.

(d) Riportare in una tabella:
- i valori $I_5, I_{10}, I_{20}, I_{40}, p(0)$;
- gli errori $|I_5 - I|, |I_{10} - I|, |I_{20} - I|, |I_{40} - I|, |p(0) - I|$.

(e) Posto $\varepsilon = |p(0) - I|$, determinare un n in modo tale che la formula dei trapezi I_n fornisca un'approssimazione di I con errore $|I_n - I| \le \varepsilon$. Calcolare successivamente I_n e verificare che effettivamente $|I_n - I| \le \varepsilon$.

SOLUZIONE:

a)

A)

$$I = \int_0^1 x^2 e^{-x} dx \underset{\substack{\downarrow \\ \text{INTEGRAZIONE} \\ \text{PER PARTI}}}{=} x^2 \cdot (-e^{-x}) - \int_0^1 -e^{-x} \cdot 2x\, dx = x^2 \cdot (-e^{-x}) + 2\int_0^1 x e^{-x} dx =$$

$$= x^2 \cdot (-e^{-x}) + 2\left(x \cdot (-e^{-x}) - \int_0^1 -e^{-x} dx\right) = x^2 \cdot (-e^{-x}) + 2\left(x \cdot (-e^{-x}) + \int_0^1 e^{-x} dx\right) =$$

$$= x^2 \cdot (-e^{-x}) + 2\left(x \cdot (-e^{-x}) - e^{-x}\right)\Big|_0^1 = -x^2 e^{-x} - 2x e^{-x} - 2e^{-x}\Big|_0^1 =$$

$$= -1^2 e^{-1} - 2e^{-1} - 2e^{-1} - (-0^2 e^{-0} - 2 \cdot 0 \cdot e^{-0} - 2e^{-0}) = -\frac{5}{e} + 2 = 0{,}160602794 1\ldots$$

IN MATLAB:

```
syms x;
f = x^2 * exp(-x);  % Definisci la funzione simbolica
integrale = int(f, x, 0, 1);  % Calcola l'integrale definito da 0 a 1
disp(integrale);  % Mostra il risultato
```

```
2 - 5*exp(-1)
```

b-c-d)

```
Tabella dei risultati:
n           In                |In - I|
--------------------------------------------
5           0.1618165768     1.2137826779e-03
10          0.1609085786     3.0578448931e-04
20          0.1606793868     7.6592668551e-05
40          0.1606219515     1.9157332069e-05
p(0)        0.1606027941     1.6237011735e-14
```

e)

e) Posto $\varepsilon = |p(0) - I| \approx 1{,}6237011735 \cdot 10^{-14}$

PER IL TEOREMA SULL' ERRORE DELLA FORMULA DEI TRAPEZI,

$$\left| I_m - \int_0^1 x^2 \cdot e^{-x}\, dx \right| = \left| -\frac{f''(\eta)}{12} \cdot \left(\frac{1}{m}\right)^2 \right| = \frac{|f''(\eta)|}{12m^2} \qquad (\eta \in [0,1])$$

CALCOLIAMO:

$\cdot\ f'(x) = 2xe^{-x} - x^2e^{-x}$; $\cdot\ f''(x) = 2e^{-x} - 4xe^{-x} + x^2e^{-x}$

$\forall x \in [0,1]$

$$|f''(x)| = |2e^{-x} - 4xe^{-x} + x^2e^{-x}| \leq 2e^{-x} + 4xe^{-x} + x^2e^{-x} \leq 2 + 4 + 1 = 7$$

DUNQUE,

$$\frac{|f''(\eta)|}{12m^2} \leq \frac{7}{12m^2}$$

IMPONGO:

$$\frac{7}{12m^2} \leq \varepsilon \iff m \geq \sqrt{\frac{7}{12\varepsilon}} = m(\varepsilon)$$

IN CONCLUSIONE, SE PRENDO $m \geq m(\varepsilon)$ ALLORA $|I_m - I| \leq \varepsilon$

NEL CASO $\varepsilon = 1{,}6237011735 \cdot 10^{-14}$ PRENDERÒ $m \geq m(1{,}6237011735 \cdot 10^{-14}) = \sqrt{\frac{7}{12 \cdot 1{,}6237011735 \cdot 10^{-14}}} =$

$= 5993842{,}747$ (QUINDI PRENDERÒ $m = 5993843$)

```
Risultato per n = 5993843:
I_n = 0.1606027941
|I_n - I| = 1.0158540675e-14
```

Problema 2.4. Consideriamo la funzione $f(x) = \frac{1}{x \log x}$ e indichiamo rispettivamente con I_n e S_n la formula dei trapezi e di Cavalieri-Simpson di ordine n per approssimare $I = \int_2^5 f(x)\mathrm{d}x$.

(a) Calcolare I prima manualmente e poi con la funzione simbolica `int` di MATLAB.

(b) Costruire una tabella che riporti vicino ad ogni valore di

$$n = 5, 10, 20, 40, 80, 160, 320, 640, 1280, 2560$$

sia le approssimazioni di I ottenute con I_n e S_n sia i relativi errori $|I_n - I|$ e $|S_n - I|$. Quale delle formule I_n e S_n converge più velocemente al valore esatto I al crescere di n?

SOLUZIONE:

a)

$$A)\ I = \int_2^5 \frac{1}{x \log x}\, dx \underset{\substack{y = \log x \\ dy = \frac{1}{x} dx}}{=} \int_{\log 2}^{\log 5} \frac{dy}{y} = \log|y| + C \Big|_{\log 2}^{\log 5} = \log(\log 5) - \log(\log 2)$$

$$= 0,84261360$$

IN MATLAB:

```
syms x
f = 1/(x*log(x));  % Definizione della funzione
integrale = int(f, x, 2, 5);  % Calcolo dell'integrale definito da 2 a 5
disp(integrale)  % Mostra il risultato
```

```
log(log(5)) - log(log(2))
```

b)

n	I_n	S_n	|I_n - I|	|S_n - I|
5	8.667092e-01	8.426174e-01	2.431127e-02	2.195180e-04
10	8.486404e-01	8.424135e-01	6.242456e-03	1.557515e-05
20	8.439702e-01	8.423989e-01	1.572295e-03	1.011061e-06
40	8.427917e-01	8.423980e-01	3.938322e-04	6.382981e-08
80	8.424964e-01	8.423979e-01	9.850591e-05	3.999568e-09
160	8.424225e-01	8.423979e-01	2.462948e-05	2.501337e-10
320	8.424041e-01	8.423979e-01	6.157557e-06	1.563594e-11
640	8.423995e-01	8.423979e-01	1.539401e-06	9.768852e-13
1280	8.423983e-01	8.423979e-01	3.848510e-07	6.228351e-14
2560	8.423980e-01	8.423979e-01	9.621279e-08	3.663736e-15

- **Cavalieri-Simpson** convergerà più velocemente all'integrale esatto I rispetto alla formula dei trapezi, soprattutto per valori di n più grandi.

Problema 2.6. Si consideri il sistema lineare $A_n\mathbf{x} = \mathbf{b}_n$, dove $\mathbf{b}_n = [1, 1, \ldots, 1]^T$ e A_n è la matrice $n \times n$ definita nel modo seguente:

$$(A_n)_{ij} = \begin{cases} 3, & \text{se } i = j, \\ -(\frac{1}{2})^{\max(i,j)-1}, & \text{se } i \neq j. \end{cases}$$

(a) Scrivere esplicitamente A_n per $n = 5$.

(b) Dimostrare che, qualunque sia n, A_n è una matrice a diagonale dominante in senso stretto per righe e per colonne. Dedurre che i metodi di Jacobi e Gauss-Seidel per risolvere un sistema lineare di matrice A_n sono convergenti.

(c) Risolvere con il comando "\" il sistema lineare $A_n\mathbf{x} = \mathbf{b}_n$ per $n = 5, 10, 20$.

(d) Risolvere il sistema lineare $A_n\mathbf{x} = \mathbf{b}_n$ per $n = 5, 10, 20$ con i metodi di Jacobi e Gauss-Seidel entro una soglia di precisione $\varepsilon = 10^{-7}$ partendo dal vettore d'innesco $\mathbf{x}^{(0)} = \mathbf{0}$.

(e) Costruire una tabella che vicino ad ogni $n = 5, 10, 20$ riporti:
- la soluzione esatta $\mathbf{x}$ del sistema $A_n\mathbf{x} = \mathbf{b}_n$ ottenuta al punto (c);
- le soluzioni approssimate $\mathbf{x}_J$ e $\mathbf{x}_G$ ottenute con i metodi di Jacobi e Gauss-Seidel al punto (d);
- gli errori $\|\mathbf{x}_J - \mathbf{x}\|_\infty$ e $\|\mathbf{x}_G - \mathbf{x}\|_\infty$;
- i numeri K_J e K_G che contano le iterazioni effettuate da Jacobi e Gauss-Seidel per calcolare $\mathbf{x}_J$ e $\mathbf{x}_G$, rispettivamente.

