Υπολογιστικές Μέθοδοι

http://eclass.uoa.gr/courses/PHYS186/

Διδάσκοντες:
Φ. Διάκονος
Δ. Φασουλιώτης

- Επίλυση συνήθων διαφορικών εξισώσεων

Ορισμός του προβλήματος Αρχικών τιμών

Πρόβλημα αρχικών τιμών:

$$\frac{dy}{dt} = f(t,y) \qquad a \le t \le b \qquad y(a) = c$$

Γενίκευση 1:

$$\frac{d\vec{y}}{dt} = f(t,\vec{y}) \qquad \vec{y}(a) = \vec{c}$$

Γενίκευση 2:

$$\frac{dy^{(n)}}{dt} = f(t,y,y',y'',\ldots,y^{(n-1)}) \qquad y^{(i)}(a) = c_i$$

Απαιτήσεις Αριθμητικής Μεθόδου → Ακρίβεια, Ευστάθεια, Απόδοση, Συμβατότητα

Πότε το πρόβλημα είναι επιλύσιμο

Πρόβλημα αρχικών τιμών
καλά δομημένο αν έχει ακριβώς μία λύση

✓f(t,y) συνεχής [α,b]

✓Συνθήκη Lipschitz $\left|\dfrac{\partial f(t,y)}{\partial y}\right| \le L$

π.χ.

$$y' = y - t^2 + 1 \quad \Rightarrow \quad \left|\frac{\partial f(t,y)}{\partial y}\right| = 1 \qquad \surd$$

$$y' = t^2 + \frac{2}{t} y \quad \Rightarrow \quad \left|\frac{\partial f(t,y)}{\partial y}\right| = \frac{2}{t} \quad \begin{cases} 1 \le t \le 2, & \surd \\ 0 \le t \le 2, & \mathbf{x} \end{cases}$$

Μέθοδος Euler

Γενικά:

Δημιουργώ διακριτό πλέγμα: $\{t_0 \equiv \alpha\ ,\ t_1, t_2, \ldots,\ t_N \equiv b\}$

$\mathbf{t_i = \alpha + i\tau}$ για $i=0,1,2,\ldots,N$ με βήμα $\boldsymbol{\tau = (b-\alpha)/N}$

Μέθοδος Euler

Σφάλμα αποκοπής

$$y(t_{i+1}) = y(t_i + \tau) = y(t_i) + \tau y'(t_i) + \frac{\tau^2}{2} y''(\xi) + \ldots$$

$$y(t_i) + \tau f(t_i, y(t_i))$$

Εξίσωση διαφορών

$$w_0 = c$$

$$w_{i+1} = w_i + \tau f(t_i, w_i)$$

Παρατηρήσεις:

- **Ακρίβεια πρώτης τάξης ως προς το χρόνο**
- **Ισοδύναμο με: $dy/dt \cong \delta y/\delta t = [y(t_{i+1}) - y(t_i)]/\delta t$ (forward derivative)**

