Esame di Laboratorio di Calcolo

Francesco Giuliano Rossi, 2001973

Giugno 2025

Indice

Sommario

In questa relazione rapporterò i passaggi adottati per affrontare il problema di esame di laboratorio di calcolo. L'obiettivo del programma realizzato è di calcolare la derivata di tre funzioni in tre punti diversi usando formule diverse e valutare la precisione di ciascun metodo.

1 Traccia d'Esame

La traccia ricevuta è la seguente

La formula per la derivata prima usando le differenze simmetriche (vedi formula ii più avanti) comporta un errore di discretizzazione proporzionale a h^2. Si può trovare una formula migliore costruendo quella combinazione lineare delle formule alle differenze simmetriche d'incremento h e $2h$ che consenta di annullare il termine dominante dell' errore. Ricavare la formula ed implementarla, confrontando i risultati con quelli delle formule:

i) $f'(x) = \frac{f(x+h)-f(x)}{h}$ [errore O(h)]

ii) $f'(x) = \frac{f(x+h)-f(x-h)}{2h}$ [errore $O(h^2) = \frac{1}{6}f'''(x)h^2 + O(h^4)$ (differenze simmetriche)]

Usarlo per valutare, per tre funzioni diverse (a sua scelta) e in tre punti diversi ciascuna, la accuratezza delle due formule con particolare riferimento a problemi di precisione numerica e discutere la scelta del valore ottimale per h.

---

2 Discussione sulla Formula

2.1 Derivazione della Formula

La consegna richiede di trovare una formula per calcolare la derivata più accurata di quelle date costruendo una combinazione lineare a partire dalla seconda equazione, usando incrementi di h e $2h$. La prima formula data è conosciuta come 'forward differencing' dato che si usa un termine $x + h$ leggermente maggiore di x, mentre si può ottenere lo stesso risultato usando la cosiddetta 'backward differencing', nel quale si prende un termine $x - h$, leggermente inferiore di x. Scritto in formule, questi sono:

$$f'(x) = \frac{f(x+h) - f(x)}{h} \quad \text{Forward Differencing}$$
$$f'(x) = \frac{f(x) - f(x-h)}{h} \quad \text{Backward Differencing}$$

In questa consegna verranno usati entrambi i metodi, ma il focus sarà principalmente sul metodo del 'forward differencing'. Tuttavia questa formula è esatta solo per equazioni lineari, per equazioni non lineari questa solo approssima il valore della derivata in tale punto. Per questo dobbiamo tenere in conto un termine d'errore dato. L'equazione di prima quindi diventa

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2}f''(\xi), \quad \xi \in (x, x+h)$$

Questo secondo termine che abbiamo introdotto rappresenta l'errore che facciamo quando non consideriamo questo piccolo valore, e quindi viene chiamato errore di troncamento. Per $h \to 0$, l'errore di troncamento dovrebbe scendere e l'approssimazione sarà ben migliore. Questo errore viene spesso rappresentato attraverso la notazione $O(h^n)$. Questa formula ha errore $O(h)$, ovvero dipende semplicemente dal valore di h, rendendola non molto accurata per h relativamente grandi. Inoltre, per il fatto che siamo obbligati a scegliere un valore di h molto piccolo, l'errore di arrotondamento diventa molto elevata.

La seconda formula, chiamata anche formula della differenza simmetrica, usa i punti $f(x + h)$ e $f(x - h)$ per rendere l'approssimazione migliore. Espandendo questi termini in serie di Taylor, si ottiene un approssimazione della formula di

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6}f'''(\xi)$$

Quindi per questa formula l'errore risulta dell'ordine di $O(h^2)$. Questo vuol dire che convergerà molto più rapidamente del primo metodo. Inoltre, essendo non obbligati a scegliere h tanto piccolo quanto per il primo

metodo, si può ottenere una precisione maggiore visto che l'errore di arrotondamento nelle operazioni sarà minore. Ora cominciamo a cercare la 'formula migliore'. Questo si fa cercando una combinazione lineare della seconda formula usando h e $2h$. Quindi posti

$$C_1 = \frac{f(x+h) - f(x-h)}{h} - \frac{h^2}{6} f''(\xi)$$
$$C_2 = \frac{f(x+2h) - f(x-2h)}{2h} - \frac{2h^2}{3} f'''(\xi)$$

e per definizione di combinazione lineare, possiamo scrivere per $A, B \in \mathbb{R}$

$$f'(x) = C = AC_1 + BC_2$$

Per trovare la 'formula migliore' vogliamo che il termine di h^2 si annulli, e che per semplicità il coefficiente di $f(x)$ è 1. Messi in sistema

$$\begin{cases} \frac{A}{6} + \frac{2}{3}B = 0 \\ A + B = 1 \end{cases}$$

e risolvendo si ottiene che $A = \frac{4}{3}$ e $B = -\frac{1}{3}$ e così otteniamo la formula

$$f'(x) = \frac{8(f(x+h) - f(x-h)) - (f(x+2h) + f(x-2h))}{12h} + \frac{h^4}{30} f^{(5)}(\xi)$$

il che avrà errore $O(h^4)$ e quindi convergerà molto più velocemente degli altri metodi essendo che l'errore di troncamento sarà molto minore, e quindi la formula in sé più accurata. Inoltre, a causa del fatto che si può scegliere h più grande, ci sarà un effetto dell'errore di arrotondamento minore. Questo metodo si chiama 'L'Estrapolazione di Richardson'.
L'errore di troncamento non è l'unica fonte di errore. C'è anche l'errore di arrotondamento da tenere in considerazione che viene dal come i computer gestiscono le operazioni tra valori che possono avere una grande differenza tra di loro. Per quando $h \to 0$ stiamo manipolando numeri con grandi differenze, alcune volte con differenze così grandi che il computer non può tenerle in conto. Per questo per $h \to 0$ l'errore diventa più grande e non possiamo direttamente prendere un h molto piccolo.

3 Implementazione

Le tre funzioni scelte sono

$$f_1(x) = (x^4 - 2x^2 + 5)(\sin(x))^2 + x^3 \sin(x)$$
$$f_2(x) = 3\cos(2x) + 5x$$
$$f_3(x) = 4\log_{10}(x) + \frac{1}{x^2 + 1}$$

In questo modo si può studiare il carattere di una funzione polinomiale dispari con funzione seno, una funzione cosinusoidale pari e una funzione logaritmica non banale (la base 10 è stata scelta perché è più facile da calcolare al livello computazionale che il logaritmo in base naturale.)
Queste funzioni sono state messe in un module chiamato 'funzioni' e ciascuna è stata definita con un procedure del tipo function.

```
module funzioni
implicit none
integer, parameter ::rk = SELECTED_REAL_KIND(3)
real, parameter ::pi = 4*atan(1.d0)

contains
function f1(x) result(res)
    real(kind=rk)::x,res
    res = (x**4-2*x**2+5)*(sin(x))**2+x**3*sin(x) !
    funzione pari con seno
end function

function f2(x) result(res)
    real(kind=rk)::x, res
    res = 3*cos(2*(x))+5*(x) !funzione dispari con coseno
end function

function f3(x) result(res)
    real(kind=rk)::x, res
    res = 4*log10(x)+1/(1+x**2) !funzione logaritmica
end function
end module
```

I tre punti scelti sono

```
    x = [2.34654, 7.36143, 50.0, &
         pi, 8.8, 7.5, &
        -pi,-8.8, 0.01]
```

