Calcolo Numerico

Mario Ambrosino

31 Maggio 2018

Indice

Capitolo 1

Elementi di Calcolo numerico

In questo capitolo raccolgo gli elementi teorici necessari per la scrittura degli esercizi richiesti. Seguo [1, 2] come riferimenti bibliografici.

1.1 Differenziazione Numerica

Assegnata una funzione a valori reali $f : \mathbb{R} \ni x \mapsto f(x) \in \mathbb{R}$, la nozione di funzione derivata è data dall'espressione

$$\lim_{\Delta x \to 0} \frac{f(x_i + \Delta x) - f(x_i)}{\Delta x}. \tag{1.1}$$

Uno strumento che utilizzeremo spesso è lo sviluppo in serie di Taylor della funzione attorno un x_0:

$$f(x) = f(x_0) + (x - x_0) f'(x_0) + \frac{(x - x_0)^2}{2!} f^{(2)}(x_0) + \cdots + \frac{(x - x_n)^2}{n!} f^{(n)}(x_0) + \dots \tag{1.2}$$

Queste due definizioni possono naturalmente essere generalizzata a funzioni a più variabili $f(x, y, \dots)$ attorno il punto $(x_0, y_0, \dots)$.

Per il calcolatore è necessario discretizzare la nozione di funzione derivata. Dividendo il dominio della funzione in un reticolo con spaziatura h, l'insieme dei valori assunti dalla funzione coincide con la mappa dell'insieme $\mathbb{R} \supset \{x_i\} \mapsto \{f_{x_i}\} \subset \mathbb{R}$. In tal caso possiamo soltanto espressioni che solo al limite con $h \to 0$ coincidono con la definizione di derivata.

1.1.1 Derivata numerica al primo ordine

L'espressione approsssimata più semplice di derivata è data da

$$f_i^{'} = \frac{f_{i+1} - f_i}{h} + \mathcal{O}(h). \tag{1.3}$$

Difatti:

$$\frac{f_{i+1} - f_i}{h} = \frac{\left(f_i + f_i^{'}h + \mathcal{O}(h^2)\right) - f_i}{h} = f_i^{'} + \mathcal{O}(h) \tag{1.4}$$

Questa formula è definita come *formula a due punti* per la derivata del primo ordine e si ricava effettuando lo sviluppo. L'accuratezza può essere migliorata sommando due espressioni con incremento di h a destra e sinistra:

$$\frac{f_{i+1} - f_{i-1}}{2h} = \frac{\left(f_i + f_i^{'}h + \frac{1}{2}f_i^{''}h^2 + \mathcal{O}(h^3)\right) - \left(f_i - f_i^{'}h + \frac{1}{2}f_i^{''}h^2 + \mathcal{O}(h^3)\right)}{2h} = f^{'}(h) + \mathcal{O}\left(h^2\right) \tag{1.5}$$

Questa formula è nota come *formula a tre punti* per la derivata del primo ordine. L'accuratezza è aumentata al costo della valutazione della funzione nei tre punti $\{x_{i-1}, x_i, x_{i+1}\}$. Possiamo estendere quanto osservato utilizzando la *formula a cinque punti.* Essa si ottiene tramite l'espressione

$$f_i^{'} = \frac{1}{12h}\left(f_{i-2} - 8f_{i-1} + 8f_{i+1} - f_{i+2}\right) + \mathcal{O}\left(h^4\right) \tag{1.6}$$

Aumentare l'accuratezza con l'aumentare dei punti non è vantaggioso a causa dei problemi ai bordi, il cui trattamento diventa più delicato all'aumentare dei punti usati nel calcolo.

