METODI STATISTICI E COMPUTAZIONALI

Stefania Spagnolo

Dipartimento di Matematica e Fisica, Univ. del Salento

LEZIONE 19-20

SOLUZIONE NUMERICA DI EQUAZIONI DIFFERENZIALI

Eq del primo ordine
Metodo di Eulero

EQUAZIONI DIFFERENZIALI

- Le equazioni differenziali intervengono nella descrizioni di sistemi fisici a partire dalla seconda legge della dinamica
- Uno strumento cruciale della fisica

- Affrontiamo le equazioni differenziali del primo ordine con condizione iniziale nota (problema di Cauchy per una equazione differenziale ordinaria)

$$y' = f(t, y) \quad \text{per } a \leq t \leq b; \quad y(a) = y_0$$

- Se la funzione f e le sue derivate sono continue, il problema di Cauchy ha soluzione e questa e' localmente unica (in un intorno di a) ossia e' possibile determinare una soluzione

$$y(t) \quad \text{tale che} \quad y' = \frac{dy}{dt} = f(t, y) \quad \text{e} \quad y(t = a) = y_0$$

EQUAZIONI DIFFERENZIALI

- Determinare una soluzione ***numerica*** di un problema di Cauchy consiste nel determinare un valore approssimato della funzione ***y(t)*** in N+1 punti equidistanti all'interno dell'intervallo [a,b].

- La soluzione numerica del problema di Cauchy consiste quindi in una successione $\{t_i,\ y_i\}$ di N+1 coppie di punti che approssimano l'andamento della soluzione esatta $y(t)$
  - h=(b-a)/N è il passo di discretizzazione.

- Il metodo numerico più semplice per risolvere un'equazione differenziale ordinaria del primo ordine è il ***metodo di Eulero***

EQUAZIONI DIFFERENZIALI

Metodo di Eulero

- La soluzione numerica del problema di Cauchy consiste quindi in una successione $\{t_i,\ y_i\}$ di N+1 coppie di punti che approssimano l'andamento della soluzione esatta $y(t)$
  - h=(b-a)/N è il passo di discretizzazione.

- Si parte dall'estremo ***a*** dove conosciamo il valore della funzione $y(a) = y_0$
- In un punto vicino ad a si ha

$$y(a+h) = y_0 + y'(a)h + \frac{1}{2}y''(\xi)h^2$$

- ***Il metodo di Eulero usa un'approssimazione al primo ordine***, e la successione di coppie si costruisce iterativamente, perciò

$$y_1 = y(a+h) = y_0 + y'(a)h = y_0 + f(a, y_0)\ h$$
$$y_2 = y(a+2h) = y_1 + y'(a+h)h = y_1 + f(a+h,\ y_1)\ h$$

Trascuro termini del secondo ordine, ***errore locale va come h^2***

EQUAZIONI DIFFERENZIALI

Metodo di Eulero

- La soluzione numerica del problema di Cauchy consiste quindi in una successione $\{t_i,\ y_i\}$ di N+1 coppie di punti che approssimano l'andamento della soluzione esatta $y(t)$
  - h=(b-a)/N è il passo di discretizzazione

$$t_0 = a \qquad y_0 = y(t_0)$$
$$t_1 = a + h \qquad y_1 = y(t_1) = y_0 + f(t_0, y_0)\ h$$
$$t_2 = a + 2h \qquad y_2 = y(t_2) = y_1 + f(t_1, y_1)\ h$$

- Al passo n+1

$$t_{n+1} = a + (n+1)h \qquad y_{n+1} = y(t_{n+1}) = y_n + f(t_n, y_n)\ h$$

EQUAZIONI DIFFERENZIALI

Metodo di Eulero

- esempio
  - Problema di Cauchy

$$y' = \frac{y^2}{2} - 2\sin t + 4 \qquad \text{con} \qquad y_0 = 0$$

  - Cerchiamo la soluzione numerica nell'intervallo 0 < x < 1

$$f(t, y) = \frac{y^2}{2} - 2\sin t + 4$$

Continua e derivable con derivate continue sia rispetto a t che a y

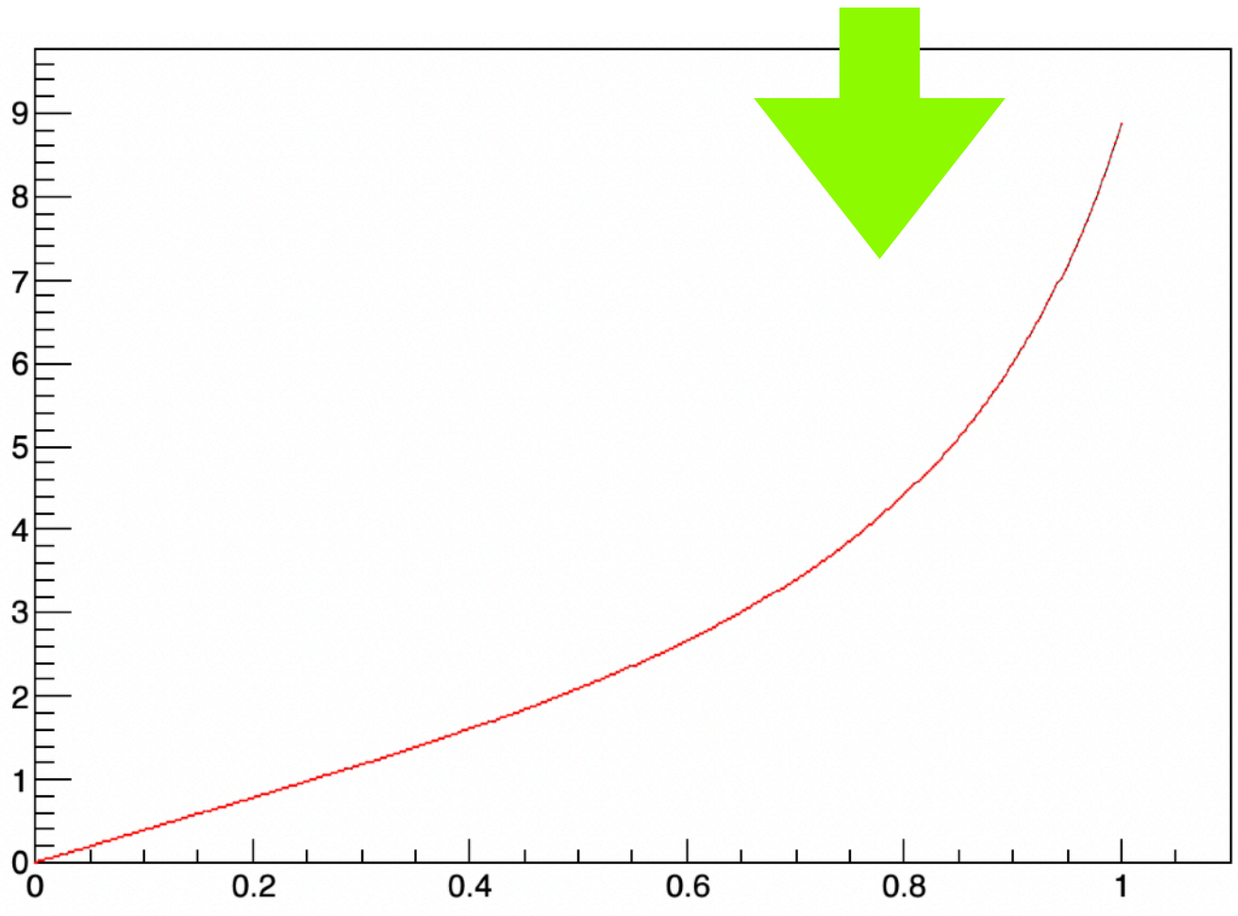
Soluzione unica nell'interno di t=0 (dove e' nota la condizione iniziale)

ESEMPIO

$$y' = \frac{y^2}{2} - 2\sin t + 4 \qquad \text{con} \qquad y_0 = 0$$

EuleroGraph.C

```
root [0] .x EuleroGraph.C(500,0.,1)
```




```
#include "TGraph.h"

// determiniamo la soluzione dell'equazione
// differenziale y'=0.5y^2-2sin(t)+4
// con la consizione iniziale y(0)=0 usando
// l'algoritmo di Eulero nell'intervallo
// 0<t<1
double f(double t, double y) {
   double f=0.5*y*y-2.*sin(t)+4.;
   return f;
}
void EuleroGraph(int NPoint=1000,
                 double a=0,
                 double b=1)
{
   TGraph * gEulero = new TGraph();
//step usato per la discretizzazione
    double h;
    h=(b-a)/double(NPoint);
// condizioni iniziali
    double t=0;
    double y=0;
    gEulero->AddPoint(t, y);
// Loop principale
    for (int i=0; i<NPoint; i++)  {
// formula di Eulero
      y=y+h*f(t,y);
      t=t+h;
      printf("t = %f ; y= %f\n",t,y);

      gEulero->AddPoint(t, y);
    }
     gEulero->SetMarkerStyle(20);
     gEulero->SetMarkerColor(kRed);
     gEulero->SetMarkerSize(0.1);
     gEulero->Draw("APC");
}
```