SOLUZIONE:

a)

$$A_5 = \begin{bmatrix} 3 & -1/2 & -1/4 & -1/8 & -1/16 \\ -1/2 & 3 & -1/4 & -1/8 & -1/16 \\ -1/4 & -1/4 & 3 & -1/8 & -1/16 \\ -1/8 & -1/8 & -1/8 & 3 & -1/16 \\ -1/16 & -1/16 & -1/16 & -1/16 & 3 \end{bmatrix}$$

b)

UNA MATRICE È DIAGONALE IN SENSO STRETTO PER RIGHE SE PER OGNI RIGA i VALE:

$$|a_{ii}| > \sum_{i \neq j} |a_{ij}|$$

ORA: - $a_{ii} = 3$

- $a_{ij} = -\left(\frac{1}{2}\right)^{\max(i,j)-1}$ CON $i \neq j$

LA SOMMA PER UNA RIGA i È: $\sum_{j \neq i} |a_{ij}| = \sum_{j \neq i} \left(\frac{1}{2}\right)^{\max(i,j)-1}$

DIVIDIAMO IL CALCOLO IN 2 CASI:

- PER $j < i$: $\max(i,j) = i \Rightarrow |a_{ij}| = \left(\frac{1}{2}\right)^{i-1}$
- PER $j > i$: $\max(i,j) = j \Rightarrow |a_{ij}| = \left(\frac{1}{2}\right)^{j-1}$

LA SOMMA TOTALE È:

$$\sum_{j \neq i} |a_{ij}| = \underbrace{\sum_{j<i} \left(\frac{1}{2}\right)^{i-1}}_{(1)} + \underbrace{\sum_{j>i} \left(\frac{1}{2}\right)^{j-1}}_{(2)}$$

(1) PER $j<i$ CI SONO $i-1$ TERMINI UGUALI A $\left(\frac{1}{2}\right)^{i-1}$, QUINDI:

$$\sum_{j<i} \left(\frac{1}{2}\right)^{i-1} = (i-1) \cdot \left(\frac{1}{2}\right)^{i-1}.$$

(2) PER $j>i$ SI HA: $\sum_{j>i} \left(\frac{1}{2}\right)^{j-1} = \left(\frac{1}{2}\right)^{i} + \left(\frac{1}{2}\right)^{i+1} + \ldots + \left(\frac{1}{2}\right)^{n-1} =$

SOMMA GEOMETRICA

$$\uparrow\!= \frac{\left(\frac{1}{2}\right)^{i}\left(1 - \left(\frac{1}{2}\right)^{n-i}\right)}{1 - 1/2} = 2 \cdot \left(\frac{1}{2}\right)^{i}\left(1 - \left(\frac{1}{2}\right)^{n-i}\right)$$

QUINDI:

$$\sum_{j \neq i} |a_{ij}| = (i-1) \cdot \left(\frac{1}{2}\right)^{i-1} + 2 \cdot \left(\frac{1}{2}\right)^{i}\left(1 - \left(\frac{1}{2}\right)^{n-i}\right) \leq$$

$$\leq (i-1) \cdot \left(\frac{1}{2}\right)^{i-1} + \left(\frac{1}{2}\right)^{i-1} = i \cdot \left(\frac{1}{2}\right)^{i-1} = f(i)$$

DERIVIAMO $f(i): i \cdot 2^{-(i-1)}$

$$f'(i) = 1 \cdot 2^{-(i-1)} + i \cdot (-2^{-(i-1)} \cdot \ln(2)) = 2^{-(i-1)} - i \cdot 2^{-(i-1)} \cdot \ln(2) =$$

$$= 2^{-(i-1)}(1 - i \cdot \ln(2)) = -2^{-i+1} \cdot (\ln(2) \cdot i - 1)$$

LA PONGO ≥ 0: $-2^{-i+1} (\ln(2) \cdot i - 1) \geq 0$

· POICHÈ -2^{-i+1} È SEMPRE MAGGIORE DI ZERO, DEVO DIMOSTRARE CHE

$$\ln(2) \cdot i - 1 \geq 0 \iff i \geq \frac{1}{\ln(2)} = 1{,}44269504\ldots$$

QUINDI $f(1{,}44269504\ldots) = 1{,}061475691\ldots < 3$

POSSIAMO CONCLUDERE CHE LA MATRICE A_m È A DIAGONALE DOMINANTE IN SENSO STRETTO PER RIGHE.

DIMOSTRIAMO CHE LA MATRICE A_m È SIMMETRICA: $(A_m)_{ij} = (A_m)_{ji} \quad \forall i,j$

SE $i \neq j$, ABBIAMO: $(A_m)_{ij} = -\left(\frac{1}{2}\right)^{\max(i,j)-1}$ E $(A_m)_{ji} = -\left(\frac{1}{2}\right)^{\max(j,i)-1}$

POICHÈ $\max(i,j) = \max(j,i)$ SI HA CHE: $(A_m)_{ij} = (A_m)_{ji}$

$\Rightarrow A_m$ È SIMMETRICA.

GRAZIE ALLA SIMMETRIA DI A_m, ESSENDO DIAGONALE DOMINANTE IN SENSO STRETTO PER RIGHE IMPLICA AUTOMATICAMENTE CHE LA MATRICE È A DIAGONALE DOMINANTE IN SENSO STRETTO PER COLONNE. PERTANTO, I METODI DI JACOBI E GAUSS-SEIDE SONO CONVERGENTI PER A_m.

c-d)

```
>> Problema6
Matrice A:
  Columns 1 through 4

   3.000000000000000  -0.500000000000000  -0.250000000000000  -0.125000000000000
  -0.500000000000000   3.000000000000000  -0.250000000000000  -0.125000000000000
  -0.250000000000000  -0.250000000000000   3.000000000000000  -0.125000000000000
  -0.125000000000000  -0.125000000000000  -0.125000000000000   3.000000000000000
  -0.062500000000000  -0.062500000000000  -0.062500000000000  -0.062500000000000

  Column 5

  -0.062500000000000
  -0.062500000000000
  -0.062500000000000
  -0.062500000000000
   3.000000000000000

n = 5:
 - uso il comando "/":
    soluzione:
   0.472839561157381
   0.472839561157381
   0.436467287222197
   0.398640122329607
   0.370433052747220

 - uso il metodo di Gauss-Seidel:
    soluzione:
   0.472839495700892
   0.472839536275474
   0.436467276654638
   0.398640117555638
   0.370433050545555

    iterazioni:
     7

    norma:
     6.545648906230994e-08
```

```
 - uso il metodo di Jacobi:
    soluzione:
   0.472839520639523
   0.472839520639523
   0.436467256695226
   0.398640103198520
   0.370433042053835

    iterazioni:
    12

    norma:
     4.051785723602208e-08

n = 10:
 - uso il comando "/":
    soluzione:
   0.482920946916211
   0.482920946916211
   0.445773181768810
   0.407139506015513
   0.378331035084876
   0.359580987851714
   0.348197124552742
   0.341556432073382
   0.337778975988719
   0.335666796313260

 - uso il metodo di Gauss-Seidel:
    soluzione:
   0.482920853681719
   0.482920910890988
   0.445773166214829
```

```
0.407139498827743
0.378331031635072
0.359580986162509
0.348197123717112
0.341556431657905
0.337778975781665
0.335666796210006
```

```
iterazioni:
 7
```

```
    norma:
     9.323449240428161e-08

 - uso il metodo di Jacobi:
    soluzione:
   0.482920898075894
   0.482920898075894
   0.445773144884688
   0.407139482755145
   0.378331021912008
   0.359580980824802
   0.348197120921409
   0.341556430227333
   0.337778975058110
   0.335666795846191

    iterazioni:
    12

    norma:
     4.884031762353302e-08

n = 20:
 - uso il comando "/":
    soluzione:
   0.483235935360422
   0.483235935360422
   0.446063940332697
   0.407405065503864
   0.378577804053791
   0.359815526975880
   0.348424238475304
   0.341779214559493
   0.337999294603625
```

0.3379929400[illegible]

0.335885737244730

0.334718624665744

0.334080313405781

0.333733894550127

0.333547084865033

0.333446888426869

0.333393396716275

0.333364954734990

0.333349885847086

0.333341927498632

```
   0.333337736383860

 - uso il metodo di Gauss-Seidel:
    soluzione:
   0.483235841376342
   0.483235899044855
   0.446063924653065
   0.407405058257840
   0.378577800575872
   0.359815525272765
   0.348424237632653
   0.341779214140388
   0.337999294394627
   0.335885737140369
   0.334718624613598
   0.334080313379717
   0.333733894537097
   0.333547084858519
   0.333446888423612
   0.333393396714646
   0.333364954734176
   0.333349885846679
   0.333341927498428
   0.333337736383758

    iterazioni:
     7

    norma:
     9.398408029603900e-08

 - uso il metodo di Jacobi:
```

```
soluzione:
0.483235886390101
0.483235886390101
0.446063903350300
0.407405042181361
0.378577790845548
0.359815519929902
0.348424234833920
0.341779212708136
0.337999293670142
0.335885736776020
0.334718624430894
0.334080313288232
0.333733894491321
0.333547084835623
0.333446888412162
0.333393396708921
0.333364954731313
0.333349885845247
0.333341927497712
0.333337736383400

iterazioni:
12

norma:
 4.89703215755810le-08
```

e)