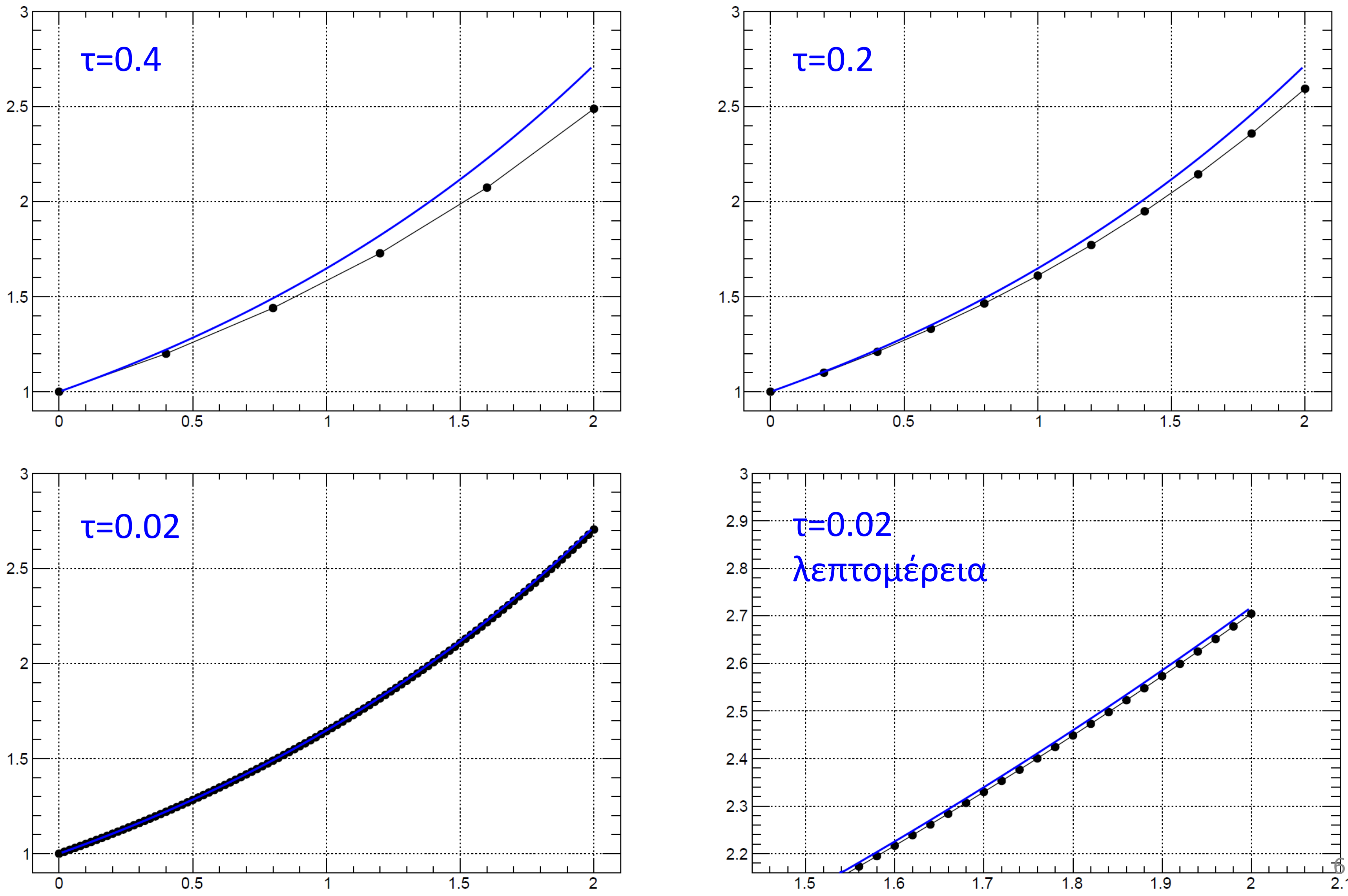
Μέθοδος Euler

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
using namespace std;

double myfunc(double y, double t)
{
  double value=0.5*y;
  return value;
}

void euler(double h)
{
double a=0;
double b=2.;
double w=1.;
double t=a;
cout<< t <<" "<< w <<endl;
int n=(b-a)/h;
for(int i=1;i<=n;i++)
    {
        w=w+h*myfunc(w,t);
        t+=h;
        cout<< t <<" "<< w <<endl;
    }
}
```

```
root [0] .x euler.c(0.1)
0 1
0.1 1.05
0.2 1.1025
0.3 1.15763
0.4 1.21551
0.5 1.27628
0.6 1.3401
0.7 1.4071
0.8 1.47746
0.9 1.55133
1 1.62889
1.1 1.71034
1.2 1.79586
1.3 1.88565
1.4 1.97993
1.5 2.07893
1.6 2.18287
1.7 2.29202
1.8 2.40662
1.9 2.52695
2 2.6533
root [1]
```

Μέθοδος Euler

$$y' = 0.5y, \quad 0 \le t \le 2, \quad y(0) = 1.$$



Μέθοδος Euler- Σφάλμα αποκοπής

```
root [0] .x euler.c(0.5)
0 1
t=0.50 w=1.25000 err=0.03403
t=1.00 w=1.56250 err=0.08622
t=1.50 w=1.95313 err=0.16388
t=2.00 w=2.44141 err=0.27688
```

