Università degli Studi di Napoli "Federico II"

Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

Giuseppe Francesco Di Cecio - M63001211
Nicola D'Ambra - M63001223

Elaborato di *Calcolo Numerico*

Metodi Numerici per la risoluzione di Equazioni Differenziali Ordinarie (ODE)

Università degli Studi di Napoli *Federico II*
Napoli
A.A 2020/2021

Indice

Capitolo 1

Metodo di Eulero - Esplicito

Il metodo di Eulero è un semplice algoritmo per ricavare una soluzione numerica ad'*equazione differenziale ordinaria* (ODE). Sia il seguente problema di Cauchy:

$$\begin{cases} \dfrac{dy}{dt} = f(t, y(t)) \\ y(t_0) = y_0 \end{cases} \tag{1.1}$$

con $t \in I = [t_0, T]$ e T prefissato.
Prima di procedere con la descrizione della tecnica si può fare una breve ripresa del concetto di Serie di Taylor.

1.1 Serie di Taylor

La serie di Taylor di una funzione in un punto è la rappresentazione della funzione come serie di termini calcolati a partire dalle derivate della funzione stessa nel punto.
Sia $f(x)$ definita a valori reali, o complessi, infinite volte derivabile, la serie di Taylor centrata nel punto x_0 è la seguente serie di potenze:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2}(x - x_0)^2 + \frac{f'''(x_0)}{6}(x - x_0)^3 + ...$$

In generale:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n \tag{1.2}$$

A questo punto si può continuare con la descrizione.

Si discretizza l'intervallo I in n sottointervalli:

In modo che:

$$t_i = t_{i-1} + h \qquad h = \frac{T - t_0}{n}$$

La funzione soluzione $y(t)$ viene approssimata alla serie di Taylor arrestata al primo ordine, in ogni intervallo. In generale vale la seguente uguaglianza:

$$y(x) = y(x_0) + y'(x_0)(x - x_0) + R(x, x - x_0)$$

Trascurando il resto $R(x, x - x_0)$, nel caso in esame:

$$h = x - x_0 \qquad t_i = x_0$$

si ha:

$$y(t_i + h) \approx y(t_i) + y'(t_i)h$$

Per la 1.1:

$$y(t_i + h) \approx y(t_i) + f(t_i, y(t_i))h \tag{1.3}$$

Quindi la funzione soluzione può essere ricava in corrispondenza di ogni punto t_i dell'intervallo I come:

1. $y(t_0) \approx y_0$
2. $y(t_1) = y(t_0 + h) \approx y(t_0) + f(t_0, y(t_0))h$
3. $y(t_2) = y(t_1 + h) \approx y(t_1) + f(t_1, y(t_1))h$
4. ...

In generale quindi::

Noto y_0, la **legge di aggiornamento** vale:

$$y_{i+1} = y_i + f(t_i, y_i)h \quad \forall i \in \{0, 1, 2, ..., n - 1\} \tag{1.4}$$

con $y_i \approx y(t_i)$ una stima del reale valore della funzione soluzione.

1.2 Interpretazione Geometrica

Dalla 1.4 si può intuire che in un intervallo qualsiasi $[t_i, t_{i+h}]$, con h passo di discretizzazione, la funzione soluzione $y(t)$ viene approssimata ad una retta tangente nel punto t_i, centro dello sviluppo di Taylor arrestato al primo ordine.

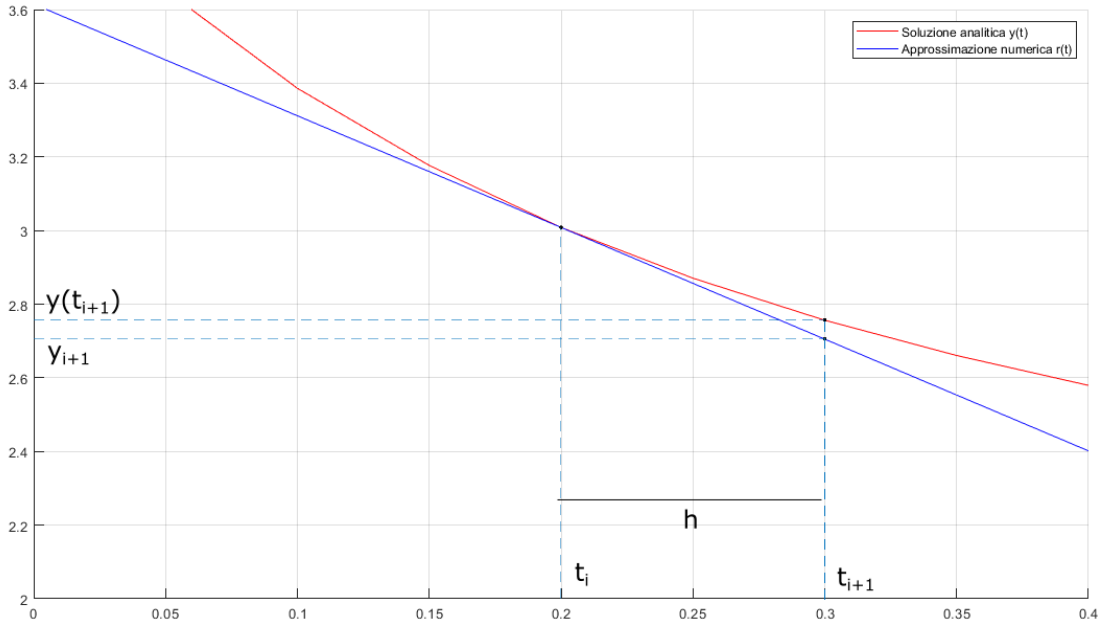


Figura 1.1: *Confronto tra soluzione numerica e soluzione analitica*

Sia la retta $r(t)$:

$$r(t) = y(t_i) + f(t_i, y(t_i))(t - t_i)$$

e $y(t)$ soluzione analitica, al passo $(i+1) - esimo$ si approssima:

$$y(t_{i+1}) \approx y_{i+1} = r(t_{i+1})$$

1.3 Implementazione Matlab

L'implementazione in Matlab è molto semplice. Prevede infatti solo la realizzazione della legge di aggiornamento della soluzione numerica.
I parametri di ingresso alla funzione sono:

- f Funzione dell'equazione differenziale di tipo 1.1
- y_0 Condizione iniziale al tempo t_0
- t_0 Tempo iniziale
- T Tempo finale
- h Passo di discretizzazione

I parametri in uscita sono:

- t Asse dei tempi discretizzato
- y Vettore soluzione

Il metodo di Eulero può essere implementato sia in un sistema aritmetico a *singola precisione* che in *doppia precisione*.

- **eulero32** implementazione del metodo con calcoli in *singola precisione* a 32bit.

- **eulero64** implementazinoe del modello con calcoli in *doppia precisione* a 64bit

```
function [y, t] = eulero32(f, y0, t0, T, h)
        t = single(t0:h:T);                 %Asse dei tempi discretizzata con passo h
        y = single(zeros(1,length(t)));     %Vettore soluzione
        y(1) = y0;                          %Condizione iniziale

        for i=1:length(t)-1
                y(i+1) = y(i) + f(t(i), y(i))*h;     %Costruzione della soluzione
        end
end

function [y, t] = eulero64(f, y0, t0, T, h)
        t = t0:h:T;                         %Asse dei tempi discretizzata con passo h
        y = zeros(1,length(t));             %Vettore soluzione
        y(1) = y0;                          %Condizione iniziale

        for i=1:length(t)-1
                y(i+1) = y(i) + f(t(i), y(i))*h;     %Costruzione della soluzione
        end
end
```

Esempio 1.3.1 *Sia la seguente equazione differenziale con* $t \in [0,1]$*:*

$$\begin{cases} \dfrac{dy}{dt} = 2y(t) - y^2(t) \\ y(t_0) = 4 \end{cases}$$

La soluzione analitica è la seguente:

$$y(t) = \frac{4}{2 - e^{-2t}}$$

La soluzione numerica può essere calcolata con la funzione MATLAB descritta precedentemente:

```
f = @(t,y) 2*y-y^2;
t0 = 0;
T = 1;
[y_eulero,t] = eulero64(f, 4, t0, T, 0.05);
y_analitica = 4./(2-exp(-2.*t));
hold on;
grid;
plot(t,y_eulero);
plot(t, y_analitica);
legend('Soluzione numerica (Eulero)', 'Soluzione analitica');
```

Il cui risultato è il seguente:

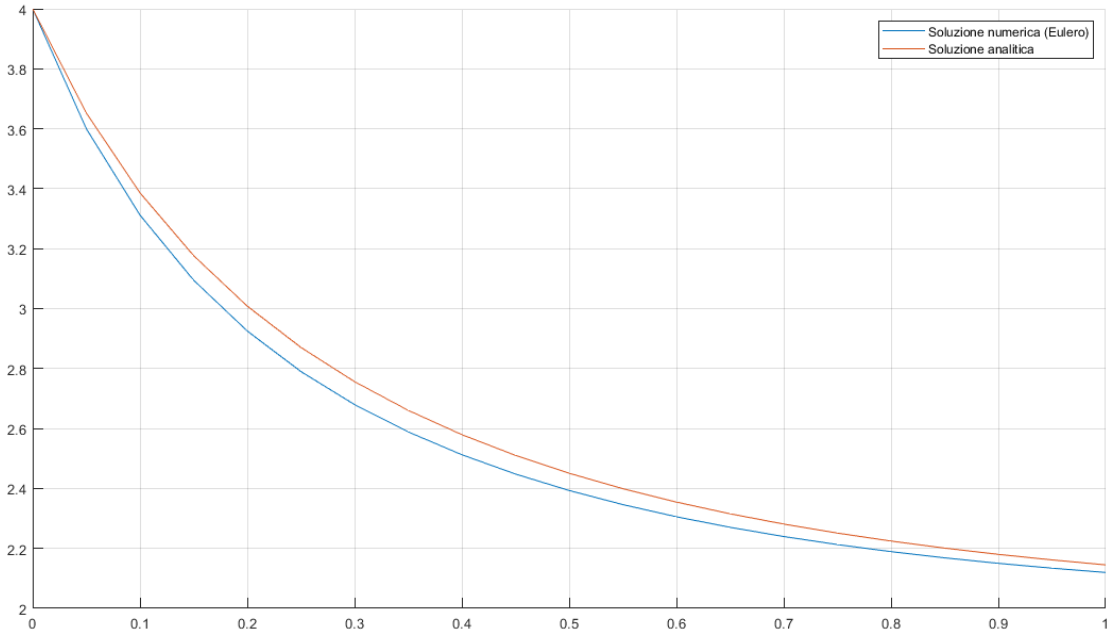


Figura 1.2: *Soluzione numerica con passo di discretizzazione h=0.05*

1.4 Errori

La legge di aggiornamento specifica che:

$$y_i \approx y(t_i)$$

quindi il metodo da luogo a degli errori.
Una soluzione numerica presenta sempre degli errori rispetto alla relativa soluzione analitica. Alcuni dovuti a delle approssimazioni altre invece a causa dell'elaborazione in un sistema aritmetico a precisione finita.

1.4.1 Errore di Discretizzazione

Sia il seguente problema di Cauchy del primo ordine:

$$M(P) : \begin{cases} \dfrac{dy}{dt} = f(t, y(t)) \\ y(t_0) = y_0 \end{cases}$$

Data la legge di aggiornamento 1.4 si ha:

$$y_{i+1} = y_i + f(t_i, y_i)h \quad \Rightarrow \quad \frac{y_{i+1} - y_i}{h} = f(t_i, y_i)$$

ovvero la derivata viene sostituita con il rapporto incrementale, avendo così un problema discretizzato con passo h:

$$M_h(P) : \begin{cases} \dfrac{y_{i+1} - y_i}{h} = f(t_i, y_i) \qquad \forall i = \{0, 1, ..., n-1\} \\ y(t_0) = y_0 \end{cases} \tag{1.5}$$

commettendo un **errore di discretizzazione**.
In generale l'errore viene definito come la differenza tra l'approssimazione e il valore reale. Nel caso del metodo di Eulero, quindi, l'errore è la differenza tra il rapporto incrementale e la derivata nel punto t:

$$T(t,h) = \frac{y(t+h) - y(t)}{h} - y'(t)$$

ovvero:

$$T(t,h) = \frac{y(t+h) - y(t)}{h} - f(t, y(t)) \tag{1.6}$$

Quindi l'errore di discretizzazione è la "distanza" tra il problema continuo $M(P)$ e il problema discreto $M_h(P)$.
Inoltre tale errore misura l'**accuratezza** fornita dal metodo.

Consistenza

Un metodo per la risoluzione di un problema ai valori iniziali si dice **consistente** se, fissato t:

$$\lim_{h \to 0} T(t,h) = 0$$

ovvero se l'errore di discretizzazione tende a 0, al tendere a 0 del passo di discretizzazione h.
La consistenza esprime il fatto che il problema discreto è una buona approssimazione del problema continuo:

$$\lim_{h \to 0} M_h(P) = M(P)$$

Inoltre un metodo di risoluzione si dice **consistente (o accurato) di ordine p** se:

$$T(t,h) = O(h^p) \tag{1.7}$$

Il problema discretizzato con il metodo di Eulero $M_h(P)$ è una buona approssimazione del problema continuo $M(P)$, poiché:

$$\lim_{h \to 0} T(t,h) = \lim_{h \to 0} \frac{y(t+h) - y(t)}{h} - y'(t) = 0$$

Il rapporto incrementale tende alla derivata per $h \to 0$.
Per misurare l'accuratezza (o consistenza), basta applicare la definizione di *O-Grande*, per $h \to 0$, su $T(t,h)$ per notare che:

$$T(t,h) = O(h)$$

Il metodo di Eulero è consistente di ordine 1.

1.4.2 Errore di Troncamento Locale

L'errore di troncamento locale (LTE) equivale a:

$$\tau_i = y(t_i) - y_i \quad con \quad y_j = y(t_j) \quad \forall j < i \tag{1.8}$$

calcolato assumendo che tutti i precedenti valori numerici ad i siano accurati.
Il metodo di Eulero approssima la funzione soluzione alla sua serie di Taylor arrestata al primo ordine:

$$\begin{cases} y(t_{i+1}) = y(t_i + h) = y(t_i) + f(t_i, y(t_i))h + O(h^2) \\ y_{i+1} = y_i + f(t_i, y_i)h \end{cases}$$

Assumendo la condizione che le soluzioni numeriche precedenti al passo i siano coincidenti con quelle analitiche, la 1.8 per il metodo di Eulero vale:

$$\tau_{i+1} = y(t_{i+1}) - y_{i+1} = O(h^2) \tag{1.9}$$

1.4.3 Errore di Troncamento Globale

Il metodo di Eulero è un metodo consistente, in cui il problema $M_h(P)$ è una buona approssimazione del problema $M(P)$.
Analizzando le soluzioni dei due problemi:

$$\begin{aligned} M(P) &\rightarrow y(t) \\ M_h(P) &\rightarrow (y_0, y_1, ..., y_{n-1}) \end{aligned}$$

bisogna capire se la soluzione discreta è una buona approssimazione della soluzione continua calcolata negli stessi punti:

$$\begin{aligned} y_0 &\approx y(t_0) \\ y_1 &\approx y(t_1) \\ ... &\approx ... \\ y_{n-1} &\approx y(t_{n-1}) \end{aligned}$$

Data la funzione $u : [t_0, T] \rightarrow \mathbb{R}$ tale che:

$$u(t_i) = y_i \qquad \forall i = 0, 1, ..., n-1 \tag{1.10}$$

essa è soluzione del problema $M_h(P)$:

$$M_h(P) : \begin{cases} \dfrac{u(t_{i+1}) - u(t_i)}{h} = f(t_i, u(t_i)) \\ u(t_0) = y(t_0) \end{cases} \qquad \forall i = \{0, 1, ..., n-1\}$$

mentre $y(t)$ è soluzione del problema continuo:

$$M(P) : \begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0 \end{cases} \qquad \forall t \in [t_0, T]$$

L'**errore globale di troncamento** (GTE) viene definito come:

$$E_i(h) = y(t_i) - u(t_i) \quad \forall i = \{0, 1, ..., n-1\} \tag{1.11}$$

ovvero al passo $i - esimo$ l'errore è la differenza tra la soluzione discreta e quella analitica, e misura l'*accuratezza* della soluzione numerica.

A differenza dell'errore di troncamento locale, in questo caso si tiene conto degli errori commessi negli step precedenti al passo $i-esimo$.
Una stima dell'errore di troncamento globale all'utimo step $N-esimo$ può essere calcolata nel seguente modo:

$$E_N \approx N\tau_i = NO(h^2) \approx O(h) \tag{1.12}$$

Dato che è noto l'errore in ogni step.

Convergenza

Un metodo per la risoluzione di un problema ai valori iniziali si dice **convergente** se, posto $E_0 = 0$:

$$\begin{cases} E_0 = y(t_0) - u(t_0) \\ u(t_0) = y(t_0) \end{cases} \quad \Rightarrow \quad E_0 = y(t_0) - y(t_0) = 0$$

fissato $t = t_0 + ih$ risulta:

$$\lim_{h\to 0} E_i(h) = 0 \qquad opp. \qquad \lim_{i\to\infty} E_i(h) = 0$$

Inoltre un metodo si dice **convergente di ordine p** se risulta:

$$E_i(h) = O(h^p) \tag{1.13}$$

La convergenza esprime il fatto che la soluzione del problema discreto è una buona approssimazione della soluzione del problema continuo. La 1.12 non è sufficiente nel dimostrare che il metodo è convergente.

Teorema 1.4.1 *Sia $y(t) \in C^2[t_0, T]$ soluzione del problema $M(P)$. Sia inoltre $y(t)$ Lipschitziana di costante L in ogni intervallo $[t_i, t_{i+1}]$, ovvero:*

$$|y(t_i) - y(t_{i+1})| \le L|t_i - t_{i+1}| \quad \Rightarrow \quad \frac{|y(t_i) - y(t_{i+1})|}{|t_i - t_{i+1}|} \le L \qquad \forall i \in \{0, 1, ..., n-1\}$$

ovvero che la pendenza in ogni intervallo non superi una determinata angolazione.
L'errore di troncamento globale viene calcolato come:

$$|E_i(h)| \le e^{(T-t_0)L}|E_0| + \frac{e^{(T-t_0)L} - 1}{L}\frac{C}{2}h$$

con

$$C = \max_{[t_0,T]} y''(t)$$

Ponendo $E_0 = 0$ si ha:

$$|E_i(h)| \le \frac{Ce^{(T-t_0)L} - 1}{2L}h \approx O(h) \tag{1.14}$$

quindi il metodo di Eulero è **convergente di ordine 1**.