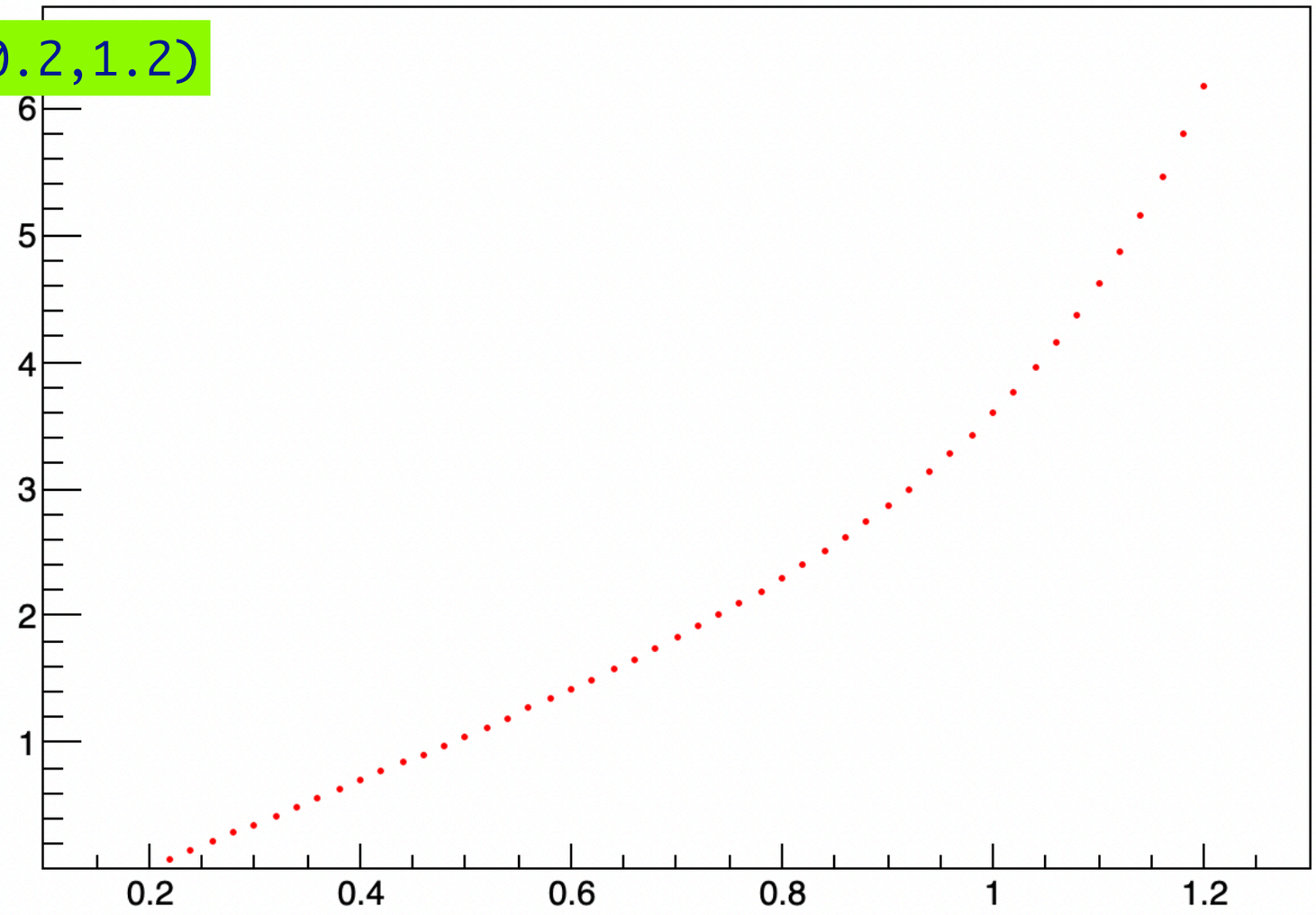
ESEMPIO

$$y' = \frac{y^2}{2} - 2\sin t + 4 \qquad \text{con} \qquad y_0 = 0$$

EuleroGraph.C

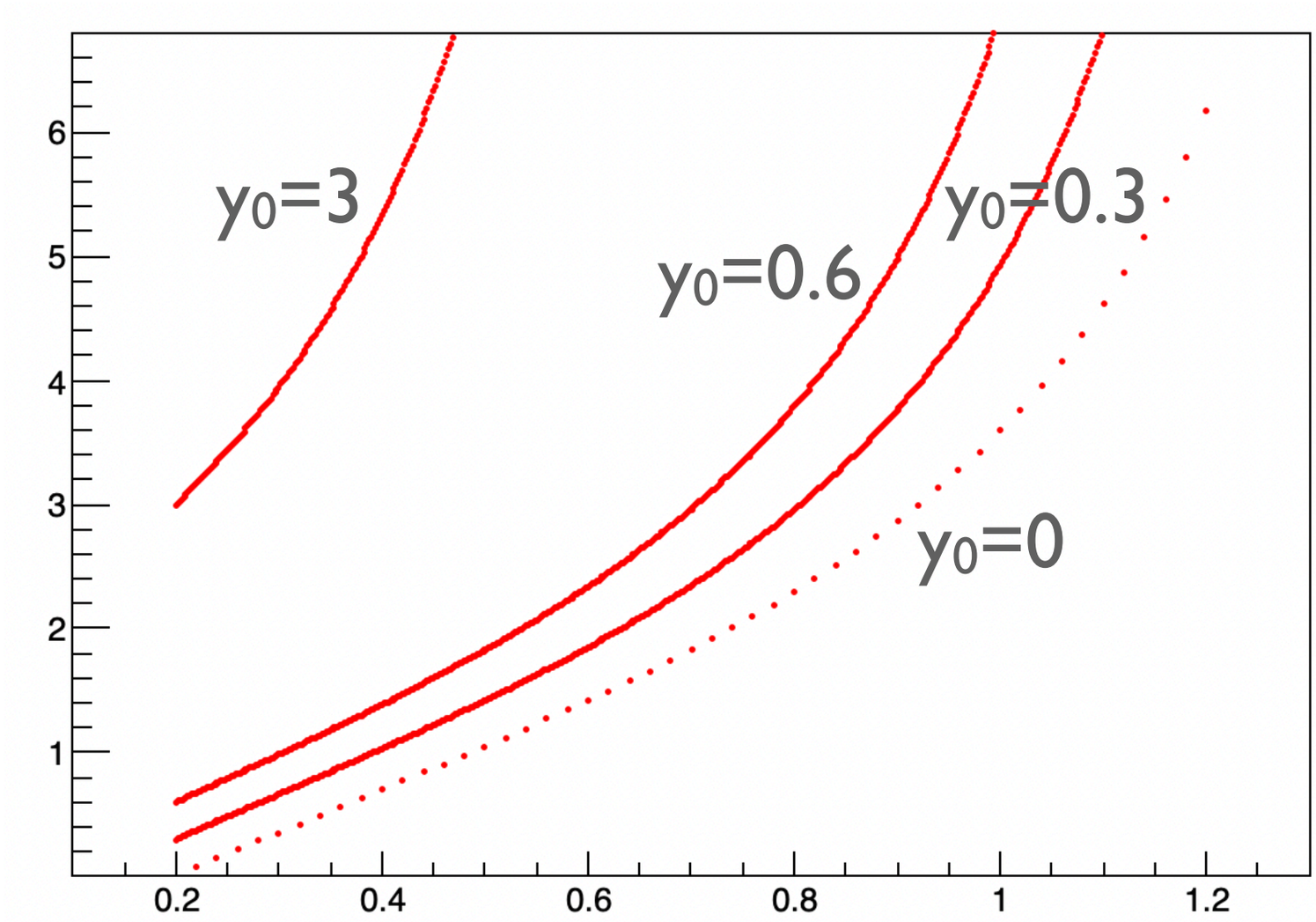
```
root [0] .x EuleroGraph.C(50,0.2,1.2)
```

```
root [0] .x EuleroGraph.C(50,0.2,1.2)
t = 0.200000 ; y= 0.000000
t = 0.220000 ; y= 0.072053
t = 0.240000 ; y= 0.143376
t = 0.260000 ; y= 0.214073
t = 0.280000 ; y= 0.284248
t = 0.300000 ; y= 0.354002
t = 0.320000 ; y= 0.423435
t = 0.340000 ; y= 0.492645
t = 0.360000 ; y= 0.561732
t = 0.380000 ; y= 0.630797
t = 0.400000 ; y= 0.699939
t = 0.420000 ; y= 0.769262
t = 0.440000 ; y= 0.838869
t = 0.460000 ; y= 0.908868
t = 0.480000 ; y= 0.979371
t = 0.500000 ; y= 1.050491
```



ESEMPIO

$$y' = \frac{y^2}{2} - 2\sin t + 4 \qquad \text{con} \qquad y_0 = 0$$



METODO DI EULERO - ERRORE

- ***Qual*** e' l'***errore*** che si commette ?
- L'algoritmo di Eulero approssima la funzione con una spezzata.
  - L'errore che si commette nell'approssimazione dipende dalla scelta del passo
  - più piccolo è il passo minore sarà l'errore.

- Analizziamo un problema di Cauchy facilmente risolubile analiticamente

$$\begin{cases} y'(t) = 0.001 \cdot y \cdot (1000 - y) \\ y(0) = 1 \end{cases}$$

Soluzione analitica

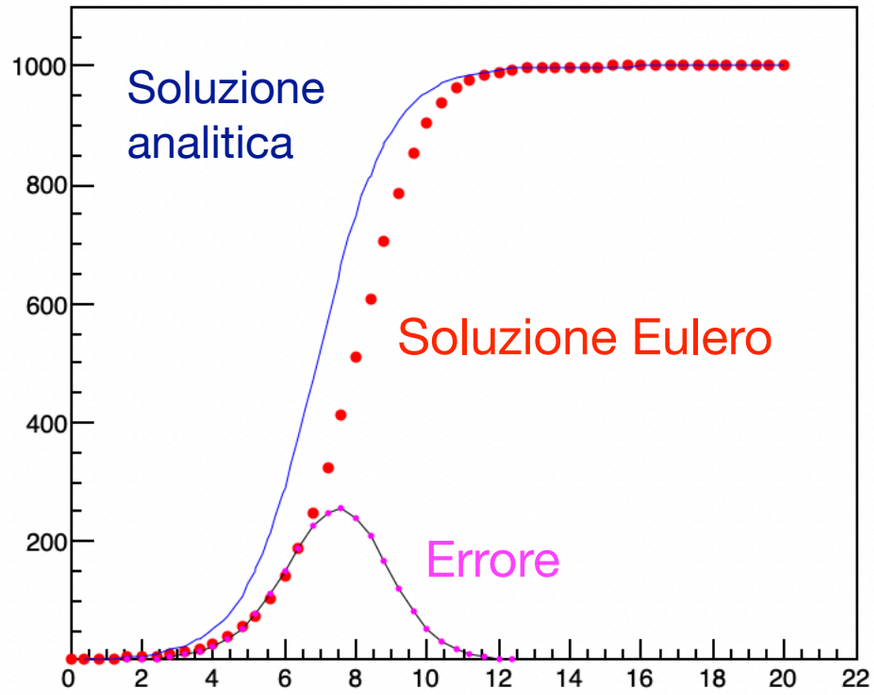
$$y(t) = \frac{1000}{999 \cdot e^{-t} + 1}$$