```
root [1] .x euler.c(0.1)
0 1
t=0.10 w=1.05000 err=0.00127
t=0.20 w=1.10250 err=0.00267
t=0.30 w=1.15763 err=0.00421
t=0.40 w=1.21551 err=0.00590
t=0.50 w=1.27628 err=0.00774
t=0.60 w=1.34010 err=0.00976
t=0.70 w=1.40710 err=0.01197
t=0.80 w=1.47746 err=0.01437
t=0.90 w=1.55133 err=0.01698
t=1.00 w=1.62889 err=0.01983
t=1.10 w=1.71034 err=0.02291
t=1.20 w=1.79586 err=0.02626
t=1.30 w=1.88565 err=0.02989
t=1.40 w=1.97993 err=0.03382
t=1.50 w=2.07893 err=0.03807
t=1.60 w=2.18287 err=0.04267
t=1.70 w=2.29202 err=0.04763
t=1.80 w=2.40662 err=0.05298
t=1.90 w=2.52695 err=0.05876
t=2.00 w=2.65330 err=0.06498
```

Αποδεικνύεται:

$$\left| y(t_i) - w_i \right| \leq \frac{\tau M}{2L}\left[e^{L(t_i - a)} - 1 \right]$$

με $\left| y''(t) \right| \leq M$

Παρατηρήσεις:

1) Γραμμική εξάρτηση σφάλματος από τ

2) Η απόκλιση μεγαλώνει όσο αυξάνει ο χρόνος t

Επίσης:

$$y''(t) = \frac{dy'}{dt} = \frac{df(t,y)}{dt} = \frac{\partial f(t,y)}{\partial t} + \frac{\partial f(t,y)}{\partial y} f(t,y)$$

Έστω:

$w_0 = c + \varepsilon_0$

$w_{i+1}=w_i+f(t_i,w_i) + \varepsilon_\iota$ με ε_ι το σφάλμα στρογγυλοποίησης

Τότε με $\varepsilon_\iota \leq \varepsilon$ προκύπτει:

$$\left|y(t_i) - w_i\right| \leq \frac{1}{L}\left(\frac{\tau M}{2} + \frac{\varepsilon}{\tau}\right)\left[e^{L(t_i - a)} - 1\right] + \varepsilon_0 e^{L(t_i - a)}$$

$$\Rightarrow \quad \lim_{\tau \to 0}\left(\frac{\tau M}{2} + \frac{\varepsilon}{\tau}\right) = \infty$$

Βέλτιστο $\boxed{\tau = \sqrt{\dfrac{2\varepsilon}{M}}}$

Μέθοδοι Taylor ανώτερης τάξης

$$y(t_{i+1}) = y(t_i) + \tau y'(t_i) + \frac{\tau^2}{2} y''(t_i) + \ldots + \frac{\tau^n}{n!} y^{(n)}(t_i) + \frac{\tau^{n+1}}{(n+1)!} y^{(n+1)}(\xi_i)$$

όπου

$$\begin{aligned} y'(t) &= f\big(t, y(t)\big) \\ y''(t) &= f'\big(t, y(t)\big) \\ y'''(t) &= f''\big(t, y(t)\big) \\ &\ldots \\ y^{(n)}(t) &= f^{(n-1)}\big(t, y(t)\big) \end{aligned}$$

$$\boxed{\begin{aligned} w_0 &= c \\ w_{i+1} &= w_i + \tau \cdot T^{(n)}(t_i, w_i) \end{aligned}}$$

όπου

$$\boxed{T^{(n)}(t_i, w_i) = f(t_i, w_i) + \frac{\tau}{2} f'(t_i, w_i) + \ldots + \frac{\tau^{n-1}}{n!} f^{(n-1)}(t_i, w_i)}$$