Esempio 1.4.1 *Sia la seguente equazione differenziale con* $t \in [0,1]$*:*

$$\begin{cases} \dfrac{dy}{dt} = -2y(t) + 1 \\ y(t_0) = 1 \end{cases}$$

La soluzione analitica è la seguente:

$$y(t) = \frac{e^{-2t} + 1}{2}$$

Uno script per valutare l'errore è il seguente:

```
%Eq. Differenziale
f = @(t,y) -2*y+1;
%Dati iniziali
t0 = 0;
y0 = 1;
T = 1;
h = [0.2 0.1 0.05];
global_error = zeros(3,1);

%Soluzione analitica da confrontare
t = 0:0.001:1;
y_analitica = (exp(-2.*t)+1)./2;
plot(t,y_analitica);

hold on;
grid;
%Calcolo delle soluzioni numeriche
for i=1:3
        [y_eulero,t] = eulero(f, y0, t0, T, h(i));
        y_analitica = (exp(-2.*t)+1)./2;
        plot(t,y_eulero);
        global_error(i,1) = max(y_analitica-y_eulero)
end
legend( 'Soluzione analitica','h=0.2', 'h=0.1', 'h=0.05');
```

Il cui risultato è il seguente:

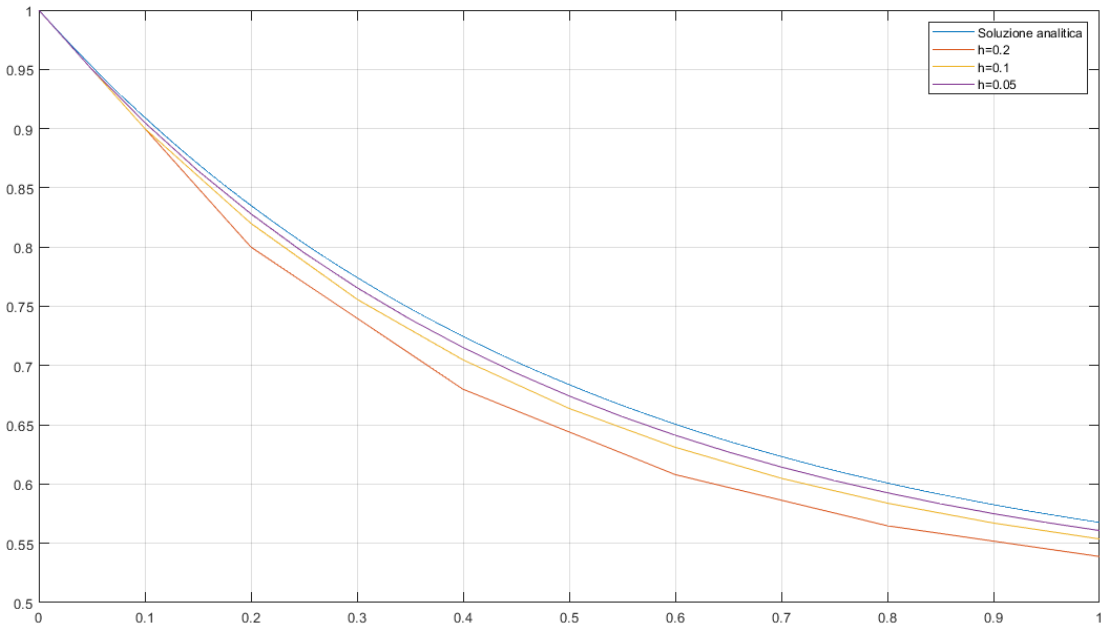


Figura 1.3: *Confronto delle soluzioni numeriche*

Il vettore global_ error $= \begin{pmatrix} 0.44 \times 10^{-1} & 0.20 \times 10^{-1} & 0.96 \times 10^{-2} \end{pmatrix}$

Come si nota dall'esempio se si vuole dimezzare l'errore (raddoppiare la precisione) bisogna dimezzare anche il passo di discretizzazione h.
In termini computazionali il problema è notevole. Se si vuole avere una precisione tale che $h = 10^{-8}$ allora:

$$N = \frac{T}{h} \approx 10^8$$

Il che richiede dunque un gran numero di iterazioni per risolvere un problema.

1.4.4 Errore di Rounf-Off

La soluzione discreta viene calcolata all'interno di un sistema aritmetico a precisione finita (computer) quindi è soggetta ad un errore di round-off.
Indicando con $\tilde{u}(t_i)$ il valore della soluzione discreta $u(t_i)$ calcolata in un sistema aritmetico floating-point, l'**errore globale di round-off** viene definito come:

$$R_i(h) = u(t_i) - \tilde{u}(t_i) \tag{1.15}$$

Teorema 1.4.2 *Sia $u(t_i)$ Lipschitziana e soluzione del problema discreto $M_h(P)$ l'errore globale di round-off del metodo di Eulero, vale:*

$$|R_i(h)| \leq e^{(T-t_0)L}|R_0| + \frac{e^{(T-t_0)L} - 1}{L}\frac{\rho}{h} \tag{1.16}$$

dove R_0 indica l'errore di round-off nel dato iniziale e:

$$\rho = \max_i(\rho_i)$$

in cui ρ_i è l'errore locale di round-off commesso al passo $i-esimo$.

Dal teorema si deduce che al crescere di h diminuisce l'errore di round-off globale.
Tale affermazione va però contro la 1.14, in cui l'errore globale diminuisce alla diminuzione di h.

1.4.5 Errore Totale

L'errore totale è l'errore che tiene in considerazione sia l'errore globale di round-off sia l'errore globale di troncamento. Per adesso quindi:

$$\begin{cases} |E_i(h)| \leq e^{(T-t_0)L}|E_0| + \dfrac{e^{(T-t_0)L}-1}{L}\dfrac{C}{2}h \\ |R_i(h)| \leq e^{(T-t_0)L}|R_0| + \dfrac{e^{(T-t_0)L}-1}{L}\dfrac{\rho}{h} \end{cases}$$

In entrambi i casi il primo addendo riguarda l'accumulo dell'errore a partire da quello iniziale, mentre il secondo addendo riguarda l'accumulo degli errori locali. Se h diminuisce:

- L'errore globale di round-off $R_i(h)$ cresce
- L'errore globale di troncamento $E_i(h)$ decresce

L'errore totale è definito come:

$$E_i^{tot}(h) = |y(t_i) - \tilde{u}(t_i)| \tag{1.17}$$

differenza della soluzione $i-esima$ del problema continuo $M(P)$ e soluzione discreta in un sistema a precisione finita. Sommando e sottraendo alla 1.17 la quantità $u(t_i)$ si ha:

$$E_i^{tot}(h) = |y(t_i) - u(t_i) + u(t_i) - \tilde{u}(t_i)| = |E_i(h) + R_i(h)| \leq |E_i(h)| + |R_i(h)| \tag{1.18}$$

combinando quindi tutte le equazioni:

$$E^{tot}(h) \leq e^{(T-t_0)L}\left(|E_0| + |R_0|\right) + \frac{e^{(T-t_0)L}-1}{L}\left(\frac{C}{2}h + \frac{\rho}{h}\right) \tag{1.19}$$

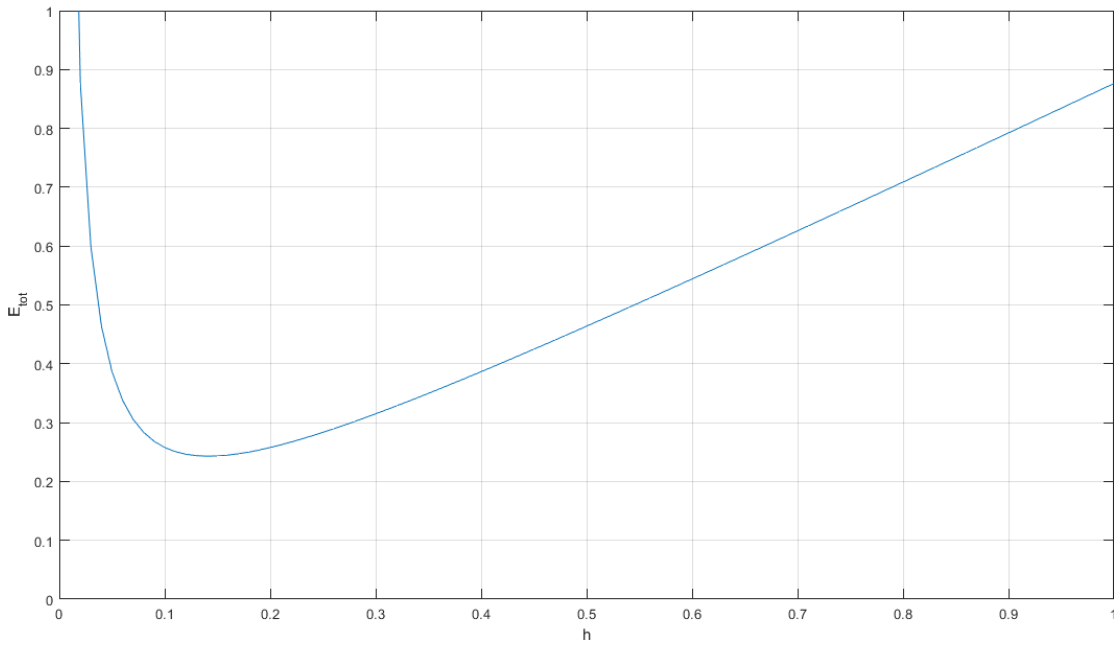


Figura 1.4: *Errore totale in funzione di h*

Il valore ottimo di h è quello che minimizza la funzione $E^{tot}(h)$. Calcolando quindi il valore per cui:

$$\frac{dE^{tot}(h)}{dh} = 0 \quad \Rightarrow \quad \frac{C}{2L}h_{opt} = \frac{\rho}{L}\frac{1}{h_{opt}}$$

in definitiva:

$$h_{opt} = \sqrt{\frac{2\rho}{C}}$$

Il valore sopra calcolato in realtà è un valore teorico. La funzione soluzione $y(t)$ infatti non è conosciuta il più delle volte, non potendo quindi calcolare ρ e C.