1.1.2 Derivata numerica del secondo ordine.

Una formula a tre punti per la valutazione numerica della derivata del secondo ordine è data dalla combinazione

$$f_{i+1} - 2f_i + f_i = h^2 f_i^{''} + \mathcal{O}\left(h^4\right) \tag{1.7}$$

che fornisce una derivate del secondo ordine del tipo

$$f_i^{''} = \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} + \mathcal{O}\left(h^2\right). \tag{1.8}$$

Una formula a cinque punti, analogamente, è data dall'espressione

$$f_i^{''} = \frac{1}{12h^2}\left(-f_{i-2} + 16f_{i-1} - 30f_i + 16f_{i+1} - f_{i+2}\right) + \mathcal{O}\left(h^4\right) \tag{1.9}$$

1.2 Integrazione Numerica

Il problema dell'integrazione numerica consiste nel valutare numericamente l'integrale su di una regione $[a, b]$

$$S = \int_a^b f(x)\, dx \tag{1.10}$$

E' necessario discretizzare il problema. Come prima scegliamo una spaziatura h *costante* e mappiamo $[a, b] \mapsto \{x_0, \ldots x_n\}$. Per l'additività del dominio di integrazione abbiamo:

$$\int_a^b f(x)\, dx = \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x)\, dx \equiv \sum_{i=0}^{n-1} I_i \tag{1.11}$$

Consideriamo separatamente ogni i-mo sezionamento dell'intervallo. L'operazione più naturale da fare in questi casi consiste nel sostituire la funzione integranda $f(x)$ in $\int_{x_i}^{x_{i+1}} f(x)\, dx$ con un'altra funzione che l'approssimi bene e di cui si sappia esprimere analiticamente la primitiva. La scelta della funzione approssimante determina naturalmente l'ordine di approssimazione.

Approssimazione di ordine zero: regola del rettangolo. Essa consiste nel sostituire alla funzione integranda una funzione costante nell'intervallo. Tale costante può essere scelta come il valore della funzione assunto ad uno degli estremi o il valor medio della funzione nel punto. Questa regola è nota come "regola del rettangolo". La miglior scelta naturalmente risulta essere quella del valor medio della funzione, la quale però ha anche un costo computazionale maggiore.

Approssimazione del primo ordine: regola del trapezio. Dopo l'approssimazione del rettangolo, la valutazione numerica più semplice che possiamo effettuare nella regione $[x_i, x_{i+1}]$ è approssimare linearmente (x) nella regione:

$$f(x) \approx f_i + (x - x_i) \frac{(f_{i+1} - f_i)}{h} \tag{1.12}$$

in tal caso si ha, integrando, che

$$S_1 = \frac{h}{2} \sum_{i=0}^{n-1} (f_i + f_{i+1}) + \mathcal{O}\left(h^2\right) \tag{1.13}$$

Questa regola è nota come "regola del trapezio".

Approssimazione del secondo ordine: regola di Cavalieri - Simpson Possiamo aumentare l'accuratezza della quadratura considerando tre punti della decomposizione (ossia considerando due intervalli). In tal caso possiamo utilizzare l'interpolazione di Lagrange della funzione $f(x)$ nella regione $[x_{i-1}, x_{i+1}]$. Si ottiene che:

$$f(x) = \frac{(x-x_i)(x-x_{i+1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})} + \frac{(x-x_{i-1})(x-x_{i+1})}{(x_i-x_{i-1})(x_i-x_{i+1})} + \frac{(x-x_{i-1})(x-x_i)}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)} + \mathcal{O}\left(h^3\right) \tag{1.14}$$

Effettuando l'integrazione per ogni coppia di intervalli adiacenti si ottiene la regola di Simpson per numero di intervalli pari

$$S_{2p} = \frac{h}{3}\sum_{j=0}^{n/2-1}\left(f_{2j} + 4f_{2j+1} + f_{2j+2}\right) + \mathcal{O}\left(h^4\right) \tag{1.15}$$

Per un numero di intervalli dispari sommiamo l'ultimo contributo separatamente

$$S_n = \frac{h}{12}\left(-f_{n-2} + 8f_{n-1} + 5f_n\right) \tag{1.16}$$

Approssimazione del secondo ordine per decomposizioni del dominio non uniformi Seguendo [2], in tal caso bisogna considerare una generalizzazione in cui si rinuncia all'ipotesi di larghezza uniforme: $h \mapsto \{h_i\}_{i=1\ldots n-1}$, ossia abbiamo che la larghezza dell'intervallo dipende dal punto i-mo del reticolo considerato. Si avrà per il contributo i-mo

$$S_i = \int_{h_{i-1}}^{h_i} f(x)\,dx = \alpha_i f_{i+1} + \beta_i f_i + \gamma_i f_{i-1} \tag{1.17}$$

con

$$\alpha = \frac{2h_i^2 + h_i h_{i-1} - h_{i-1}^2}{6h_i},\ \beta = \frac{(h_i + h_{i-1})^3}{6h_i h_{i-1}},\ \gamma = \frac{-h_i^2 + h_i h_{i-1} + 2h_i^2}{6h_i} \tag{1.18}$$