Questi numeri sono stati pensati tenendo in mente il carattere pari o dispari della funzione. La prima colonna di questa 'matrice' (in realtà un array a rango 1, di dimensione 9), rappresenta i punti provati sulla prima funzione. Inoltre, i primi numeri delle prime due 'colonne' sono i valori dove la funzione si dovrebbe annullare (o approssimano al meglio possibile).
Per calcolare la derivata nel punto, si crea una subroutine che usa le equazioni trovate nella sezione prima e le formule nel modulo 'funzioni' per calcolare la derivata usando ciascun metodo. Il codice è come segue

```
module derivate
use funzioni
use err_check
implicit none

contains
subroutine df1(x, h, derivvalue, counter)
    real(kind=rk),intent(in)::x,h, derivvalue
    integer, intent(in)::counter
    real(kind=rk):: deriv1, deriv2, deriv3, err1, err2,
    err3, deriv4, min_err
    character(len=11)::metodo

    !calcolo della derivata con i tre metodi
    deriv1 = (f1(x+h)-f1(x))/h
    deriv4 = (f1(x)-f1(x-h))/h
    deriv2 = (f1(x+h)-f1(x-h))/(2*h)
```

```
121     deriv3 = (8*(f1(x+h)-f1(x-h))-(f1(x+2*h)-f1(x-2*h)))
        /(12*h)
122
123     !calcolo dell'errore assoluto
124     err1 = deriv1 - derivvalue
125     err2 = deriv2 - derivvalue
126     err3 = deriv3 - derivvalue
127
128     min_err = min(abs(err1), abs(err2), abs(err3))
129
130     if(min_err == abs(err1)) then
131         metodo = 'Incremento'
132     else if(min_err == abs(err2)) then
133         metodo = 'Simmetrica'
134     else if(min_err == abs(err3)) then
135         metodo = 'Richardson'
136     end if
137
138     call err_checker(min_err, h, counter, metodo)
139     !scrittura di tutto su un file di testo distinto per
        ogni punto
140     !sono commentati cosi' non generano 40Gb di file
141     !write(unit=counter, fmt=*) h, x, deriv1, deriv2,
        deriv3, deriv4, &
142     err1, err2, err3
143 end subroutine
```

Due altre subroutine identiche sono state create per valutare un qualsiasi punto per le altre due funzioni. Sono state suddivise per poter facilmente differenziare tra le tre funzioni scelte. Questa subroutine, dopo aver eseguito i calcoli, calcola l'errore (calcolato a mano e inserita nell'array chiamato 'derivvalue' noto e passato alla subroutine con la variabile derivvalue) e lo scrive su un file 'fort.counter' dove 'counter' rappresenta un numero unico passato alla subroutine che dipende dal punto preso in considerazione, come si può vedere più avanti. In questo modo si separano i risultati di ogni punto per fare un'analisi dati più approfondita. I comandi di write sono stati commentati tutti a parte il file 'fort.20', visto che il focus di questo programma è di trovare il valore ottimale di h.
Prima di scrivere sul file, la subroutine verifica quale degli errori è il più vicino allo zero attraverso la funzione intrinseca 'min()'. In base a quale metodo è più accurato, i dati vengono passati alla subroutine 'err_checker'. Il codice per questa subroutine è:

```
24 module err_check
25 use funzioni
26 implicit none
27
28 real(kind=rk), dimension(9), save::min_err_fp=1000
29 real(kind=rk), dimension(9), save::h_optimal_fp=0.0
30 character(len=11), dimension(9), save::metodo_migliore_fp
31
32 contains
33 subroutine err_checker(min_err, h, counter, metodo)
34     real(kind=rk), intent(in):: min_err, h
35     integer, intent(in)::counter
36     character(len=11)::metodo
37     !per associare ad un elemento dell'array 'min_err_fp'
        a partire dal counter
38     integer::id
```

```
39
40      id = counter-9
41      !check per vedere se l'errore e' minore di quello
        globale si usa il valore di
42      !id per prendere il valore minimo corrispondente al
        counter e verificare se
43      !l'errore calcolato da quella iterazione do e' minore
         di quello globale salvato
44      select case (counter)
45      case(10)
46          if (min_err<min_err_fp(id)) then
47              min_err_fp(id) = min_err
48              h_optimal_fp(id) = h
49              metodo_migliore_fp(id)=metodo
50          end if
51      case(11)
52          if (min_err<min_err_fp(id)) then
53              min_err_fp(id) = min_err
54              h_optimal_fp(id) = h
55              metodo_migliore_fp(id)=metodo
56          end if
57      case(12)
58          if (min_err<min_err_fp(id)) then
59              min_err_fp(id) = min_err
60              h_optimal_fp(id) = h
61              metodo_migliore_fp(id)=metodo
62          end if
63      case(13)
64          if (min_err<min_err_fp(id)) then
65              min_err_fp(id) = min_err
66              h_optimal_fp(id) = h
67              metodo_migliore_fp(id)=metodo
68          end if
69      case(14)
70          if (min_err<min_err_fp(id)) then
71              min_err_fp(id) = min_err
72              h_optimal_fp(id) = h
73              metodo_migliore_fp(id)=metodo
74          end if
75      case(15)
76          if (min_err<min_err_fp(id)) then
77              min_err_fp(id) = min_err
78              h_optimal_fp(id) = h
79              metodo_migliore_fp(id)=metodo
80          end if
81      case(16)
82          if (min_err<min_err_fp(id)) then
83              min_err_fp(id) = min_err
84              h_optimal_fp(id) = h
85              metodo_migliore_fp(id)=metodo
86          end if
87      case(17)
88          if (min_err<min_err_fp(id)) then
89              min_err_fp(id) = min_err
90              h_optimal_fp(id) = h
91              metodo_migliore_fp(id)=metodo
92          end if
93      case(18)
94          if (min_err<min_err_fp(id)) then
95              min_err_fp(id) = min_err
96              h_optimal_fp(id) = h
97              metodo_migliore_fp(id)=metodo
```

```
98          end if
99      end select
100 end subroutine
101 end module
```

Ho usato 'select case' per evitare l'uso di tanti if. Questo rende il programma più efficiente, perché usando select case, trova solo il caso nella quale la condizione è vera(in questo caso quanto counter è uguale ad un certo valore), invece di provare tutte le dichiarazioni if. Inoltre, gli array 'min_err_fp' e h_optimal_fp' sono state create per contenere i valori migliori. Hanno tutti l'attributo save cosi è sicuro che il loro valore persiste tra chiamate della subroutine.
Per valutare la convergenza dei valori, ho usato un ciclo do che a ogni iterazione di i trova un corrispondente valore di h. In questo modo possiamo provare tanti h senza doverli inserire manualmente. Inoltre ho usato una variabile 'scale' per cambiare più velocemente quante iterazioni si facevano. Per tutta la relazione è stato usato un valore di $scale = 1E7$. Questo ciclo do si fa andare al contrario per partire da un numero di h grande (relativamente, si parte da un valore di $h = 0.1$) e arrivare a uno più piccolo. In questo modo, provando tutti i valori di h, possiamo trovare qual'è quello ottimale.

```
205 do i = 1*scale, 1 ,-1
206     h=i*(0.1/scale)
207     call df1(x(1), h, derivvalue(1), 10) !cella 1,1,
        funzione 1
208     call df2(x(2), h, derivvalue(2), 11) !cella 1,2,
        funzione 2
209     call df3(x(3), h, derivvalue(3), 12) !cella 1,3,
        funzione 3
210
211     call df1(x(4), h, derivvalue(4), 13) !cella 2,1,
        funzione 1
212     call df2(x(5), h, derivvalue(5), 14) !cella 2,2,
        funzione 2
213     call df3(x(6), h, derivvalue(6), 15) !cella 2,3,
        funzione 3
214
215     call df1(x(7), h, derivvalue(7), 16) ! cella 3,1,
        funzione 1
216     call df2(x(8), h, derivvalue(8), 17) !cella 3,2
        funzione 2
217     call df3(x(9), h, derivvalue(9), 18) !cella 3,3
        funzione 3
218 end do
```