Tabella risultati per n = 5:

x_Direct	**x_Gauus-Seidel**	**x_Jacobi**
0.472839561157381	0.472839495700892	0.472839520639523
0.472839561157381	0.472839536275474	0.472839520639523
0.436467287222197	0.436467276654638	0.436467256695226
0.398640122329607	0.398640117555638	0.39864010319852
0.37043305274722	0.370433050545555	0.370433042053835

k_Gauss-Seidel	**k_Jacobi**
7	12

norma_Gauss-Seidel	**norma_Jacobi**
6.54564890623099e-08	4.05178572360221e-08

Tabella risultati per n = 10:

x_Direct	x_Gauus-Seidel	x_Jacobi
0.482920946916211	0.482920853681719	0.482920898075894
0.482920946916211	0.482920910890988	0.482920898075894
0.44577318176881	0.445773166214829	0.445773144884688
0.407139506015513	0.407139498827743	0.407139482755145
0.378331035084876	0.378331031635072	0.378331021912008
0.359580987851714	0.359580986162509	0.359580980824802
0.348197124552742	0.348197123717112	0.348197120921409
0.341556432073382	0.341556431657905	0.341556430227333
0.337778975988719	0.337778975781665	0.33777897505811
0.33566679631326	0.335666796210006	0.335666795846191

k_Gauss-Seidel	k_Jacobi
7	12

norma_Gauss-Seidel	norma_Jacobi
9.32344924042816e-08	4.8840317623533e-08

Tabella risultati per n = 20:

x_Direct	x_Gauus-Seidel	x_Jacobi
0.483235935360422	0.483235841376342	0.483235886390101
0.483235935360422	0.483235899044855	0.483235886390101
0.446063940332697	0.446063924653065	0.4460639033503
0.407405065503864	0.4074050582578 4	0.407405042181361
0.378577804053791	0.378577800575872	0.378577790845548
0.35981552697588	0.359815525272765	0.359815519929902
0.348424238475304	0.348424237632653	0.34842423483392
0.341779214559493	0.341779214140388	0.341779212708136
0.337999294603625	0.337999294394627	0.337999293670142
0.33588573724473	0.335885737140369	0.33588573677602
0.334718624665744	0.334718624613598	0.334718624430894
0.334080313405781	0.334080313379717	0.334080313288232
0.333733894550127	0.333733894537097	0.333733894491321
0.333547084865033	0.333547084858519	0.333547084835623
0.333446888426869	0.333446888423612	0.333446888412162
0.333393396716275	0.333393396714646	0.333393396708921
0.33336495473499	0.333364954734176	0.333364954731313
0.333349885847086	0.333349885846679	0.333349885845247
0.333341927498632	0.333341927498428	0.333341927497712
0.33333773638386	0.333337736383758	0.3333377363834

k_Gauss-Seidel	k_Jacobi
7	12

norma_Gauss-Seidel	norma_Jacobi
9.3984080296039e-08	4.8970321575581e-08

Problema 2.7. Si consideri il sistema lineare $A_n\mathbf{x} = \mathbf{b}_n$, dove $\mathbf{b}_n = [-1, 1, -1, 1, \dots, (-1)^n]^T$, $A_n = (n+1)I_n - B_n$, I_n è la matrice identità $n \times n$ e $B_n = \left[\sin \frac{\pi i}{n+1} \sin \frac{\pi j}{n+1}\right]_{i,j=1}^{n}$.

(a) Dimostrare che, qualunque sia n, A_n è una matrice a diagonale dominante in senso stretto per righe e per colonne. Dedurre che i metodi di Jacobi e Gauss-Seidel per risolvere un sistema lineare di matrice A_n sono convergenti.

(b) Dimostrare che, qualunque sia n e qualunque sia la partizione $[n_1, n_2, \dots, n_m]$ di n che si considera, il metodo di Jacobi a blocchi $[n_1, n_2, \dots, n_m]$ è applicabile per risolvere il sistema $A_n\mathbf{x} = \mathbf{b}_n$.

(c) Fissiamo $n = 9$ e la partizione di 9 data da $[3, 3, 3]$. Immaginiamo di risolvere il sistema $A_n\mathbf{x} = \mathbf{b}_n$ con i metodi di Jacobi, Jacobi a blocchi $[3, 3, 3]$ e Gauss-Seidel innescati con il vettore nullo $\mathbf{x}^{(0)} = \mathbf{0}$. Costruire una tabella che riporti vicino ad ogni $\varepsilon \in \{10^{-1}, 10^{-2}, \dots, 10^{-10}\}$:

- i numeri K_ε^J, $K_\varepsilon^{J_B}$, K_ε^G che contano le iterazioni effettuate da Jacobi, Jacobi a blocchi $[3, 3, 3]$ e Gauss-Seidel per convergere entro la precisione ε;
- le soluzioni approssimate $\mathbf{x}_\varepsilon^J$, $\mathbf{x}_\varepsilon^{J_B}$, $\mathbf{x}_\varepsilon^G$ calcolate da Jacobi, Jacobi a blocchi $[3, 3, 3]$ e Gauss-Seidel;
- le norme ∞ degli errori $\|\mathbf{x}_\varepsilon^J - \mathbf{x}\|_\infty$, $\|\mathbf{x}_\varepsilon^{J_B} - \mathbf{x}\|_\infty$, $\|\mathbf{x}_\varepsilon^G - \mathbf{x}\|_\infty$, essendo $\mathbf{x}$ la soluzione esatta del sistema.

SOLUZIONE:

a)

LA MATRICE $A_m = (m+1) I_m - B_m$, DOVE I_m È LA MATRICE IDENTITÀ $m \times m$ E B_m HA ELEMENTI $b_{ij} = \left[\sin\left(\frac{\pi i}{m+1}\right)\sin\left(\frac{\pi j}{m+1}\right)\right]_{i,j=1}^{m}$.

L'ELEMENTO DIAGONALE DI A_m È $a_{ii} = (m+1) - b_{ii}$.
ORA $b_{ii} = \sin^2\left(\frac{i\pi}{m+1}\right)$.

ALLORA $a_{ii} = (m+1) - \sin^2\left(\frac{\pi i}{m+1}\right)$.

GLI ELEMENTI FUORI DALLA DIAGONALE SONO:

$$a_{ij} = -b_{ij} \quad \text{PER } i \neq j$$

ALLORA LA SOMMA DEGLI ELEMENTI FUORI DALLA DIAGONALE DELLA i-ESIMA RIGA SONO:

$$\sum_{j\neq i} |a_{ij}| = \sum_{j \neq i} |b_{ij}| = \sum_{j\neq i} \left|\sin\left(\frac{\pi i}{m+1}\right)\sin\left(\frac{\pi j}{m+1}\right)\right|$$

PER LA PROPRIETÀ DEL SENO (DEVE ESSERE ≤ 1) SI HA CHE $|b_{ij}| \leq 1$:

$$\sum_{j\neq i} |b_{ij}| \leq m-1$$

QUINDI:

$$|a_{ii}| = (m+1) - \sin^2\left(\frac{\pi i}{m+1}\right) > m-1$$

VERO, POICHÉ $\sin^2\left(\frac{\pi i}{m+1}\right) \leq 1 \Rightarrow (m+1) - \sin^2\left(\frac{\pi i}{m+1}\right) \geq m$ CHE È MAGGIORE STRETTO DI $m-1$.

· LA MATRICE B_m È SIMMETRICA PERCHÈ:

$$B_{i,j} = \sin\left(\frac{\pi i}{m+1}\right)\left(\frac{\pi j}{m+1}\right) = B_{j,i}.$$

QUINDI, PER COSTRUZIONE, B_m È SIMMETRICA.

LA MATRICE A_m È COSTRUITA COSÌ: $A_m = (m+1) I_m - B_m$.

- GLI ELEMENTI DELLA MATRICE IDENTITÀ I_m SONO: $\begin{cases} 1 & \text{se } i=j \\ 0 & \text{se } i \neq j \end{cases}$.