Μέθοδοι Taylor

✓ Μικρό τοπικό σφάλμα αποκοπής

x Αναλυτικοί υπολογισμοί

$$
\begin{aligned}
T^{(2)}(t,y) &= f(t,y)+\frac{\tau}{2}f'(t,y)\\
&= f(t,y)+\frac{\tau}{2}\left[\frac{\partial f(t,y)}{\partial t}+\frac{\partial f(t,y)}{\partial y}y'\right]\\
&= f(t,y)+\frac{\tau}{2}\frac{\partial f(t,y)}{\partial t}+\frac{\tau}{2}f(t,y)\frac{\partial f(t,y)}{\partial y}
\end{aligned}
$$

$$
\begin{aligned}
T^{(2)}(t,y) &= f(t,y)+\frac{\tau}{2}f'(t,y) \\
&= f(t,y)+\frac{\tau}{2}\left[\frac{\partial f(t,y)}{\partial t}+\frac{\partial f(t,y)}{\partial y}y'\right] \\
&= f(t,y)+\frac{\tau}{2}\frac{\partial f(t,y)}{\partial t}+\frac{\tau}{2}f(t,y)\frac{\partial f(t,y)}{\partial y}
\end{aligned}
$$

Μοιάζει 2-διάστατο ανάπτυγμα Taylor

$$
f(t+a,y+b)=f(t,y)+a\frac{\partial f}{\partial t}+b\frac{\partial f}{\partial y}
$$

$$
\begin{aligned}
T^{(2)}(t,y) &= f(t,y)+\frac{\tau}{2}f'(t,y) \\
&= f(t,y)+\frac{\tau}{2}\left[\frac{\partial f(t,y)}{\partial t}+\frac{\partial f(t,y)}{\partial y}y'\right] \\
&= f(t,y)+\frac{\tau}{2}\frac{\partial f(t,y)}{\partial t}+\frac{\tau}{2}f(t,y)\frac{\partial f(t,y)}{\partial y}
\end{aligned}
$$

Μοιάζει 2-διάστατο ανάπτυγμα Taylor

$$
f(t+a,y+b)=f(t,y)+a\frac{\partial f}{\partial t}+b\frac{\partial f}{\partial y}
$$

όπου

α=τ/2

b=τ/2 f(t,y)

$$
\begin{aligned}
T^{(2)}(t,y) &= f(t,y)+\frac{\tau}{2}f'(t,y)\\
&= f(t,y)+\frac{\tau}{2}\left[\frac{\partial f(t,y)}{\partial t}+\frac{\partial f(t,y)}{\partial y}y'\right]\\
&= f(t,y)+\frac{\tau}{2}\frac{\partial f(t,y)}{\partial t}+\frac{\tau}{2}f(t,y)\frac{\partial f(t,y)}{\partial y}
\end{aligned}
$$

Μοιάζει 2-διάστατο ανάπτυγμα Taylor

$$
f(t+a,y+b)=f(t,y)+a\frac{\partial f}{\partial t}+b\frac{\partial f}{\partial y}
$$

όπου

α=τ/2

b=τ/2 f(t,y)

$$
\boxed{\begin{aligned}
w_0 &= c\\
w_{i+1} &= w_i+\tau\cdot f\left(t_i+\frac{\tau}{2},\ \ w_i+\frac{\tau}{2}\cdot f(t_i,w_i)\right)
\end{aligned}}
$$

Μέθοδος Midpoint ή Runge Kutta 2ης τάξης

Μέθοδοι Runge-Kutta 3ης τάξης

$$T^{(3)}(t_i, w_i) = f(t_i, w_i) + \frac{\tau}{2} f'(t_i, w_i) + \frac{\tau^2}{2 \cdot 3} f''(t_i, w_i)$$