Sommando i vari contributi si ottiene, nel caso della decomposizione in un numero pari di intervalli

$$S_n = \int_0^{h_{n-1}} f(x)\,dx = \alpha f_n + \beta f_{n-1} + \gamma f_{n-2} \tag{1.19}$$

dove

$$\alpha = \frac{h_{n-1}}{6}\left(3 - \frac{h_{n-1}}{h_{n-1}+h_{n-2}}\right),\ \beta = \frac{h_{n-1}}{6}\left(3 + \frac{h_{n-1}}{h_{n-2}}\right),\ \gamma = -\frac{h_{n-1}}{6}\frac{h_{n-1}^2}{h_{n-2}(h_{n-1}+h_{n-2})} \tag{1.20}$$

Controllo dell'errore tramite schema di integrazione adattivo. Se espandiamo l'integranda $f(x)$ in serie di Taylor attorno $x = a$, abbiamo che, assumendo $h = b - a$:

$$S = \int_a^b f(x)\, dx = \int_a^b \left[\sum_{k=0}^{\infty} \frac{(x-a)^k}{k!} f^{(k)}(x) \right] dx = \sum_{k=0}^{\infty} \left[\frac{h^{k+1}}{(k+1)!} f^{(k)}(a) \right] \tag{1.21}$$

Confrontiamo con l'espressione di S con regola di Simpson di ordine 0 (ossia quando $x_{i-1} = a$, $x_i = c = (a+b)/2$, $x_{i+1} = b$. Si ottiene, espandendo in serie di Taylor $f(c)$ e $f(b)$

$$S_0 = \frac{h}{6}\left(f(a) + 4f(c) + f(b)\right) = \tag{1.22}$$

$$= \frac{h}{6}\left(f(a) + \sum \frac{h^k}{k!}\left(\frac{1}{2^{k-2}} + 1\right) f^k(a) \right). \tag{1.23}$$

Effettuando la differenza tra S e S_0 otteniamo la seguente stima dell'errore

$$\Delta_0 = S - S_0 \approx -\frac{h^5}{2880} f^{(4)}(a) \tag{1.24}$$

Effettuando la stessa differenza per l'ordine successivo, in cui si decompongono i due intervalli $[a, c]$ e $[c, b]$ in altrettanti sotto intervalli, si ottiene che

$$\Delta_1 \approx -\frac{1}{2^4}\frac{h^5}{2880} f^{(4)}(a) \tag{1.25}$$

In tal modo si ha che

$$\Delta_1 - \Delta_0 \approx -\frac{15}{16}\frac{h^5}{2880} f^{(4)}(a) \approx \frac{15}{16}\Delta_0 \approx 15\Delta_1 \tag{1.26}$$

Ciò ci permette di definire un algoritmo adattivo di integrazione: consideriamo di voler un errore sull'integrazione minore o uguale a $\delta \in \mathbb{R}$:

$$|\Delta S| \leq \delta \tag{1.27}$$

allora se si ha che $|S_1 - S_0| \leq 15\delta \implies \Delta_1 \leq \delta$ e si può utilizzare il valore di S_1 come valore dell'integrazione numerica controllata dal parametro δ. Altrimenti si bisezionano tutti gli intervalli e si itera il processo di controllo.