- GLI ELEMENTI DELLA MATRICE A_m SONO: $a_{i,j} = (m+1)(I_m)_{i,j} - b_{i,j}$

$$\Rightarrow a_{i,j} = \begin{cases} (m+1) - b_{i,j} & \text{se } i=j \\ -b_{i,j} & \text{se } i \neq j \end{cases}$$

VERIFICHIAMO LA SIMMETRIA:

1. SE $i=j$ GLI ELEMENTI DIAGONALI SONO UGUALI PER DEFINIZIONE:

$$a_{i,i} = (m+1) - b_{i,i}$$

2. SE $i \neq j$ GLI ELEMENTI FUORI DALLA DIAGONALE DI A_m SONO:

$$a_{i,i} = -b_{i,j}$$

MA POICHÈ B_m È SIMMETRICA, ABBIAMO: $a_{i,j} = -b_{i,j} = -b_{j,i} = a_{j,i}$.

$\Rightarrow a_{i,j} = a_{j,i} \quad \forall i,j$. QUINDI, LA MATRICE A_m È SIMMETRICA

POICHÈ A_m È SIMMETRICA, LA DOMINANZA DIAGONALE PER RIGHE IMPLICA AUTOMATI-CAMENTE LA DOMINANZA DIAGONALE PER COLONNE.

POICHÈ ABBIAMO DIMOSTRATO CHE A_m È A DIAGONALE DOMINANTE IN SENSO STRETTO SIA PER RIGHE CHE PER COLONNE, CONCLUDIAMO CHE SIA JACOBI CHE GAUSS-SEIDEL CONVERGONO

b)

NEL METODO DI JACOBI A BLOCCHI, LA MATRICE A_m È DIVISA IN BLOCCHI IN BASE ALLA PARTIZIONE $[m_1, m_2, \dots, m_m]$, DOVE $m_1+m_2+\dots+m_m=M$.

ORA $A_m = D + R$, DOVE
- D = È LA PARTE A BLOCCHI DIAGONALE
- R = BLOCCHI FUORI ALLA DIAGONALE

IL METODO DI JACOBI A BLOCCHI È APPLICABILE SE:

1. D È INVERTIBILE;
2. LA PARTIZIONE SCELTA È VALIDA, OSSIA D MANTIENE LA SUA STRUTTURA.

GRAZIE ALLA PROPRIETÀ DI DIAGONALE DOMINANTE, LA MATRICE A_m È INVERTIBILE

POSSIAMO DIVIDERE LA MATRICE A_m IN m BLOCCHI DIAGONALI $D_1, D_2, \dots, D_m$:

$$D = \begin{bmatrix} D_1 & 0 & \cdots & 0 \\ 0 & D_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_m \end{bmatrix}$$

OGNI BLOCCO D_i È UNA SOTTOMATRICE DI A_m LUNGO LA DIAGONALE PRINCIPALE, ED È INVERTIBILE POICHÈ:

1. LA DIAGONALE DI A_m È DOMINANTE IN SENSO STRETTO.
2. I SOTTOBLOCCHI DELLA MATRICE DIAGONALE DOMINANTE EREDITANO LA DOMINANZA DIAGONALE IN SENSO STRETTO E QUINDI DI CONSEGUENZA L'INVERTIBILITÀ.

PER ESEMPIO CON $m=4$ E UNA PARTIZIONE DI A_m IN DUE BLOCCHI DIAGONALI D_1 E D_2 DI DIMENSIONE 2x2 SI HA CHE:

- $$A_m = \begin{bmatrix} 4.6545 & -0.5590 & -0.5590 & -0.3455 \\ -0{,}5590 & 4.0955 & -0.9045 & -0.5590 \\ -0{,}5590 & -0.9045 & 4.0955 & -0.5590 \\ -0.3455 & -0.5590 & -0.5590 & 4.6545 \end{bmatrix}$$

- I BLOCCHI DIAGONALI: $D_1 = \begin{bmatrix} 4.6545 & -0.5590 \\ -0.5590 & 4.0955 \end{bmatrix}$, $D_2 = \begin{bmatrix} 4.0955 & -0.5590 \\ -0.5590 & 4.6545 \end{bmatrix}$

SI NOTA CHE ENTRAMBI I BLOCCHI D_1 E D_2 SONO A DIAGONALE DOMINANTE IN SENSO STRETTO.

=> I BLOCCHI SONO INVERTIBILI E IL METODO DI JACOBI A BLOCCHI È APPLICABILE PER QUESTA PARTIZIONE.

c)

```
Risultati per epsilon = 0.1
Soluzioni approssimate:
```

Soluzione_Jacobi_a_blocchi	Soluzione_Gauss_Seidel	Soluzione_Jacobi
-1.0184e-01	-1.0096e-01	-1.0096e-01
9.6500e-02	1.0168e-01	1.0358e-01
-1.0482e-01	-1.0453e-01	-1.0700e-01
1.1193e-01	1.0409e-01	1.0994e-01
-8.7455e-02	-1.0633e-01	-1.1111e-01
1.1193e-01	1.0332e-01	1.0994e-01
-1.0482e-01	-1.0398e-01	-1.0700e-01
9.6500e-02	1.0058e-01	1.0358e-01
-1.0184e-01	-1.0065e-01	-1.0096e-01

```
Numero di iterazioni (k):
```

Jacobi_a_blocchi	Gauss_Seidel	Jacobi
1	1	1

```
Norma infinito degli errori:
```

Errore_Jacobi_a_blocchi	Errore_Gauss_Seidel	Errore_Jacobi
1.5713e-02	7.1015e-03	1.2957e-02

Risultati per epsilon = 0.01

Soluzioni approssimate:

Soluzione_Jacobi_a_blocchi	Soluzione_Gauss_Seidel	Soluzione_Jacobi
-1.0089e-01	-1.0065e-01	-1.0048e-01
9.8314e-02	9.8656e-02	9.8913e-02
-1.0232e-01	-1.0166e-01	-1.0087e-01
9.8324e-02	9.7446e-02	9.7364e-02
-1.0176e-01	-1.0228e-01	-1.0052e-01
9.8324e-02	9.7285e-02	9.7364e-02
-1.0232e-01	-1.0219e-01	-1.0087e-01
9.8314e-02	9.8329e-02	9.8913e-02
-1.0089e-01	-1.0088e-01	-1.0048e-01

Numero di iterazioni (k):

Jacobi_a_blocchi	Gauss_Seidel	Jacobi
3	2	2

Norma infinito degli errori:

Errore_Jacobi_a_blocchi	Errore_Gauss_Seidel	Errore_Jacobi
1.4058e-03	9.0100e-04	2.6517e-03

Risultati per epsilon = 0.001

Soluzioni approssimate:

Soluzione_Jacobi_a_blocchi	**Soluzione_Gauss_Seidel**	**Soluzione_Jacobi**
-1.0095e-01	-1.0088e-01	-1.0093e-01
9.8190e-02	9.8309e-02	9.8226e-02
-1.0249e-01	-1.0237e-01	-1.0245e-01
9.7127e-02	9.7186e-02	9.7121e-02
-1.0302e-01	-1.0303e-01	-1.0303e-01
9.7127e-02	9.7097e-02	9.7121e-02
-1.0249e-01	-1.0249e-01	-1.0245e-01
9.8190e-02	9.8187e-02	9.8226e-02
-1.0095e-01	-1.0095e-01	-1.0093e-01

Numero di iterazioni (k):

Jacobi_a_blocchi	**Gauss_Seidel**	**Jacobi**
5	3	5

Norma infinito degli errori:

Errore_Jacobi_a_blocchi	**Errore_Gauss_Seidel**	**Errore_Jacobi**
1.4635e-04	1.9823e-04	1.3687e-04

Risultati per epsilon = 0.0001

Soluzioni approssimate:

Soluzione_Jacobi_a_blocchi	**Soluzione_Gauss_Seidel**	**Soluzione_Jacobi**
-1.0097e-01	-1.0097e-01	-1.0097e-01
9.8147e-02	9.8148e-02	9.8147e-02
-1.0255e-01	-1.0255e-01	-1.0255e-01
9.7004e-02	9.6998e-02	9.7000e-02
-1.0315e-01	-1.0316e-01	-1.0315e-01
9.7004e-02	9.6993e-02	9.7000e-02
-1.0255e-01	-1.0256e-01	-1.0255e-01
9.8147e-02	9.8141e-02	9.8147e-02
-1.0097e-01	-1.0098e-01	-1.0097e-01

Numero di iterazioni (k):

Jacobi_a_blocchi	**Gauss_Seidel**	**Jacobi**
7	5	8

Norma infinito degli errori:

Errore_Jacobi_a_blocchi	**Errore_Gauss_Seidel**	**Errore_Jacobi**
1.7362e-05	1.1193e-05	1.3475e-05

```
Risultati per epsilon = 1e-05
Soluzioni approssimate:
```

Soluzione_Jacobi_a_blocchi	Soluzione_Gauss_Seidel	Soluzione_Jacobi
-1.0098e-01	-1.0098e-01	-1.0098e-01
9.8139e-02	9.8139e-02	9.8139e-02
-1.0256e-01	-1.0256e-01	-1.0256e-01
9.6989e-02	9.6988e-02	9.6989e-02
-1.0317e-01	-1.0317e-01	-1.0317e-01
9.6989e-02	9.6988e-02	9.6989e-02
-1.0256e-01	-1.0256e-01	-1.0256e-01
9.8139e-02	9.8138e-02	9.8139e-02
-1.0098e-01	-1.0098e-01	-1.0098e-01

```
Numero di iterazioni (k):
```