Δυστυχώς από το ανάπτυγμα της μεθόδου Taylor 3ης τάξης, ο όρος

$$\frac{\tau^2}{6}\left[\frac{\partial f}{\partial y}(t, y)\right]^2 f(t, y))$$

δεν μπορεί να απαλειφθεί. Μπορούμε όμως να κατασκευάσουμε κάποιες μεθόδους Runge-Kutta, χρησιμοποιώντας 3 υπολογισμούς της ***f(t,y)*** οι οποίες όμως δεν είναι ισοδύναμες με Taylor 3ης τάξης. Η πιο σημαντική είναι η λεγόμενη **Modified Euler**

$$\boxed{\begin{aligned} w_0 &= c \\ w_{i+1} &= w_i + \frac{\tau}{2} \cdot \left[f(t_i, w_i) + f\left(t_{i+1}, \quad w_i + \tau \cdot f(t_i, w_i)\right)\right] \end{aligned}}$$

Η χρυσή μέθοδος για όμως βασίζεται στην Taylor 4ης τάξης

$$T^{(4)}(t_i, w_i) \quad = f(t_i, w_i) + \frac{\tau}{2} f'(t_i, w_i) + \frac{\tau^2}{2 \cdot 3} f''(t_i, w_i) + \frac{\tau^3}{2 \cdot 3 \cdot 4} f'''(t_i, w_i)$$

και είναι η ακόλουθη:

Runge Kutta 4ης τάξης

$$\begin{aligned} w_0 &= c \\ w_{i+1} &= w_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned}$$

$$\begin{aligned} k_1 &= \tau\ f(t_i, w_i) \\ k_2 &= \tau\ f(t_i + \tau/2, w_i + k_1/2) \\ k_3 &= \tau\ f(t_i + \tau/2, w_i + k_2/2) \\ k_4 &= \tau\ f(t_i + \tau, w_i + k_3) \end{aligned}$$

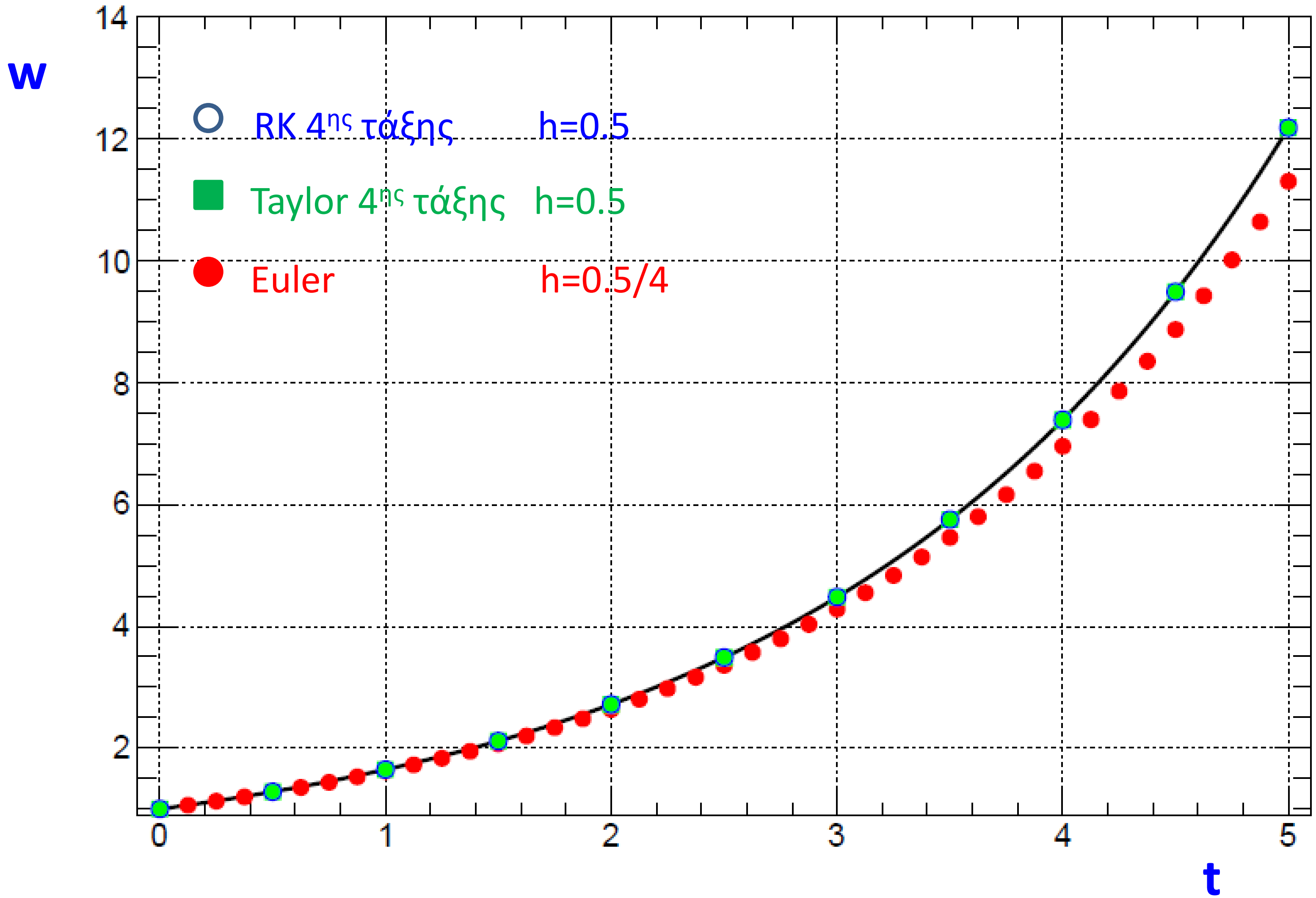
- 4 υπολογισμοί f(t,y)
- Ακρίβεια ≈ τ^4

Runge Kutta 4ης τάξης

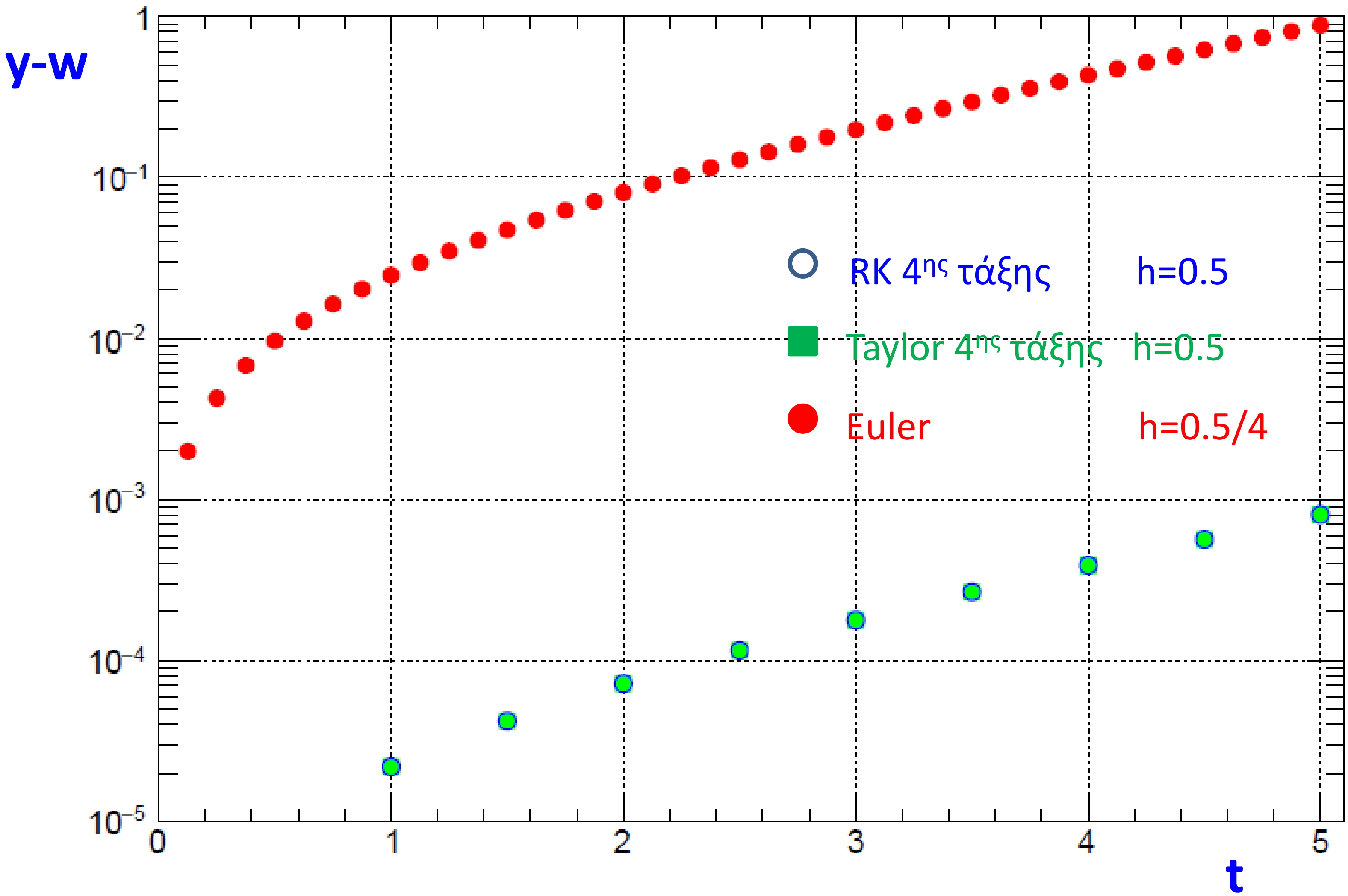
```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <TH1F.h>
#include <TF1.h>
double myf(double t, double y)
{
  double value=0.5*y; // f(t,y)
  return value;
}
void rk4()
{
   double wrk4[100];  double t[100];
// Αρχικοποίηση
   double a=0; double b=5;
   double h=0.5; int N=(b-a)/h;
   t[0]=a;  wrk4[0]=1.;
// Εκτέλεση αλγορίθμου
   for(int i=0; i<N; i++)
      {
         double k1=h*myf(t[i],wrk4[i]);
         double k2=h*myf(t[i]+h/2, wrk4[i]+k1/2);
         double k3=h*myf(t[i]+h/2, wrk4[i]+k2/2);
         double k4=h*myf(t[i]+h, wrk4[i]+k3);
         wrk4[i+1]= wrk4[i]+(k1+2*k2+2*k3+k4)/6;
         t[i+1]=t[i]+h;
      }
}
```

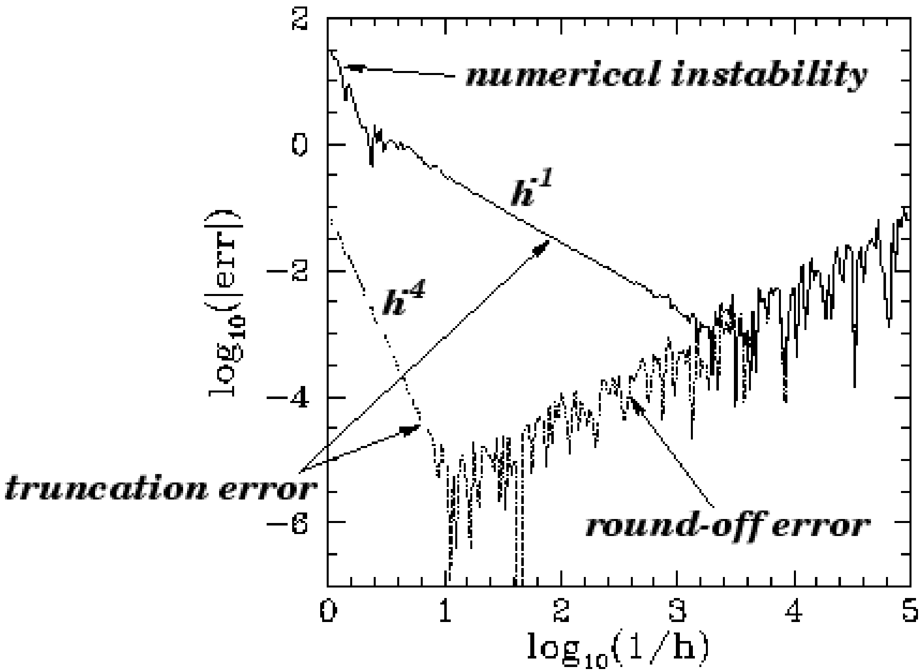
Μέθοδοι Runge-Kutta 4ης τάξης – Παράδειγμα 1

$$y' = \frac{1}{2} y, \quad 0 \le t \le 5, \quad y(0) = 1.$$