1.3 Ricerca degli zeri di un'equazione

Un problema comune dell'analisi è la ricerca degli zeri di una funzione. Ciò equivale alla richiesta di trovare il kernel di una qualsiasi funzione f:

$$x \in S : \ f(x) = 0 \tag{1.28}$$

Metodo di bisezione Supponiamo di sapere che nell'intervallo $[a, b]$ esiste un solo zero x_r per l'equazione $f(x) = 0$. Allora sappiamo che $f(a) f(b) < 0$. Possiamo allora dividere la regione in due parti: $[a, b] = [a, x_0] \cup [x_0, b]$ con $x_0 = (a + b)/2$. Abbiamo analogamente che o $f(a) f(x_0) < 0$ o $f(x_0) f(b) < 0$. Scelta la regione che soddisfa tale richiesta, iteriamo il processo di bisezione fino a quando $|f(x_n)| \leq \delta$.

Metodi di Newton Esso si basa sull'approssimazione lineare di una funzione liscia attorno la sua radice. Effettuiamo uno sviluppo in serie di Taylor della funzione f attorno la radice x_r,dove x_k è un punto vicino ottenuto tramite la $k-$ma iterazione del metodo ricorsivo di ricerca dello zero.

$$f(x_r) \simeq f(x_k) + (x_k - x_r) f'(x) + \ldots \tag{1.29}$$

Definiamo allora la ricerca dello zero sulla funzione linearizzata attorno il punto: consideriamo ossia l'equazione della retta

$$0 = f(x_{k+1}) = f(x_k) + (x_{k+1} - x_k) f'(x_k) \tag{1.30}$$

Si ottiene che:

$$x_{k+1} = x_k + \Delta x_k = x_k - f_k / f_k^{'} \tag{1.31}$$

Metodo della secante Quando non è possibile effettuare una stima analitica della derivata prima $f_k^{'}$ è necessario sostituire tale espressione con una stima della derivata.

$$\frac{1}{f_k^{'}} \mapsto \frac{x_k - x_{k-1}}{f_k - f_{k-1}}. \tag{1.32}$$

1.4 Ricerca degli estremi di una funzione

Come sappiamo, si definisce massimo (minimo) di una funzione (ad una variabile) un punto $X \in [a, b] \subset \mathbb{R}$ tale che

$$\frac{df}{dx}(X) = 0 \qquad \wedge \qquad \frac{d^2 f}{dx^2}(X) \gtrless 0 \tag{1.33}$$

Avendo presentato il problema della ricerca degli zeri di una funzione, la ricerca numerica degli estremi di una funzione consiste in una ricerca numerica degli zeri della derivata con valutazione di controllo sulla positività o negatività della variazione della funzione stessa: è necessario muoversi verso il massimo (minimo) della funzione.

Nel caso non si avesse a disposizione l'espressione della derivata prima naturalmente sarà necessario ricorrere alla valutazione numerica della derivata.

Formalmente, la sequenza ricorsiva di ricerca del minimo (massimo) è data da:

$$x_{k+1=} x_k + \Delta x_k = x_k \mp a \cdot g^{'}(x_k) \tag{1.34}$$

Esso può essere esteso al caso a più variabili considerando la sequenza di vettori ottenuta ricorsivamente con la regola

$$\boldsymbol{x}_{k+1=} \boldsymbol{x}_k + \Delta \boldsymbol{x}_k = \boldsymbol{x}_k \mp a \cdot \boldsymbol{\nabla} g(x_k) / |\boldsymbol{\nabla} g(x_k)| . \tag{1.35}$$

Capitolo 2

Risoluzione degli esercizi

Presenterò in questo capitolo alcune semplici applicazioni:

1. Scrivere un listato per confrontare i metodi di derivazione e integrazione l'efficienza all'aumentare dell'ordine (o del metodo).
2. Scrivere un listato per la valutazione di zeri (con confronto tra i tre metodi).
3. Scrivere un listato per la valutazione di estremi per una funzione.

Come nella precedente tesina, ho caricati i sorgenti utili su GitHub presso l'indirzzo `https://github.com/mario-ambrosino/numerical-calculus`. Farò riferimento ai listati contenuti all'interno della cartella `sources`.

Per il primo esercizio presenterò sarà necessario far riferimento ai due sorgenti `test_derivative.c` e `test_integration.c`; per il secondo esercizio invece `zero_search.c` e infine per l'ultimo `minima_search.c`.