1.5 Stabilità

L'algoritmo di Eulero è un metodo *consistente* e *convergente*. In generale, però, il valore h scelto per ricavare la soluzione indice molto sulla stessa. Per valori di h abbastanza grandi, infatti, l'errore potrebbe divergere rendendo la soluzione inaccettabile.

Esempio 1.5.1 *Sia la seguente equazione differenziale con* $t \in [0, 0.8]$*:*

$$\begin{cases} \dfrac{dy}{dt} = -15y(t) \\ y(t_0) = 0.6 \end{cases}$$

La soluzione analitica è la seguente:

$$y(t) = 0.6e^{-15t}$$

Calcolando la soluzione numerica con vari valori di h si ha:

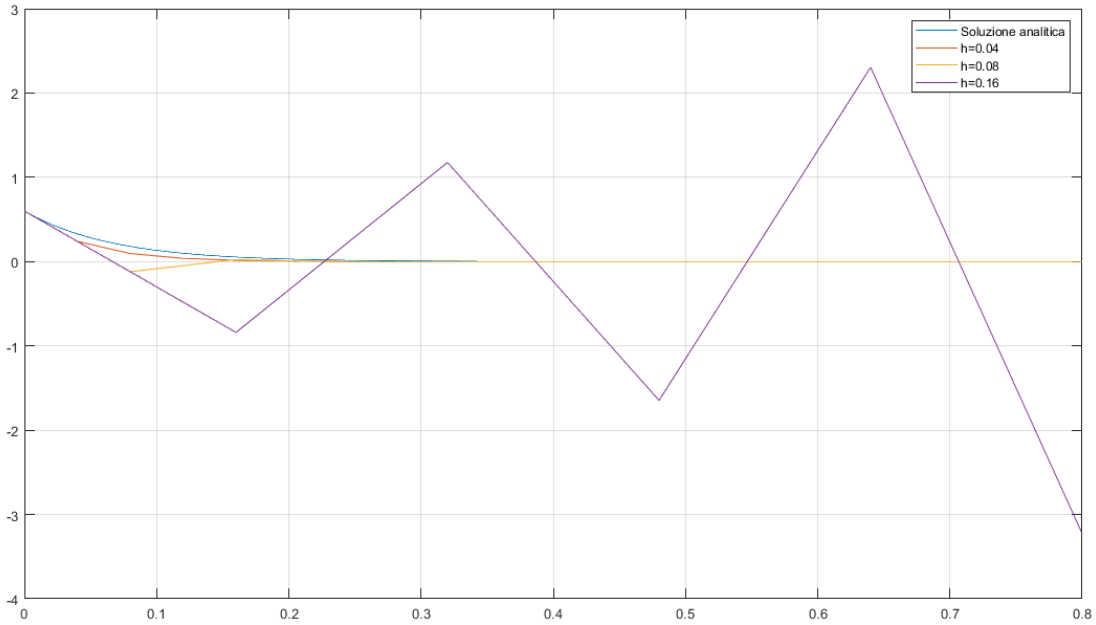


Figura 1.5: *Soluzione calcolata con h=0.04, h=0.08, h=0.16*

Quindi con $h = 0.16$ *la soluzione è del tutto inaccettabile, rendendo il metodo di Eulero un metodo per niente accurato.*

LA stabilità di un metodo è data dal fatto che esso stesso non deve propagare un errore iniziale durante le successive iterazioni.
Sia $M_h(P)$ un problema discreto e $M_h^e(P)$ lo stesso problema discretto con un errore sul dato iniziale:

$$M_h(P): \begin{cases} \dfrac{u(t_{i+1}) - u(t_i)}{h} = f(t_i, u(t_i)) \\ u(t_0) = y(t_0) \end{cases} \qquad \forall i = \{0, 1, ..., n-1\}$$

e perturbando il problema con un errore ϵ_0:

$$M_h^e(P): \begin{cases} \dfrac{v(t_{i+1}) - v(t_i)}{h} = f(t_i, v(t_i)) \\ v(t_0) = y(t_0) + \epsilon_0 = v_0 \end{cases} \qquad \forall i = \{0, 1, ..., n-1\}$$

l'**errore di propagazione** al passo $i - esimo$ viene calcolato come:

$$\epsilon_i(h) = v(t_i) - u(t_i) \tag{1.20}$$

Partendo da questo calcolo si può definire la stabilità assoluta.

1.5.1 Stabilità assoluta

Siano $u(t_i)$ e $v(t_i)$ le soluzioni determinate con un metodo per la risoluzione di un problema ai valori iniziali rispettivamente ai dati iniziali y_0 e v_0, il metodo utilizzato si dice

assolutamente stabile per un dato valore di h se:

$$|\epsilon_0| \leq C \qquad \Rightarrow \qquad |\epsilon_i(h)| \leq C \quad \forall i \in \{0, 1, ..., n-1\}$$

con C costante (upper bound).
Intuitivamente significa che l'errore in ogni intervallo di calcolo soddisfa la stessa limitazione dell'errore iniziale in tutto l'intervallo $[t_0, +\infty]$.

1.5.2 Indice di condizionamento

In riferimento dell'esempio precedente si intuisce che il valore di h incide molto sulla soluzione. Il sistema di calcolo è un sistema aritmetico a precisione finita, quindi è importante dare una stima dell'indice di condizionamento. Se si riesce a trovare una funzione g_i tale che associa al dato iniziale y_0 il valore della soluzione del problema discreto $M_h(P)$ allora:

$$C_A(g_i, y_0) = |g_i'(y_0)|$$

corrisponde all'indice di condizionamento assoluto.
Sulla base di ciò dato un problema di test come il seguente:

$$\begin{cases} \dfrac{dy}{dt} = \lambda y(t) \\ y(t_0) = y_0 \end{cases} \tag{1.21}$$

Per la legge di aggiornamento di Eulero si ha:

$$y_{i+1} = y_i + hf(t_i, y_i) \quad con \quad f(t_i, y_i) = \lambda y_i$$

$$y_{i+1} = y_i + h\lambda y_i = (1 + h\lambda)y_i = (1 + h\lambda)^2 y_{i-1} = ... = (1 + h\lambda)^i y_0$$

quindi la funzione per calcolare l'indice di condizionamento del problema vale:

$$g_i(y_0) = (1 + h\lambda)^i y_0 \tag{1.22}$$

Proposizione 1.5.1 (Condizionamento) *Il metodo di Eulero è ben condizionato se e solo se:*

$$\lambda = 0 \qquad opp. \qquad \left(\lambda < 0 \ \wedge \ h < -\frac{2}{\lambda}\right) \tag{1.23}$$

Dimostrazione. Indicata con $g_i(y_0)$ la funzione che al dato iniziale y_0 associa il valore della soluzione y_i del problema discreto, relativo al problema test 1.21, nel punto t_i, si ha per la 1.22:

$$g_i(y_0) = y_i = (1 + h\lambda)^i y_0$$

L'indice di condizionamento assoluto vale quindi:

$$C_A(g_i, y_0) = |g_i'(y_0)| = |(1 + h\lambda)^i|$$

Il problema è ben condizionato se:

$$|(1 + h\lambda)|^i \leq 1$$

Dato che $i \in \mathbb{N}$ allora:

$$|(1+h\lambda)| \leq 1 \quad \Rightarrow \quad -1 \leq (1+h\lambda) \leq 1$$

quindi, valutando le tre possibili situazioni:

- λ=**0** Il problema è ben condizionato.
- λ>**0** L'equazione non ammette soluzioni accettabili, poiché richiede che h sia negativo:
$$|(1+h\lambda)| \leq 1 \quad \Rightarrow \quad (1+h\lambda) \leq 1 \quad \Rightarrow \quad h \leq 0$$
quindi il problema è sempre mal condizionato per qualunque h.
- λ<**0** Il problema è ben condizionato solo per alcuni valori di h:
$$|(1+h\lambda)| \leq 1 \quad \Rightarrow \quad |(1-h|\lambda|)| \leq 1 \quad \Rightarrow \quad -\frac{2}{|\lambda|} \leq -h \leq 0 \quad \Rightarrow \quad 0 \leq h \leq \frac{2}{|\lambda|}$$
Si ha dunque la tesi.