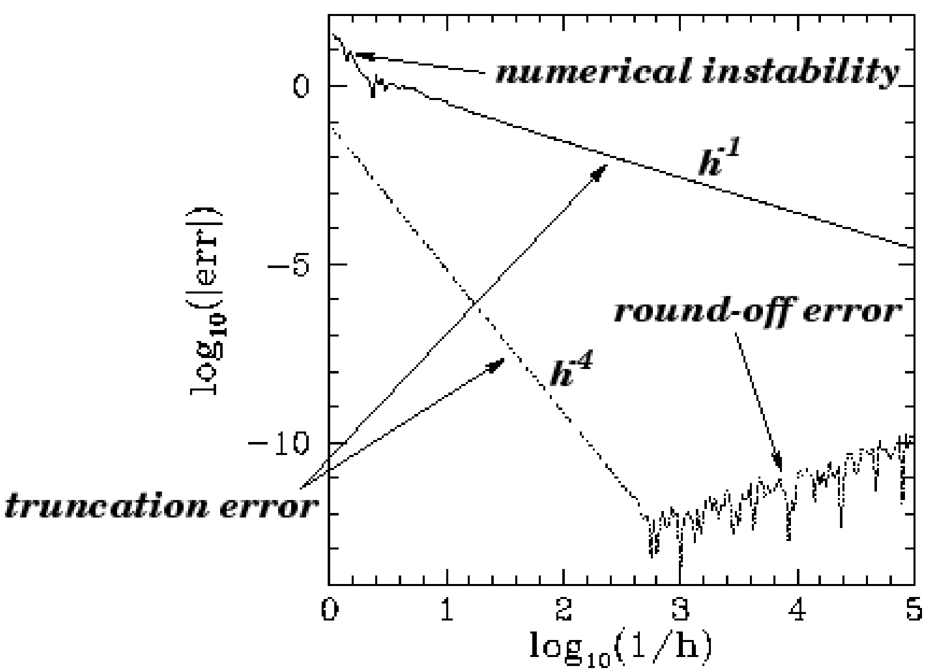


$$y' = \frac{1}{2}y, \quad 0 \le t \le 5, \quad y(0) = 1.$$





Υπολογισμός Single Precision



Υπολογισμός Double Precision

Μέθοδοι Runge-Kutta – Αβεβαιότητα

✓ Με τις μεθόδους Runge-Kutta δεν χρειάζεται να κάνουμε αναλυτικούς υπολογισμούς για την επίλυση μιας ΣΔΕ

❑ Πως θα γνωρίζουμε όμως το σφάλμα των υπολογισμών μας χωρίς αναλυτικές πράξεις;

Μία διέξοδος θα ήταν (όπως και στην περίπτωση της αριθμητικής ολοκλήρωσης) να κάνουμε τον υπολογισμό με την ίδια μέθοδο αλλά διαφορετικό βήμα, εφόσον γνωρίζουμε την εξάρτηση του τοπικού σφάλματος από το βήμα $\boldsymbol{\tau}$

Μέθοδοι Runge-Kutta – Αβεβαιότητα

✓ Με τις μεθόδους Runge-Kutta δεν χρειάζεται να κάνουμε αναλυτικούς υπολογισμούς για την επίλυση μιας ΣΔΕ

❑ Πως θα γνωρίζουμε όμως το σφάλμα των υπολογισμών μας χωρίς αναλυτικές πράξεις;

Μία δεύτερη προσέγγιση θα αποτελούσε η χρήση δύο μεθόδων διαφορετικής τάξης ακρίβειας …

Έστω ότι εφαρμόζουμε δύο διαφορετικής τάξης μεθόδους για την επίλυση του ίδιου προβλήματος

$$y(t_{i+1}) = y(t_i) + \tau\Phi(t_i, y(t_i), \tau) + \propto (\tau^{n+1})$$

$$y(t_{i+1}) = y(t_i) + \tau\widetilde{\Phi}(t_i, y(t_i), \tau) + \propto (\tau^{n+k})$$

με τοπικό σφάλμα αντίστοιχα

$$\varepsilon_{i+1}(\tau) = \frac{1}{\tau}\left(y(t_{i+1}) - w_{i+1}\right) \propto \tau^{n} \qquad \widetilde{\varepsilon}_{i+1}(\tau) = \frac{1}{\tau}\left(y(t_{i+1}) - \widetilde{w}_{i+1}\right) \propto \tau^{n+k-1}$$

Με αντικατάσταση προκύπτει

$$\varepsilon_{i+1}(\tau) = \widetilde{\varepsilon}_{i+1}(\tau) + \frac{1}{\tau}\left(\widetilde{w}_{i+1} - w_{i+1}\right) \Rightarrow \boxed{\varepsilon_{i+1}(\tau) \approx \frac{1}{\tau}\left(\widetilde{w}_{i+1} - w_{i+1}\right)}$$

Προσαρμοζόμενες Μέθοδοι (adaptive ή error control)

Χρησιμοποιώντας την προηγούμενη προσέγγιση, μπορούμε να αναρωτηθούμε αν θα μπορούσαμε να δημιουργήσουμε μεθόδους που να μην έχουν απαραίτητα σταθερό βήμα, αλλά να μπορούν προσαρμόζουν το βήμα τους σύμφωνα με την επιδιωκόμενη ακρίβεια

Ας γράψουμε το τοπικό σφάλμα της μεθόδου τάξης n

$$\varepsilon_{i+1}(\tau) \approx K\tau^n$$

