# Elaborato di Calcolo Numerico 2023-2024

Autore: Jonathan Colombo Matricola: 7011579 Email: jonathan.colombo@edu.unifi.it
Autore: Matteo Pascuzzo Matricola: 7072913 Email: matteo.pascuzzo@edu.unifi.it

00/06/2024

## Esercizio 1

Dimostrare che:

$$\frac{25f(x) - 48f(x-h) + 36f(x-2h) - 16f(x-3h) + 3f(x-4h)}{12h} = f'(x) + O(h^4)$$

**Soluzione** Per verificare l'equazione basta scrivere gli sviluppi di Taylor centrati in x delle funzioni presenti:

$$f(x-h) = \tfrac{f^{(0)}(x)}{0!} \cdot (x-h-x)^0 + \tfrac{f'(x)}{1!} \cdot (x-h-x)^1 + \tfrac{f''(x)}{2!} \cdot (x-h-x)^2 + \tfrac{f'''(x)}{3!} \cdot (x-h-x)^3 + \tfrac{f^{(4)}}{4!} \cdot (x-h-x)^4 + O(h^5)$$

$$f(x-2h) = \tfrac{f^{(0)}(x)}{0!} \cdot (x-2h-x)^0 + \tfrac{f'(x)}{1!} \cdot (x-2h-h)^1 + \tfrac{f''(x)}{2!} \cdot (x-2h-x)^2 + \tfrac{f'''(x)}{3!} \cdot (x-2h-x)^3 + \tfrac{f^{(4)}(x)}{4!} \cdot (x-2h-x)^4 + O(h^5)$$

$$f(x-3h) = \tfrac{f^{(0)}(x)}{0!} \cdot (x-3h-x)^0 + \tfrac{f'(x)}{1!} \cdot (x-3h-h)^1 + \tfrac{f''(x)}{2!} \cdot (x-3h-x)^2 + \tfrac{f'''(x)}{3!} \cdot (x-3h-x)^3 + \tfrac{f^{(4)}(x)}{4!} \cdot (x-3h-x)^4 + O(h^5)$$

$$f(x-4h) = \tfrac{f^{(0)}(x)}{0!} \cdot (x-4h-x)^0 + \tfrac{f'(x)}{1!} \cdot (x-4h-h)^1 + \tfrac{f''(x)}{2!} \cdot (x-4h-x)^2 + \tfrac{f'''(x)}{3!} \cdot (x-4h-x)^3 + \tfrac{f^{(4)}(x)}{4!} \cdot (x-4h-x)^4 + O(h^5)$$

Riscrivo il numeratore sostituendo le funzioni con gli sviluppi appena ottenuti:

$$\begin{aligned}
&25f(x) - 48 \cdot (f(x) + hf'(x) + \frac{h^2 f''(x)}{2} + \frac{h^3 f'''(x)}{6} + \frac{h^4 f''''(x)}{24} + O(h^5)) \\
&+ 36 \cdot (f(x) - 2hf'(x) + \frac{4h^2 f''(x)}{2} - \frac{8h^3 f'''(x)}{6} + \frac{16h^4 f''''(x)}{24} + O(h^5)) \\
&- 16 \cdot (f(x) - 3hf'(x) + \frac{9h^2 f''(x)}{2} - \frac{27h^3 f'''(x)}{6} + \frac{81h^4 f''''(x)}{24} + O(h^5)) \\
&+ 3 \cdot (f(x) - 4hf'(x) + \frac{16h^2 f''(x)}{2} - \frac{64h^3 f'''(x)}{6} + \frac{256h^4 f''''(x)}{24} + O(h^5))
\end{aligned}$$

Proseguo lo sviluppo del numeratore:

$$
\begin{aligned}
&25f(x) - 48f(x) - 48h'f(x) - \frac{48h^2 f''(x)}{2} - \frac{48h^3 f'''(x)}{6} - \frac{48h^4 f''''(x)}{24} + O(h^5) \\
&+ 36f(x) - 72hf'(x) + 72h^2 f''(x) - 48h^3 f'''(x) + 24h^4 f''''(x) + O(h^5) \\
&- 16f(x) + 48hf'(x) - 72h^2 f''(x) + 72h^3 f'''(x) - 54h^4 f''''(x) + O(h^5) \\
&+ 3f(x) - 12hf'(x) + 24h^2 f''(x) - 32h^3 f'''(x) + 32h^4 f''''(x) + O(h^5) \\
&= 12hf'(x) + O(h^5)
\end{aligned}
$$

Quindi l'equazione iniziale diventa:

$$\frac{12hf'(x) + O(h^5)}{12h} = f'(x) + O(h^4)$$

come volevamo dimostrare.