2.1 Metodi di Derivazione

Ho implementato gli algoritmi precedentemente presentati nel listato all'indirizzo `https://github.com/mario-ambrosino/numerical-calculus/blob/master/sources/test_derivative.c`.

Il listato contiene la funzione $sin(x)$ e restituisce il valore della sua derivata nell'intervallo $[-\pi, +\pi]$.

In realtà sono stati utilizzati i due script `test_derivative.sh` e `analysis_derivative.sh` nella cartella principale su GitHub, che automatizzano il processo di test.

I file in output sono del tipo "`modeN_M.dat`" dove `N` indica il metodo utilizzato e `M` indica l'esponente della spaziatura nella definizione del metodo numerico di derivata, definita come $\Delta = 1 \cdot 10^{-M}$

I risultati dell'analisi sono riportati nelle immagini (Cfr. **Figura 2.1**). Non riporto i vari metodi poiché differiscono di poco i risultati. Inoltre ho notato un'anomalia per il metodo a cinque punti.

2.2 Metodi di Integrazione

Gli algoritmi presentati sono stati utilizzati nel listato all'indirizzo `https://github.com/mario-ambrosino/numerical-calculus/blob/master/sources/test_integration.c`

Il programma ammette come parametri in input gli estremi inferiore e superiore di integrazione (in unità di π), il numero minimo e il numero massimo di suddivisioni dell'intervallo e infine il metodo di integrazione. Ho implementato il metodo dei rettangoli, dei trapezi e di Simpson. Le funzioni scelte sono $sin(x)$, $sin\,(1/x)$ e $sin\,(x)\,/x$, cambiati tramite riscrittura del sorgente. L'intervallo di integrazione invece è $[-\pi, +\pi]$. In output si osserva sia il valore assunto dall'integrale che la differenza rispetto al valore atteso.

I file in output sono del tipo "`modeN_M.dat`" dove `N` indica il metodo utilizzato e `M` indica l'esponente della spaziatura nella definizione del metodo numerico di derivata, definita come $\Delta = 1 \cdot 10^{-M}$

Anche in questo caso i risultati sono riportati nelle immagini seguenti (Cfr. **Figura 2.2**).

2.3 Metodi di ricerca degli zeri.

E' possibile leggere il listato `zero_search.c` commentato presso l'indirizzo `https://github.com/mario-ambrosino/numerical-calculus/blob/master/sources/zero_search.c`. Il suo tipico output mostra che i tre algoritmi sono accurati e che l'ordine di efficienza della ricerca di zeri è tale che il Metodo di Newton sia il più efficiente, seguito dal metodo delle tangenti che richiede qualche passo ricorsivo in più per poi passare al metodo di bisezione, molto meno efficiente dei primi due.

Anche in questo caso abbiamo utilizzato la funzione $f\,(x) = sin\,(x)$, con intervallo di ricerca in $[0, 2]$