Αν αντί για το βήμα ***τ*** χρησιμοποιήσουμε ένα βήμα ***qτ***, τότε:

$$\varepsilon_{i+1}(q\tau) \approx K(q\tau)^n = q^n\left(K\tau^n\right) \approx \frac{q^n}{\tau}\left(\tilde{w}_{i+1} - w_{i+1}\right)$$

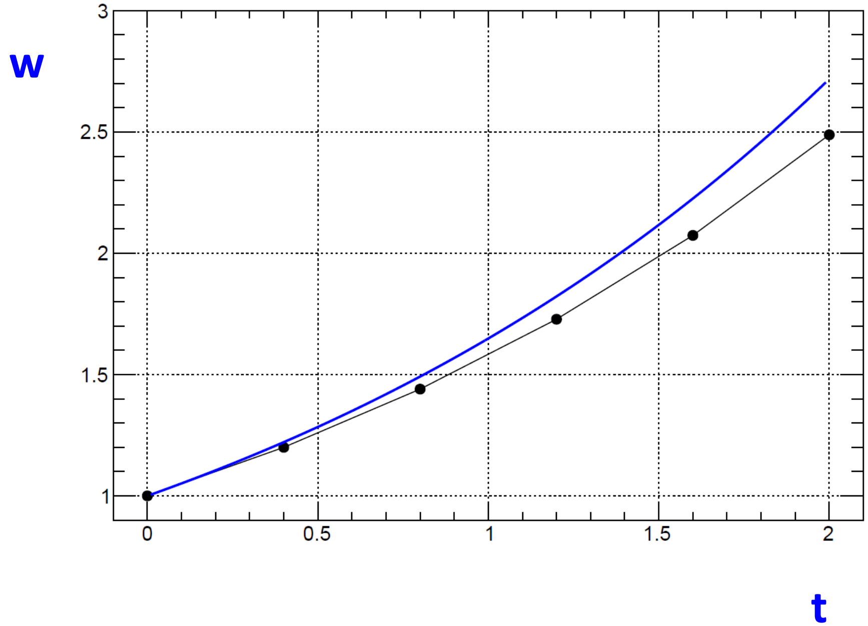
Και απαιτώντας το σφάλμα να είναι μικρότερο από μια τιμή, έστω ***ε***, τότε:

$$\frac{q^n}{\tau}\left|\tilde{w}_{i+1} - w_{i+1}\right| \le \varepsilon \quad \Rightarrow \quad \boxed{q \le \left(\frac{\varepsilon\tau}{\left|\tilde{w}_{i+1} - w_{i+1}\right|}\right)^{1/n}}$$

Γνωρίζουμε ότι καθώς προχωράμε στο χρόνο η προσέγγιση της λύσης ανεξάρτητα από τη μέθοδο που χρησιμοποιούμε έχει μεγαλύτερο σφάλμα

π.χ. για την Euler

$$\left| y(t_i) - w_i \right| \le \frac{\tau M}{2L} \left[e^{L(t_i - a)} - 1 \right]$$



Γιατί λοιπόν να βασίζουμε την εκτίμησή μας για το σημείο ***i+1*** μόνο στο σημείο ***i*** που είναι το πιο ανακριβές που έχουμε στη διάθεσή μας;

Στη γενική περίπτωση μπορούμε να ορίσουμε την πολύ-βηματική μέθοδο m-βημάτων ως:

$$
\begin{aligned}
w_{i+1} = \quad & a_{m-1}w_i + a_{m-2}w_{i-1} + \ldots + a_0 w_{i+1-m} + \\
& \tau\left[b_m f(t_{i+1}, w_{i+1}) + b_{m-1} f(t_i, w_i) + \ldots + b_0 f(t_{i+1-m}, w_{i+1-m})\right]
\end{aligned}
$$

όπου $\quad i = m-1, m, \ldots, N-1 \qquad \tau = \dfrac{(b-a)}{N}$

και $\quad a_0, a_1, \ldots, a_{m-1} \;\; \kappa\alpha\iota \;\; b_0, b_1, \ldots, b_m \quad$ γνωστές σταθερές

Επίσης δίνονται οι αρχικές τιμές

$$w_0 = c, \;\; w_1 = c_1, \;\; w_2 = c_2, \;\ldots\; w_{m-1} = c_{m-1}$$

Στη γενική περίπτωση μπορούμε να ορίσουμε την πολύ-βηματική μέθοδο m-βημάτων ως:

$$
\begin{aligned}
w_{i+1} = \quad & a_{m-1} w_i + a_{m-2} w_{i-1} + \ldots + a_0 w_{i+1-m} + \\
& \tau \left[b_m f(t_{i+1}, w_{i+1}) + b_{m-1} f(t_i, w_i) + \ldots + b_0 f(t_{i+1-m}, w_{i+1-m}) \right]
\end{aligned}
$$

όπου $\quad i = m-1, m, \ldots, N-1 \qquad \tau = \dfrac{(b-a)}{N}$

και $\quad a_0, a_1, \ldots, a_{m-1} \;\; \kappa\alpha\iota \;\; b_0, b_1, \ldots, b_m \quad$ γνωστές σταθερές

Επίσης δίνονται οι αρχικές τιμές

$$
w_0 = c, \;\; w_1 = c_1, \;\; w_2 = c_2, \;\; \ldots \;\; w_{m-1} = c_{m-1}