METODO DI EULERO - ERRORE

$$\begin{cases} y'(t) = 0.001 \cdot y \cdot (1000 - y) \\ y(0) = 1 \end{cases}$$

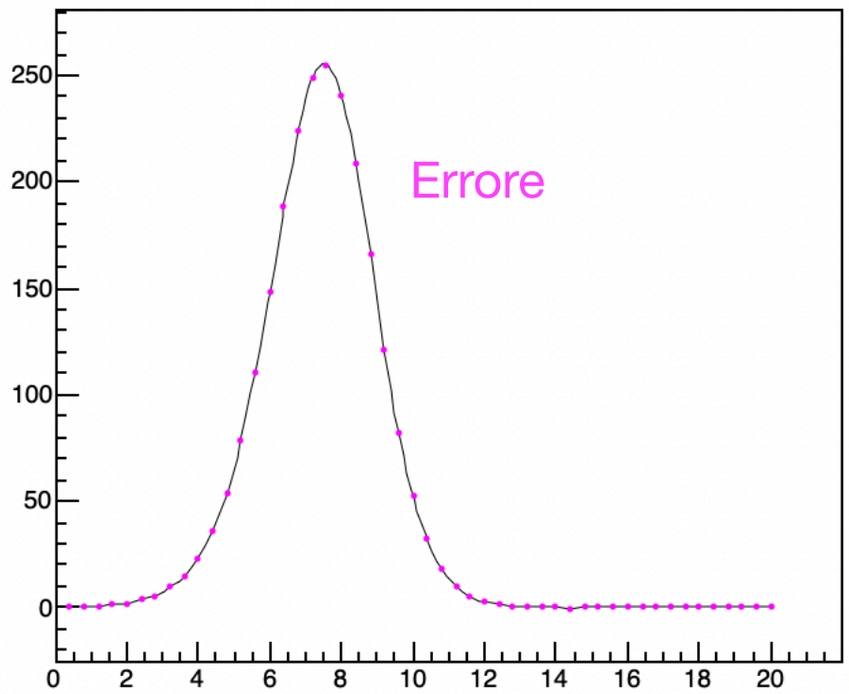
Soluzione analitica

$$y(t) = \frac{1000}{999 \cdot e^{-t} + 1}$$



NPoints = 50

- L'algoritmo di Eulero approssima la funzione con una spezzata. L'errore che si commette nell'approssimazione dipende dalla scelta del passo.
- Piu' piccolo e' il passo minore e' l'errore (teorico)

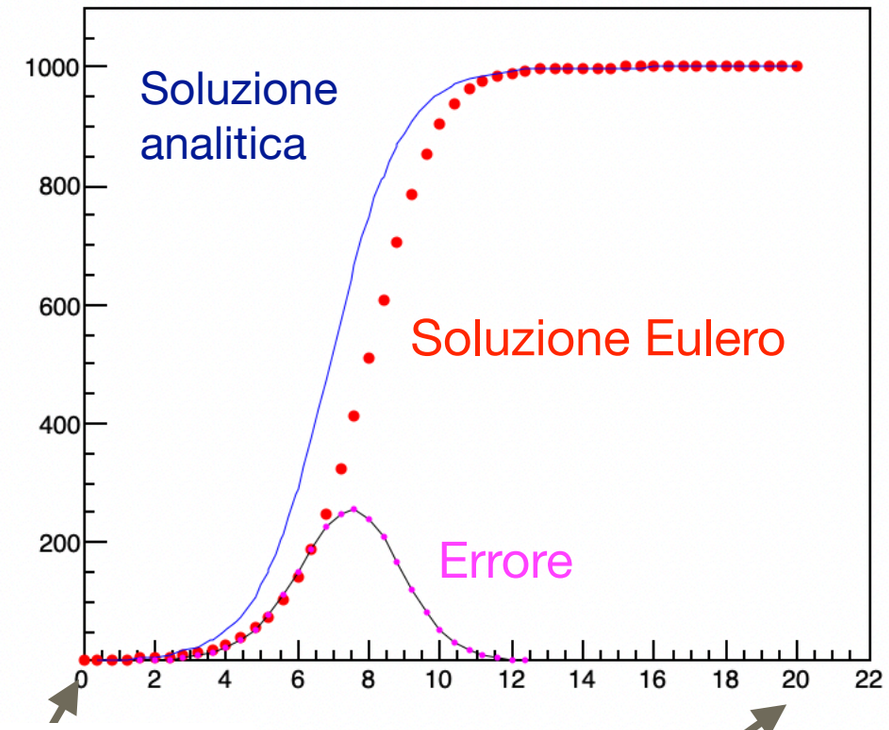


METODO DI EULERO - ERRORE

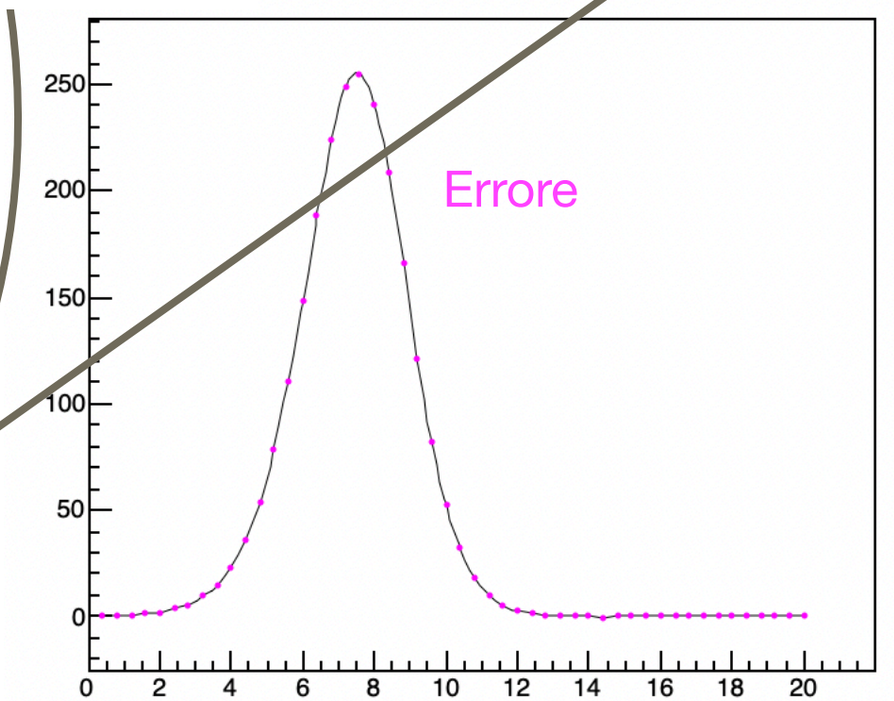
$$\begin{cases} y'(t) = 0.001 \cdot y \cdot (1000 - y) \\ y(0) = 1 \end{cases}$$

Soluzione analitica

$$y(t) = \frac{1000}{999 \cdot e^{-t} + 1}$$



NPoints = 50



- L'algoritmo di Eulero approssima la funzione con una spezzata. L'errore che si commette nell'approssimazione dipende dalla scelta del passo.
- Piu' piccolo e' il passo minore e' l'errore (teorico)

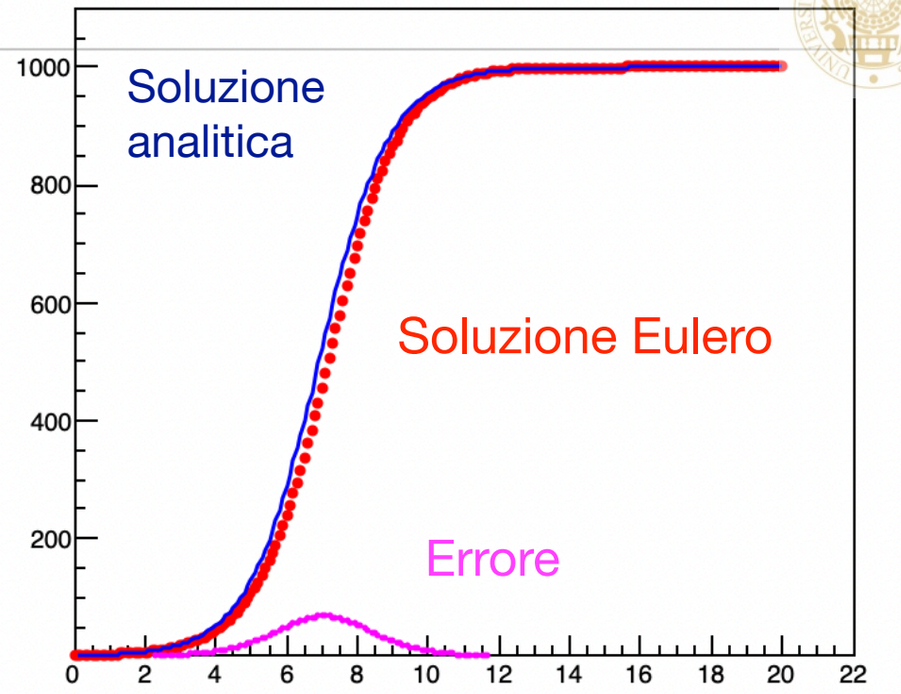
```
root [0] .x EuleroGraphErr1.C(50,0.,20.,1.)
```

METODO DI EULERO - ERRORE

$$\begin{cases} y'(t) = 0.001 \cdot y \cdot (1000 - y) \\ y(0) = 1 \end{cases}$$

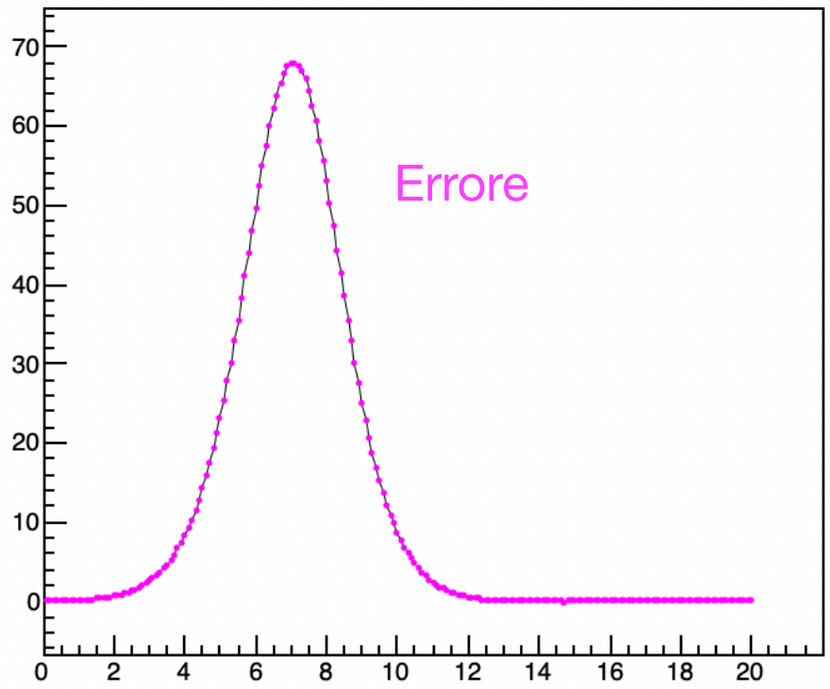
Soluzione analitica

$$y(t) = \frac{1000}{999 \cdot e^{-t} + 1}$$



NPoints = 200

- L'algoritmo di Eulero approssima la funzione con una spezzata. L'errore che si commette nell'approssimazione dipende dalla scelta del passo.
- Piu' piccolo e' il passo minore e' l'errore (teorico)