Esempio 1.5.2 *Riprendendo quindi l'esempio precedente, la divergenza si aveva per:*

$$h = 0.16 \qquad \lambda = -15$$

Questo perchè l'indice di condizionamento è maggiore di 1, andando ad amplificare l'errore, infatti:

$$\frac{2}{|\lambda|} = \frac{2}{15} = 0.1\bar{3} < 0.16 = h$$

Dunque il problema era mal condizionato. Per gli altri valori di h invece era ben condizionato.

1.5.3 Zero-Stabilità

A differenza della stabilità assoluta, che studia l'andamento della soluzione con h prefissato e in un intervallo $[t_0, +\infty]$, la zero-stabilità studia la soluzione numerica nell'intervallo $[t_0, T]$ con $h \to 0$. Un metodo di risoluzione a step è **zero-stabile** se esistono $h_0 > 0$ e $K > 0$ tali che:

$$|\epsilon_i(h)| \leq K|\epsilon_0| \quad per \quad h \leq h_0$$

Il metodo di Eulero è zero-stabile poiché:

$$|\epsilon_i(h)| \leq e^{(T-t_0)L}|\epsilon_0| \quad con \quad e^{(T-t_0)L} = K$$

La zero-stabilità dipende inoltre dall'errore iniziale ϵ_0. La propagazione si può mantenere sotto ad una determinata soglia se ϵ_0 è "sufficientemente piccolo".

Teorema 1.5.2 (Teorema di Lax-Dahlquist) *Un metodo numerico è consistente e zero-stabile se e solo se è convergente. Vale il viceversa.*
Inoltre l'ordine di consistenza coincide con l'ordine di convergenza.

Il teorema sopra citato è molto importante. Attraverso la conoscenza di stabilità e consistenza, parametri più semplici da calcolare, si possono ricavare informazioni sulla convergenza, parametro invece più difficile da calcolare.

1.6 Esempio semplice

Si consideri il seguente problema in $t = [0, 1]$:

$$\begin{cases} \dfrac{dy}{dt} = -10y(t) \\ y(t_0) = 1 \end{cases}$$

La soluzione analitica è conosciuta e vale:

$$y(t) = e^{-10t}$$

1.6.1 Stabilità

In prima analisi bisogna fissare un passo di discretizzazione h (o il numero di intervalli N) limite, in modo da mantenersi in una regione di stabilità. Per la 1.23 si ha:

$$h < \frac{2}{10} = 0.2$$

oppure:

$$h = \frac{T - t_0}{N} \quad \Rightarrow \quad N = \frac{T - t_0}{h} > \frac{1}{0.2} > 5$$

Eseguendo il seguente script infatti (per N=5 e N=20):

```
lambda = -10;

f = @(t,y) lambda*y;   %Soluzione y(t) = e^(lambda*t)
N = [5 10^7];          %Numero di iterazioni
t0 = 0; T=1; y0=1;     %Condizioni iniziali
h=(T-t0)./N;           %Passo di discretizzazione

t = t0:h(end):T;      %Asse dei tempi per la soluzione analitica
y_analitica = exp(lambda.*t);
plot(t,y_analitica,'LineWidth', 2, 'Color', 'black');
hold on;

t = t0:h(1):T;        %Asse dei tempi
y_numerica = eulero32(f, y0, t0, T, h(1));
plot(t,y_numerica, 'LineWidth', 2);
xlabel('Tempo');
ylabel('Soluzione');
legend('Soluzione Analitica', 'h=5');
grid;
```

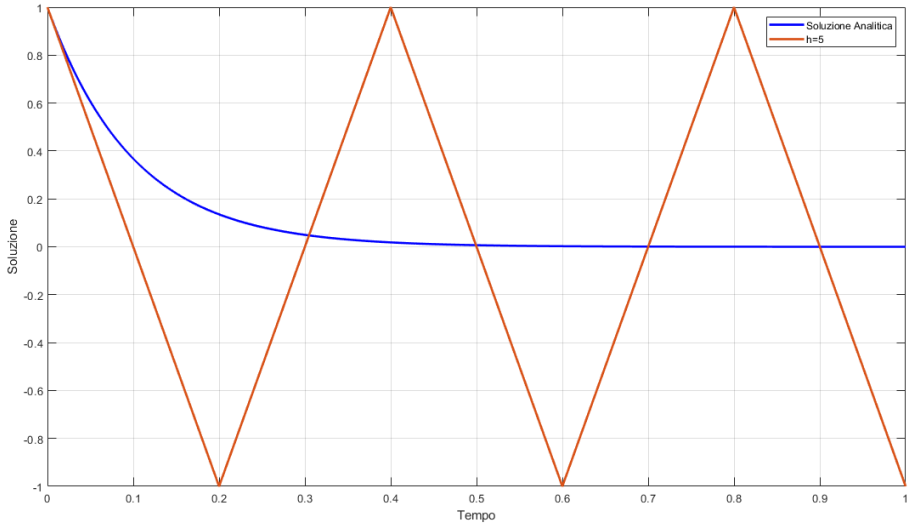


Figura 1.6: *N=5*

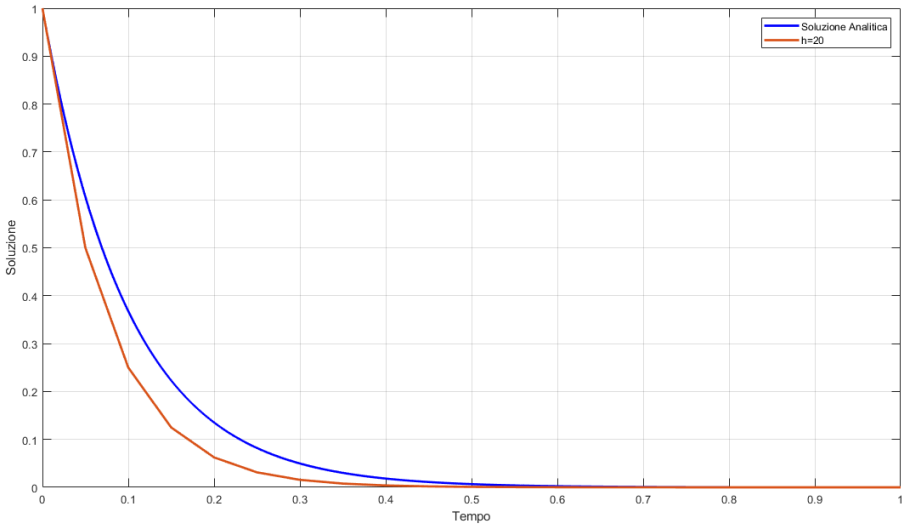


Figura 1.7: *N=20*

1.6.2 Errore Totale

L'errore totale prevede una componente di *errore di troncamento globale* (diminuisce al crescere di N) e una componente di *errore di roundoff* (aumenta al crescere di N). Per dare una visione grafica di quest'errore si sono calcolate varie soluzioni numeriche con diversi valori di N.

```
lambda = -10;

f = @(t,y) lambda*y;     %Soluzione y(t) = e^(lambda*t)
N = [10^2 10^3 10^4 10^5 10^6 10^7];
```

```
t0 = 0; T=1; y0=1;
h=(T-t0)./N;

e_tot = zeros(1,length(N));
t = t0:h(end):T;
y_analitica = exp(lambda.*t);
plot(t,y_analitica,'LineWidth', 2, 'Color', 'blue');

hold on;
for i=1:length(N)
t = t0:h(i):T;
y_numerica = eulero32(f, y0, t0, T, h(i)));
e_tot(i) = abs(y_numerica(end)-y_analitica(end));
plot(t,y_numerica, 'LineWidth', 2);
end

xlabel('Tempo');
ylabel('Soluzione');
legend('Soluzione Analitica','h=10^2', 'h=10^3', 'h=10^4','h=10^5','h=10^6','h=10^7
grid;
```

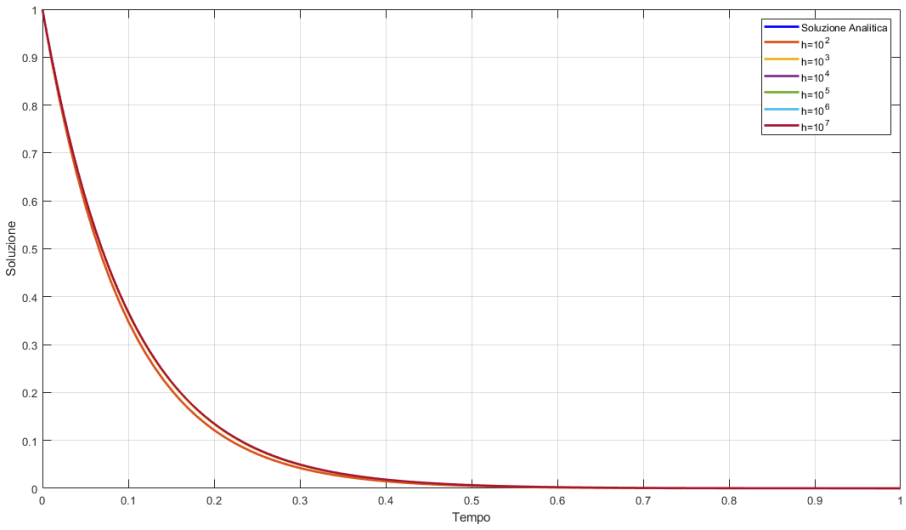


Figura 1.8: *Grafico di tutte le soluzioni numeriche calcolate*

Si può effettuare uno zoom sulle curve per vedere meglio i confronti:

