

Elaborato Calcolo Numerico

Alessio Santoro - 7029440

A.A. 2022/2023

Nota: Per gli esercizi che prevedono delle *function* Matlab, si specifica nella relativa risposta al quesito i file tra gli allegati a cui essa si riferisce.

1

Si vuole dimostrare la seguente espressione:

$$-\frac{1}{4}f(x-h) - \frac{5}{6}f(x) + \frac{3}{2}f(x+h) - \frac{1}{2}f(x+2h) + \frac{1}{12}f(x+3h) = hf'(x) + O(h^5)$$

Si considera lo sviluppo delle funzioni $f(x-h)$, $f(x+h)$, $f(x+2h)$, $f(x+3h)$:

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f^{(3)}(x) + \frac{h^4}{24}f^{(4)}(x) + O(h^5)$$

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f^{(3)}(x) + \frac{h^4}{24}f^{(4)}(x) + O(h^5)$$

$$f(x+2h) = f(x) + 2hf'(x) + \frac{4h^2}{2}f''(x) + \frac{8h^3}{6}f^{(3)}(x) + \frac{16h^4}{24}f^{(4)}(x) + O(h^5)$$

$$f(x+3h) = f(x) + 3hf'(x) + \frac{9h^2}{2}f''(x) + \frac{27h^3}{6}f^{(3)}(x) + \frac{81h^4}{24}f^{(4)}(x) + O(h^5)$$

Si sostituiscono le espressioni così trovate nella parte sinistra dell'equazione iniziale e si ottiene la seguente espressione:

$$\begin{aligned} & -\frac{1}{4} \left[f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f^{(3)}(x) + \frac{h^4}{24}f^{(4)}(x) + O(h^5) \right] + \\ & -\frac{5}{6} [f(x)] + \\ & +\frac{3}{2} \left[f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f^{(3)}(x) + \frac{h^4}{24}f^{(4)}(x) + O(h^5) \right] + \\ & -\frac{1}{2} \left[f(x) + 2hf'(x) + \frac{4h^2}{2}f''(x) + \frac{8h^3}{6}f^{(3)}(x) + \frac{16h^4}{24}f^{(4)}(x) + O(h^5) \right] + \\ & +\frac{1}{12} \left[f(x) + 3hf'(x) + \frac{9h^2}{2}f''(x) + \frac{27h^3}{6}f^{(3)}(x) + \frac{81h^4}{24}f^{(4)}(x) + O(h^5) \right] \end{aligned}$$

Si procede a moltiplicare i coefficienti di ogni espressione e poi raccogliere i termini che contengono le derivate dello stesso ordine, una volta raccolti i termini

assumono i seguenti valori che, stando all'equazione iniziale dovranno poi essere sommati:

$$f(x) \left[-\frac{1}{4} - \frac{5}{6} + \frac{3}{2} - \frac{1}{2} + \frac{1}{12} \right] = 0 \quad (1)$$

$$f'(x) \cdot h \left[\frac{1}{4} + \frac{3}{2} - \frac{1}{2} 2 + \frac{1}{12} 3 \right] = h f'(x) \quad (2)$$

$$f''(x) \cdot \frac{h^2}{2} \left[-\frac{1}{4} + \frac{3}{2} - \frac{1}{2} 4 + \frac{1}{12} 9 \right] = 0 \quad (3)$$

$$f^{(3)}(x) \cdot \frac{h^3}{6} \left[-\frac{1}{4} + \frac{3}{2} - \frac{1}{2} 8 + \frac{1}{12} 27 \right] = 0 \quad (4)$$

$$f^{(4)}(x) \cdot \frac{h^4}{24} \left[-\frac{1}{4} + \frac{3}{2} - \frac{1}{2} 16 + \frac{1}{12} 81 \right] = 0 \quad (5)$$

Dalle espressioni (1)...(5) e dalle proprietà degli "O-grande" di moltiplicazione per una costante segue l'asserto.

2

La doppia precisione dello standard IEEE 754 è una rappresentazione in base binaria, in forma normalizzata (1.f) che approssima per arrotondamento e occupa 64 bit, di cui 52 dedicati alla frazione (53 alla mantissa).