## Esercizio 2

La funzione:

$$f(x) = 1 + x^2 + \frac{log(|3(1-x)+1|)}{80}, x \in [1, 5/3]$$

ha un asintoto in x = 4/3, in cui tende a $\infty$. Graficarla in Matlab, utilizzando x = linspace(1, 5/3, 100001)

**Soluzione**

Di seguito è riportato il codice Matlab e il grafico ottenuto dall'esecuzione del programma:

```
clc, clearvars, close all
x = linspace(1,5/3, 100001);
y = 1 + x.^2 + (log(abs(3*(1 - x) + 1))/80);
plot(x,y);
grid on;
xlabel('ascissa x');
ylabel('ordinata f(x)');
title('Grafico della funzione f(x)');
[min_value, min_index] = min(y);
min_x = x(min_index);
disp(['Il minimo della funzione si verifica in x = ', num2str(min_x), ' con valore f(x) = ', num2str(min_value)]);
x_right = 4/3 + 0.001;
x_left = 4/3 - 0.001;
lim_right = 1 + x_right^2 + log(abs(3*(1 - x_right) + 1))/80;
lim_left = 1 + x_left^2 + log(abs(3*(1 - x_left) + 1))/80;
disp(['Limite della funzione mentre x si avvicina a 4/3 da destra: ', num2str(lim_right)]);
disp(['Limite della funzione mentre x si avvicina a 4/3 da sinistra: ', num2str(lim_left)]);
```

Codice Matlab esercizio 2

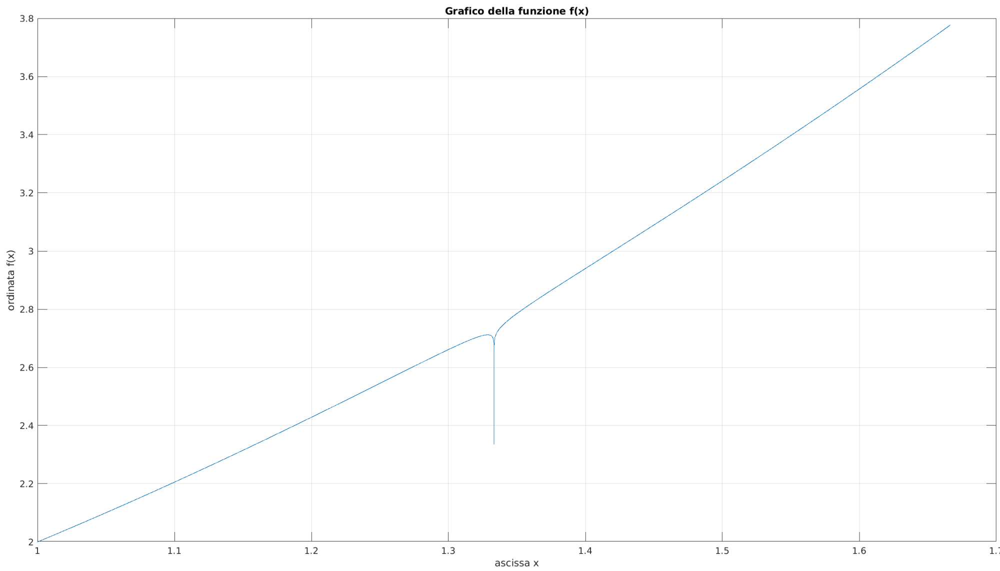


Grafico della funzione

# Esercizio 3

Spiegare in modo esaustivo il fenomeno della cancellazione numerica. Fare un esempio che la illustri, spiegandone i dettagli.

**Soluzione**

La cancellazione numerica consiste nella perdita di cifre significative nel risultato derivante dalla somma di addendi quasi opposti. Questo rispecchia il malcondizionamento di questa operazione. Infatti, se x e y sono i due numeri da sommare, il numero di condizionamento è dato da:

$$k = \frac{|x1| + |x2|}{|x1 + x2|}$$

**Esempio:** Abbiamo i seguenti due numeri reali p1 e p2: p1 = 0.12345789, p2 = 0.12345666 la cui differenza d vale:

$$d = p1 - p2 = 0.00000123 = 1.23 \cdot 10^{-6}$$

Supponiamo che l'architettura del calcolatore ci permetta di memorizzare solo le prime 6 cifre dopo la virgola, quindi i due numeri per essere rappresentati all'interno del calcolatore, verrebbero troncati appena dopo la sesta cifra: t1 = 0.123457 e t2 = 0.123456 Effettuando la sottrazione e riscrivendo il risultato nella notazione floating point si ha:

$$dt = t1 - t2 = 0.000001 = 1 \cdot 10^{-6}$$

che è un risultato diverso rispetto a d.

# Esercizio 4

Scrivere una function Matlab che implementi in modo efficiente il metodo di bisezione.

**Soluzione:** di seguito è riportato il codice Matlab.

```
%Esercizio 4 Scrivere una function Matlab che implementi in modo efficiente
% il metodo di bisezione.
function [x, it, count] = bisezione( a, b, f, tolx )
%
% x = bisezione( a, b, f, tolx ) Metodo di bisezione per calcolare
% una radice di f(x), interna ad [a,b], con tolleranza tolx.
%
if a >= b
    error('Estremi intervallo errati');
end
if tolx<=0
    error('Tolleranza non appropriata');
end
count = 0;
fa = feval(f,a);
fb = feval(f,b);
if fa*fb>=0
    error('Intervallo di confidenza non appropriato')
end
imax = ceil( log2(b-a)-log2(tolx) );
if imax<1
    x = (a+b)/2;
    return
end
for i = 1:imax
    x = (a+b)/2;
    fx = feval( f, x );
    f1x = abs(fb-fa)/(b-a);
    count=count+2;
    if abs(fx)<=tolx*f1x
        break
    elseif fa*fx<0
        b = x; fb = fx;
    else
        a = x; fa = fx;
    end
end
it =  i;
return
```

Codice Matlab metodo di bisezione

Scrivere function Matlab distinte che implementino efficientemente i metodi di Newton e delle secanti per la ricerca degli zeri di una funzione f (x).