I valori di derivvalue sono stati calcolate separatamente dal programma, valutando le tre derivate nei punti specifici. Le derivate di ciascuna funzione risultano essere:

$$f_1'(x) = x^3\cos(x) + 3x^2\sin(x) + 2(5 - 2x^2 + x^4)\cos(x)\sin(x) + (-4x + 4x^3)(\sin(x))^2$$

$$f_2'(x) = 5 - 6\sin(2x)$$

$$f_3'(x) = \frac{-2x}{(1+x^2)^2} + \frac{4}{x\ln(10)}$$

Queste sono state usate per calcolare i valori presente nell'array 'derivvalue'

```
219     derivvalue = [0.0, 0.00002492587, 0.03472757134, &
220                   -31.0062766803, 10.69306698751,
        0.22704715192, &
221                   31.0062766803, -0.69306698751,
        173.697796760]
```

4 Risultati

4.1 Tentativo 4 Decimali di Precisione

In questa sezione, parlerò del tentativo dove ho usato la funzione intrinseca SELECTED_REAL_KIND(4) per avere una precisione di 4 cifre decimali.

4.1.1 Convergenza delle Derivate

Durante l'esecuzione del programma, tolte il '!' davanti alle righe con un comando write, il programma genera nove file 'fort.10-fort.18', uno per ogni punto di ogni funzione. Questi file hanno rapportati i valori di: x, h, i valori delle derivate valutate usando ogni metodo, e quanto scosta dal valore atteso la derivata numerica. Da questi file, usando gnuplot sono stati generati dei grafici del valore della derivata un funzione di h. I grafici si trovano nella sezione 6.1.1. Come possiamo vedere, usando il metodo del rapporto incrementale per le derivate, risulta non molto accurato per h relativamente grandi per tutti e tre le funzioni scelte, e anche rispetto agli altri due metodi per il calcolo della derivata, converge lentamente. Tuttavia questo non ci sorprende per il fatto che, come detto prima, l'errore per il metodo del rapporto incrementale è dell'ordine di $O(h)$ e quindi è molto sensibile a errori di troncamento. Gli altri due metodi per il calcolo delle derivate, invece, risultano essere più accurate fin da subito, anche se il metodo della differenza simmetrica soffre dello stesso problema del rapporto incrementale. Ovvero che per h relativamente grandi non è molto preciso ma poi man mano che h diminuisce converge più rapidamente.
Inoltre è interessante vedere come per h molto piccoli (tendenzialmente dopo l'ordine del 10^{-3}, 10^{-4}), tutti e tre i metodi iniziano a dare valori apparentemente a caso. Come spiegato prima, questo è dovuto a errori di arrotondamento, il che può essere evitato specificando una maggior numero di numeri decimali di precisione con SELECTED_REAL_KIND.
Per vedere quanto è distante il valore della valore calcolata numericamente e la derivata analitica, ho inserito questo frammento di codice:

```
136     err1 = deriv1 - derivvalue
137     err2 = deriv2 - derivvalue
138     err3 = deriv3 - derivvalue
139     err4 = deriv4 - derivvalue
```

dove derivvalue è il valore della derivata calcolata analiticamente e deriv1, deriv2, e deriv3, sono tutti e tre valori calcolati al momento dalla subroutine come mostrato prima. Il problema con questo metodo è che i valori analitici sono comunque soggetti ad errori di arrotondamento, non solo nel valore di deriv1, deriv2, deriv3, e deriv4, ma anche nel valore di derivvalue e dei risultati err1, err2, err3, e err4.

4.2 Tentativo 33 Decimali di Precisione

In questa sezione, discuterò il tentativo fatto con la funzione intrinseca SELECTED_REAL_KIND(33), tenendo il numero di iterazioni uguali alla prima. Questo è stato fatto principalmente per curiosità personale per vedere come avrebbe influito i calcoli e l'errore di arrotondamento.

4.2.1 Convergenza delle Derivate

Analizzando i grafici nella sezione 6.2.1 possiamo vedere, per h che tende a valori sempre più piccoli vediamo chiaramente che il rapporto incrementale è il metodo che converge più lentamente di tutti. Il fatto che questo metodo non converge veloce quanto gli altri ha molto senso perché il suo errore principale non è errore di arrotondamento, ma di troncamento. Questo errore non cambierà in base al kind, è un errore innato e pertanto per h grandi avrà sempre un errore grande. Inoltre, come ci aspettavamo, il metodo della differenza simmetrica converge più velocemente del rapporto incrementale ma meno veloce dell'estrapolazione di Richardson. Una cosa che mi ha sorpreso è come l'estrapolazione di Richardson converge quasi istantaneamente. Solo guardando con molta attenzione sono riuscito a vedere che per grandi valori di h, circa dell'ordine del 10^{-1}, si vedeva leggermente una linea blu. Questa poi non si vede più a causa del fatto che l'errore è talmente piccolo che le due righe diventano uguali.
Un caso particolare che vorrei sottolineare è la terza funzione valutata nel terzo punto. Come si può notare dalla figura 18, l'estrapolazione di Richardson inizia ad assumere valori solo per $h < 0.01$. Ricordo che la funzione tre è la seguente

$$f_3(x) = 4\log_{10}(x) + \frac{1}{x^2+1} \tag{1}$$

e inoltre ricordo che per calcolare le derivate, il metodo della differenza simmetrica ed estrapolazione di Richardson devono poter valutare la funzione in $f(x-h)$. Tuttavia il dominio del logaritmo è $(0, +\infty)$ e quindi quando il programma prova a valutare $f(x-h)$ con $x = 0.01$ e $h > 0.01$ la funzione restituisce NaN, e quindi la derivata in quel punto usando questo metodo non si può calcolare. Per questo nelle figure 9 e 18 solo il metodo del 'forward differencing' assume valori prima di $h = 0.01$.
Per le altre funzioni e gli altri punti, è interessante vedere come i valori alla fine non 'esplodono' come per un kind più piccolo. Questo potrebbe essere causato dal fatto che stiamo prendendo valori di h così talmente piccoli che $x + h \cong x$. Questo, come effetto, fa venire valori identici per $f(x+h), f(x)$, e in tal modo la differenza tra i due valori viene nulla. Questo si chiama l'epsilon di macchina, e rappresenta il numero più basso che il computer può distinguere, senza che si verifichi una cancellazione di dati. Infatti, analizzando i file generati, per tanti i valori con h piccolo, il valore della derivata risulta zero. Questo rafforza ancora di più l'ipotesi proposta.