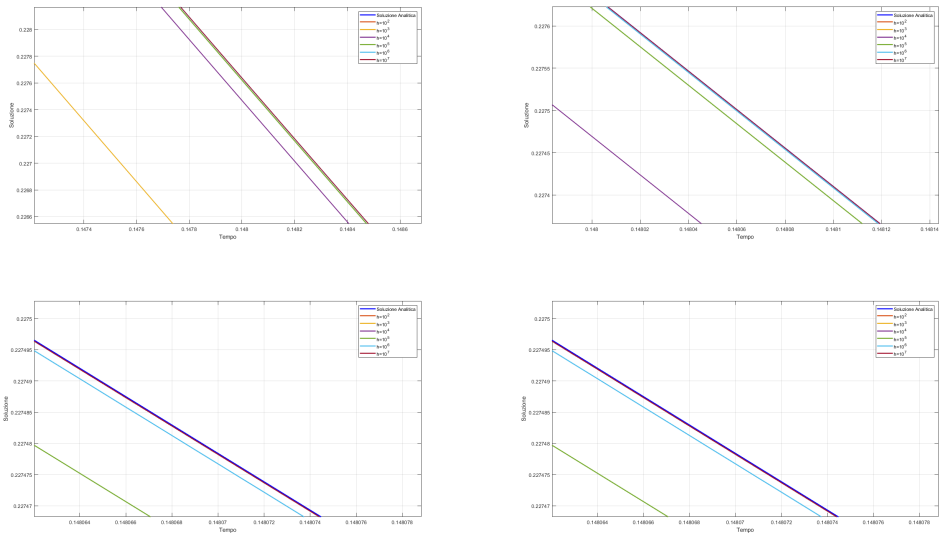


Figura 1.9: *Zoom del grafico in corrispondenza delle varie curve*

A questo punto si può studiare l'andamento dell'errore. Esso si propaga fino all'ultimo step. Per ogni curva numerica quindi si può calcolare l'errore totale come:

$$e_{tot} = |y_{numerica}(end) - y_{analitica}(end)|$$

Dato che l'interesse è sull'ordine di grandezza, il grafico può essere fatto in scala logaritmica. A partire dallo script precedente quindi:

```
loglog(N, e_tot,'LineWidth',2);
hold on;
loglog(N, e_tot, '+');
grid;
```

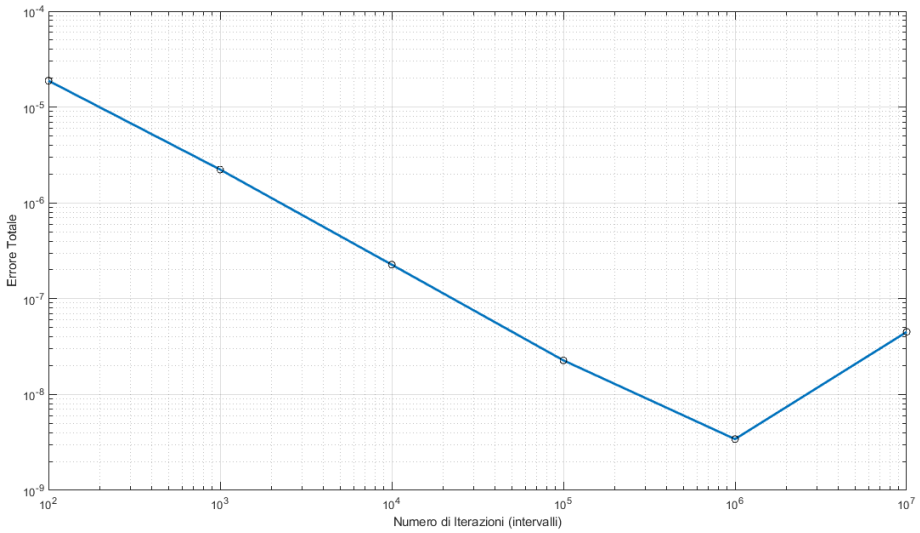


Figura 1.10: *Errore calcolato sull'ultima componente di ogni soluzione numerica rispetto alla soluzione analitica*

Nella figura si notato due andamenti diversi:

1. $N < 10^5$ si ha un andamento proporzionale a $O(h)$. Questo perché per valori piccoli di N (e grandi di h) prevale l'errore di troncamento invece dell'errore di roundoff.

2. $N > 10^5$ si inizia a perdere la linearità e si ha un'andamento in cui l'errore inizia a crescere, ciò significa che l'errore di roundoff prevale sull'errore di troncamento, poiché si tratta di valori di N molto grandi (e piccoli di h).

Errore di Troncamento Globale

Le soluzioni numeriche calcolate in precedenza sono state calcolate utilizzando una rappresentazione in *singola precisione*.
Per ricavare il contributo dell'errore di troncamento si può assumere che la soluzione in *doppia precisione* sia la soluzione numerica corretta, ovvero senza errore di roundoff.
Quindi l'errore vale:

$$gte = |y_{numerica\_double}(end) - y_{analitica}(end)|$$

Dato che si propaga, basta considerare l'ultimo elemento del vettore soluzione.

```
gte = zeros(1,length(N));
e_tot = zeros(1, length(N));
t = t0:h(end):T;
y_analitica = exp(lambda.*t);

hold on;
for i=1:length(N)
t = t0:h(i):T;  %Asse dei tempi
y_numerica = eulero32(f, y0, t0, T, h(i));
y_numerica_double = eulero64(f, y0, t0, T, h(i));
e_tot(i) = abs(y_analitica(end)-y_numerica(end));
gte(i) = abs(y_analitica(end)-y_numerica_double(end));
end

figure;
p1 = loglog(N, e_tot, 'LineWidth', 2);
hold on;
loglog(N, e_tot, 'o', 'Color', 'Black');
p2 = loglog(N, gte, '--', 'Color', 'Red', 'LineWidth', 2);
xlabel('Numero di Iterazioni (intervalli)');
ylabel('Errore Totale');
legend([p1 p2], {'Errore Totale', 'GTE'});
```

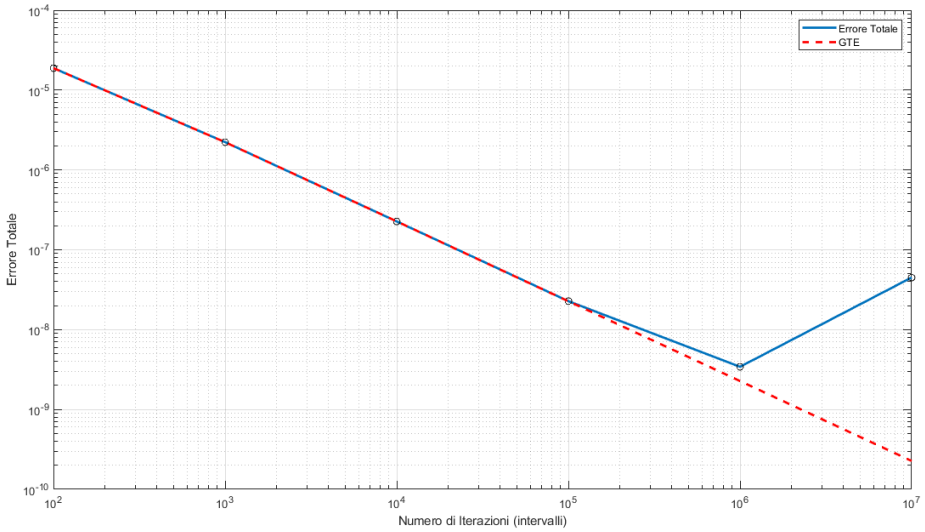


Figura 1.11: *Errore di troncamento calcolato sull'ultima componente di ogni soluzione numerica rispetto alla soluzione analitica*

Errore di Round-Off

Allo stesso modo si può calcolare l'errore di roundoff come differenza tra la soluzione in *singola precisione* e in *doppia precisione*:

$$roundoff = |y_{numerica\_double}(end) - y_{numerica}(end)|$$

```
gte = zeros(1,length(N));
e_tot = zeros(1, length(N));
roundoff = zeros(1,length(N));
t = t0:h(end):T;
y_analitica = exp(lambda.*t);

hold on;
for i=1:length(N)
t = t0:h(i):T;  %Asse dei tempi
y_numerica = eulero32(f, y0, t0, T, h(i));
y_numerica_double = eulero64(f, y0, t0, T, h(i));
e_tot(i) = abs(y_analitica(end)-y_numerica(end));
gte(i) = abs(y_analitica(end)-y_numerica_double(end));
roundoff(i) = abs(y_numerica(end) - y_numerica_double(end));
end

figure;
p1 = loglog(N, e_tot, 'LineWidth', 2);
hold on;
loglog(N, e_tot, 'o', 'Color', 'Black');
p2 = loglog(N, gte, '--', 'Color', 'Red', 'LineWidth', 2);
p3 = loglog(N, roundoff, '--','Color', 'Black', 'LineWidth', 2);
xlabel('Numero di Iterazioni (intervalli)');
ylabel('Errore Totale');
legend([p1 p2 p3], {'Errore Totale', 'GTE', 'RoundOff'});
```