Jacobi_a_blocchi	Gauss_Seidel	Jacobi
9	7	11

```
Norma infinito degli errori:
```

Errore_Jacobi_a_blocchi	Errore_Gauss_Seidel	Errore_Jacobi
2.2482e-06	6.3909e-07	1.3102e-06

Risultati per epsilon = 1e-06

Soluzioni approssimate:

Soluzione_Jacobi_a_blocchi	**Soluzione_Gauss_Seidel**	**Soluzione_Jacobi**
-1.0098e-01	-1.0098e-01	-1.0098e-01
9.8138e-02	9.8138e-02	9.8138e-02
-1.0256e-01	-1.0256e-01	-1.0256e-01
9.6987e-02	9.6987e-02	9.6987e-02
-1.0317e-01	-1.0317e-01	-1.0317e-01
9.6987e-02	9.6987e-02	9.6987e-02
-1.0256e-01	-1.0256e-01	-1.0256e-01
9.8138e-02	9.8138e-02	9.8138e-02
-1.0098e-01	-1.0098e-01	-1.0098e-01

Numero di iterazioni (k):

Jacobi_a_blocchi	**Gauss_Seidel**	**Jacobi**
12	8	14

Norma infinito degli errori:

Errore_Jacobi_a_blocchi	**Errore_Gauss_Seidel**	**Errore_Jacobi**
1.1324e-07	1.5243e-07	1.2742e-07

Risultati per epsilon = 1e-07

Soluzioni approssimate:

Soluzione_Jacobi_a_blocchi	Soluzione_Gauss_Seidel	Soluzione_Jacobi
-1.0098e-01	-1.0098e-01	-1.0098e-01
9.8138e-02	9.8138e-02	9.8138e-02
-1.0256e-01	-1.0256e-01	-1.0256e-01
9.6987e-02	9.6987e-02	9.6987e-02
-1.0317e-01	-1.0317e-01	-1.0317e-01
9.6987e-02	9.6987e-02	9.6987e-02
-1.0256e-01	-1.0256e-01	-1.0256e-01
9.8138e-02	9.8138e-02	9.8138e-02
-1.0098e-01	-1.0098e-01	-1.0098e-01

Numero di iterazioni (k):

Jacobi_a_blocchi	Gauss_Seidel	Jacobi
14	10	17

Norma infinito degli errori:

Errore_Jacobi_a_blocchi	Errore_Gauss_Seidel	Errore_Jacobi
1.5980e-08	8.6713e-09	1.2391e-08

Risultati per epsilon = 1e-08

Soluzioni approssimate:

Soluzione_Jacobi_a_blocchi	**Soluzione_Gauss_Seidel**	**Soluzione_Jacobi**
-1.0098e-01	-1.0098e-01	-1.0098e-01
9.8138e-02	9.8138e-02	9.8138e-02
-1.0256e-01	-1.0256e-01	-1.0256e-01
9.6987e-02	9.6987e-02	9.6987e-02
-1.0317e-01	-1.0317e-01	-1.0317e-01
9.6987e-02	9.6987e-02	9.6987e-02
-1.0256e-01	-1.0256e-01	-1.0256e-01
9.8138e-02	9.8138e-02	9.8138e-02
-1.0098e-01	-1.0098e-01	-1.0098e-01

Numero di iterazioni (k):

Jacobi_a_blocchi	**Gauss_Seidel**	**Jacobi**
16	11	20

Norma infinito degli errori:

Errore_Jacobi_a_blocchi	**Errore_Gauss_Seidel**	**Errore_Jacobi**
2.2676e-09	2.0682e-09	1.2051e-09

Risultati per epsilon = 1e-09
Soluzioni approssimate:

Soluzione_Jacobi_a_blocchi	Soluzione_Gauss_Seidel	Soluzione_Jacobi
-1.0098e-01	-1.0098e-01	-1.0098e-01
9.8138e-02	9.8138e-02	9.8138e-02
-1.0256e-01	-1.0256e-01	-1.0256e-01
9.6987e-02	9.6987e-02	9.6987e-02
-1.0317e-01	-1.0317e-01	-1.0317e-01
9.6987e-02	9.6987e-02	9.6987e-02
-1.0256e-01	-1.0256e-01	-1.0256e-01
9.8138e-02	9.8138e-02	9.8138e-02
-1.0098e-01	-1.0098e-01	-1.0098e-01

Numero di iterazioni (k):

Jacobi_a_blocchi	Gauss_Seidel	Jacobi
19	13	22

Norma infinito degli errori:

Errore_Jacobi_a_blocchi	Errore_Gauss_Seidel	Errore_Jacobi
1.2150e-10	1.1765e-10	2.5484e-10

```
Risultati per epsilon = 1e-10
Soluzioni approssimate:
    Soluzione_Jacobi_a_blocchi    Soluzione_Gauss_Seidel    Soluzione_Jacobi
    __________________________    ______________________    ________________

           -1.0098e-01                 -1.0098e-01             -1.0098e-01
            9.8138e-02                  9.8138e-02              9.8138e-02
           -1.0256e-01                 -1.0256e-01             -1.0256e-01
            9.6987e-02                  9.6987e-02              9.6987e-02
           -1.0317e-01                 -1.0317e-01             -1.0317e-01
            9.6987e-02                  9.6987e-02              9.6987e-02
           -1.0256e-01                 -1.0256e-01             -1.0256e-01
            9.8138e-02                  9.8138e-02              9.8138e-02
           -1.0098e-01                 -1.0098e-01             -1.0098e-01

Numero di iterazioni (k):
    Jacobi_a_blocchi    Gauss_Seidel    Jacobi
    ________________    ____________    ______

           21                15           25

Norma infinito degli errori:
    Errore_Jacobi_a_blocchi    Errore_Gauss_Seidel    Errore_Jacobi
    _______________________    ___________________    _____________

          1.7326e-11               6.6925e-12          2.4783e-11
```

Problema 2.8. Consideriamo i seguenti due casi:

- $f(x) = x^3 + 3x - 1 - e^{-x^2}$ e $[a, b] = [0, 1]$;
- $f(x) = \cos x - x$ e $[a, b] = [0, \pi]$.

Per ciascuno di questi due casi, risolvere i seguenti punti.

(a) Verificare che $f(a)f(b) < 0$.

(b) Tracciare il grafico di $f(x)$ su $[a, b]$ e verficare che $f(x)$ ha un unico zero ζ nell'intervallo (a, b).

(c) Dimostrare analiticamente che $f(x)$ ha un'unico zero ζ nell'intervallo (a, b).

(d) Costruire una tabella che riporti vicino ad ogni $\varepsilon \in \{10^{-1}, 10^{-2}, \ldots, 10^{-10}\}$:

- un'approssimazione ξ_ε di ζ, calcolata con il metodo di bisezione, che soddisfa $|\xi_\varepsilon - \zeta| \leq \varepsilon$;
- il numero d'iterazioni K_ε effettuate dal metodo di bisezione per calcolare l'approssimazione ξ_ε;
- il valore $f(\xi_\varepsilon)$.

SOLUZIONE:

a)

CALCOLIAMO $f(a)$ E $f(b)$:

1. PER $x=a=0$: $f(0)=0^3+3\cdot 0-1-e^{-0^2}=-2$

2. PER $x=b=1$: $f(1)=1^3+3\cdot 1-1-e^{-1^2}=3-e^{-1}\approx 2,6321>0$

$\Rightarrow f(0)\cdot f(1)=(-2)\cdot(2,6321)=-5,2642<0.$

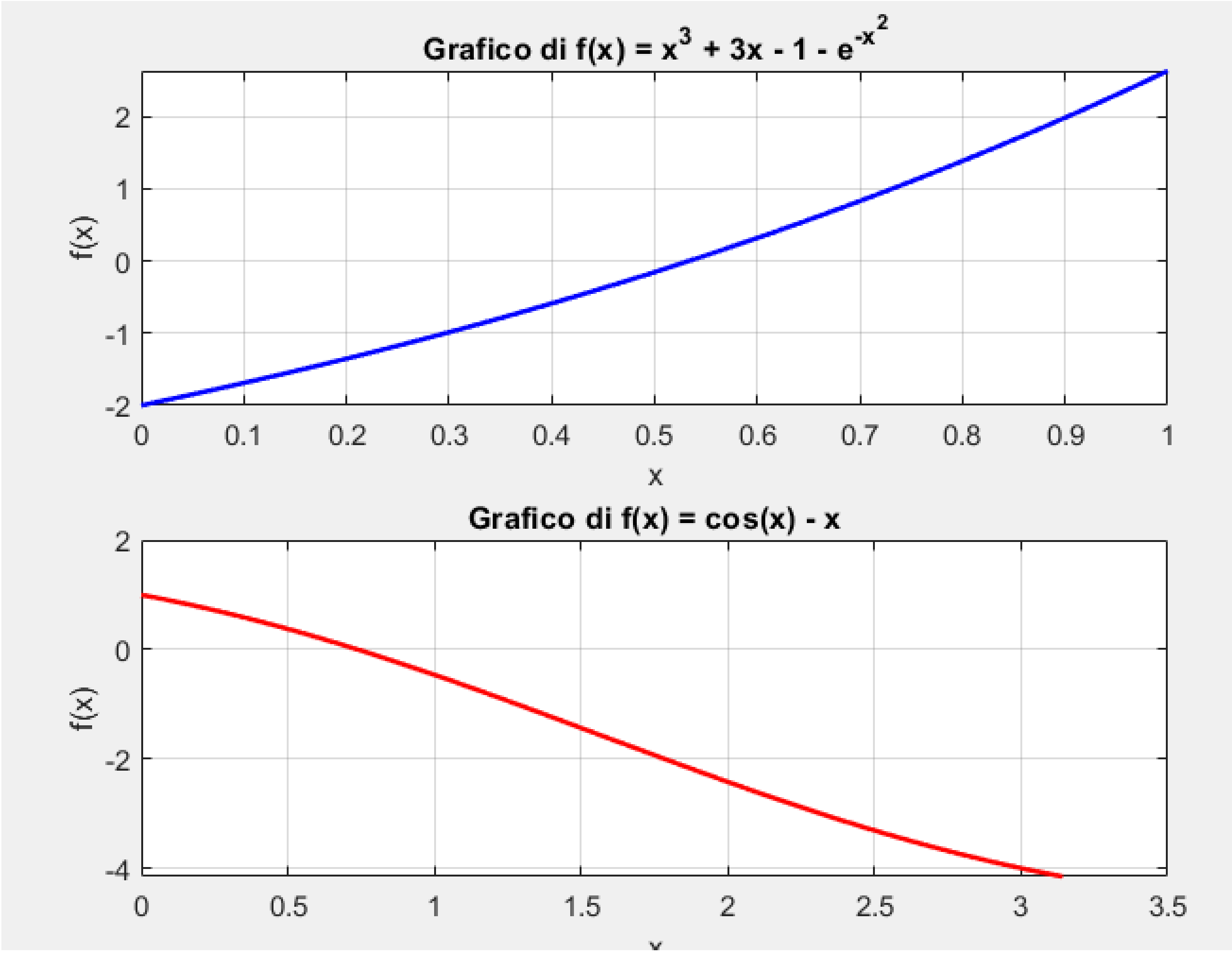
CASO 2: $f(x)=\cos(x)-x$, $[a,b]=[0,\pi]$

1. PER $x=a=0$: $f(0)=\cos(0)-0=1-0=1$

2. PER $x=b=\pi$: $f(\pi)=\cos(\pi)-\pi=-1-\pi=-1-3,1416\approx -4,1416<0$

$\Rightarrow f(0)\cdot f(\pi)=1\cdot(-4,1416)=-4,1416<0.$

b)



c)

CASO 1: $f(x) = x^3+3x-1-e^{-x^2}$, $[a,b]=[0,1]$

- $f(x)$ È LA SOMMA DI POLINOMI E FUNZIONI ESPONENZIALI $\Rightarrow$ È CONTINUA SU $[0,1]$.
- DAL PUNTO (A) ABBIAMO VISTO CHE $f(0)<0$ E $f(1)>0$ $\Rightarrow \exists$ ALMENO UNO ZERO IN $(0,1)$ PER IL TEOREMA DI BOLZANO.
- DERIVIAMO:

$$f'(x) = 3x^2+3+2xe^{-x^2}>0 \quad \forall x \in [0,1].$$

POICHÈ $f'(x)>0 \Rightarrow f(x)$ È STRETT. CRESCENTE SU $[0,1]$ E PUÒ AVERE AL PIÙ UNO ZERO.

CASO 2: $f(x)=\cos x - x$, $[a,b]=[0,\pi]$

- $f(x)$ È LA COMBINAZIONE DI FUNZIONI CONTINUE (OSSIA $\cos x$ E x) $\Rightarrow$ È CONTINUA SU $[0,\pi]$
- DAL PUNTO (A), $f(0)>0$ E $f(\pi)<0$ $\Rightarrow \exists$ ALMENO UNO ZERO IN $(0,\pi)$ PER IL TEOREMA DI BOLZANO.

- DERIVIAMO: $f'(x) = -\sin(x)-1$ CHE È SEMPRE NEGATIVA.

POICHÈ $f'(x)<0 \Rightarrow f(x)$ È STRETT. DECRESCENTE SU $[0,\pi] \Rightarrow$ PUÒ AVERE AL PIÙ UNO ZERO

ENTRAMBE LE FUNZIONI $f(x)$ HANNO UN UNICO ZERO NEI RISPETTIVI INTERVALLI.

d)

```
Caso 1: f(x) = x^3 + 3x - 1 - e^{-x^2} in [0, 1]
Epsilon     Iterazioni   Approssimazione      f(Approssimazione)
-------------------------------------------------------------
1.0e-01     4            0.5312500000         -1.0419952430e-02
1.0e-02     7            0.5351562500         7.7653125829e-03
1.0e-03     10           0.5336914062         9.3895595480e-04
1.0e-04     14           0.5334777832         -5.5864090477e-05
1.0e-05     17           0.5334892273         -2.5746125594e-06
1.0e-06     20           0.5334897041         -3.5420670641e-07
1.0e-07     24           0.5334897935         6.2119488442e-08
1.0e-08     27           0.5334897824         1.0078712531e-08
1.0e-09     30           0.5334897800         -7.6311579278e-10
1.0e-10     34           0.5334897802         -8.5501605795e-11

Caso 2: f(x) = cos(x) - x in [0, pi]
Epsilon     Iterazioni   Approssimazione      f(Approssimazione)
-------------------------------------------------------------
1.0e-01     5            0.7363107782         4.6403471699e-03
1.0e-02     9            0.7393787398         -4.9141530026e-04
1.0e-03     12           0.7389952446         1.5043574205e-04
1.0e-04     15           0.7390431815         7.0210305791e-05
1.0e-05     19           0.7390881223         -5.0025832334e-06
1.0e-06     22           0.7390855008         -6.1512370841e-07
1.0e-07     25           0.7390851731         -6.6691625000e-08
1.0e-08     29           0.7390851350         -3.0343347834e-09
1.0e-09     32           0.7390851332         2.6112001450e-11
1.0e-10     35           0.7390851332         -5.0399240337e-11
```

Problema 2.9. Un'immagine in bianco e nero viene spesso rappresentata come una matrice A a componenti in $[0,1]$: ogni componente (i,j) di A rappresenta un pixel e il valore A_{ij} rappresenta il colore del pixel (i,j), cioè una tonalità di grigio in $[0,1]$. Il valore 0 corrisponde al nero, il valore 1 corrisponde al bianco, i valori intermedi in $(0,1)$ corrispondono alle varie sfumature (toni) di grigio che diventano via via più chiari man mano che ci si sposta dallo 0 (nero) all'1 (bianco).

(a) Attraverso il comando "rand", costruire una matrice A di dimensioni 100×200 le cui componenti sono numeri casuali in $(0,1)$. Visualizzare l'immagine corrispondente alla matrice A mediante il comando "imshow(A)" e salvare l'immagine sul proprio computer.