4.3 Ottimizzazione di h

In questa sezione discuterò la scelta ottimale del valore di h. Ovviamente scegliendo un kind molto alto si può prendere un valore più piccolo di h, ma al costo di dimensioni del file e complessità computazionale. Dall'altra parte non possiamo scegliere un valore di h troppo piccolo, sennò domina l'errore di troncamento. Per questo trovare il miglior valore di

h non è affatto facile. Usando il codice descritto in precedenza, ho trovato i valori ottimali di h per ogni funzione in ogni punto per un SELECTED_REAL_KIND(4) e il programma gli scrive su un file chiamato 'fort.20'. I risultati sono i seguenti:

Funzione	Punto	Valore di h	Metodo
1	1	$2.841\,603\,01 \times 10^{-2}$	Richardson
1	2	$5.100\,845\,92 \times 10^{-2}$	Richardson
1	3	$9.999\,451\,79 \times 10^{-2}$	Simmetrica
2	1	$3.482\,162\,95 \times 10^{-2}$	Richardson
2	2	$8.042\,383\,19 \times 10^{-2}$	Richardson
2	3	$9.999\,784\,08 \times 10^{-2}$	Richardson
3	1	$3.482\,162\,95 \times 10^{-2}$	Richardson
3	2	$6.613\,016\,13 \times 10^{-2}$	Richardson
3	3	$5.152\,400\,12 \times 10^{-4}$	Richardson

Mentre i valori ottimali di h per un SELECTED_REAL_KIND(33) sono:

Funzione	Punto	Valore di h	Metodo
1	1	$9.999\,999\,939\,225\,290\,290\,778\,502\,821\,922\,302\,25 \times 10^{-9}$	Richardson
1	2	$9.999\,999\,939\,225\,290\,290\,778\,502\,821\,922\,302\,25 \times 10^{-9}$	Richardson
1	3	$4.829\,999\,852\,518\,085\,390\,329\,360\,961\,914\,062\,50 \times 10^{-6}$	Incremento
2	1	$9.999\,999\,939\,225\,290\,290\,778\,502\,821\,922\,302\,25 \times 10^{-9}$	Richardson
2	2	$5.999\,999\,785\,899\,490\,234\,442\,055\,225\,372\,314\,45 \times 10^{-8}$	Richardson
2	3	$4.499\,999\,874\,951\,754\,463\,836\,550\,712\,585\,449\,22 \times 10^{-7}$	Incremento
3	1	$1.800\,000\,006\,824\,120\,646\,342\,635\,154\,724\,121\,09 \times 10^{-7}$	Incremento
3	2	$4.000\,000\,046\,744\,389\,692\,321\,419\,715\,881\,347\,66 \times 10^{-7}$	Incremento
3	3	$9.118\,999\,878\,410\,249\,948\,501\,586\,914\,062\,500\,00 \times 10^{-5}$	Richardson

Come ci si aspettava, il valore ottimale di h cambia in base al kind. Questo è dovuto ai tipi di errori che affliggono questi metodi per calcolare la derivata numericamente. Quando si aumenta il kind, si minimizza l'errore di troncamento aumentando il numero di cifre precise nel numero floating point, e quindi permette di valutare le funzioni con h sempre più piccole e quindi con valori sempre più vicini a quelli analitici. Una cosa abbastanza inaspettata era che il metodo dell'estrapolazione di Richardson non era sempre il metodo più accurato. Tuttavia, potrebbe essere che il metodo del rapporto incrementale, per alcuni valori di h, dia un valore più accurato, ma in media l'estrapolazione di Richardson era più accurata. Infatti, con un kind più basso questo metodo risulta quasi sempre il più preciso.

5 Conclusione

La maggior difficoltà incontrata è stata nel come implementare tutte le chiamate alle subroutine. Essendo che si dovevano valutare tre funzioni diverse in tre punti diverse, pensavo all'inizio che una subroutine bastasse, ma in quel modo differenziare tra quale delle tre funzioni valutare era impossibile. Condensare tutto in solo una subroutine avrebbe reso il codice più pulito e forse anche più efficiente.

6 Grafici

6.1 Tentativo 4 Decimali di Precisione

Tutti questi grafici vanno letti da destra verso sinistra. I valori di h grandi si trovano a destra mentre i valori piccoli di h sono a sinistra. In questi prossimi grafici si potrà vedere la convergenza dei metodi al variare di h.