METODO DI EULERO - ERRORE

- Stimiamo l'errore

Metodo di Eulero $y_{i+1} = y_i + hf(x_i, y_i)$

Sviluppo in serie di Taylor $y(x_{i+1}) = y(x_i + h) = y(x_i) + hf(x_i, y(x_i)) + \frac{1}{2}h^2 y''(\xi)$

Sottraendo membro a membro

$$y(x_{i+1}) - y_{i+1} = y(x_i) - y_i + h(f(x_i, y(x_i)) - f(x_i, y_i)) + \frac{1}{2}h^2 y''(\xi)$$

L'***errore commesso al passo i-esimo*** nella mia approssimazione e_i vale:

$e_i = y(x_i) - y_i$ *Valore della soluzione effettiva (soluzione analitica) - approssimazione di Eulero*

METODO DI EULERO - ERRORE

- Stimiamo l'errore

Metodo di Eulero $y_{i+1} = y_i + hf(x_i, y_i)$

Sviluppo in serie di Taylor $y(x_{i+1}) = y(x_i + h) = y(x_i) + hf(x_i, y(x_i)) + \frac{1}{2}h^2 y''(\xi)$

Sottraendo membro a membro

e_{i+1} $y(x_{i+1}) - y_{i+1} = y(x_i) - y_i + h(f(x_i, y(x_i)) - f(x_i, y_i)) + \frac{1}{2}h^2 y''(\xi)$

L'***errore commesso al passo i-esimo*** nella mia approssimazione e_i vale:

$e_i = y(x_i) - y_i$ *Valore della soluzione effettiva (soluzione analitica) - approssimazione di Eulero*

$$e_{i+1} = e_i + h(f(x_i, y(x_i)) - f(x_i, y_i)) + \frac{1}{2}h^2 y''(\xi)$$

Osserviamo che

$$e_o = 0$$

$$e_1 = \frac{1}{2}h^2 y''(\xi)$$

METODO DI EULERO - ERRORE

- Dal termine successivo a i=1, interviene nell'errore $f(x_i, y(x_i)) - f(x_i, y_i)$

  - = differenza della funzione f(x,y) tra due punti che differiscono solo in y !! Poiché la f è continua e derivabile rispetto alla variabile y, posso applicare il teorema di Lagrange, per cui:

$$f(x_i, y(x_i)) - f(x_i, y_i) = (y(x_i) - y_i)\frac{df}{dy}(x_i, \eta_i) =$$

$$= e_i f_y(x_i, \eta_i)$$

con $\eta_i \in (y(x_i), y_i)$

Il termine e_i è proprio l'errore al passo i-esimo.

Con $f_y(x_i, \eta_i)$ indico la derivata della f rispetto a y calcolata nel punto (x_i, η_i)

L'errore al passo i+1 (slide precedente) diventa:

$$e_{i+1} = e_i + h e_i f_y(x_i, \eta_i) + \frac{1}{2} h^2 y''(\xi)$$

METODO DI EULERO - ERRORE

Poiché la f è continua e derivabile in un intervallo chiuso ***sarà possibile identificare*** in tale intervallo ***un massimo della sua derivata*** *(sia della derivata parziale rispetto a y, che della derivata totale rispetto a x)*.

$$f_y(x_i, \eta_i) < \mu$$

Esistono quindi un μ e un τ tali che:

$$y''(\xi_i) = \frac{df}{dt}(\xi_i, y(\xi_i)) < \tau$$

$$\forall x_i, \eta_i, \xi_i$$

Allora l'errore al passo i+1

$$e_{i+1} = e_i + h e_i f_y(x_i, \eta_i) + \frac{1}{2} h^2 y''(\xi)$$

può essere maggiorato in questo modo

$$e_{i+1} \leq e_i + h e_i \mu + h^2 \tau = e_i (1 + h\mu) + h^2 \tau$$

Per valutare l'errore in funzione di h mi basta applicare la formula precedente in maniera ricorsiva

METODO DI EULERO - ERRORE

$$e_{i+1} \le e_i + h e_i \mu + h^2 \tau = e_i (1 + h\mu) + h^2 \tau$$

$$e_i \le e_{i-1}(1+h\mu) + h^2\tau \le (1+h\mu)(e_{i-2}(1+h\mu) + h^2\tau) + h^2\tau =$$

$$= h^2\tau(1 + (1+h\mu)) + e_{i-2}(1+h\mu)^2 \le h^2\tau(1 + (1+h\mu)) + (1+h\mu)^2(e_{i-3}(1+h\mu) + h^2\tau) =$$

$$= h^2\tau(1 + (1+h\mu) + (1+h\mu)^2) + e_{i-3}(1+h\mu)^3 \le \ldots$$

$$\le h^2\tau(1 + (1+h\mu) + \ldots + (1+h\mu)^{i-1}) + e_0(1+h\mu)^i$$

Ma e_0=0, quindi

B^{i-1} $B^{i-2}A^1$ A^{i-1}

$$e_i \le h^2\tau(1 + (1+h\mu) + \ldots + (1+h\mu)^{i-1})$$

Se si considera lo sviluppo notevole:

$$A^n - B^n = (A-B)(A^{n-1} + A^{n-2}B + \ldots + B^{n-1})$$

Assumendo $B = 1 \quad A = 1 + h\mu$

METODO DI EULERO - ERRORE

$$e_{i+1} \le e_i + h e_i \mu + h^2 \tau = e_i (1 + h\mu) + h^2 \tau$$

$$e_i \le e_{i-1}(1+h\mu) + h^2\tau \le (1+h\mu)(e_{i-2}(1+h\mu) + h^2\tau) + h^2\tau =$$
$$= h^2\tau(1 + (1+h\mu)) + e_{i-2}(1+h\mu)^2 \le h^2\tau(1 + (1+h\mu)) + (1+h\mu)^2(e_{i-3}(1+h\mu) + h^2\tau) =$$
$$= h^2\tau(1 + (1+h\mu) + (1+h\mu)^2) + e_{i-3}(1+h\mu)^3 \le \ldots$$
$$\le h^2\tau(1 + (1+h\mu) + \ldots + (1+h\mu)^{i-1}) + e_0(1+h\mu)^i$$

Ma e_0=0, quindi

B^{i-1} $B^{i-2}A^1$ A^{i-1}

$$e_i \le h^2\tau(1 + (1+h\mu) + \ldots + (1+h\mu)^{i-1})$$

Se si considera lo sviluppo notevole:

$$A^n - B^n = (A-B)(A^{n-1} + A^{n-2}B + \ldots + B^{n-1})$$

Assumendo $B = 1 \quad A = 1 + h\mu$ posso scrivere

A^i B^i

$$(1 + (1+h\mu) + \ldots + (1+h\mu)^{i-1}) = \frac{((1+h\mu)^i - 1)}{1 + h\mu - 1} = \frac{((1+h\mu)^i - 1)}{h\mu}$$

METODO DI EULERO - ERRORE

Sostituendo nella espressione di partenza sull'errore al passi i-esimo

$$e_i \le h^2\tau\left(1 + (1+h\mu) + \ldots + (1+h\mu)^{i-1}\right)$$

$$e_i \le h^2\tau\frac{(1+h\mu)^i - 1}{h\mu} = h\tau\frac{(1+h\mu)^i - 1}{\mu} \le h\tau\frac{(1+h\mu)^i}{\mu}$$

Inoltre, dal momento che $1 + x \le e^x$ si puo' scrivere $1 + h\mu \le e^{h\mu}$

$$(1+h\mu)^i \le e^{ih\mu}$$

Osserviamo che ***ih*** rappresenta ***i step di dimensione h,*** e dal momento che mi muovo all'interno dell'intervallo [a,b], ***ih <= b-a*** <—— ampiezza dell'intervallo

Quindi $(1+h\mu)^i \le e^{(b-a)\mu}$

e infine $e_i \le h\frac{\tau}{\mu}e^{(b-a)\mu}$

EQUAZIONI DIFFERENZIALI

$$e_i \leq h \frac{\tau}{\mu} e^{(b-a)\mu}$$

- L'errore che si commette con l'algoritmo di Eulero è proporzionale ad h.
- Quindi l'algoritmo di Eulero è convergente in h
  - Al diminuire di h l'errore diminuisce linearmente con h.

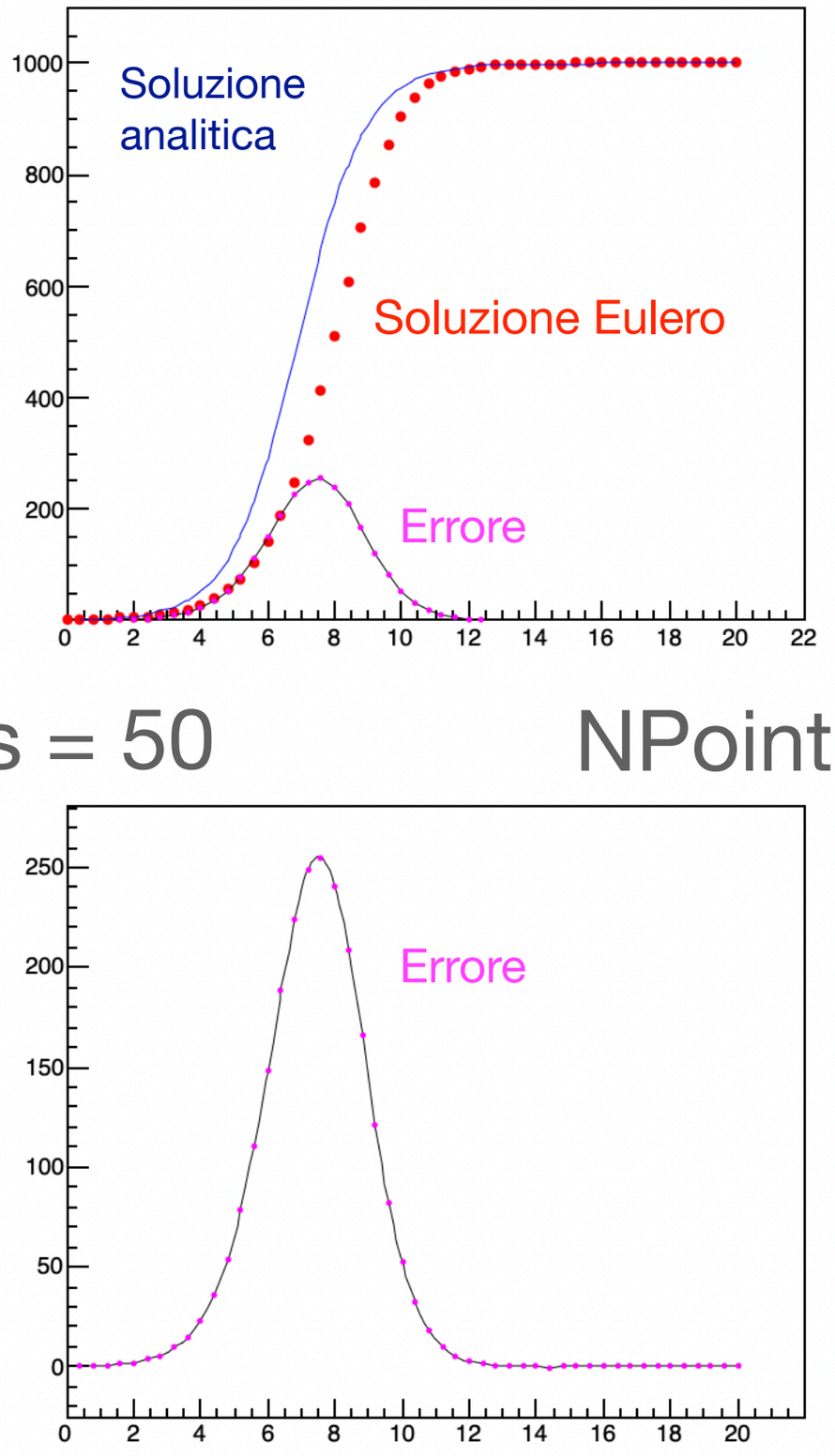
- Un ***algoritmo*** di risoluzione di equazioni differenziali si dice di ***ordine n se il suo errore diminuisce con h come una potenza di n***