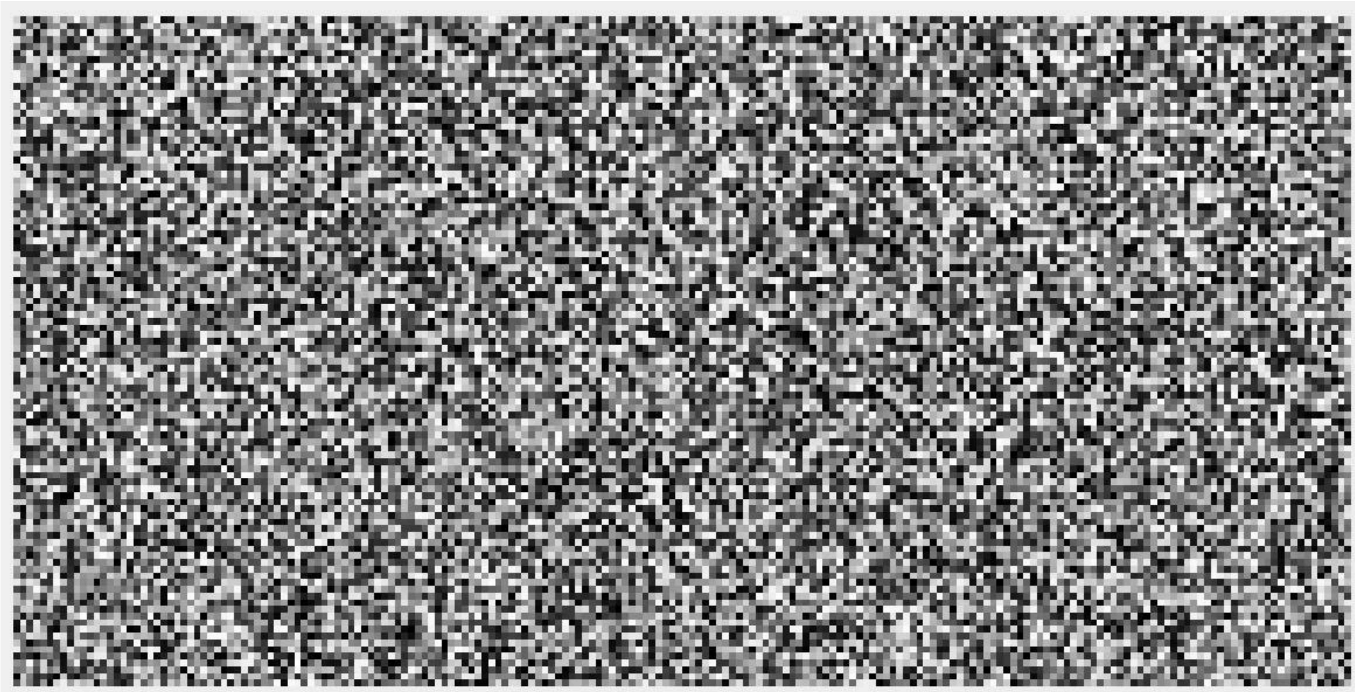
(b) Costruire la versione compressa B di A utilizzando una griglia di compressione $\left\{0, \frac{1}{n}, \frac{2}{n}, \ldots, \frac{n-1}{n}, 1\right\}$ con $n = 20$. Visualizzare e salvare l'immagine corrispondente alla matrice B.

(c) Costruire la versione compressa C di A utilizzando una griglia di compressione $\left\{0, \frac{1}{n}, \frac{2}{n}, \ldots, \frac{n-1}{n}, 1\right\}$ con $n = 5$. Visualizzare e salvare l'immagine corrispondente alla matrice C.

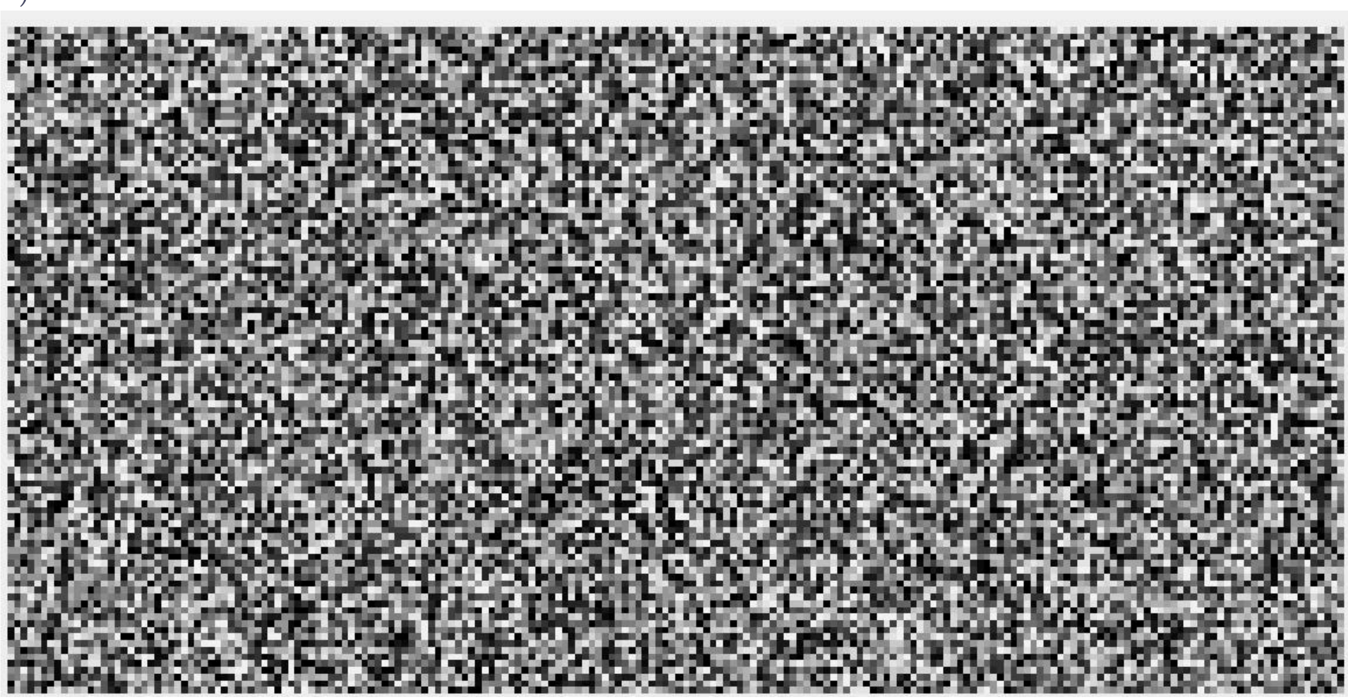
(d) Confrontare le dimensioni delle tre immagini corrispondenti ad A, B, C: che cosa si osserva?

SOLUZIONE:

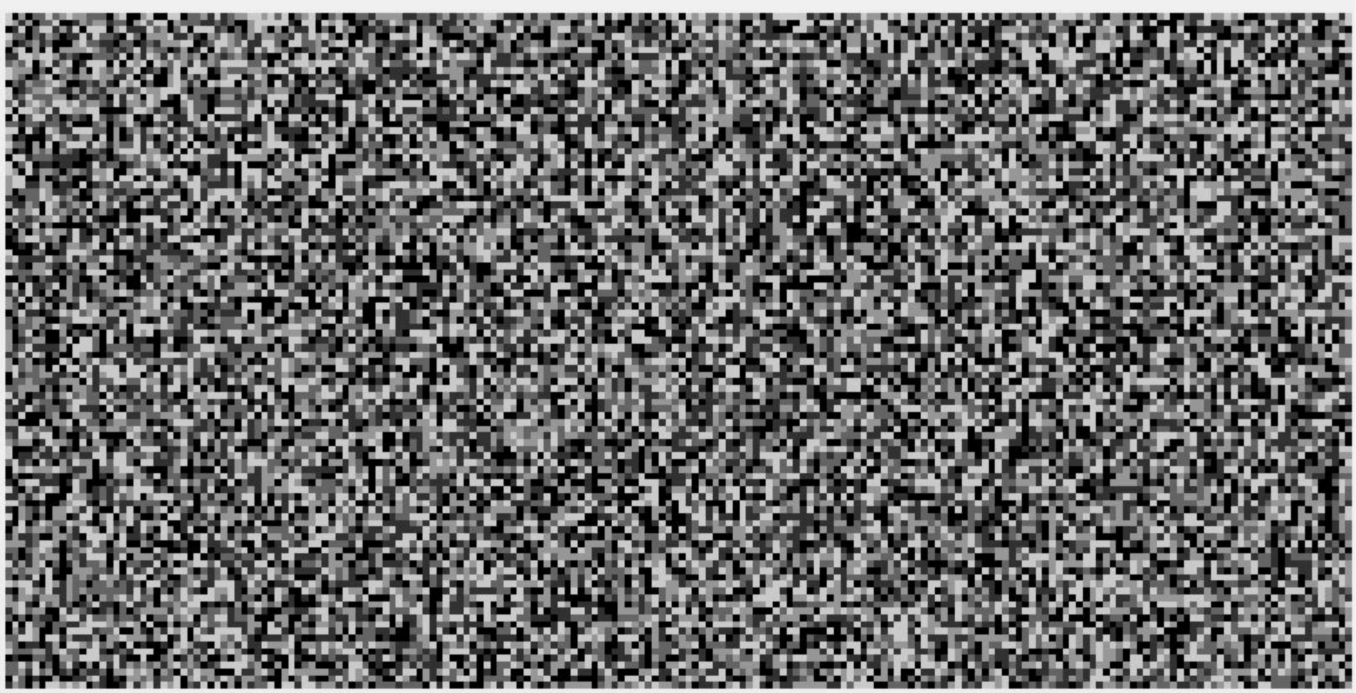
a)



b)



c)



d)

Le dimensioni delle immagini dipendono dalla quantità di informazioni salvate, ovvero il numero di pixel e la loro rappresentazione binaria.
L'immagine A ha un peso totale di : 20KB.
L'immagine B è compressa usando $n = 20$; Ha un peso totale di 15KB.
L'immagine C è compressa usando $n = 5$; Ha un peso totale di 9KB.

La compressione riduce lo spazio occupato a discapito della qualità. A valori più bassi di n, la compressione è più aggressiva, ma si osserva perdita di dettagli.