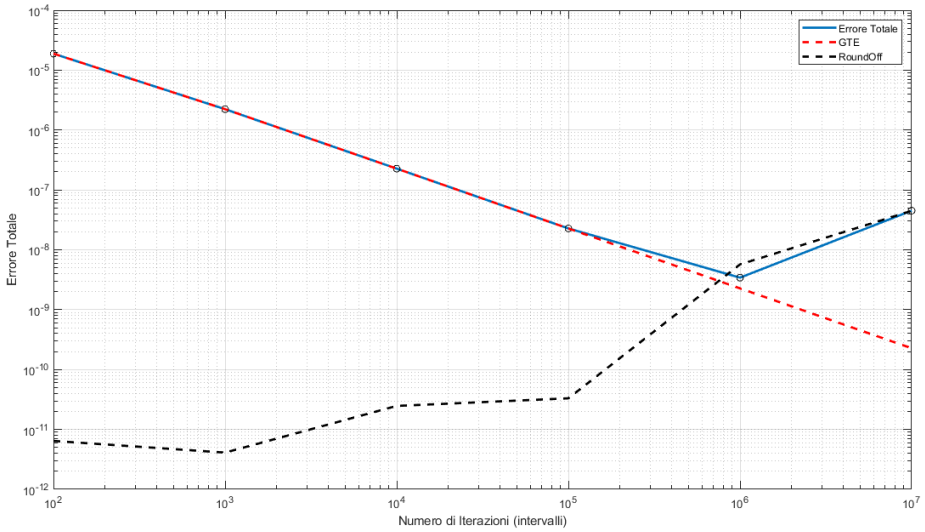


Figura 1.12: *Errore di roundoff calcolato sull'ultima componente di ogni soluzione numerica*

Nell'estremo destro del grafico si può notare come la curva dell'errore di roundoff diventa dominante nel errore totale.
Per verificare l'equazione 1.18 si può aggiungere al grafico la somma dell'errore di roundoff e l'errore di troncamento globale calcolato in precedenza.

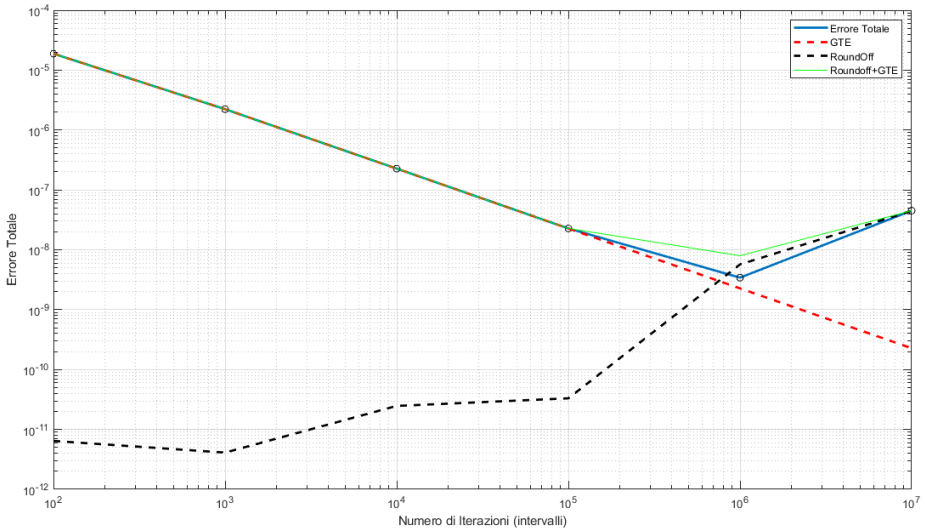


Figura 1.13: *Grafico degli errori calcolati*

Si nota quindi che:

$$e_{tot} < |roundoff| + |gte|$$

1.6.3 Numero di intervalli ottimo

Per avere una visione più chiara di ciò che succede intorno a 10^6, si possono calcolare le soluzioni e gli errori in tale intorno:

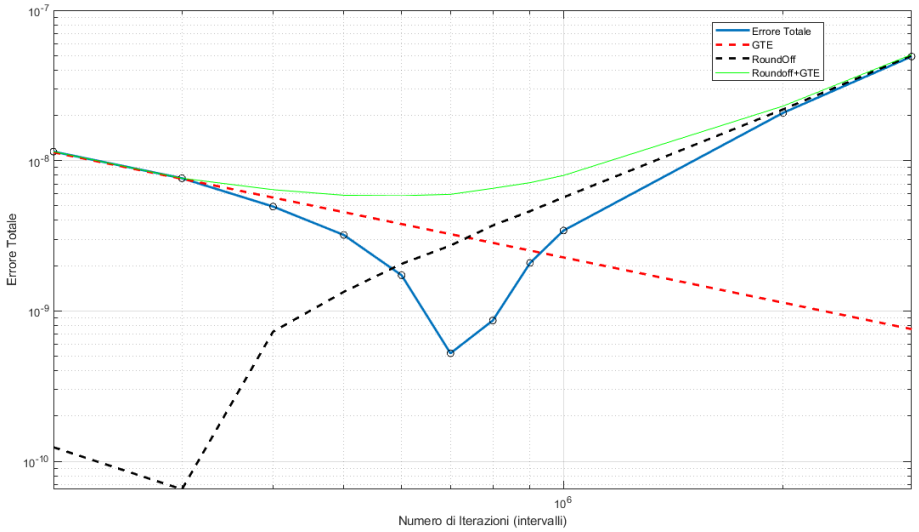


Figura 1.14: *Grafico degli errori calcolati*

In verde si è messo in risalto la funzione maggiorata dell'errore totale, molto simile alla funzione descritta in Fig.1.4.
Il valore ottimo di N (e di h) equivale all'incrocio delle due curve degli errori, poiché si ottiene la massima accuratezza.

$$N_{opt} \approx 7.2 \times 10^5 \qquad h_{opt} \approx 0.138 \times 10^{-5}$$

L'errore minimo che si commette è di circa 10^{-9}.
Se l'errore non è accettabile nel contesto in cui viene calcolata la soluzione, allora il metodo di Eulero non è il metodo più adatto. In tal caso bisogna affidarsi a metodi più precisi.

Capitolo 2

Applicazioni

2.1 Sistemi di Equazioni Differenziali

Fin ora sono stati tratti solo problemi di Cauchy del primo ordine, del tipo 1.1. Il più delle volte però si è costretti ad analizzare equazioni differenziali di ordine superiore al primo:

$$y^n(t) = f(t, y(t), y'(t), y''(t), ..., y^{n-1}(t)) \tag{2.1}$$

con $n-1$ condizioni iniziali.
Introducendo delle variabili ausiliarie:

$$z_1(t) = y(t) \quad z_2(t) = y'(t) \quad ... \quad z_N(t) = y^{n-1}(t)$$

si può ottenere una equazione differenziale del primo ordine equivalente:

$$\mathbf{z}'(t) = \begin{pmatrix} z_1'(t) \\ z_2'(t) \\ \vdots \\ z_n'(t) \end{pmatrix} = \begin{pmatrix} y'(t) \\ y''(t) \\ \vdots \\ y^n(t) \end{pmatrix} = \begin{pmatrix} z_2(t) \\ z_3(t) \\ \vdots \\ f(t, z_1(t), z_2(t), ..., z_n(t)) \end{pmatrix} = \mathbf{g}(t, \mathbf{z}(t))$$

A questo punto si può applicare il metodo di Eulero alla variabile $\mathbf{z}(t)$ multidimensionale.

Esempio 2.1.1 *Sia il seguente problema di Cauchy:*

$$x''(t) = -\lambda x(t)$$

con $x(0) = 0$ *e* $x'(0) = 0$ *Utilizzando* $\boldsymbol{v}(t)$ *come variabile ausiliaria:*

$$\boldsymbol{v}(t) = \begin{pmatrix} x(t) \\ x'(t) \end{pmatrix} = \begin{pmatrix} v_1(t) \\ v_2(t) \end{pmatrix}$$

La nuova equazione differenziale diventa:

$$\boldsymbol{v}'(t) = \begin{pmatrix} x'(t) \\ x''(t) \end{pmatrix} = \begin{pmatrix} v_2(t) \\ -\lambda v_1(t) \end{pmatrix} = \boldsymbol{g}(t, \boldsymbol{v}(t))$$

in cui $\boldsymbol{g}(t)$ *è la nuova funzione vettoriale che descrive* $\boldsymbol{v}'(t)$

L'implementazione in MATLAB prevede un cambiamento rispetto alla normale implementazione descritta in precedenza.