6.1.1 Convergenza della Derivata

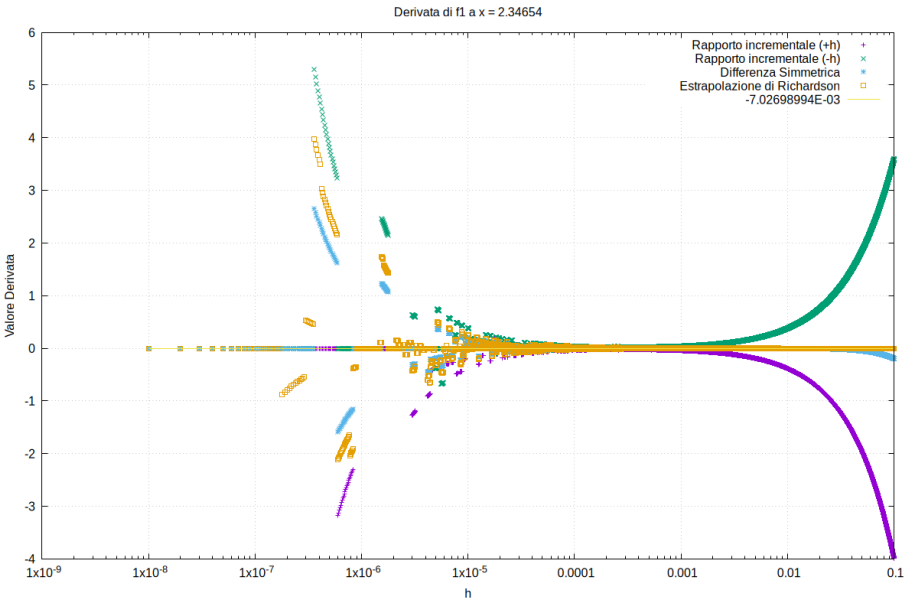

Figura 1: Punto 1, Funzione 1

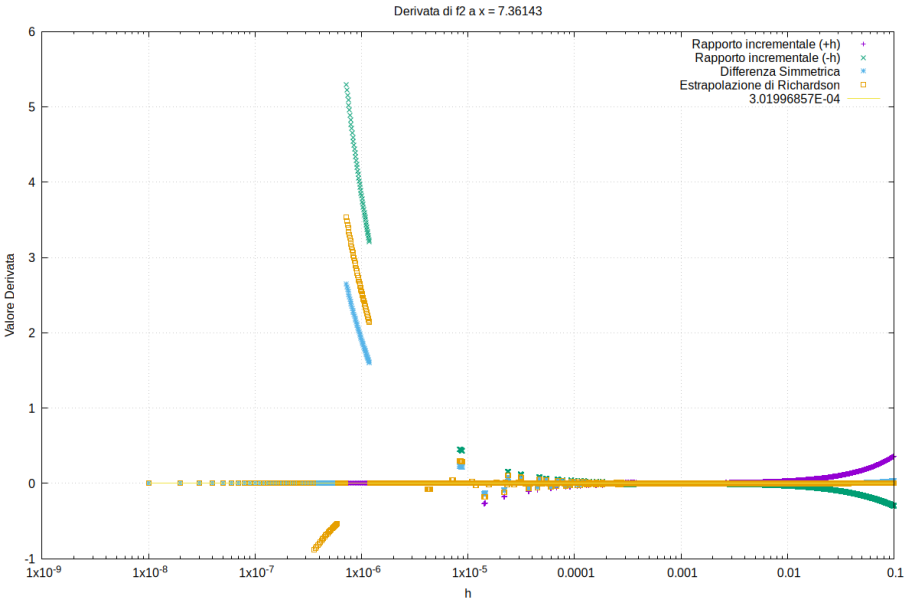

Figura 2: Punto 1, Funzione 2

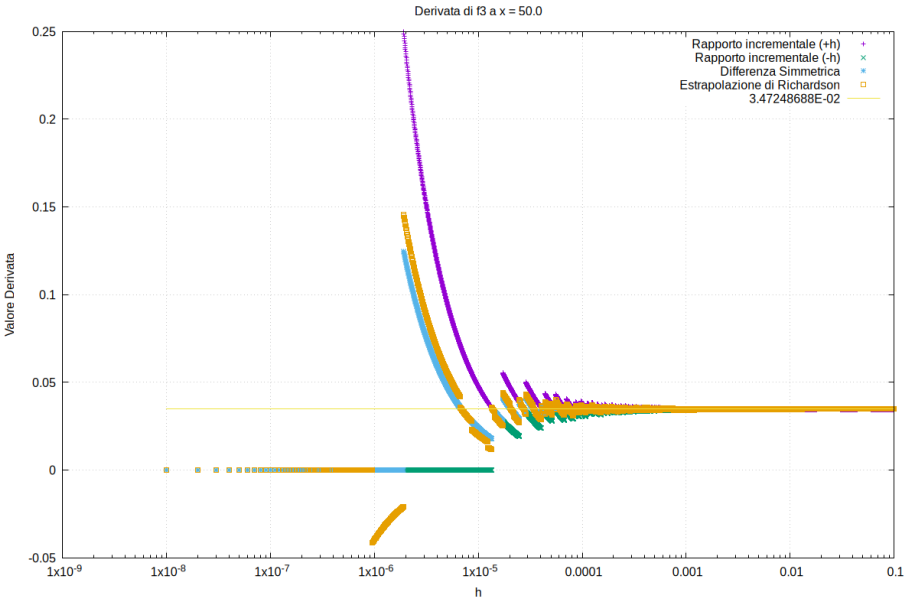

Figura 3: Punto 1, Funzione 3

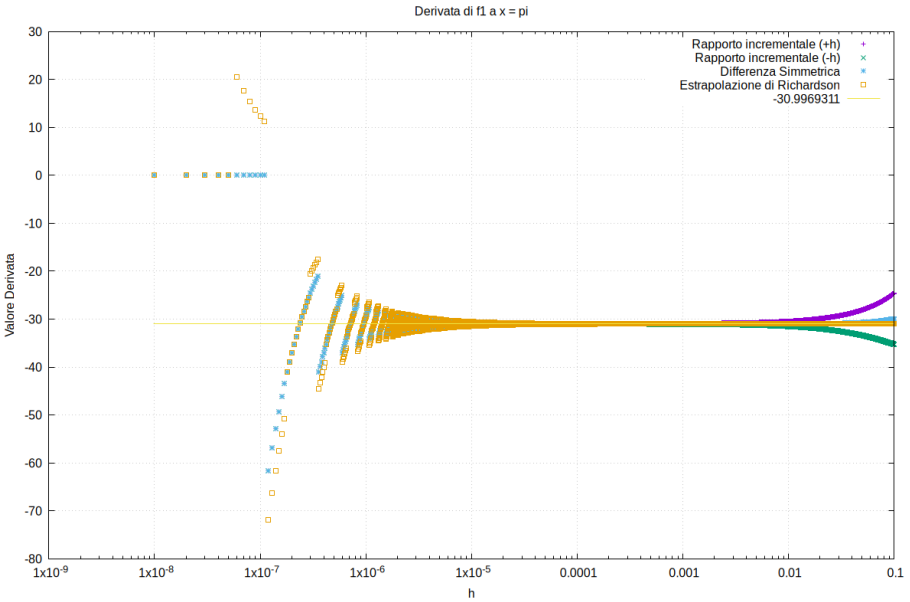

Figura 4: Punto 2, Funzione 1

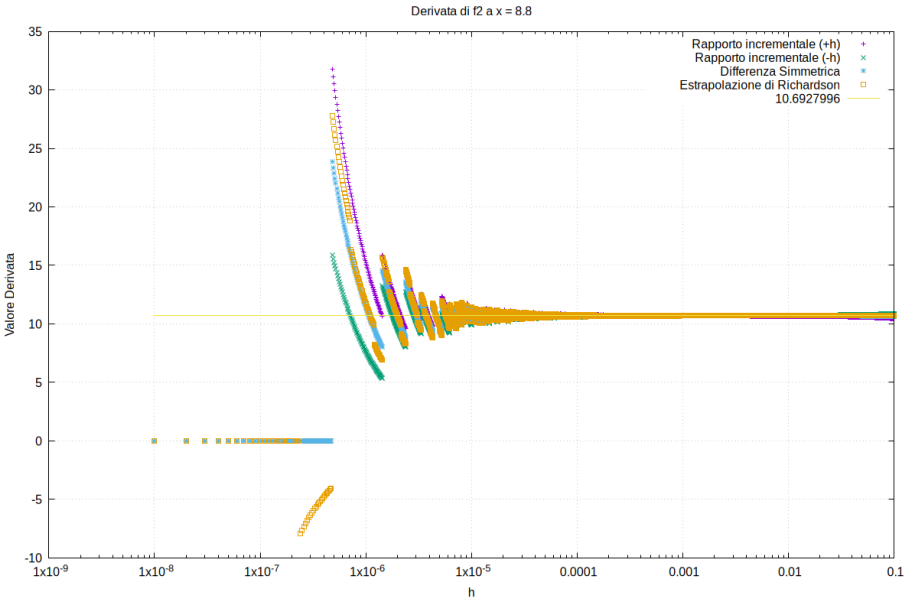

Figura 5: Punto 2, Funzione 2

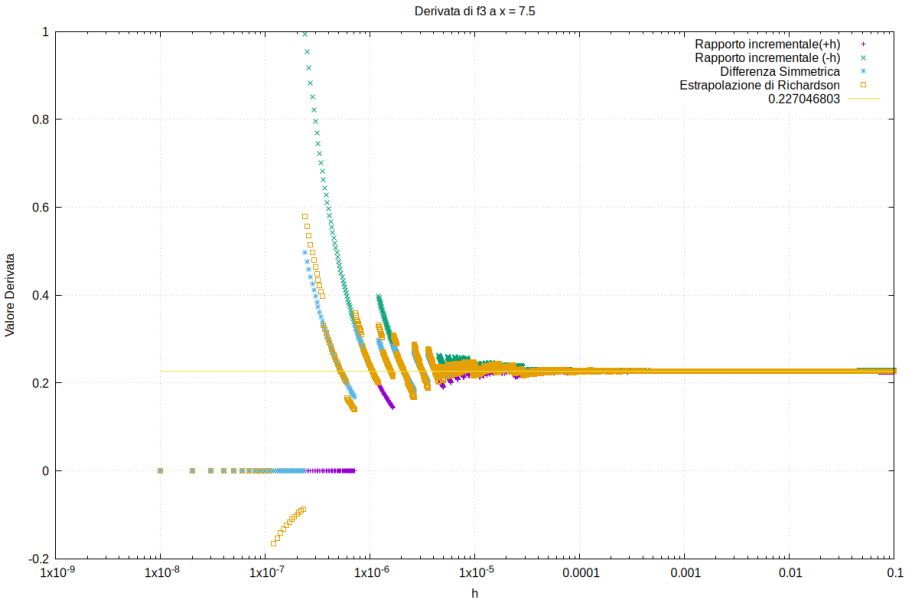

Figura 6: Punto 2, Funzione 3