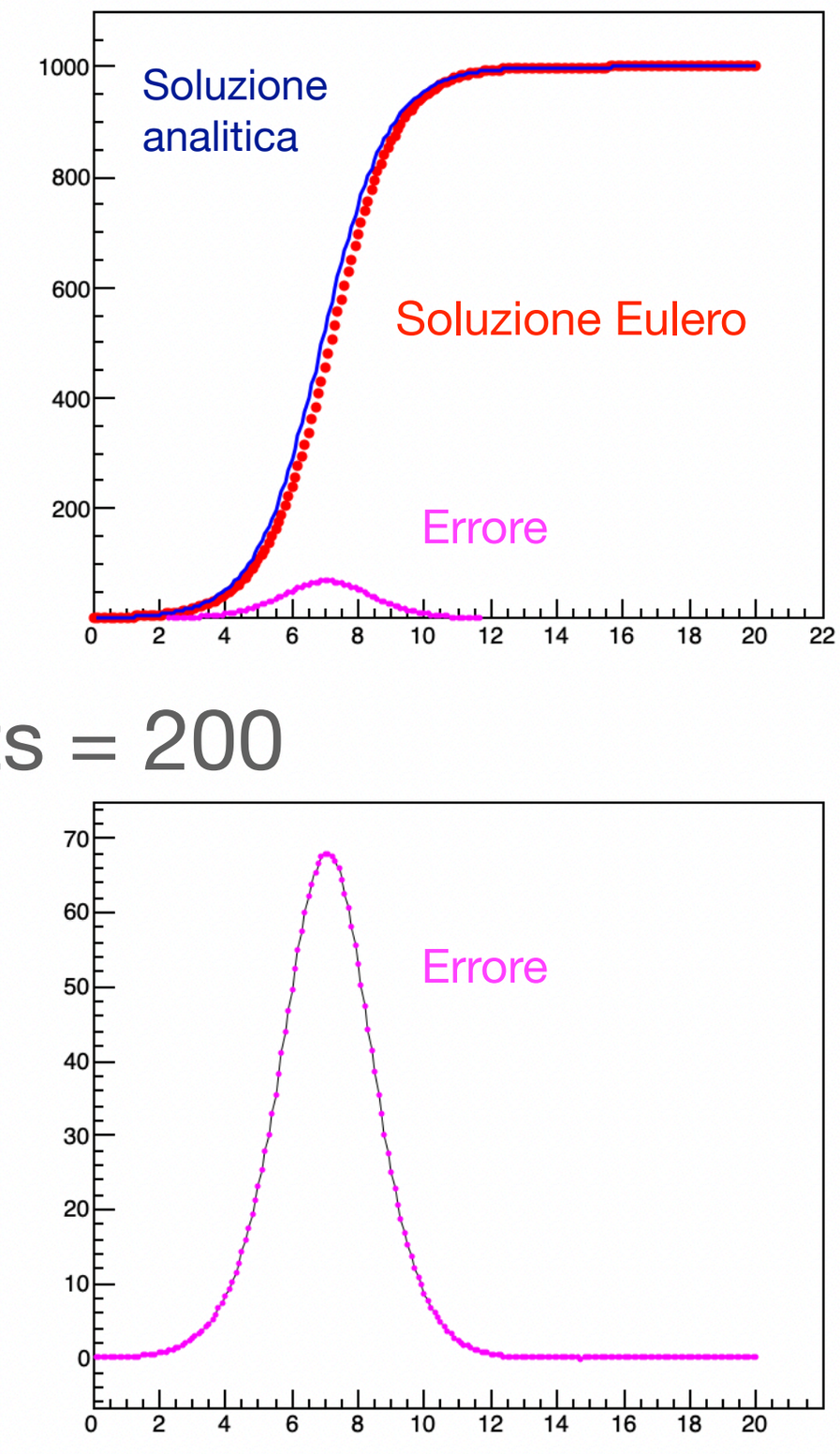
$$e_i \leq G h^n$$

- ***Quindi l'algoritmo di Eulero è un algoritmo del primo ordine.***

METODO DI EULERO - ERRORE

$$\begin{cases} y'(t) = 0.001 \cdot y \cdot (1000 - y) \\ y(0) = 1 \end{cases}$$





NPoints = 50

NPoints = 200

METODO DI EULERO - ERRORE

- passo h = 0.025 errore massimo = 17.2046
- passo h = 0.05 errore massimo = 34.2932
- passo h = 0.1 errore massimo = 68.062
- passo h = 0.2 errore massimo = 133.765
- passo h = 0.4 errore massimo = 255.278
- passo h = 0.8 errore massimo = 453.624

NPoints = 200

NPoints = 50

$$e_i \leq h \frac{\tau}{\mu} e^{(b-a)\mu}$$

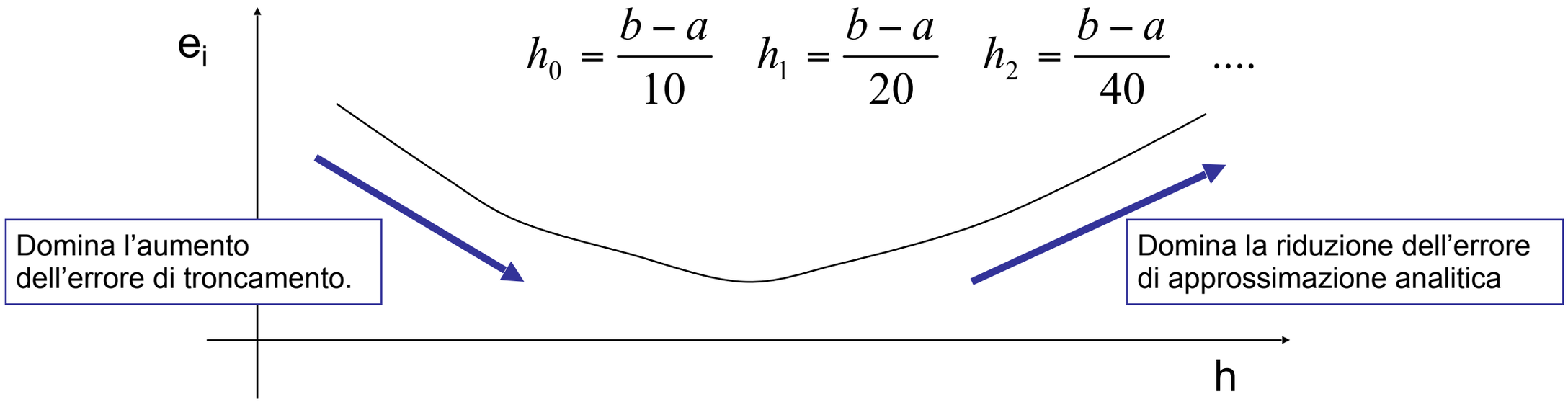
EQUAZIONI DIFFERENZIALI

Sino ad ora non abbiamo preso in considerazione il fatto che oltre all'errore dovuto all'approssimazione analitica della soluzione dell'equazione differenziale in ogni l'algoritmo numerico esiste anche un errore di arrotondamento nei calcoli tipico dell'utilizzo di un calcolatore.

Al diminuire di h aumenta il numero di operazioni che occorre eseguire e quindi aumenta l'errore di arrotondamento dovuto alla precisione della macchina. Supponiamo di dimezzare progressivamente h e di valutare ogni volta quanto vale l'errore al passo i-esimo.