Si può dunque ottenere il valore della precisione di macchina (u) dalla seguente espressione, dove: $b = 2$ rappresenta la base, e $m = 53$ la mantissa:

$$u = \frac{1}{2} b^{1-m} = 2^{-53}$$

Invece **eps** è definito dalla stessa funzione **help** di Matlab come la distanza tra 1.0 e il maggior valore a doppia precisione successivo disponibile, ovvero 2^{-52} .

Si osserva infatti che, considerato il valore $x = 1 + u = 1 + 2^{-53} \neq 1$ e sia fl la funzione di floating, allora vale che $fl(x) = 1$, poichè $u = 2^{-53} < 2^{-52} = \mathbf{eps}$.

Vi è dunque un errore di rappresentazione del valore x (ε_x), determinato dalla seguente espressione:

$$\varepsilon_x = \frac{|x - fl(x)|}{|x|} = \frac{|1 + 2^{-53} - 1|}{|1 + 2^{-53}|} = \frac{|2^{-53}|}{|1 + 2^{-53}|} < |2^{-53}| = u$$

3

La cancellazione numerica è quel fenomeno in cui, sommando in aritmetica finita due numeri quasi opposti si verifica la perdita di cifre significative. Questo è dovuto all'espressione del numero di condizionamento della somma in aritmetica finita (k) che per due valori x e y è dato da:

$$k = \frac{|x| + |y|}{|x + y|}$$

Infatti, se $x \rightarrow -y$ allora $k \rightarrow \infty$ e la somma tra x e y risulta mal condizionata.

4

Sia $x^* \in \mathbb{R}$ il valore di cui si ricerca la radice sesta.

Per calcolarlo si definisce una funzione $f(x)$ come segue:

$$f(x) = x^6 - x^*$$

La cui derivata è:

$$f'(x) = 6x^5$$

La funzione $f(x)$ si annulla solo nella radice sesta di x^* , quindi avendo un'approssimazione iniziale x_0 si può applicare il metodo di Newton alla funzione $f(x)$ per ricercarne una radice che coinciderà con il valore cercato:

$$x_{i+1} = x_i + \frac{f(x_i)}{f'(x_i)} = x_i - \frac{x_i^6 - x^*}{6x_i^5} = \frac{1}{6} \left[5x_i + \frac{x^*}{x_i} \right]$$

La function che implementa il metodo presentato è contenuta nel file `radice.m`:

```
function root = radice(x)
%
%   root = radice(x)
%
%   Questa funzione calcola la radice sesta di un valore non negativo
%   attraverso il metodo iterativo di Newton utilizzando solo operazioni elementari
%
%   Input:
%       x: valore di cui si vuole calcolare la radice sesta
%   Output:
%       root; risultato del calcolo
if(x<0), error("Value x must be not negative");end
if(x==0)
    root = 0;
    return;
end

root = x;
er = 1;

while(er >= eps*(1+abs(x)))
    xi = (5*root+x/root^5)/6;
    er = abs(root - xi);
    root = xi;
end
return;
end
```

I dati sul confronto tra il risultato offerto dalla funzione e il valore $x^{1/6}$ sono contenuti nel file `table_5.txt`:

x	radice(x)	$x^{1/6}$	errore
-----	-----	-----	-----
1e-10	0.021544	0.021544	3.4694e-18
1.1288e-09	0.032268	0.032268	6.9389e-18
1.2743e-08	0.048329	0.048329	6.9389e-18
1.4384e-07	0.072385	0.072385	1.3878e-17
1.6238e-06	0.10841	0.10841	4.1633e-17
1.833e-05	0.16238	0.16238	0
0.00020691	0.2432	0.2432	2.7756e-17
0.0023357	0.36425	0.36425	5.5511e-17
0.026367	0.54556	0.54556	0
0.29764	0.81711	0.81711	1.1102e-16
3.3598	1.2238	1.2238	2.2204e-16
37.927	1.833	1.833	0
428.13	2.7453	2.7453	4.4409e-16
4832.9	4.1118	4.1118	8.8818e-16
54556	6.1585	6.1585	0
6.1585e+05	9.2239	9.2239	1.7764e-15
6.9519e+06	13.815	13.815	1.7764e-15
7.8476e+07	20.691	20.691	3.5527e-15
8.8587e+08	30.99	30.99	3.5527e-15
1e+10	46.416	46.416	7.1054e-15