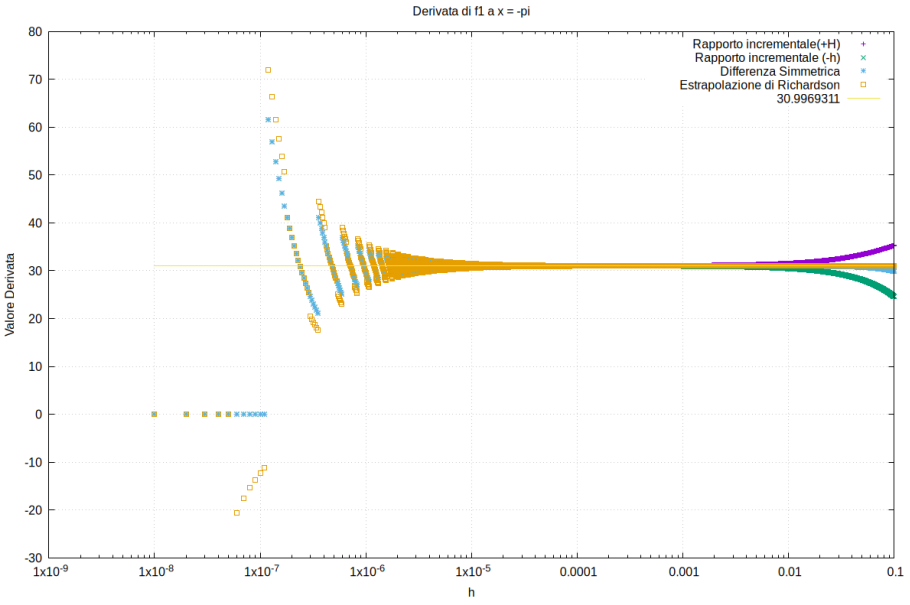

Figura 7: Punto 3, Funzione 1

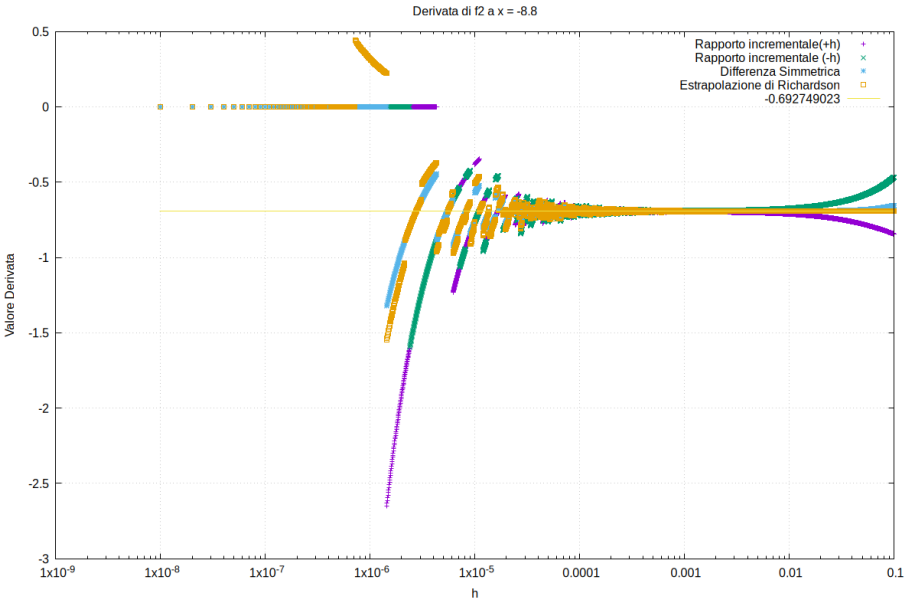

Figura 8: Punto 3, Funzione 2

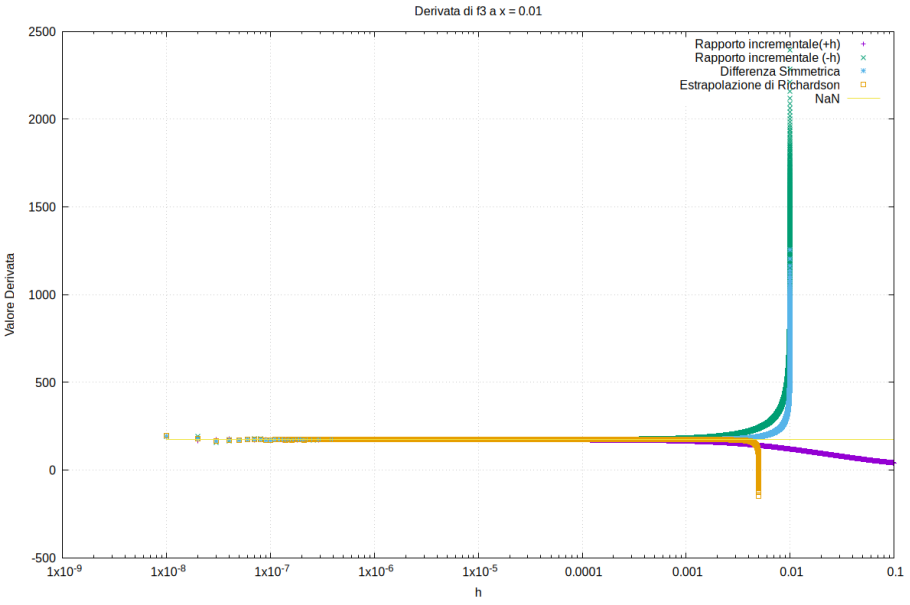

Figura 9: Punto 3, Funzione 3

6.2 Tentativi 33 Decimali di Precisione

Tutti questi grafici vanno letto da sinistra verso destra. I valori di h grandi si trovano a destra mentre i valori piccoli di h sono a sinistra. In questi prossimi grafici si potrà vedere la convergenza dei metodi al variare di h.