$$h_n = \frac{b-a}{10 \cdot 2^n}$$

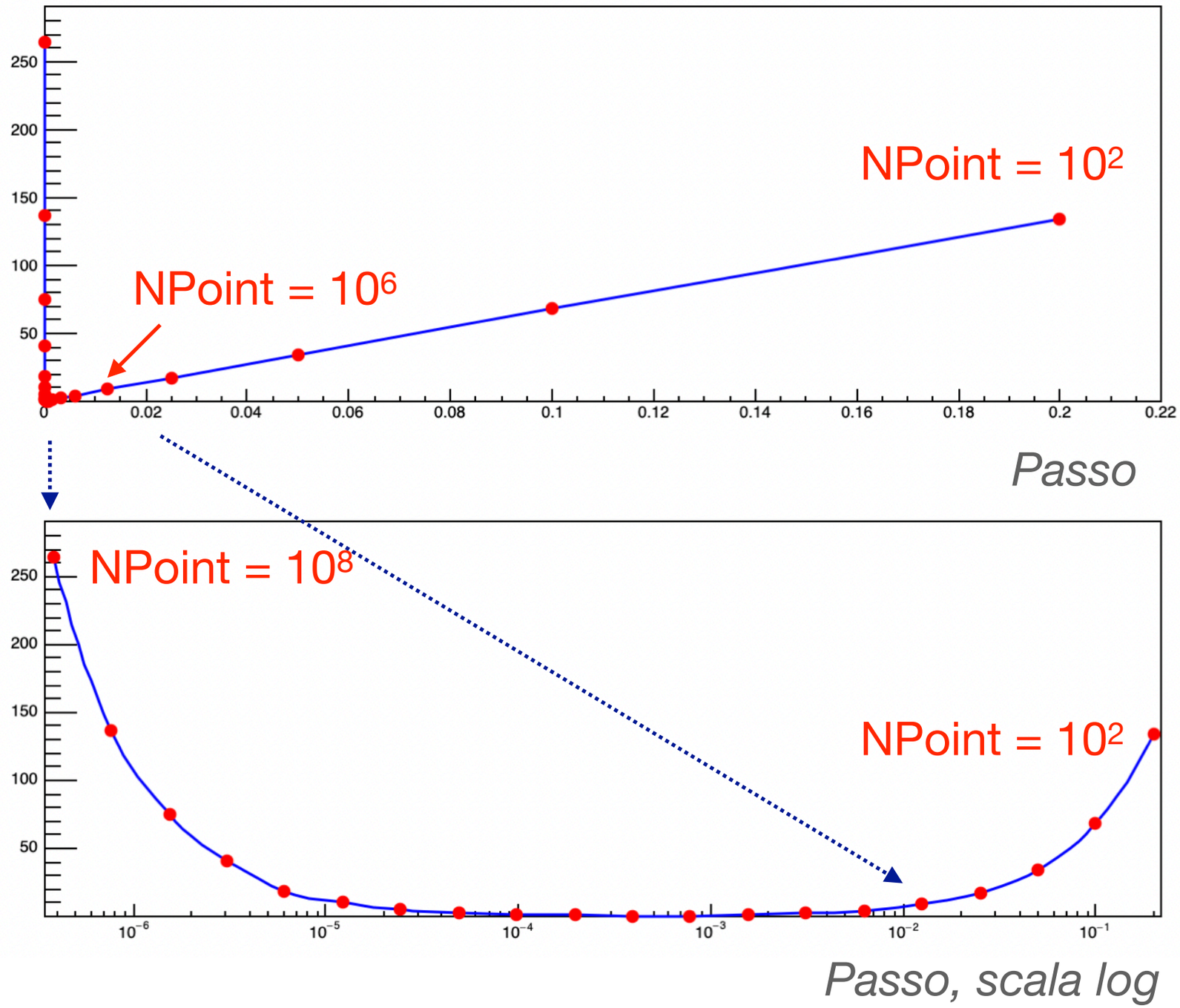
$$h_0 = \frac{b-a}{10} \quad h_1 = \frac{b-a}{20} \quad h_2 = \frac{b-a}{40} \quad ....$$



METODO DI EULERO - ERRORE

```
root [0] .x EuleroGraphErr2.C(0.,20.,1.)
```

$$\begin{cases} y'(t) = 0.001 \cdot y \cdot (1000 - y) \\ y(0) = 1 \end{cases}$$

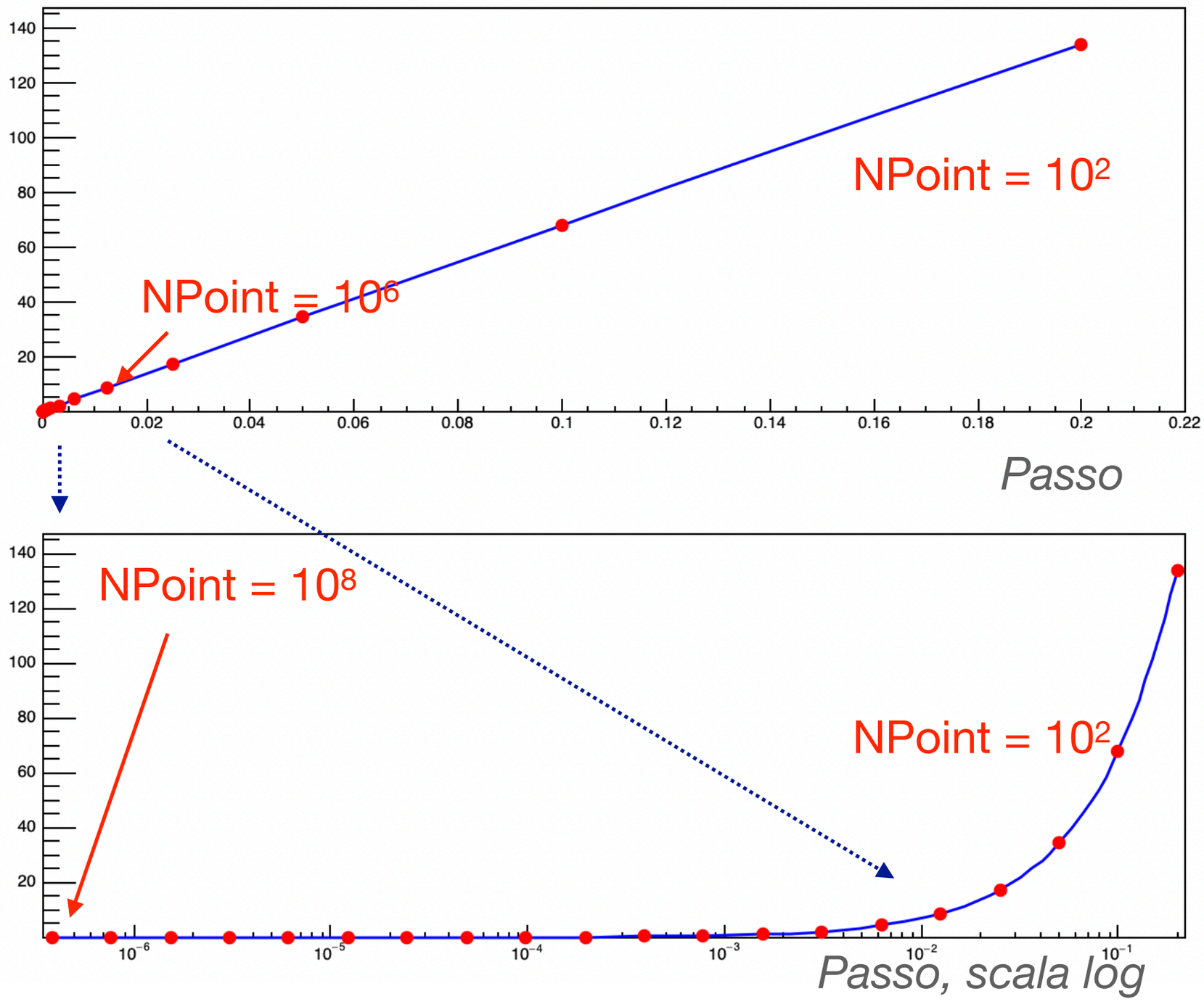


NOTA: calcoli con variabili floating point

METODO DI EULERO - ERRORE

```
root [0] .x EuleroGraphErr2Double.C(0.,20.,1.)
```

$$\begin{cases} y'(t) = 0.001 \cdot y \cdot (1000 - y) \\ y(0) = 1 \end{cases}$$



NOTA: calcoli con variabili in ***doppia precisione***

EQUAZIONI DIFFERENZIALI

Un errore di arrotondamento, una volta generato, si propaga attraverso i passi successivi del calcolo.

Questo conduce al concetto di stabilità numerica di un algoritmo.
Un algoritmo si dice ***numericamente stabile*** se un errore, una volta che è stato generato, non cresce eccessivamente durante il calcolo.

Nel caso dell'algoritmo di Eulero il termine responsabile della stabilità dell'algoritmo è:

$$e_{i+1} = e_i(1 + hf_y(x_i, \eta_i)) + \frac{1}{2}h^2 y''(\xi)$$

$$1 + hf_y(x_i, \eta_i) < 1 + h\mu$$

L'approssimazione sara' numericamente stabile se

$$|1 + h\mu| < 1$$

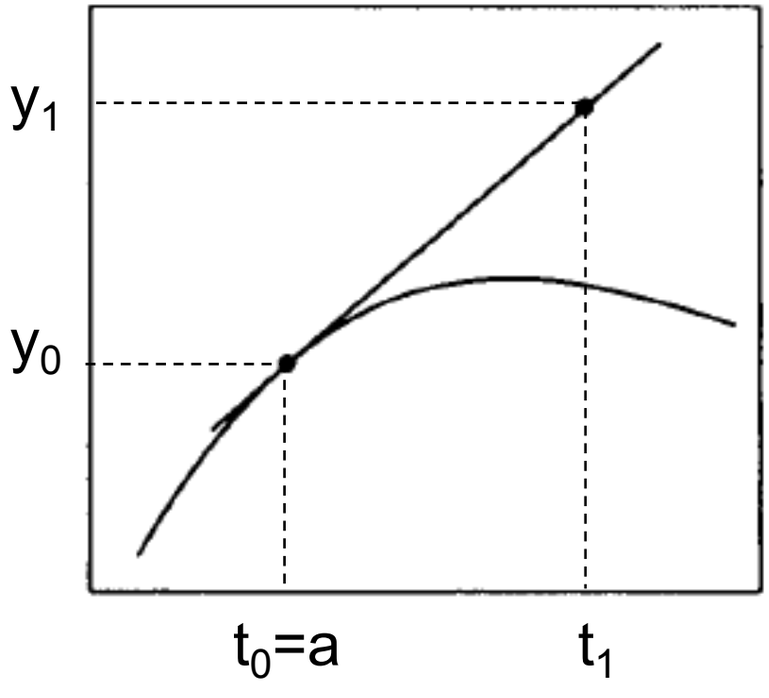
$$-1 < 1 + h\mu < 1$$

cioe' se la derivata della funzione f fatta rispetto ad y e' negativa nell'intervallo in cui si cerca la soluzione