**Soluzione:**

Di seguito è riportato il codice Matlab del metodo di Newton.

```
function [x, it, count] = newton( f, f1, x0, tolx, maxit )

```

```
%[x, it, count] = newton( f, f1, x0, tolx, maxit)
%Metodo di Newton per determinare una approssimazione
%della radice di f(x)=0 con tolleranza (mista) tolx, a
%partire da x0, entro maxit iterationi (default = 100).
%f1 implementa f%(x) mentre in uscita flag vale -1 se
%tolleranza non soddisfatta entro maxit iterate o
%la derivata si annulla, altrimenti ritorna il numero
%di iterazioni richieste.

if maxit <0
    maxit=100;
end
if tolx<0
    error('Tolleranza non idonea');
end
it = 0;
count=0;
x = x0 ;
for i =1: maxit
    x0 = x ;
    fx = feval (f , x0 );
    f1x = feval ( f1 , x0 );
    count=count+2;
    if f1x==0
        break
    end
    x = x0 - (fx / f1x) ;
    %x = x0 - m *(fx / f1x) ; riga da scommentare per il metodo di newton modificato,
                           % dell'esercizio 7 con m = 5
    if abs (x - x0 ) <= tolx *(1+ abs ( x ))
        break
    end
    it = it + 1;
end
if ( abs (x - x0 ) > tolx *(1+ abs ( x )))
    disp ('Il metodo non converge ')
end
end
```

Codice Matlab metodo di Newton

Di seguito è riportato il codice Matlab del metodo delle secanti.

```
function [x,it,count] = secanti(f,x0,x1,maxIt,tolx)
% [x, it, count] = secanti(f,x0,x1,maxIt,tolx)
% Calcola uno zero della funzione f usando il metodo delle secanti.
% Input: f - funzione di cui voglio determinare la radice,
% x0 - prima approssimazione iniziale della radice x1
% - seconda approssimazione iniziale della radice,
% maxIt - numero massimo d'iterazioni [DEFAULT 100],
% tolx - tolleranza [DEFAULT 10^-3]
```

```
% Output: x - approssimazione della radice,
%it - numero di iterazioni eseguite,
%count - numero di valutazioni funzionali eseguite

if maxIt <0
    maxIt=100; end
if tolx<0
    tolx=1e-3; end
count=0;
f0=feval(f,x0);
f1=feval(f,x1);
count=count+2;
if f1==0
    x=x1; return; end
x=(x0*f1-x1*f0)/(f1-f0);
for i=1:maxIt
    if abs(x-x1)<= tolx*(1+abs(x1))
        break
    end
    x0=x1;
    f0=f1;
    x1=x;
    f1= feval(f,x);
    count=count+1;
    if f1==0
        break
    end
    x=(x0*f1-x1*f0)/(f1-f0);
end
it=i;
if abs(x-x1)>tolx*(1+abs(x1))
    disp('Il metodo non converge');
end
end
```

Codice Matlab metodo delle secanti

# Esercizio 6

Utilizzare le function dei precedenti esercizi per determinare una approssimazione della radice della funzione:

$$f(x) = e^x - cos(x),$$

per tol = $10^{-3}$ , $10^{-6}$ , $10^{-9}$ , $10^{-12}$ , partendo da x0 = 1 (e x1 = 0.9 per il metodo delle secanti). Per il metodo di bisezione, usare l'intervallo di confidenza iniziale [-0.1, 1]. Tabulare i risultati, in modo da confrontare il costo computazionale di ciascun metodo.

**Soluzione:**

Risultati dei metodi di Secanti, Bisezione e Newton

| Metodo | Tolleranza | Radice | Iterazioni | Numero di Valutazioni Funzionali |
|---|---|---|---|---|
| Secanti | $10^{-3}$ | 1.1522e-06 | 6 | 7 |
| Secanti | $10^{-6}$ | 2.0949e-16 | 8 | 9 |
| Secanti | $10^{-9}$ | 2.0949e-16 | 8 | 9 |
| Secanti | $10^{-12}$ | -1.2557e-17 | 9 | 10 |
| Bisezione | $10^{-3}$ | 0.00097656 | 9 | 38 |
| Bisezione | $10^{-6}$ | 9.5367e-07 | 19 | 38 |
| Bisezione | $10^{-9}$ | 9.3132e-10 | 29 | 58 |
| Bisezione | $10^{-12}$ | 9.0949e-13 | 39 | 78 |
| Newton | $10^{-3}$ | 2.8423e-09 | 5 | 10 |
| Newton | $10^{-6}$ | 3.5748e-17 | 6 | 12 |
| Newton | $10^{-9}$ | 3.5748e-17 | 6 | 12 |
| Newton | $10^{-12}$ | 3.5748e-17 | 6 | 12 |

## Esercizio 7

Applicare gli stessi metodi e dati del precedente esercizio, insieme al metodo di Newton modificato, per la funzione

$$f(x) = e^x - cos(x) + sin(x) - x(x + 2)$$

Tabulare i risultati, in modo da confrontare il costo computazionale e l'accuratezza di ciascun metodo. Commentare i risultati ottenuti.