6.2.1 Convergenza Derivate

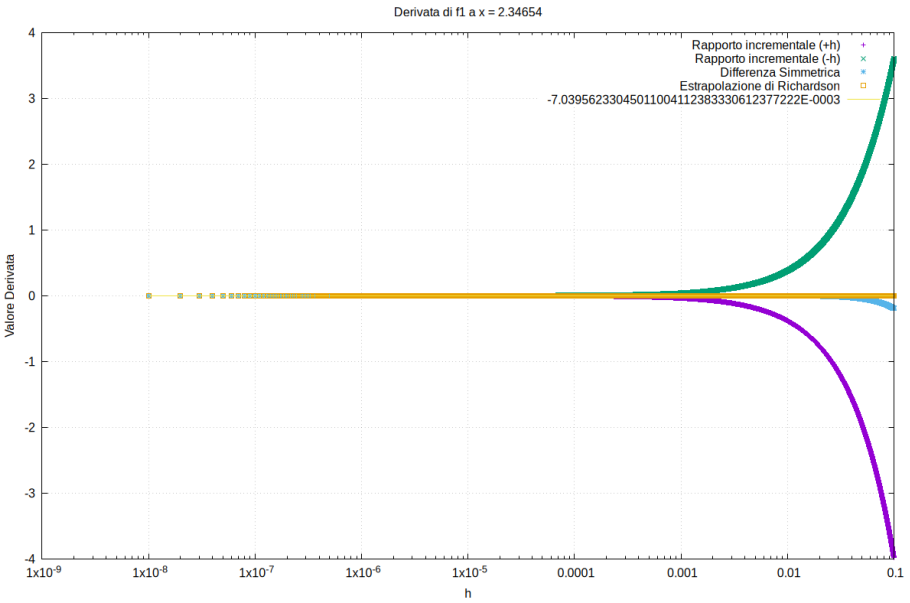

Figura 10: Punto 1, Funzione 1

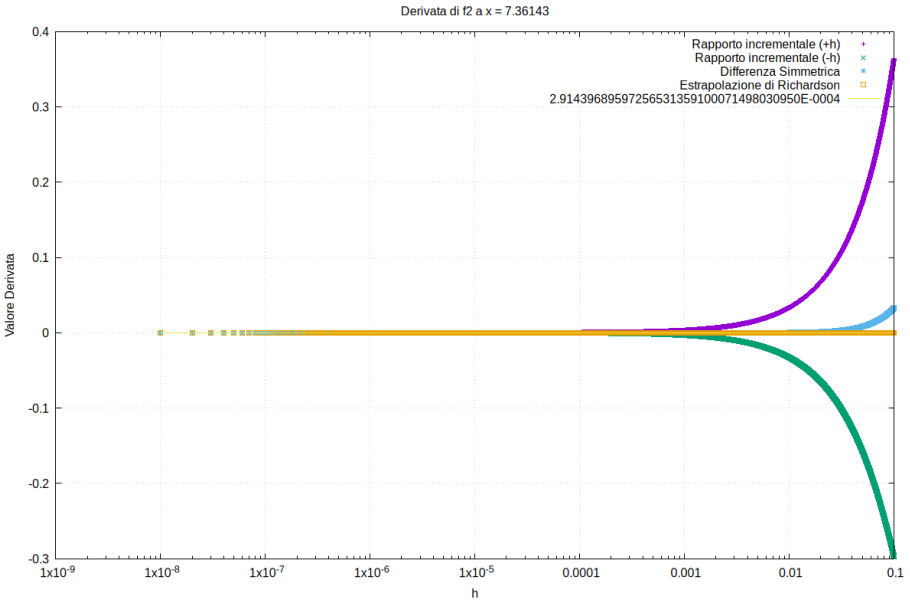

Figura 11: Punto 1, Funzione 2

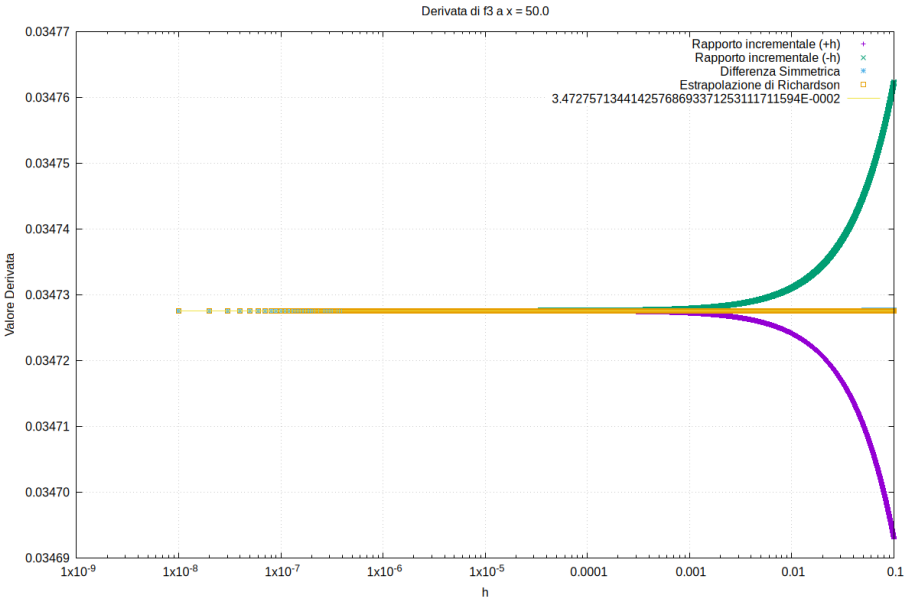

Figura 12: Punto 1, Funzione 3

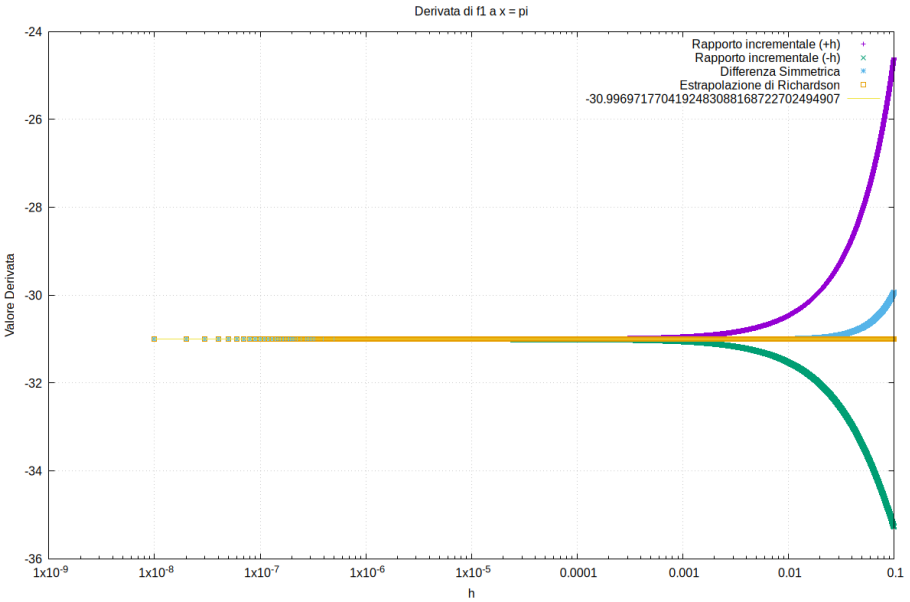

Figura 13: Punto 2, Funzione 1

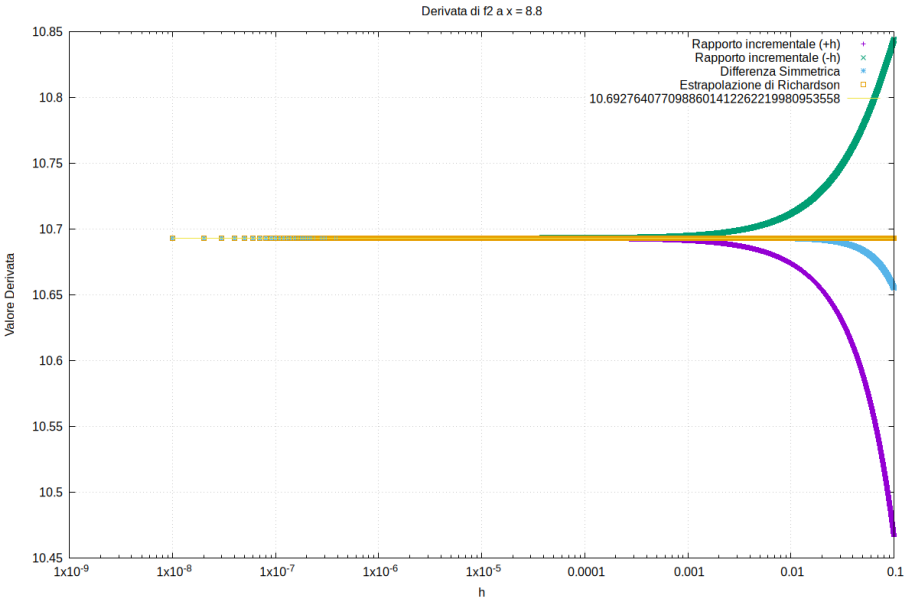