I parametri di ingresso alla funzione sono:

- f Funzione vettoriale dell'equazione differenziale di tipo 1.1
- y_0 Vettore delle condizioni iniziali al tempo t_0
- t_0 Tempo iniziale
- T Tempo finale
- h Passo di discretizzazione

I parametri in uscita sono:

- t Asse dei tempi discretizzato
- y Matrice soluzione soluzione

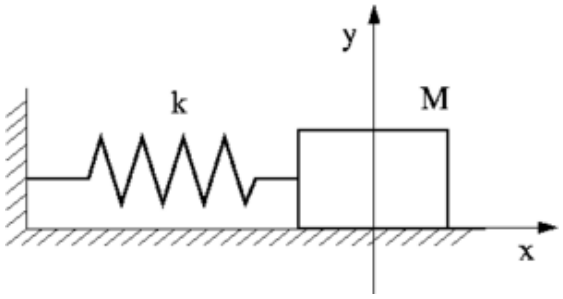
La soluzione $y(t)$ viene vista come un vettore di vettori. Definito il passo di discretizzazione h e, quindi, N numero di intervalli, la dimensione della soluzione è $n \times N$ (con n ordine dell'equazione differenziale).

```
function [y, t] = eulero32(f, y0, t0, T, h)
        t = single(t0:h:T);                 %Asse dei tempi discretizzata con passo h
        y = single(zeros(length(y0),length(t)));     %Vettore soluzione
        y(:,1) = y0;                    %Condizione iniziale

        for i=1:length(t)-1
                y(:,i+1) = y(:,i) + f(t(i), y(:,i))*h;    %Costruzione della soluzione
        end
end

function [y, t] = eulero64(f, y0, t0, T, h)
        t = t0:h:T;                   %Asse dei tempi discretizzata con passo h
        y = zeros(zeros(length(y0),length(t));     %Vettore soluzione
        y(:,1) = y0;                    %Condizione iniziale

        for i=1:length(t)-1
                y(:,i+1) = y(:,i) + f(t(i), y(:,i))*h;    %Costruzione della soluzione
        end
end
```

Il metodo di Eulero viene calcola insieme tutte le soluzioni. Ogni riga rappresenta un vettore soluzione.
Questa modifica rende il metodo di Eulero più generale. Infatti una equazione differenziale del primo ordine può essere vista come un sistema di equazioni vettoriali con una sola equazione.

2.2 Sistema Massa-Molla

Il sistema massa-molla è un sistema meccanico composto da un corpo di massa m, attaccato ad una molla con constante elastica k.



Il modello è descritto dalla seguente equazione differenziale:

$$m\frac{d^2x}{dt^2} = -kx$$

La soluzione analitica è conosciuta e vale:

$$x(t) = Acos(\omega_0 t) \quad con \quad \omega_0 = \frac{k}{m}$$

A questo punto si hanno tutti gli strumenti per analizzare l'errore prodotto dal metodo di Eulero.
Si inizia con il calcolo della soluzione numerica per vari N:

```
%Parametri
k=1;
m=1;

%Condizioni iniziali
t0=0; T=2;
x0=0; v0=1;
y0=[x0,v0];

%Passo di discretizzazione
N = [10^2 10^3 10^4 10^5 10^6 10^7];
h=(T-t0)./N;

%Soluzione anlitica
tempo = t0:h(end):T;
y_analitica = [sin(tempo); cos(tempo)];
plot(tempo, y_analitica(1,:), 'Color', 'Black', 'LineWidth',2);

%Inizializzazione degli errori
gte = zeros(1,length(N));
e_tot = zeros(1, length(N));
e_tot_equivalente = zeros(1, length(N));
roundoff = zeros(1,length(N));
plot(tempo,y_analitica(1,:));

%Soluzione numerica
hold on;
f = @(t,y) [y(2); -k/m*y(1)];
for i=1:length(N)
        t = t0:h(i):T;  %Asse dei tempi
        y_numerica = eulero32(f, y0, t0, T, h(i));
        y_numerica_double = eulero64(f, y0, t0, T, h(i));
        roundoff(i) = abs(y_numerica_double(1,end) - y_numerica(1,end));
        e_tot(i) = abs(y_analitica(1,end)-y_numerica(1,end));
```

```
        e_tot_equivalente(i) =  abs(y_numerica_double(1,end) -
↪  y_numerica(1,end) + y_analitica(1,end)-y_numerica_double(1,end));
        gte(i) = abs(y_analitica(1,end)-y_numerica_double(1,end));
        plot(t,y_numerica(1,:), 'LineWidth', 2);
end
legend('Soluzione Analitica', 'N=10^2', 'N=10^3','N=10^4', 'N=10^5',
↪  'N=10^6', 'N=10^7');
xlabel('Tempo');
ylabel('Soluzione');
grid;
```

L'output è il seguente:

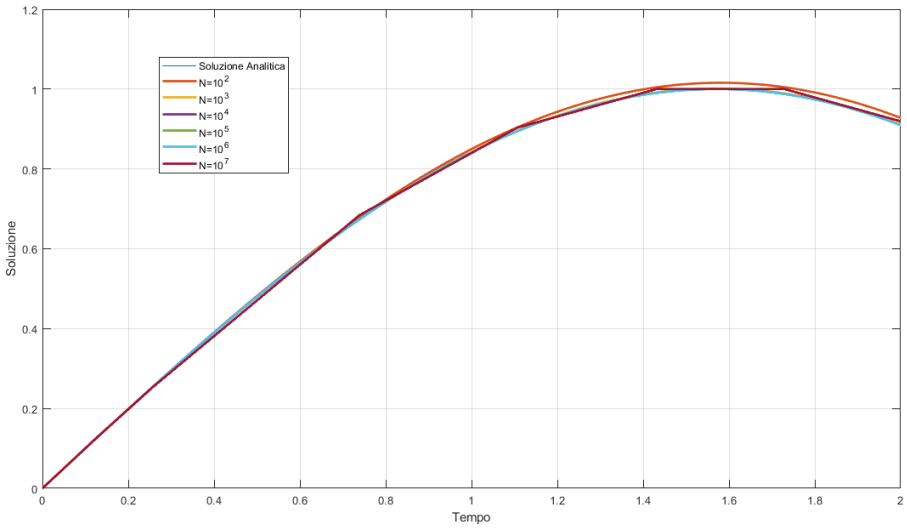


Figura 2.1: *Confronto tra le varie soluzioni calcolate*

Effettuando uno zoom del grafico si possono notare delle particolarità.

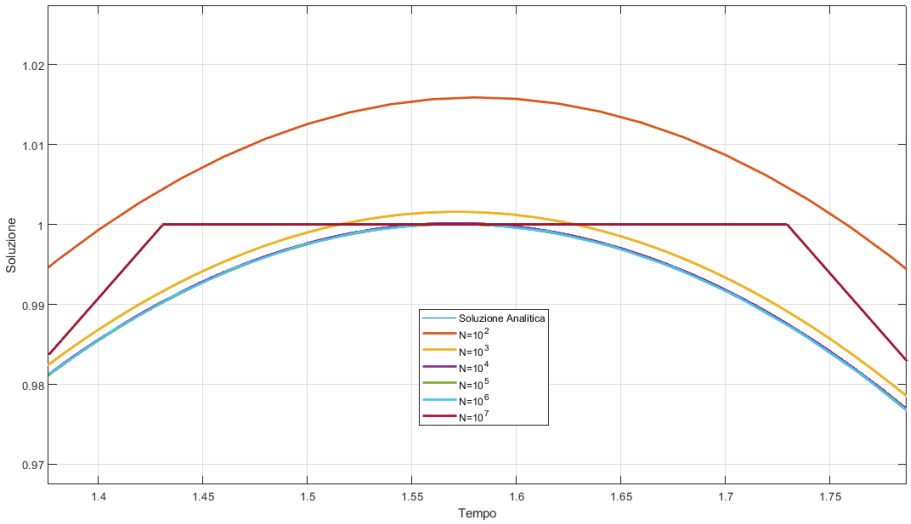


Figura 2.2: *Zoom delle soluzioni*

Già dal grafico si può notare l'influenza dell'errore di roundoff. Infatti per $N = 10^7$ l'andamento della soluzione numerica è molto diverso rispetto all'andamento della soluzione analitica.
Andando ad effettuare l'analisi degli errori infatti:

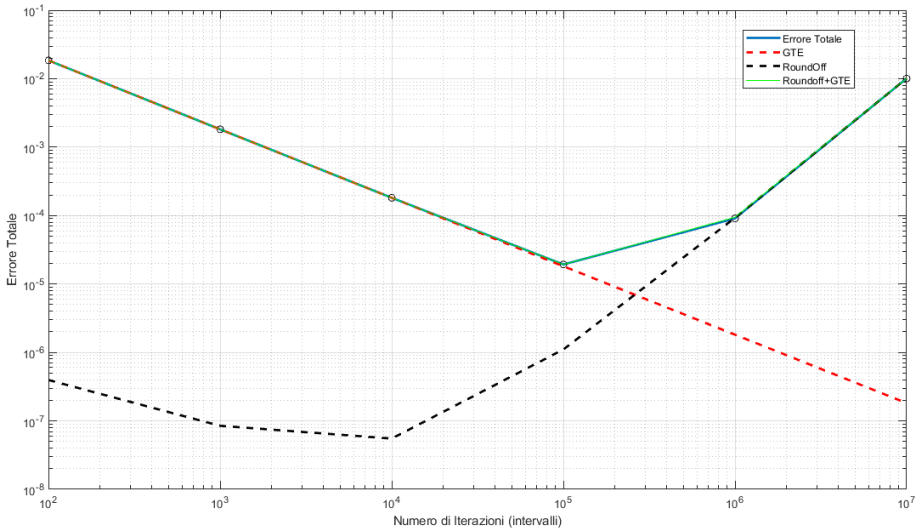


Figura 2.3: *Confronto degli errori*

Per $N = 10^7$ si ha un errore di 10^{-2}.
In questo caso il valore di N ottimo è:

$$10^5 < N_{opt} < 10^6$$