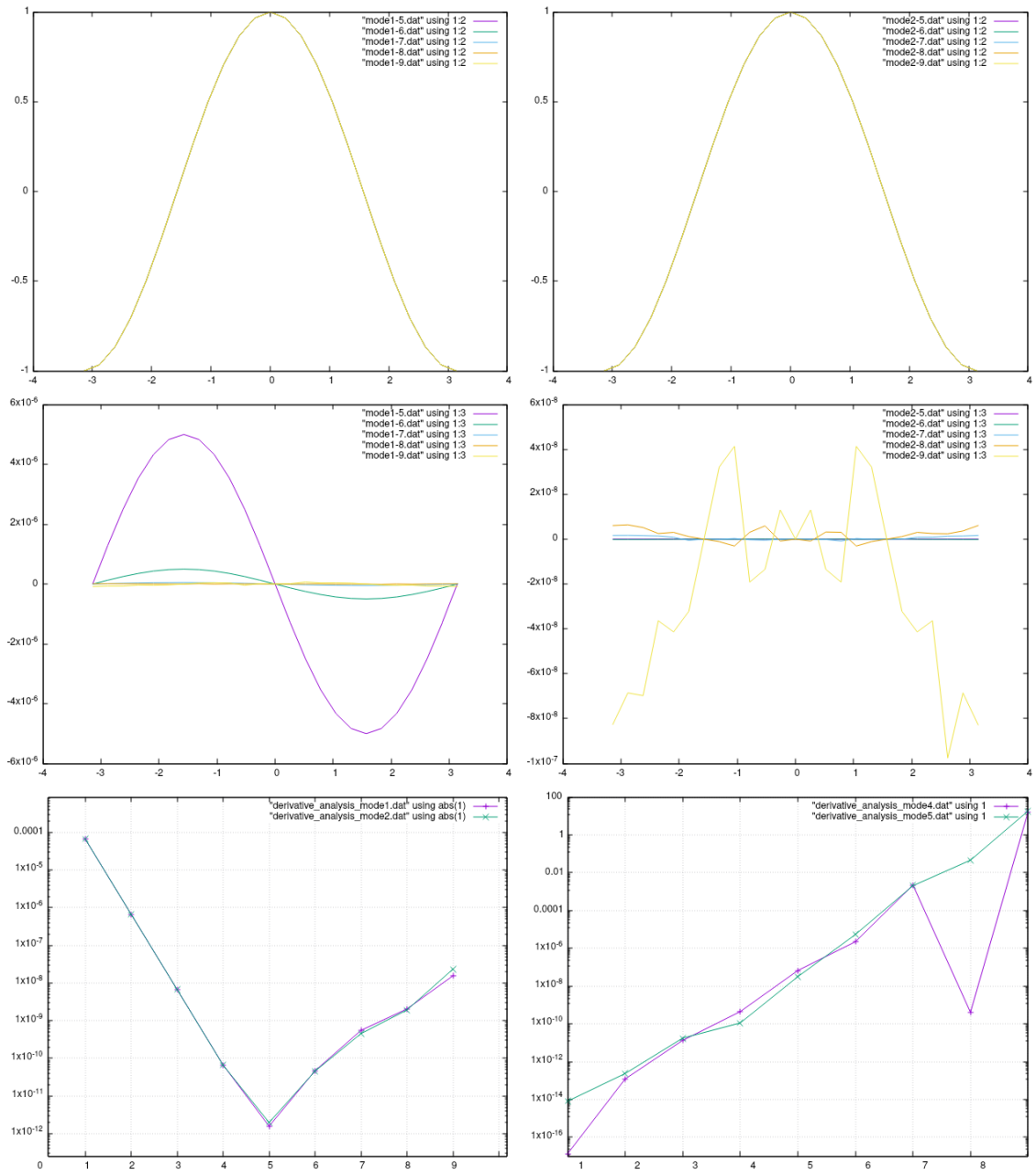

Figura 2.1: Analisi dei risultati per il metodo di derivazione a due e a tre punti al variare della spaziatura $h = \Delta x$. I primi due grafici rappresentano la valutazione della derivata della funzione $sin\,(x)$ per i due metodi, i secondi due rappresentano la differenza tra derivata teorica e derivata numerica, gli ultimi l'errore medio al variare della spaziatura.