**Soluzione:**

La molteplicità per il metodo di Newton è stata calcolata osservando in quale ordine di derivazione la funzione non si annulla per x = 0. Calcolo delle derivate:

$$f(x) = e^x - cos(x) + sin(x) - x(x + 2)$$

$$f'(x) = e^x + sin(x) + cos(x) - 2x - 2$$

$$f''(x) = e^x + cos(x) - sin(x) - 2$$

$$f'''(x) = e^x - sin(x) - cos(x)$$

$$f^{(4)}(x) = e^x - cos(x) + sin(x)$$

$$f^{(5)}(x) = e^x + sin(x) + cos(x)$$

Risultati dei metodi di Secanti, Bisezione, Newton e Newton modificato

| Metodo | Tolleranza | Radice | Iterazioni | Numero di Valutazioni Funzionali |
|---|---|---|---|---|
| Secanti | $10^{-3}$ | 0.005576 | 33 | 34 |
| Secanti | $10^{-6}$ | -0.0010403 | 61 | 62 |
| Secanti | $10^{-9}$ | -0.001075 | 89 | 90 |
| Secanti | $10^{-12}$ | -0.0010751 | 100 | 101 |
| Bisezione | $10^{-3}$ | 0.0375 | 3 | 6 |
| Bisezione | $10^{-6}$ | 0.003125 | 5 | 10 |
| Bisezione | $10^{-9}$ | 0.0011163 | 31 | 62 |
| Bisezione | $10^{-12}$ | 0.0011163 | 32 | 64 |
| Newton | $10^{-3}$ | 0.0039218 | 25 | 50 |
| Newton | $10^{-6}$ | -8.0477e-05 | 42 | 84 |
| Newton | $10^{-9}$ | -8.0477e-05 | 42 | 84 |
| Newton | $10^{-12}$ | -8.0477e-05 | 42 | 84 |
| Newton modificato | $10^{-3}$ | 2.6016e-05 | 2 | 4 |
| Newton modificato | $10^{-6}$ | 2.6016e-05 | 2 | 4 |
| Newton modificato | $10^{-9}$ | 2.6016e-05 | 2 | 4 |
| Newton modificato | $10^{-12}$ | 2.6016e-05 | 2 | 4 |

## Esercizio 8

Scrivere una function Matlab, function x = mialu(A,b) che, data in ingresso una matrice A ed un vettore b, calcoli la soluzione del sistema lineare Ax = b con il metodo di fattorizzazione LU con pivoting parziale. Curare particolarmente la scrittura e l'efficienza della function, e validarla su un congruo numero di esempi significativi, che evidenzino tutti i suoi possibili output.

**Soluzione:** di seguito è riportato il codice Matlab.

```matlab
function x = mialu(A,b)
% x = mialu(A,b)
% Data in ingresso una matrice A ed un vettore b, calcola la soluzione del sistema lineare Ax=b con il metodo di fattorizzazione
% LU con pivoting parziale
% Input: A = matrice dei coefficienti, b = vettore dei termini noti
% Output: x = soluzione del sistema lineare
[m,n] = size(A);
if m ~= n
    error('Errore: matrice in input non quadrata');
end
if n~=length(b)
    error(' Errore: la dimensione del vettore b non coincide con la dimensione della matrice A ');
end
if size(b,2)>1
    error(' Errore: il vettore b non ha la struttura di un vettore colonna ');
end
```

```
p = (1:n).';
for i=1:n
    [mi,ki]=max(abs(A(i:n,i)));
    if mi==0
        error(' Errore: matrice singolare! ');
    end
    ki = ki+i-1;
    if ki>i
        A([i,ki],:) = A([ki,i],:);
        p([i,ki]) = p([ki,i]);
    end
    A(i+1:n,i) = A(i+1:n,i)/A(i,i);
    A(i+1:n,i+1:n) = A(i+1:n,i+1:n)-A(i+1:n,i)*A(i,i+1:n);
end
x = b(p);
for i=1:n
    x(i+1:n) = x(i+1:n)-A(i+1:n,i)*x(i);
end
for i=n:-1:1
    x(i) = x(i)/A(i,i);
    x(1:i-1) = x(1:i-1)-A(1:i-1,i)*x(i);
end
end
```

Codice Matlab metodo mialu

**Primo esempio:** Il codice restituisce correttamente un messaggio di errore nel caso in cui venga data in input una matrice singolare A. Data una matrice dei coefficienti singolare

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

lo script restituisce: "Errore: matrice singolare!".

**Secondo esempio:** Il codice restituisce correttamente un messaggio di errore nel caso in cui venga data in input una matrice non quadrata. Data una matrice dei coefficienti

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \in \mathbb{R}^{2\times 3}$$

lo script restituisce: "Errore: matrice in input non quadrata".

**Terzo esempio:** Il codice restituisce correttamente la soluzione del sistema lineare. Data una matrice dei coefficienti

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 4 \\ 6 & 7 & 8 \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

e un vettore

$$\mathbf{b} = \begin{pmatrix} 10 \\ 20 \\ 30 \end{pmatrix} \in \mathbb{R}^{3}$$

lo script restituisce le soluzioni: -0.5000, 3.0000 e 1.5000 .