Figura 14: Punto 2, Funzione 2

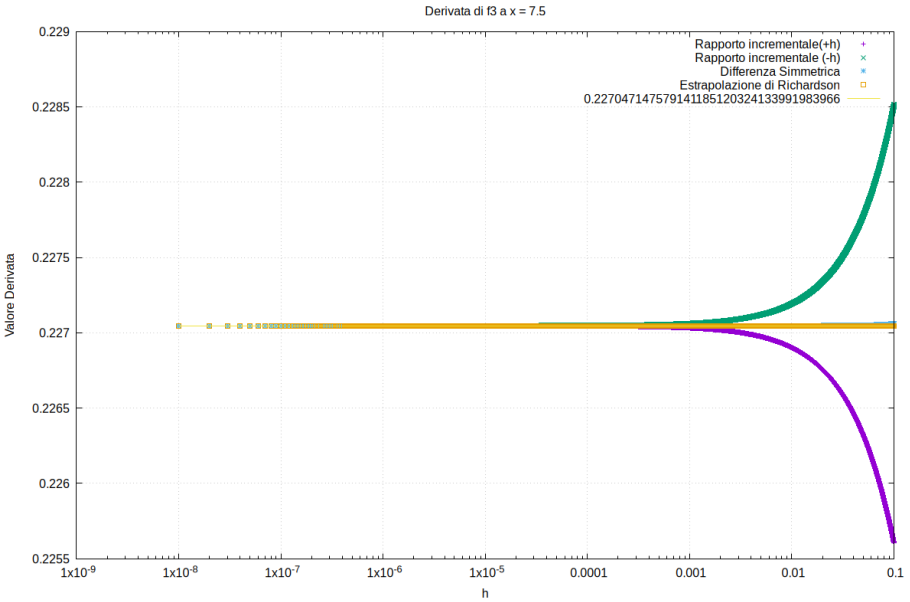

Figura 15: Punto 2, Funzione 3

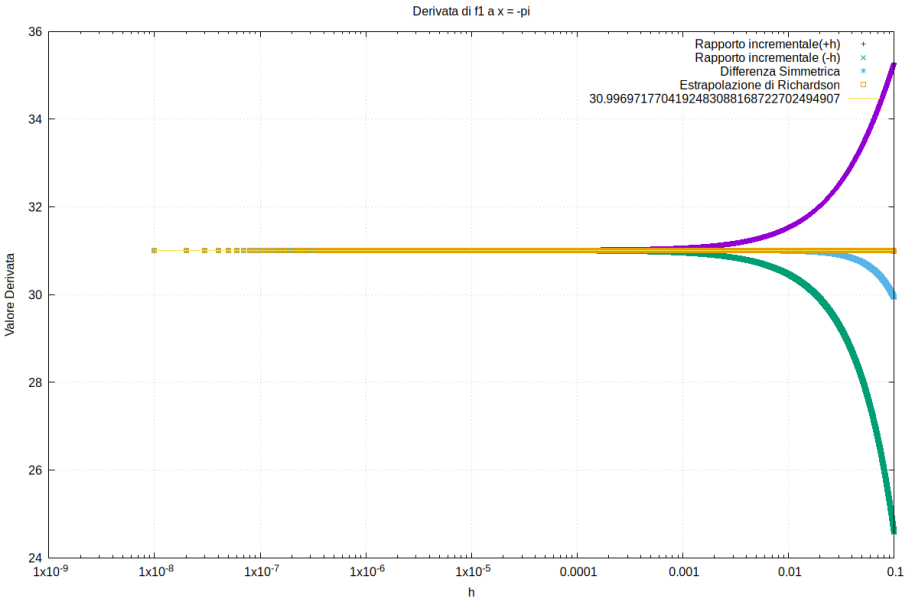

Figura 16: Punto 3, Funzione 1

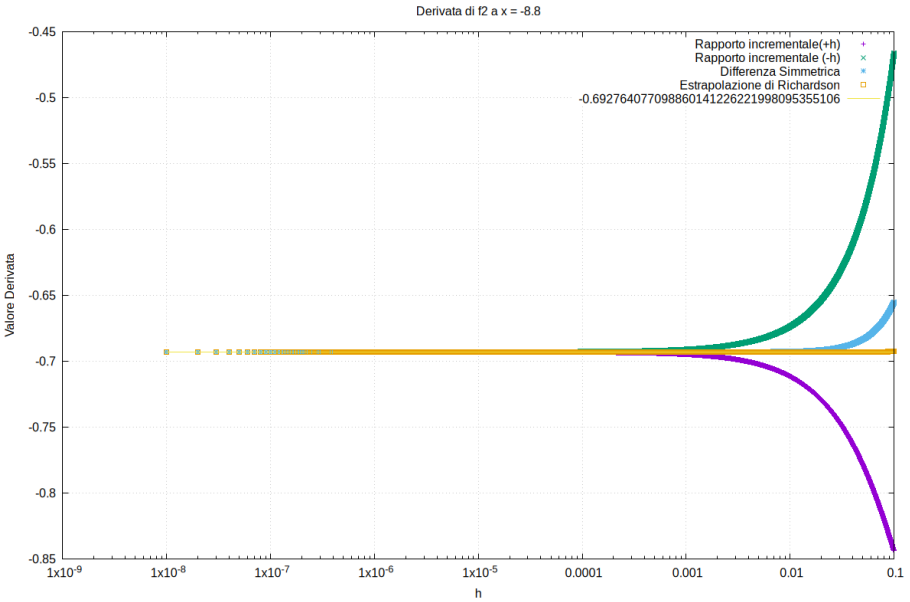

Figura 17: Punto 3, Funzione 2

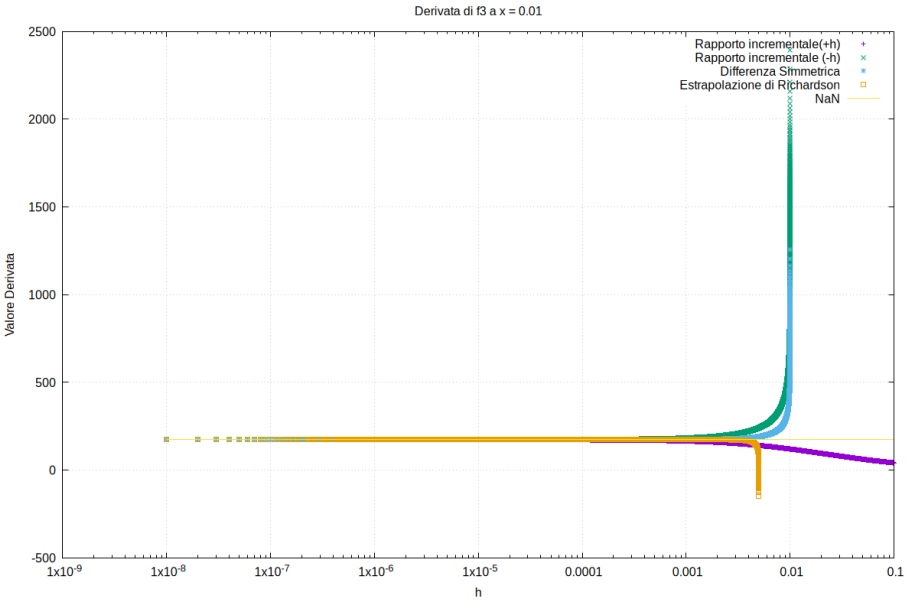

Figura 18: Punto 3, Funzione 3