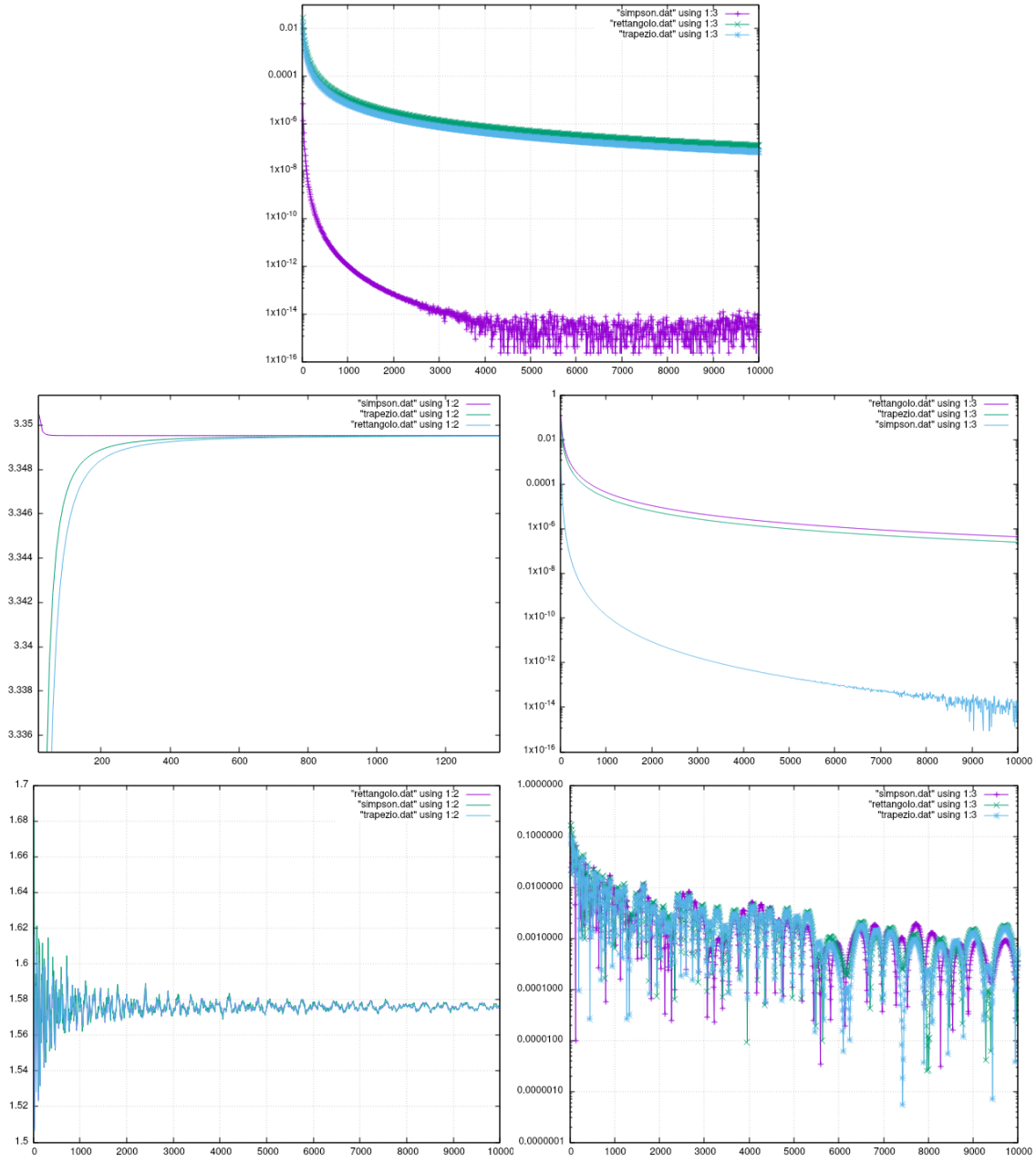

Figura 2.2: Nella prima figura andamento dell'integrazione di $sin\,(x)$ nell'intervallo $[-\pi, \pi]$ al variare del numero di intervalli. Esso coincide con il grafico dell'andamento dell'errore. Nelle due figure sottostanti, grafico dell'andamento dell'integrazione di $sinc\,(x)$ e di $sin\,(1/x)$ al variare del numero di intervalli.

```
###OUTPUT METODO BISEZIONE###
############################################################
i   A             B             C             Error
############################################################
01; 1.000000e+00; 2.000000e+00; 1.500000e+00; 5.707963e-01
02; 1.500000e+00; 2.000000e+00; 1.750000e+00; 7.079633e-02
03; 1.500000e+00; 1.750000e+00; 1.625000e+00; 1.792037e-01
04; 1.500000e+00; 1.625000e+00; 1.562500e+00; 5.420367e-02
05; 1.562500e+00; 1.625000e+00; 1.593750e+00; 8.296327e-03
06; 1.562500e+00; 1.593750e+00; 1.578125e+00; 2.295367e-02
07; 1.562500e+00; 1.578125e+00; 1.570312e+00; 7.328673e-03
08; 1.570312e+00; 1.578125e+00; 1.574219e+00; 4.838268e-04
09; 1.570312e+00; 1.574219e+00; 1.572266e+00; 3.422423e-03
10; 1.570312e+00; 1.572266e+00; 1.571289e+00; 1.469298e-03
11; 1.570312e+00; 1.571289e+00; 1.570801e+00; 4.927357e-04
12; 1.570312e+00; 1.570801e+00; 1.570557e+00; 4.454455e-06
13; 1.570557e+00; 1.570801e+00; 1.570679e+00; 2.396862e-04
14; 1.570679e+00; 1.570801e+00; 1.570740e+00; 1.176159e-04
15; 1.570740e+00; 1.570801e+00; 1.570770e+00; 5.658070e-05

###OUTPUT METODO NEWTON###
############################################################
i   A             B             C             Error
############################################################
1; 0.000000e+00; 1.642093e+00; 8.210463e-01; 7.129629e-02
2; 1.570675e+00; 1.642093e+00; 1.606384e+00; 1.210496e-04
3; 1.570675e+00; 1.570796e+00; 1.570736e+00; 5.913528e-13
4; 1.570796e+00; 1.570796e+00; 1.570796e+00; 1.092876e-16

###OUTPUT METODO TANGENTI###
############################################################
i   A             B             C             Error
############################################################
1; 1.564904e+00; 2.000000e+00; 1.782452e+00; 5.891951e-03