**Quarto esempio:** Il codice restituisce correttamente un messaggio di errore nel caso in cui la lunghezza di b non sia compatibile con il numero delle righe/colonne della matrice A. Data una matrice

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 4 \\ 6 & 7 & 8 \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

e un vettore

$$\mathbf{b} = \begin{pmatrix} 10 \\ 20 \\ 30 \\ 40 \end{pmatrix} \in \mathbb{R}^4$$

lo script restituisce: "Errore: la dimensione del vettore b non coincide con la dimensione della matrice A".

# Esercizio 9

Scrivere una function Matlab, function x = mialdl(A,b) che, dati in ingresso una matrice sdp A ed un vettore b, calcoli la soluzione del corrispondente sistema lineare utilizzando la fattorizzazione LDL$^T$. Curare particolarmente la scrittura e l'efficienza della function, e validarla su un congruo numero di esempi significativi, che evidenzino tutti i suoi possibili output.

**Soluzione:** di seguito è riportato il codice Matlab

```
function x = mialdl(A,b)
% mialdl(A,b) calcola la soluzione del sistema lineare Ax = b con il metodo
% di fattorizzazione LDLt
% Input:
% A = matrice simmetrica definita positiva da fattorizzare
% b = vettore dei termini noti
% Output:
% x = soluzione del sistema

[m,n] = size(A);
if m~=n
    error("Errore: La matrice deve essere quadrata");
end
if A(1,1)<=0
    error('Errore: La matrice deve essere sdp');
end
A(2:n,1)=A(2:n,1)/A(1,1); %fattorizzazione LDLT
for i=2:n
    v = (A(i,1:i-1).').*diag(A(1:i-1,1:i-1));
    A(i,i) = A(i,i)-A(i,1:i-1)*v;
    if A(i,i)<=0
        error("Errore: La matrice deve essere sdp");
    end
    A(i+1:n,i) = (A(i+1:n,i)-A(i+1:n,1:i-1)*v)/A(i,i);
end
x=b;
for i=1:n
    x(i+1:n) = x(i+1:n)-(A(i+1:n,i)*x(i));
```

```
end
x = x./diag(A);
for i=n:-1:2
    x(1:i-1) = x(1:i-1)-A(i,1:i-1).'*x(i);
end
end
```

Codice Matlab metodo mialdl

**Primo esempio:** Il codice restituisce correttamente un messaggio di errore nel caso in cui venga data in input una matrice A non quadrata. Data una matrice dei coefficienti non quadrata

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \in \mathbb{R}^{2\times 3}$$

lo script restituisce: "Errore: La matrice deve essere quadrata".

**Secondo esempio:** Il codice restituisce correttamente un messaggio di errore nel caso in cui la lunghezza di b non sia compatibile con il numero delle righe/colonne della matrice A. Data una matrice

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 4 \\ 6 & 7 & 8 \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

e un vettore

$$\mathbf{b} = \begin{pmatrix} 10 \\ 20 \\ 30 \\ 40 \end{pmatrix} \in \mathbb{R}^{4}$$

lo script restituisce: "Errore: la dimensione del vettore b non coincide con la dimensione della matrice A".

**Terzo esempio:** Il codice restituisce correttamente un messaggio di errore nel caso in cui la matrice A sia simmetrica ma non definita positiva. Per questo esempio si può utilizzare uno script per osservare il comportamento corretto della funzione mialdl.

```
% Dimensioni della matrice A
n = 3;

% Generazione di una matrice simmetrica ma non definita positiva
A = randn(n,n);
A = 0.5 * (A + A'); % Garantisce la simmetria

disp('Matrice A:');
disp(A);

% Lunghezza desiderata del vettore dei termini noti b
lunghezza_b = 3;

% Generazione del vettore dei termini noti b
b = rand(lunghezza_b, 1);

disp('vettore dei termini noti');
disp(b);

```

```
% Chiamata alla funzione mialdl per calcolare la soluzione del sistema lineare Ax = b
x = mialdl(A, b);
```

Codice Matlab terzo esempio

**Quarto esempio:** Il codice restituisce correttamente un messaggio di errore nel caso in cui la matrice A definita positiva ma non simmetrica. Data una matrice

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \in \mathbb{R}^{3\times3}$$

e un vettore

$$\mathbf{b} = \begin{pmatrix} 10 \\ 20 \\ 30 \end{pmatrix} \in \mathbb{R}^3$$

lo script restituisce: "Errore: matrice A non simmetrica".

**Quinto esempio:** Il codice restituisce correttamente le soluzioni del sistema lineare. Data una matrice dei coefficienti simmetrica definita positiva

$$A = \begin{bmatrix} 6 & 2 & 1 \\ 2 & 5 & 2 \\ 1 & 2 & 4 \end{bmatrix} \in \mathbb{R}^{3\times3}$$

e un vettore dei termini noti

$$\mathbf{b} = \begin{pmatrix} 0.9294 \\ 0.7757 \\ 0.4868 \end{pmatrix} \in \mathbb{R}^3$$

lo script restituisce le soluzioni: 0.1172, 0.0891 e 0.0478.