APPENDICE:

- ESERCIZIO 1:

```
function [V] = ValPol(X,Y,T)
% X = vettore a componenti reali tutti distinti
% Y = vettore a componenti reali della stessa lunghezza di X
% T = vettore contenente i punti in cui verrà calcolato p(x)
%
% output:
% V = vettore che contiene le valutazioni nei punti del vettore T del
%    polinomio p(x) interpolante i valori Y sui nodi X.

    Z = zeros(lenght(Y));
    % Z = tabelle delle differenze divise
    for i =1:length(Y)
```

```
        Z(i,1) = Y(i);
    end

    C = CNewton(X,Y);
    %C = vettore dei coefficienti di Newton

    for k = 1:length(T)
        V(k) = RH(T(k),C,X);
    end
    %V = vettore dei polinomi calcolati con t



    function [f] = CNewton(X,Y)
        Z = zeros(length(Y));
        %Z = tabelle delle differenze divise
        for i = 1:length(Y)
            Z(i,1) = Y(i);
        end
        %f(1) = Z(1,1);
        for j = 2:length(Y)
            for i = j:length(Y)
                Z(i,j) = (Z(i,j-1)-Z(j-1,j-1))/(X(i)-X(j-1));
        %       if i == j
        %           f(j) = Z(i,j);
        %       end
            end
        end
        f = diag(Z);
    end

    function [g] = RH(t,C,X)
        g = 0;
        n = length(C);
        for i = n:-1:1
            g = g * (t - X(i)) + C(i);
        end
    end
end
```

- ESERCIZIO 2:

```
function [app] = Trapezi(a,b,n,f)
% input:
% a = estremo sinistro dell'intervallo
% b = estremo destro dell'intervallo
% n = numero naturale >= 1
% f = funzione integrabile su [a,b]
%
% output:
% app = approssimazione dell'integrale su [a,b] della
% funzione f ottenuta mediante la formula dei trapezi
% di ordine n

    r = 0;
    h = (b-a)/n;
    for j = 1:(n-1)
        r = r+f(a+j*h);
    end
    app = ((f(a)+f(b))/2+r)*h;
end
```

- ESERCIZIO 3:

```
function [p0] = Estrapolazione(a,b,f,N)
% input:
% a = estremo sinistro dell'intervallo
% b = estremo destro dell'intervallo
% f = funzione definita su [a,b]
% N = vettore di numeri >= 1 tutti distinti
%
% output:
% p0 = valore estrapolato p0, dove p(x) è il polinomio d'interpolazione
%      dei dati (h(k)^2,g(k)) dove gli h(k) sono i passi di
%      discretizzazione dei g(k); i g(k) sono le formule dei
%      trapezi di ordine N(k) per approssimare l'integrale di f su [a,b].



    for k = 1:length(N)
        h(k) = (b-a)/N(k);
        g(k) = Trapezi(a,b,N(k),f);
```

```
        % g contiene i valori di tutte le ln per ogni valore n in N
    end

    p0 = ValPol(h.^2,g,0);
end
```

ESERCIZIO 4:

```
function [Sn] = CavaSimp(a,b,f,n)
% input:
% a = estremo sinistro dell'intervallo
% b = estremo destro dell'intervallo
% f = funzione integrabile su [a,b]
% n = numero naturale >= 1
%
% output:
% Sn = approssimazione dell'integrale su [a,b] della
%   funzione f ottenuta mediante la formula di Cavalieri-Simpson di ordine n

    h = (b-a)/n;
    % s1 rappresenta la prima sommatoria della formula
    s1 = 0;
    for j = 1:(n-1)
        xj = a+(j*h);
        s1 = s1+f(xj);
    end

    %s2 rappresenta la seconda sommatoria della formula
    s2 = 0;
    for j = 0:(n-1)
        x0 = a + (j*h);
        x1 = a + ((j+1)*h);
        x2 = (x0+x1)/2;
        s2 = s2 + f(x2);
    end

    % Sn rappresenta la formula di Cavalieri-Simpson di ordine n
    Sn = (h/6)*(f(a)+f(b)+2*s1+4*s2);
end
```

ESERCIZIO 5:

```
function [x,k,norma] = Jacobi(A,b,ep,x,Nmax)
% input:
% A = matrice del sistema lineare
% b = vettore dei termini noti del sistema lineare
% ep = soglia di precisione del metodo
% x = vettore d'innesco del metodo
% Nmax = numero massimo di iterazioni consentito
%
% output:
% x = vettore delle soluzioni del sistema lineare calcolato
%   con il metodo di Jacobi
% k = numero di iterazioni effettuate dal metodo per il
%   calcolo del vettore delle soluizioni
%  norma = norma del residuo a cui si arresta il metodo
    k = 0;
    z = zeros(length(A),1);
    for i = 0:Nmax-1
        r = b-(A*x);
        if norm(r,2)<=ep*norm(b,2)
            break;
        end
        for j = 1:length(r)
            z(j,1) = r(j)/A(j,j);
        end
        x = x+z;
        k = k+1;
    end
    norma = norm(r,2);
end
```

ESERCIZIO 6:

```
function [x,k,norma] = GaussSeidel(A,b,ep,x,Nmax)
% input :
% A = matrice del sistema lineare
% b = vettore dei termini noti del sistema lineare
% ep = soglia di precisione del metodo
% x = vettore d'innesco del metodo
```

```
% Nmax = numero massimo di iterazioni consentito
%
% output:
% x = vettore delle soluzioni del sistema lineare calcolato
%  con il metodo di Jacobi
% k = numero di iterazioni effettuate dal metodo per il
%  calcolo del vettore delle soluzioni
% n = norma del residuo a cui si arresta il metodo
    k = 0;
    z = zeros(length(A),1);
    for l = k:Nmax-1
        r = b-(A*x);
        if norm(r,2)<=ep*norm(b,2)
            break;
        end
        for i = 1:length(z)
            S = 0;
            for j = 1:i-1
                S = S+(A(i,j)*z(j,1));
            end
            z(i,1) = (r(i-1)-S)/A(i,i);
        end
        x = x+z;
        k = k+1;
    end
    n = norm(r,2);
end
```

ESERCIZIO 7:

```
function [x,k,norma] = JacobiABlocchi(A,b,n,ep,x,Nmax)
% input :
% A = matrice del sistema lineare
% b = vettore dei termini noti del sistema lineare
% n = partizione della dimensione di A
% ep = soglia di precisione del metodo
% x = vettore d'innesco del metodo
% Nmax = numero massimo di iterazioni consentito
%
% output:
```

```
% x = vettore delle soluzioni del sistema lineare calcolato
%  con il metodo di Jacobi
% k = numero di iterazioni effettuate dal metodo per il
%   calcolo del vettore delle soluzioni
% norma = norma del residuo a cui si arresta il metodo

    l = 1;
    M = zeros(length(A));
    % nel seguente ciclo, costruisco la matrice M
    for i = 1:length(n)
        for j = l:l+n(i)-1
            for k = l:l+n(i)-1
                M(j,k) = A(j,k);
            end
        end
        l = l+n(i);
    end



    k = 0;
    % nel seguente ciclo, applico il metodo iterativo per calcolare x
    for i = 1:Nmax
        r = b-(A*x);
        if norm(r,2)<=ep*norm(b,2)
            break;
        end
        z = M\r;
        x = x+z;
        k = k+1;
    end
    norma = norm(r,2);
end
```

ESERCIZIO 8:

```
function [app,k,val] = Bisezione(a,b,f,ep)
% input :
% a = estremo sinistro dell'intervallo
% b = estremo destro dell'intervallo
% f = funzione continua su [a,b], con f(a)f(b)<0 e con un unico zero
```

```
%  S nell'intervallo
% ep = soglia di precisione >0
%
% output :
% app = approssimazione di uno zero della funzione f sull'intervallo
%      [a,b] ottenuto con il metodo della bisezione
% k = indice di arresto del metodo
% val = valore di f(app) che sarà all'incirca pari a 0
    ak = a;
    bk = b;
    k = 0;
    while (bk-ak)>=ep
        if (f(ak)*f((ak+bk)/2))<=0
            bk = (ak+bk)/2;
        else
            ak = (ak+bk)/2;
        end
        k = k+1;
    end
    app = (ak+bk)/2;
    val = f(app);
end
```

ESERCIZIO 9:

```
function [B] = Compressione(A,n)
% input :
% A = matrice con tutte le componenti nell'intervallo [0,1]
% n = numero naturale >=1
%
% output :
% B = matrice ottenuta "comprimendo" le informazioni contenute in A



    % nel seguente ciclo, comprimo ogni componente di A in B secondo le
    % istruzioni date confrontando ogni componente con il vettore v (griglia)
    s = size(A);
    B = A;
    for i = 1:s(1)
        for j = 1:s(2)
```

```
            B(i,j) = floor(A(i,j)*10)/10;
        end
    end
end
```