2; 1.564904e+00; 1.571132e+00; 1.568018e+00; 3.361645e-04
3; 1.570796e+00; 1.571132e+00; 1.570964e+00; 1.834032e-09
4; 1.570796e+00; 1.571132e+00; 1.570964e+00; 5.963112e-17
5; 1.570796e+00; 1.571132e+00; 1.570964e+00; 6.722053e-18
```

Algorithm 1: Tabelle con confronto dei tre metodi ottenute tramite il listato `zero_search.c`

2.4 Metodi di ricerca del minimo.

Il listato `minima_search.c` commentato è visitabile presso l'indirizzo `https://github.com/mario-ambrosino/numerical-calculus/blob/master/sources/minima_search.c`.

La funzione scelta per la ricerca di minimo locale è

$$f(x) = x^4 + 5x^3 + 2x^2 + 2x + 1 \tag{2.1}$$

e, scegliendo come punto di inizio $x = 6$ si ottiene un minimo a $x = -3.50542 \pm 0.000002$ dopo 19 passi.

Il risultato ottenuto tramite Mathematica con il comando `FindMinimum[`$1+2x+2x^2+5x^3+x^4$`, {{`x`,6}}]` è dato da $x = -3.50542$.

Ho introdotto una piccola modifica per migliorare la convergenza dell'algoritmo: il moltiplicatore cambia al variare del numero di step, quindi all'aumentare delle iterazioni accorcia linearmente la lunghezza di spostamento (è meglio utilizzare un moltiplicatore relativamente alto). La mia scelta per i parametri è la seguente:

```
./exec/minima_search.exe 5 .000002 0.01 100000
```

Riporto qui la funzione centrale dell'algoritmo.

```
double gradesc((*f)(double), x, error, mult, max_step) {
    // omessi i tipi per semplificare

    double curr_err = error + 1;   // rende vera la condizione del primo while
     long step = 0;
     double prev;

    while((error < curr_err) && (step < max_step)) {
          prev = x;
          x -= NumDer(f, x) * mult / (step + 1);      // discesa del gradiente
          curr_err = fabs(prev-x);
          printf("\n%li %f %f", step, x, curr_err);
          step++;     double
     }

     return x;

}
```

Algorithm 2: La funzione che realizza la discesa del gradiente

Bibliografia

[1] L. Barone, E. Marinari, G. Organtini, F. Ricci-Tersenghi
Programmazione Scientifica
Pearson Education - (2006) - ISBN: 978-8-8719-2242-3

[2] Tao Pang
An Introduction to Computational Physics - Second Edition
Cambridge University Press - (2006) - ISBN: 978-0-521-53276-1