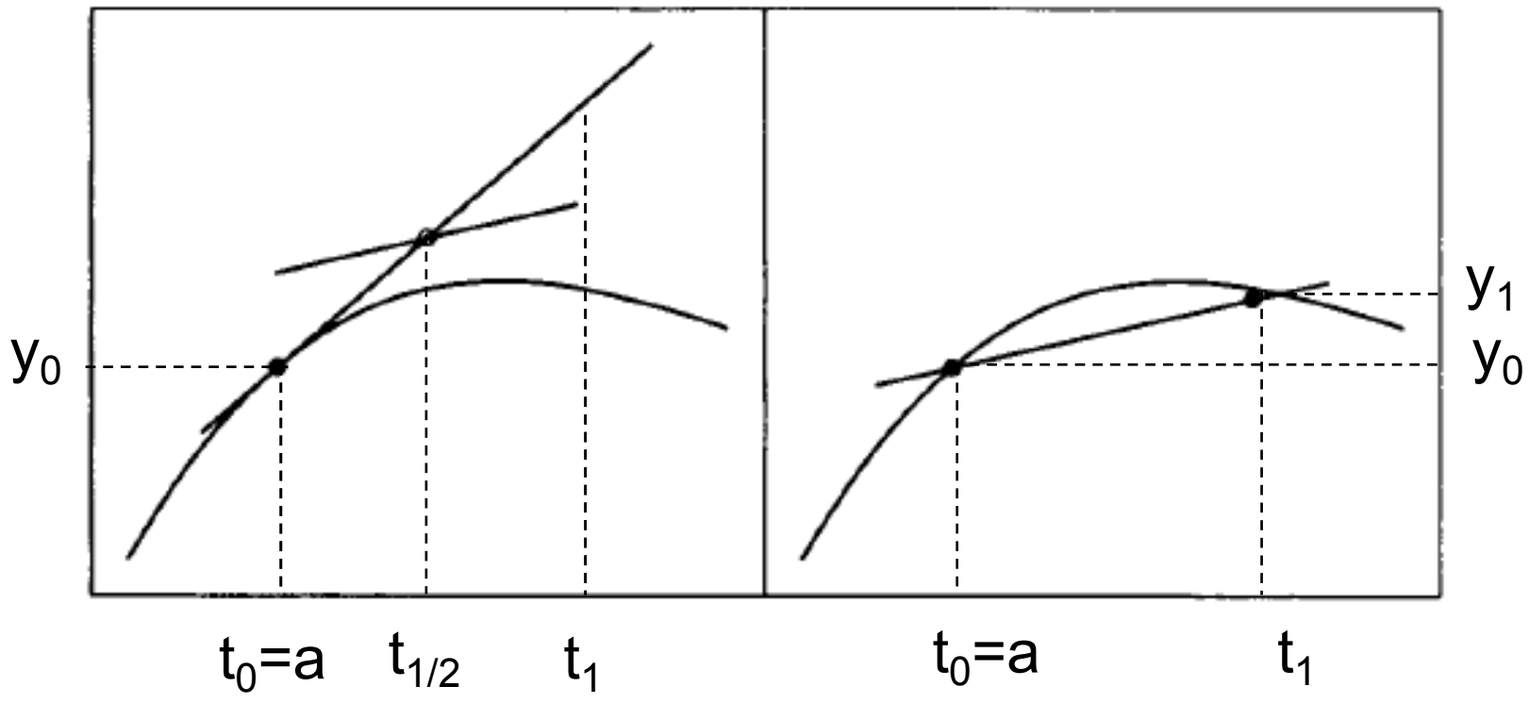
SOLUZIONE NUMERICA DI EQUAZIONI DIFFERENZIALI

Eq del primo ordine
Metodo di Runge-Kutta

DA EULERO A RUNGE-KUTTA



- Un metodo intuitivo per migliorare l'algoritmo di Eulero potrebbe consistere nel considerare non il valore che la funzione f assume all'inizio dell'intervallo (i,i+1) ma nel punto medio.
- Questo equivale ad utilizzare la pendenza della retta tangente alla mia funzione nel centro dell'intervallo e non alla fine



- Per fare questo occorre stimare la y al centro dell'intervallino $\{t_i, t_{i+1}\}$

DA EULERO A RUNGE-KUTTA

L'algoritmo ricorsivo proposto diventa quindi:

$$\begin{cases} t_{i+1} = t_i + h \\ y_{i+1} = y_i + hf(y_{i+\frac{1}{2}}, t_{i+\frac{1}{2}}) \end{cases}$$

Per determinare $y_{i+\frac{1}{2}}$ uso un algoritmo di Eulero con passo h/2

$$y_{i+\frac{1}{2}} = y_i + \frac{h}{2} f(y_i, t_i)$$

Se indico con:

$$k_1 = f(y_i, t_i)$$

$$k_2 = f(y_i + \frac{h}{2} k_1, t_i + \frac{h}{2})$$

Allora il termine successivo di passo h della procedura iterativa diventa:

$$\begin{cases} t_{i+1} = t_i + h \\ y_{i+1} = y_i + hk_2 \end{cases}$$

METODO DI RUNGE-KUTTA

Carl Runge
(1856-1927)

Martin W. Kutta
(1867-1944)

$$\begin{cases} t_{i+1} = t_i + h \\ y_{i+1} = y_i + hk_2 \end{cases} \quad \text{con} \quad \begin{aligned} k_1 &= f(y_i, t_i) \\ k_2 &= f(y_i + \frac{h}{2}k_1, t_i + \frac{h}{2}) \end{aligned}$$

- Questo algoritmo è detto di Runge-Kutta del secondo ordine
- E' possibile dimostrare che l'errore commesso con questo algoritmo va come h^2, mentre l'errore locale (ad un passo) commesso va come h^3. Esattamente come nel caso dell'algoritmo di Eulero l'errore locale andava come h^2 mentre l'errore globale come h.
- L'algoritmo di Eulero può essere considerato un algoritmo di R-K del primo ordine.

Vediamo che l'errore locale va come h^3
Stimiamo y_{i+1} e y_i a partire da $y_{i+1/2}$ fermandoci al secondo ordine

METODO DI RUNGE-KUTTA

$$y_i = y(t_{i+1/2} - \frac{h}{2}) = y_{i+1/2} - \frac{h}{2}y'(t_{i+\frac{1}{2}}) + \left(\frac{h}{2}\right)^2 \frac{1}{2}y''(t_{i+\frac{1}{2}}) - \left(\frac{h}{2}\right)^3 \frac{1}{6}y'''(\xi)$$

$$y_{i+1} = y(t_{i+1/2} + \frac{h}{2}) = y_{i+1/2} + \frac{h}{2}y'(t_{i+\frac{1}{2}}) + \left(\frac{h}{2}\right)^2 \frac{1}{2}y''(t_{i+\frac{1}{2}}) + \left(\frac{h}{2}\right)^3 \frac{1}{6}y'''(\xi)$$

Sottraendo membro a membro le due relazioni si ottiene:

METODO DI RUNGE-KUTTA

$$y_i = y(t_{i+1/2} - \frac{h}{2}) = y_{i+1/2} - \frac{h}{2}y'(t_{i+\frac{1}{2}}) + \left(\frac{h}{2}\right)^2 \frac{1}{2}y''(t_{i+\frac{1}{2}}) - \left(\frac{h}{2}\right)^3 \frac{1}{6}y'''(\xi)$$

$$y_{i+1} = y(t_{i+1/2} + \frac{h}{2}) = y_{i+1/2} + \frac{h}{2}y'(t_{i+\frac{1}{2}}) + \left(\frac{h}{2}\right)^2 \frac{1}{2}y''(t_{i+\frac{1}{2}}) + \left(\frac{h}{2}\right)^3 \frac{1}{6}y'''(\xi)$$

Sottraendo membro a membro le due relazioni si ottiene:

$$y_{i+1} - y_i = hy'(t_{i+\frac{1}{2}}) + \left(\frac{h}{2}\right)^3 \frac{1}{3}y'''(\xi)$$

$$y_{i+1} = y_i + hy'(t_{i+\frac{h}{2}}) + \left(\frac{h}{2}\right)^3 \frac{1}{3}y'''(\xi)$$

$$y_{i+1} = y_i + hf(y_{i+1/2}, t_i + \frac{h}{2}) + \left(\frac{h}{2}\right)^3 \frac{1}{3}y'''(\xi)$$

Trascurando il termine di ordine dal 3 in su si ottiene l'espressione introdotta in precedenza in maniera grafica

L'errore locale è quindi del terzo ordine.

METODO DI RUNGE-KUTTA

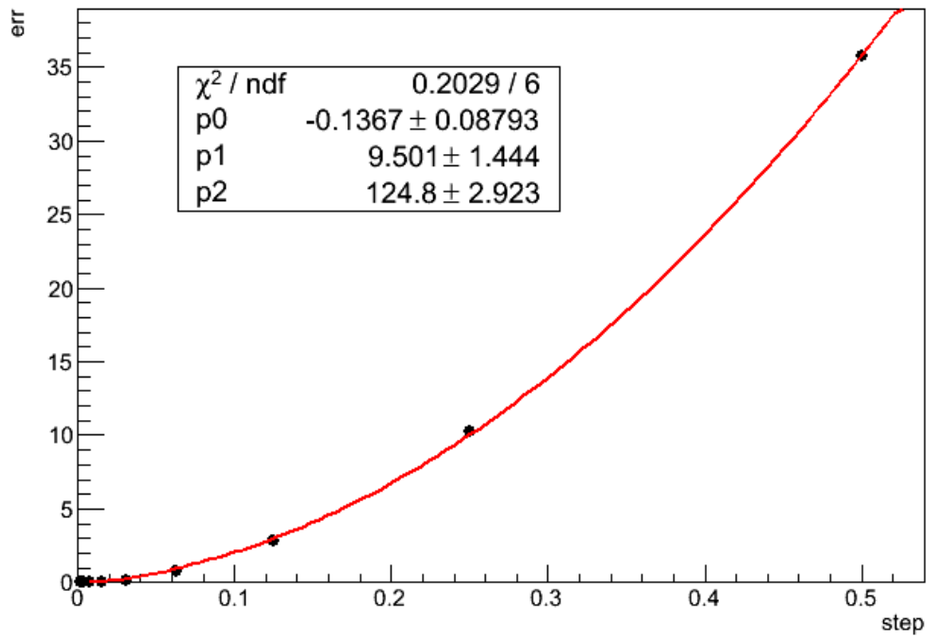
Per verificare in maniera empirica che l'algoritmo proposto è effettivamente del secondo ordine, consideriamo come esempio il problema di Cauchy a soluzione analitica nota

$$\begin{cases} y' = 0.001 \cdot y \cdot (1000 - y) & 0 \le t \le 20 \\ y(0) = 1 \end{cases}$$

$$y(t) = \frac{1000.}{999 \cdot e^{-t} + 1}$$

soluzione

Verifichiamo che l'errore massimo commesso ha un andamento del tipo : $err \approx Gh^2$



Code RKErrore1.C

METODO DI RUNGE-KUTTA DEL QUARTO ORDINE

Utilizzando una metodologia analoga si arriva a scrivere l'algoritmo di Runge-Kutta del **quarto ordine**.