5

Il seguente testo è contenuto nel file `newtonMethod.m` e rappresenta il metodo di Newton:

```
function x_n = newtonMethod(f,df, x0, tol)
%
%   x = newtonMefunthod(f,df,x0,tol, itmax)
%
%   Ricerca la radice di una funzione di cui è nota la derivata a partire
%   da un approssimazione iniziale mediante il metodo di Newton
%
%   Input:
%       f: funzione di cui si ricercano gli 0
%       df: derivata della funzione f
%       x0: approssimazione iniziale della radice
%       tol: errore assoluto ammissibile
%   Output:
%       x: approssimazione della radice di f
%       n: numero di iterazioni eseguite

%controllo valori input
if nargin ~= 4, error("Missing arguments"); end
```

```

if tol<0, error("Invalid arguments: tolerance must be non negative"); end
%if itmax<=0, error("Invalid argument: iterations number must be positive");end

fx = feval(f, x0);
dfx = feval(df,x0);

x = x0 - fx/dfx;

n = 0;
while abs(x-x0) > tol*( 1 + abs(x0))
    n = n + 1;

    x0 = x;

    fx = feval(f,x0);
    dfx = feval(df, x0);

    if dfx==0
        error("Value of derivative function is 0, invalid first approximation");
    end

    x = x0 - fx /dfx; %calcolo effettivo
end
%if n>=itmax, warning("Max number of iterations reached");end
x_n= [x,n];
return
end

```

Da qui in poi viene presentato il contenuto del file `secantsMethod.m` che rappresenta il metodo delle secanti:

```

function x_i = secantsMethod(f, x0, x1, tol)
%
%   x = secantsMethod(f,df,x0,tol, itmax)
%
%   Ricerca la radice di una funzione di cui è nota la derivata a partire
%   da un approssimazione iniziale mediante il metodo delle secanti
%
%   Input:
%       f: funzione di cui si ricercano gli 0
%       x0: prima approssimazione iniziale della radice
%       x1: seconda approssimazione iniziale della radice
%       tol: errore assoluto ammissibile
%   Output:
%       x: approssimazione della radice di f
%       i: numero di iterazioni eseguite

%controllo valori input
if nargin ~= 4, error("Missing arguments"); end
if tol<0, error("Invalid arguments: tolerance must be non negative"); end

```

```

fx0 = feval(f,x0);
fx1= feval(f,x1);

x = x1-(fx1*(x1-x0))/(fx1-fx0);

i = 1;
while abs(x-x1) > tol*( 1 + abs(x1))
    i = i+1;

    x0 = x1;
    x1 = x;
    fx0 = fx1;
    fx1 = feval(f,x1);
    if fx0 == fx1, error("Invalid initial approximations: "+ ...
        "Function assume same value in different points");end
    x = x1-(fx1*(x1-x0))/(fx1-fx0));
end
%if i >= itmax, warning("Max number of iterations reached");end
x_i = [x,i];
return
end

```

6

Nel file `results_6.txt` è contenuta la tabella dei risultati delle funzioni precedentemente mostrate:

tolerance	Newton	num iterations N	Secant	num iterations S
-----	-----	-----	-----	-----
0.001	0.73909	3	0.7391	4
1e-06	0.73909	4	0.73909	6
1e-09	0.73909	4	0.73909	7
1e-12	0.73909	5	0.73909	7

Per entrambi i metodi, la parte più costosa computazionalmente è la valutazione funzionale, dato che tutte le altre operazioni che vengono svolte sono operazioni elementari.

Il metodo di Newton esegue due valutazioni iniziali e poi due in ogni iterazione. Sia n il numero di iterazioni, il costo computazionale del metodo di Newton è dato da $2(n+1)$.

Il metodo delle secanti esegue due valutazioni iniziali e poi una per ogni iterazioni, quindi il suo costo computazionale è dato da $n+2$.

Tolleranza	Iterazioni Newton	Costo Newton	Iterazioni secanti	Costo secanti
10^{-3}	3	8	4	6
10^{-6}	4	10	6	8
10^{-9}	4	10	6	8
10^{-12}	5	12	7	9