## Esercizio 10

Scrivere una function Matlab, function [x,nr] = miaqr(A,b) che, data in ingresso la matrice A m $\times$ n, con m $\geq$ n = rank(A), ed un vettore b di lunghezza m, calcoli la soluzione del sistema lineare Ax = b nel senso dei minimi quadrati e, inoltre, la norma, nr, del corrispondente vettore residuo. Curare particolarmente la scrittura e l'efficienza della function, e validarla su un congruo numero di esempi significativi, che evidenzino tutti i suoi possibili output.

**Soluzione:** di seguito è riportato il codice Matlab

```
function [x,nr] = miaqr(A,b)

% La funzione miaqr(A,b) calcola la fattorizzazione QR del sistema lineare
% Ax = b sovradimensionato restituendo,oltre alla fattorizzazione,la norma
% euclidea del vettore residuo.
% Input:
% A = matrice da fattorizzare
% b = vettore dei termini noti
% Output:
% x = soluzione del sistema
```

```
% nr = norma euclidea del vettore residuo

[m,n] = size(A);
if(n>m)
    error('Errore: sistema in input non sovradeterminato');
end
k = length(b);
if k~=m
    error('Errore: La dimensione della matrice e del vettore non coincidono'); end
for i = 1:n
    a = norm(A(i:m,i),2);
    if a==0
        error('Errore: La matrice non ha rango massimo');
    end
    if A(i,i)>=0
        a = -a;
    end
    v1 = A(i,i)-a;
    A(i,i) = a;
    A(i+1:m,i) = A(i+1:m,i)/v1;
    beta = -v1/a ;
    A(i:m,i+1:n) = A(i:m,i+1:n)-(beta*[1;A(i+1:m,i)])*...
        ([1;A(i+1:m,i)]'*A(i:m,i+1:n));
end
for i=1:n
    v = [1;A(i+1:m,i)];
    beta = 2/(v'*v);
    b(i:k) = b(i:k)-(beta*(v'*b(i:k)))*v;
end
for j=n:-1:1
    b(j) = b(j)/A(j,j);
    b(1:j-1) = b(1:j-1)-A(1:j-1,j)*b(j);
end
x = b(1:n);
nr = norm(b(n+1:m));
end
```

Codice Matlab metodo miaqr

**Primo esempio:** Il codice restituisce correttamente un messaggio di errore nel caso in cui la matrice dei coefficienti A abbia un numero di colonne maggiore di quello delle righe. Data una matrice dei coefficienti A

$$A = \begin{bmatrix} 6 & 2 & 1 \\ 2 & 5 & 2 \end{bmatrix} \in \mathbb{R}^{2\times 3}$$

e un vettore

$$\mathbf{b} = \begin{pmatrix} 4 \\ 1 \end{pmatrix} \in \mathbb{R}^2$$

lo script restituisce: ”Errore: sistema in input non sovradeterminato”.

**Secondo esempio:** Il codice restituisce correttamente un messaggio di errore nel caso in cui il vettore dei termini b abbia un numero diverso di righe rispetto ad A. Data una matrice dei coefficienti

$$A = \begin{bmatrix} 6 & 2 \\ 1 & 1 \\ 5 & 4 \end{bmatrix} \in \mathbb{R}^{3\times 2}$$

e un vettore

$$\mathbf{b} = \begin{pmatrix} 1 \\ 4 \\ 5 \\ 8 \end{pmatrix} \in \mathbb{R}^{2}$$

lo script restituisce: "Errore: La dimensione della matrice e del vettore non coincidono".

**Terzo esempio:** Il codice restituisce correttamente un messaggio di errore nel caso in cui la matrice A non abbia rango massimo. Data una matrice dei coefficienti

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 6 & 8 \\ 3 & 6 & 9 & 12 \\ 3 & 6 & 9 & 12 \end{bmatrix} \in \mathbb{R}^{4\times 4}$$

e un vettore

$$\mathbf{b} = \begin{pmatrix} 1 \\ 4 \\ 5 \\ 8 \end{pmatrix} \in \mathbb{R}^{4}$$

lo script restituisce: "Errore: La matrice non ha rango massimo".

**Quarto esempio:** Il codice restituisce le soluzioni corrette e norma del vettore residuo uguale a 0 data una matrice una matrice quadrata e un vettore dei termini noti in input. Data una matrice dei coefficienti

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \in \mathbb{R}^{3\times 3}$$

e un vettore

$$\mathbf{b} = \begin{pmatrix} 10 \\ 11 \\ 12 \end{pmatrix} \in \mathbb{R}^{3}$$

lo script restituisce correttamente le soluzioni: -3.3333. -2.3333 e 6.0000. Norma euclidea del vettore residuo: 0.