L'incremento di y da i a i+1 e' calcolato usando una sorta di media pesata della pendenza della funzione y calcolata 4 volte:

- all'inizio dell'intervallo (k_1)
- 2 volte al centro dell'intervallo (k_2 e k_3)
- alla fine dell'intervallo (k_4)

$$k_1 = f(y_i, t_i)$$

$$k_2 = f\left(y_i + \frac{h}{2}k_1, t_i + \frac{h}{2}\right)$$

$$k_3 = f\left(y_i + \frac{h}{2}k_2, t_i + \frac{h}{2}\right)$$

$$k_4 = f(y_i + hk_3, t_i + h)$$

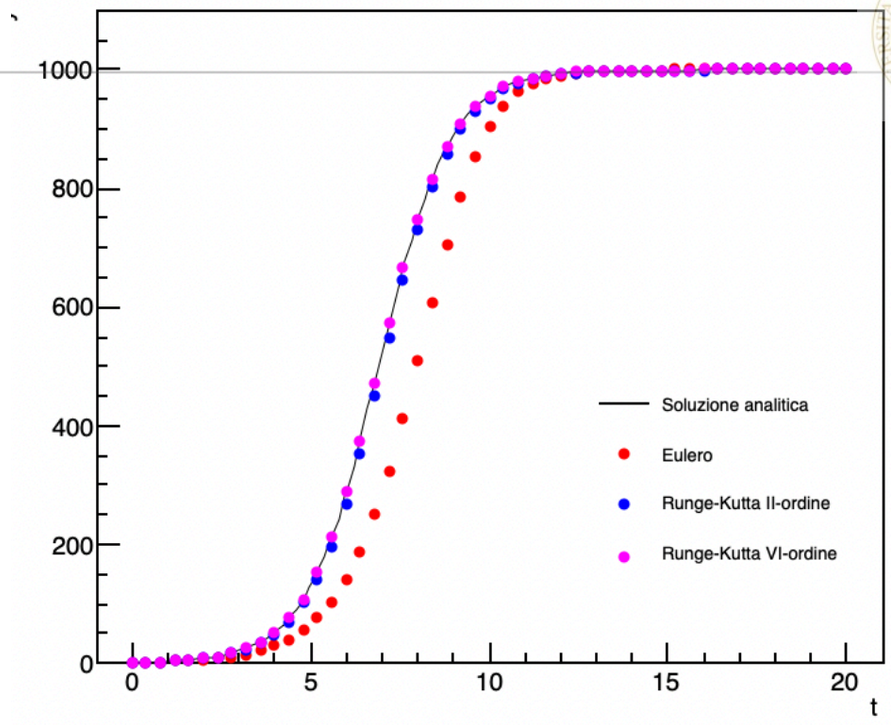
$$y_{i+1} = y_i + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4)$$

Si dimostra che non è possibile aumentare ulteriormente l'ordine dell'algoritmo

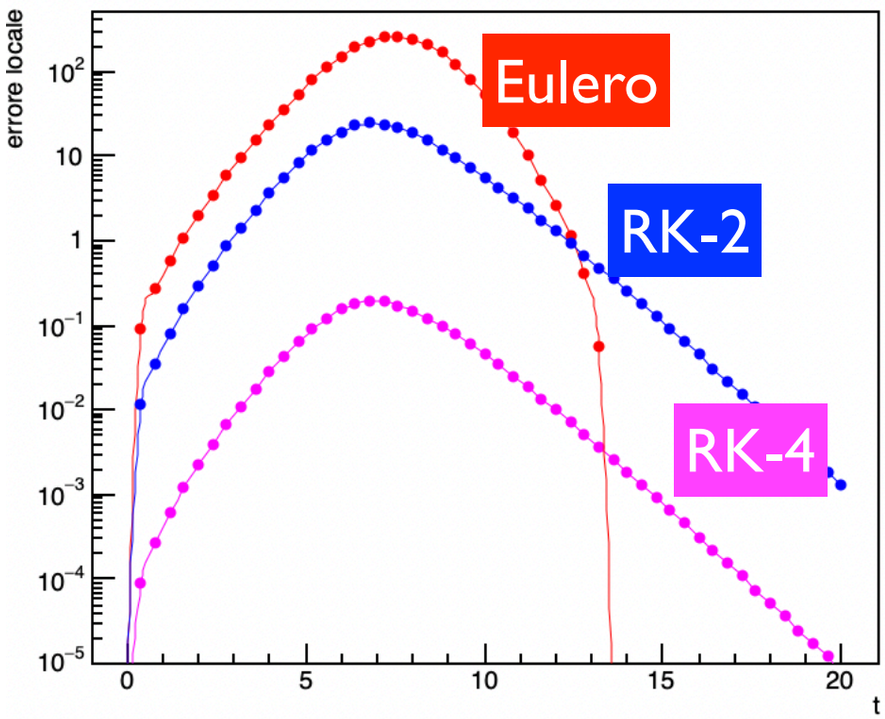
ESEMPI EuleroRK_r5.C

- Studiamo l'***errore locale delle soluzioni numeriche del problema di Cachy,***
  - ***per N = 50m, h=0.4***, in 0<x<20

$$\begin{cases} y'(t) = 0.001 \cdot y \cdot (1000 - y) \\ y(0) = 1 \end{cases}$$



errore massimo
Eulero = 255.278 ;
RK 2o ordine 24.1906 ;
RK 4o ordine 0.189715



ESEMPI

- Studiamo l'errore massimo (globale) vs step (h) delle soluzioni numeriche del problema di Cachy

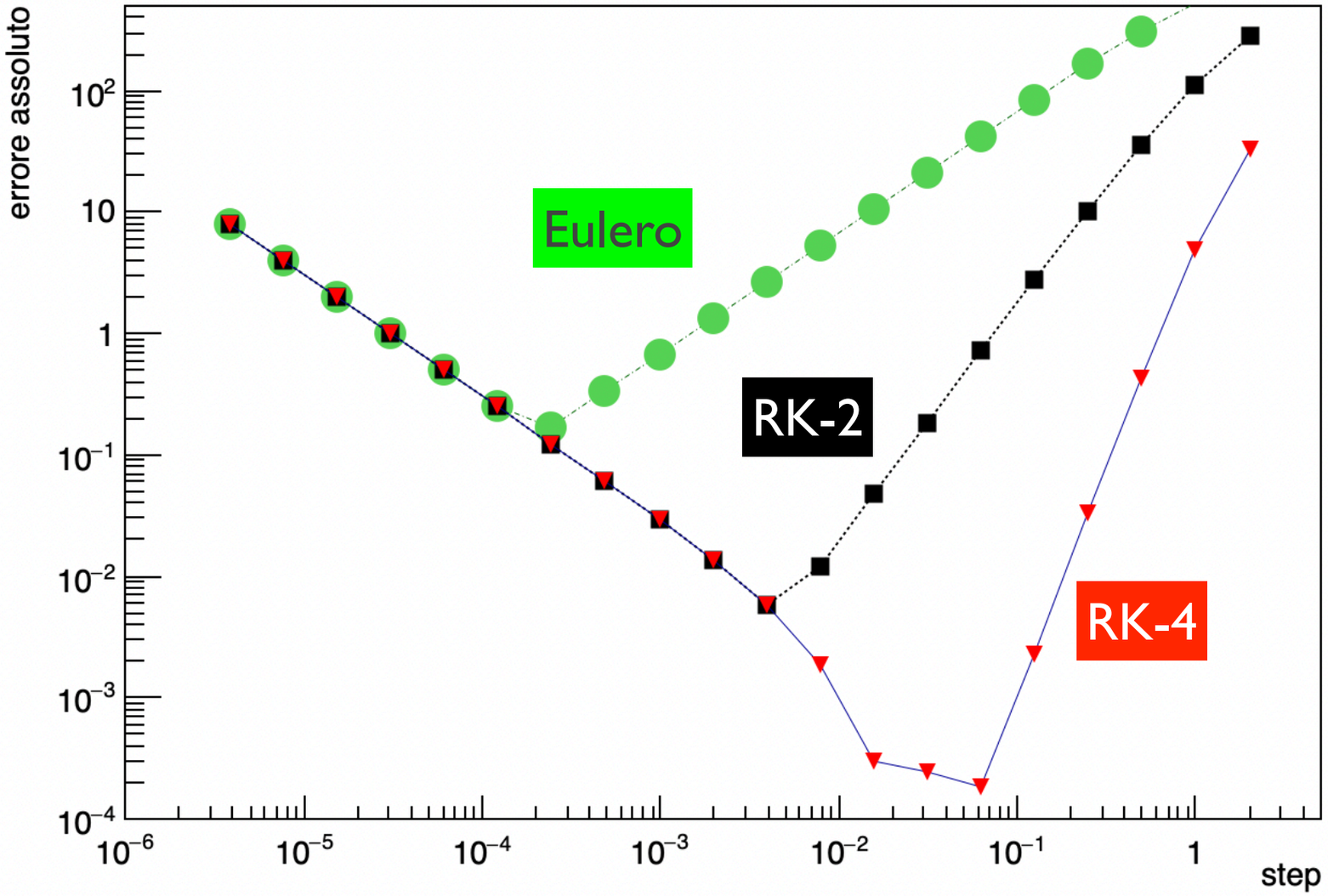
$$\begin{cases} y'(t) = 0.001 \cdot y \cdot (1000 - y) \\ y(0) = 1 \end{cases}$$

- Osserviamo che l'errore di arrotondamento e' uguale in tutti e tre i casi

RKErrore1.C

Scrive "RKErr1.root" con TTree

Errore globale assoluto vs step



Apro il file

Chiedo l'elenco degli oggetti nel file

Stampo su schermo la struttura dei dai nel TTree, i branch

```
[HOME] root -l RKErr1.root
root [0]
Attaching file RKErr1.root as _file0...
(TFile *) 0x11d86f320
root [1] .ls
TFile**         RKErr1.root
 TFile*         RKErr1.root
  KEY: TTree    T;1  Equazioni Differenziali
root [2] T->Print()
******************************************************************************
*Tree    :T         : Equazioni Differenziali                                 *
*Entries :       20 : Total =            3569 bytes  File  Size =       1373 *
*        :          : Tree compression factor =   1.07                       *
******************************************************************************
*Br    0 :NPoint    : NPoint/F                                                *
*Entries :       20 : Total  Size=        639 bytes  File Size  =        127 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.18     *
*                                                                            *
 ............................................................................
*Br    1 :step      : step/F                                                  *
*Entries :       20 : Total  Size=        629 bytes  File Size  =        123 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.20     *
*............................................................................*
*Br    2 :errEU     : errEU/F                                                 *
*Entries :       20 : Total  Size=        634 bytes  File Size  =        149 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.00     *
*............................................................................*
*Br    3 :errRK2    : errRK2/F                                                *
*Entries :       20 : Total  Size=        639 bytes  File Size  =        150 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.00     *
*............................................................................*
*Br    4 :errRK4    : errRK4/F                                                *
*Entries :       20 : Total  Size=        639 bytes  File Size  =        150 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.00     *
*............................................................................*
```

ESEMPI

```
TFile * f = new TFile(“RKErr1.root”);
TH2F* h2 = new TH2F("h2","; step; errore assoluto”,100,1.e-6,5.,100,0.0001,500.);
h2->Draw();
T->Draw(“errEU:step”,"","same");
T->Draw(“errRK2:step","","same")
T->Draw(“errRK4:step","","same")
```

Definisco il box
Disegno il box
Disegno (sul box, “SAME”) l’errore max di Eulero in funzione del passo
Disegno (sul box, “SAME”) l’errore max di RK-2 in funzione del passo
Disegno (sul box, “SAME”) l’errore max di RK-4 in funzione del passo