## Esercizio 11

Risolvere i sistemi lineari, di dimensione n,

$$A_n x_n = b_n, n = 1, ..., 15$$

di cui

$$A_n = \begin{bmatrix} 1 & 1 & \dots & \dots & \dots & 1 \\ 10 & \ddots & \ddots & \dots & \dots & 1 \\ 10^2 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & \vdots & \dots & \dots & \dots & 1 \\ 10^{n-1} & 1 & \dots & \dots & 10 & 1 \end{bmatrix} \in \mathbb{R}^{n\times n}$$

$$\mathbf{b}_n = \begin{pmatrix} n-1+\frac{10^1-1}{9} \\ n-2+\frac{10^2-1}{9} \\ n-3+\frac{10^3-1}{9} \\ \vdots \\ 0+\frac{10^n-1}{9} \end{pmatrix} \in \mathbb{R}^n$$

la cui soluzione è il vettore $x_n = (\,1, \,.\,.\,.\,, 1\,)^T \in \mathbb{R}^n$ , utilizzando la function mialu. Tabulare e commentare l'accuratezza dei risultati ottenuti, dandone spiegazione esaustiva.

**Soluzione:** di seguito è riportato il codice Matlab

```
% Generazione del sistema lineare An
n = 15; % dimensione della matrice
A = ones(n); % inizializza una matrice di uno

% Genera la matrice desiderata
for i = 1:n
    for j = 1:i
        A(i, j) = 10^(i-j); % assegna il valore 10^(i-j) alla posizione (i, j)
    end
end

disp(A); % visualizza la matrice generata

%Generazione del vettore dei termini noti
b = zeros(1, n);
for i = 1:n
    b(i) = n - i + ((10^(i) - 1 )/9);
end

disp(b); % visualizza il vettore generato
b = b.'; % vettore b trasposto
% Richiamo della funzione mialu
x = mialu(A,b);

% Stampa del risultato
disp(x);
```

Codice Matlab esercizio 11

Per spiegazione esaustiva, calcolare il numero di condizionamento. Prendo la matrice A, calcolo la norma di A e la moltiplico per la norma di A alla meno uno.

## Esercizio 12

Fattorizzare, utilizzando la function mialdlt, le matrici sdp

$$A_n = \begin{bmatrix} n & -1 & \dots & -1 \\ -1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1 \\ -1 & \vdots & -1 & n \end{bmatrix} \in \mathbb{R}^{n\times n}, n = 1, ..., 100.$$

Graficare, in un unico grafico, gli elementi diagonali del fattore D, rispetto all'indice diagonale.

**Soluzione:** di seguito è riportato il codice Matlab per la function mialdlt

```
function A = mialdlt(A)

% mialdl(A,b) effettua la fattorizzazione della matrice A con il metodo
% di fattorizzazione LDLt
% Input:
% A = matrice simmetrica definita positiva da fattorizzare
% Output:
% A = matrice fattorizata

[m,n] = size(A);
if m~=n
    error("Errore: La matrice deve essere quadrata");
end
if A(1,1)<=0
    error('Errore: La matrice deve essere sdp');
end
A(2:n,1)=A(2:n,1)/A(1,1); %fattorizzazione LDLT
for i=2:n
    v = (A(i,1:i-1).').*diag(A(1:i-1,1:i-1));
    A(i,i) = A(i,i)-A(i,1:i-1)*v;
    if A(i,i)<=0
        error("Errore: La matrice deve essere sdp");
    end
    A(i+1:n,i) = (A(i+1:n,i)-A(i+1:n,1:i-1)*v)/A(i,i);
end
return
```

Codice Matlab mialdlt

Di seguito è riportato il codice Matlab utilizzato per l'esercizio 12.

```
% Generazione del sistema lineare An
n = 100; % dimensione della matrice
listA = cell(n,1);% genero una lista di 100 elementi eterogenei

```

```
for i = 1:n
    A = -1 * ones(i);
    for j= 1:i
        A(j,j) = i; %faccio l'assegnazione del valore n-esimo sull' elemento diagonale
    end
listA{i} = A;
end

listResults = cell(n,1);
listDiag = cell(n,1);

for i = 1:n
    listResults{i} = mialdlt(listA{i});
    listDiag{i} = diag(listResults{i});
end

x = 1;
y = listDiag{1};
plot(x,y, "*");
hold on; %crea una sola finestra graficando tutte le curve
for i=2:n
    x = 1:i;
    y = listDiag{i};
    plot(x,y, "*");
end

hold off;% fa si che il grafico sia completato in un unica finestra
```

Codice Matlab esercizio 12

Di seguito è riportato il grafico della esecuzione.

## Esercizio 13

Utilizzare la function miaqr per risolvere, nel senso dei minimi quadrati, il sistema lineare sovradeterminato Ax = b, in cui

$$A = \begin{bmatrix} 7 & 2 & 1 \\ 8 & 7 & 8 \\ 7 & 0 & 7 \\ 4 & 3 & 3 \\ 7 & 0 & 10 \end{bmatrix},$$

$$\mathbf{b} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}$$

dove viene minimizzata la seguente norma pesata del residuo r = (r1 , . . . , r5 $)^T$ :

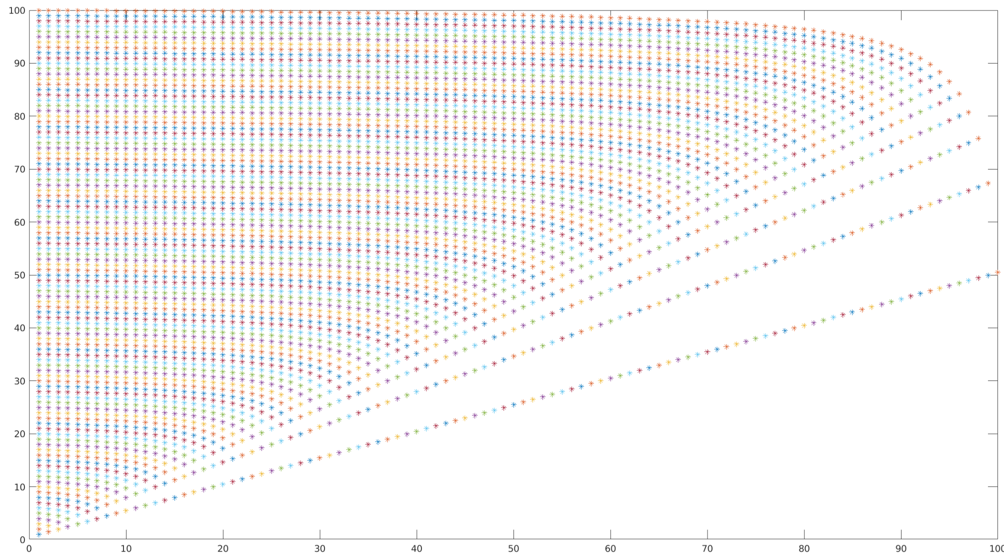


Grafico della funzione

$$p_w^2 := \sum_{i=1}^{5} w_i r_i^2,$$

con w1 = w2 = 0.5, w3 = 0.75, w4 = w5 = 0.25. Dettagliare l'intero procedimento, calcolando, in uscita, anche $p_w$.

**Soluzione:** Soluzione calcolata: 0.1645 -0.1326 0.3392

Norma calcolata: 3.0264

# Esercizio 14

Scrivere una function Matlab, [x,nit] = newton(fun,x0,tol,maxit) che implementi efficientemente il metodo di Newton per risolvere sistemi di equazioni non lineari. Curare particolarmente il criterio di arresto. La seconda variabile, se specificata, ritorna il numero di iterazioni eseguite. Prevedere opportuni valori di default per gli ultimi due parametri di ingresso (rispettivamente, la tolleranza per il criterio di arresto, ed il massimo numero di iterazioni). La function fun deve avere sintassi: [f,jacobian]=fun(x), se il sistema da risolvere è f(x)=0.

**Soluzione:** di seguito è riportato il codice Matlab.

```
function [x, nit] = newton(fun, x0, tol, maxit)
% Questo metodo risolve sistemi non lineari di equazioni
% attraverso l'uso del metodo di Newton.
% Restituisce inoltre il numero di iterazioni eseguite.
% Input: fun = vettore delle funzioni, jacobian = matrice jacobiana di fun
    ,
% x0 = vettore delle approssimazioni iniziali,
% tol = tolleranza specificata, maxit = numero di iterazioni massime,
% se non specificata maxit = 1000
% Output: x = vettore delle soluzioni, nit = numero di iterazioni svolte
```

```
if(nargin == 3)
    tol = eps;
end
if(nargin == 4)
    maxit = 1000;
end
if (tol<0)
    error('Errore! Tolleranza in input minore di 0.');
end
if (maxit<=0)
    error('Errore! Massimo numero di iterazioni minore di 0.');
end
x = x0;
for nit=1:maxit
    x0 = x;
    [fx, jacobian] = feval(fun,x0);
    dx = mialu(jacobian, -fx);
    x = x + dx;
    if (norm(x-x0)<=tol*(1+norm(x0)))
        disp('Tolleranza desiderata raggiunta.');
        break
    end
end
if not(norm(x-x0)<=tol*(1+norm(x0)))
    disp('Metodo non convergente.');
end
end
```

Codice Matlab esercizio 14

## Esercizio 15

Usare la function del precedente esercizio per risolvere, a partire dal vettore iniziale nullo, il sistema non lineare derivante dalla determinazione del punto stazionario della funzione:

$$f(x) = \frac{1}{2}x^T Qx + e^T[\cos(\alpha x) + \beta \exp(-x)]$$

$$e = (1, ..., 1)^T \in R^{50},$$

$$Q = \begin{bmatrix} 4 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & 4 \end{bmatrix} \in \mathbb{R}^{50\times 50}, \alpha = 2, \beta = -1.1,$$

utilizzando tolleranze tol $= 10^{-3}$, $10^{-8}$, $10^{-13}$ (le function cos e exp sono da intendersi in modo vettoriale). Graficare la soluzione e tabulare in modo conveniente i risultati ottenuti.

**Soluzione:**

## Esercizio 16

**Soluzione:**

## Esercizio 17

**Soluzione:**

## Esercizio 18

**Soluzione:**

## Esercizio 19

**Soluzione:**

## Esercizio 20

**Soluzione:**

## Esercizio 21

**Soluzione:**

## Esercizio 22

**Soluzione:**

## Esercizio 23

**Soluzione:**

## Esercizio 24

**Soluzione:**

## Esercizio 25

**Soluzione:**

## Esercizio 26

**Soluzione:**

## Esercizio 27

**Soluzione:**

## Esercizio 28

**Soluzione:**

## Esercizio 29

**Soluzione:**

## Esercizio 30

**Soluzione:**