$$

Αν $\boldsymbol{b_m = 0}$ → άμεση μέθοδος (explicit)

Αν $\boldsymbol{b_m \neq 0}$ → έμμεση μέθοδος (implicit)

Πολύ-βηματικές Μέθοδοι 4-ης τάξης

Άμεση μέθοδος **Adams-Bashforth** 4ης τάξης

$$w_0 = c,\;\; w_1 = c_1,\;\; w_2 = c_2,\;\; w_3 = c_3$$

$$w_{i+1} = w_i + \frac{\tau}{24}\left[55 f(t_i, w_i) - 59 f(t_{i-1}, w_{i-1}) + 37 f(t_{i-2}, w_{i-2}) - 9 f(t_{i-3}, w_{i-3})\right]$$

Έμμεση μέθοδος **Adams-Moulton** 4ης τάξης

$$w_0 = c,\;\; w_1 = c_1,\;\; w_2 = c_2$$

$$w_{i+1} = w_i + \frac{\tau}{24}\left[9 f(t_{i+1}, w_{i+1}) + 19 f(t_i, w_i) - 5 f(t_{i-1}, w_{i-1}) + f(t_{i-2}, w_{i-2})\right]$$

Adams-Bashforth Two-Step Explicit Method

$$w_0 = \alpha, \quad w_1 = \alpha_1,$$

$$w_{i+1} = w_i + \frac{h}{2}[3f(t_i, w_i) - f(t_{i-1}, w_{i-1})],$$

Adams-Bashforth Three-Step Explicit Method

$$w_0 = \alpha, \quad w_1 = \alpha_1, \quad w_2 = \alpha_2,$$

$$w_{i+1} = w_i + \frac{h}{12}[23f(t_i, w_i) - 16f(t_{i-1}, w_{i-1}) + 5f(t_{i-2}, w_{i-2})],$$

Adams-Bashforth Four-Step Explicit Method

$$w_0 = \alpha, \quad w_1 = \alpha_1, \quad w_2 = \alpha_2, \quad w_3 = \alpha_3,$$

$$w_{i+1} = w_i + \frac{h}{24}[55f(t_i, w_i) - 59f(t_{i-1}, w_{i-1}) + 37f(t_{i-2}, w_{i-2}) - 9f(t_{i-3}, w_{i-3})],$$

Adams-Bashforth Five-Step Explicit Method

$$w_0 = \alpha, \quad w_1 = \alpha_1, \quad w_2 = \alpha_2, \quad w_3 = \alpha_3, \quad w_4 = \alpha_4,$$

$$w_{i+1} = w_i + \frac{h}{720}[1901f(t_i, w_i) - 2774f(t_{i-1}, w_{i-1})$$
$$+ 2616f(t_{i-2}, w_{i-2}) - 1274f(t_{i-3}, w_{i-3}) + 251f(t_{i-4}, w_{i-4})],$$

Adams-Moulton Two-Step Implicit Method

$$w_0 = \alpha, \quad w_1 = \alpha_1,$$

$$w_{i+1} = w_i + \frac{h}{12}[5f(t_{i+1}, w_{i+1}) + 8f(t_i, w_i) - f(t_{i-1}, w_{i-1})],$$

Adams-Moulton Three-Step Implicit Method

$$w_0 = \alpha, \quad w_1 = \alpha_1, \quad w_2 = \alpha_2,$$

$$w_{i+1} = w_i + \frac{h}{24}[9f(t_{i+1}, w_{i+1}) + 19f(t_i, w_i) - 5f(t_{i-1}, w_{i-1}) + f(t_{i-2}, w_{i-2})],$$

Adams-Moulton Four-Step Implicit Method

$$w_0 = \alpha, \quad w_1 = \alpha_1, \quad w_2 = \alpha_2, \quad w_3 = \alpha_3,$$

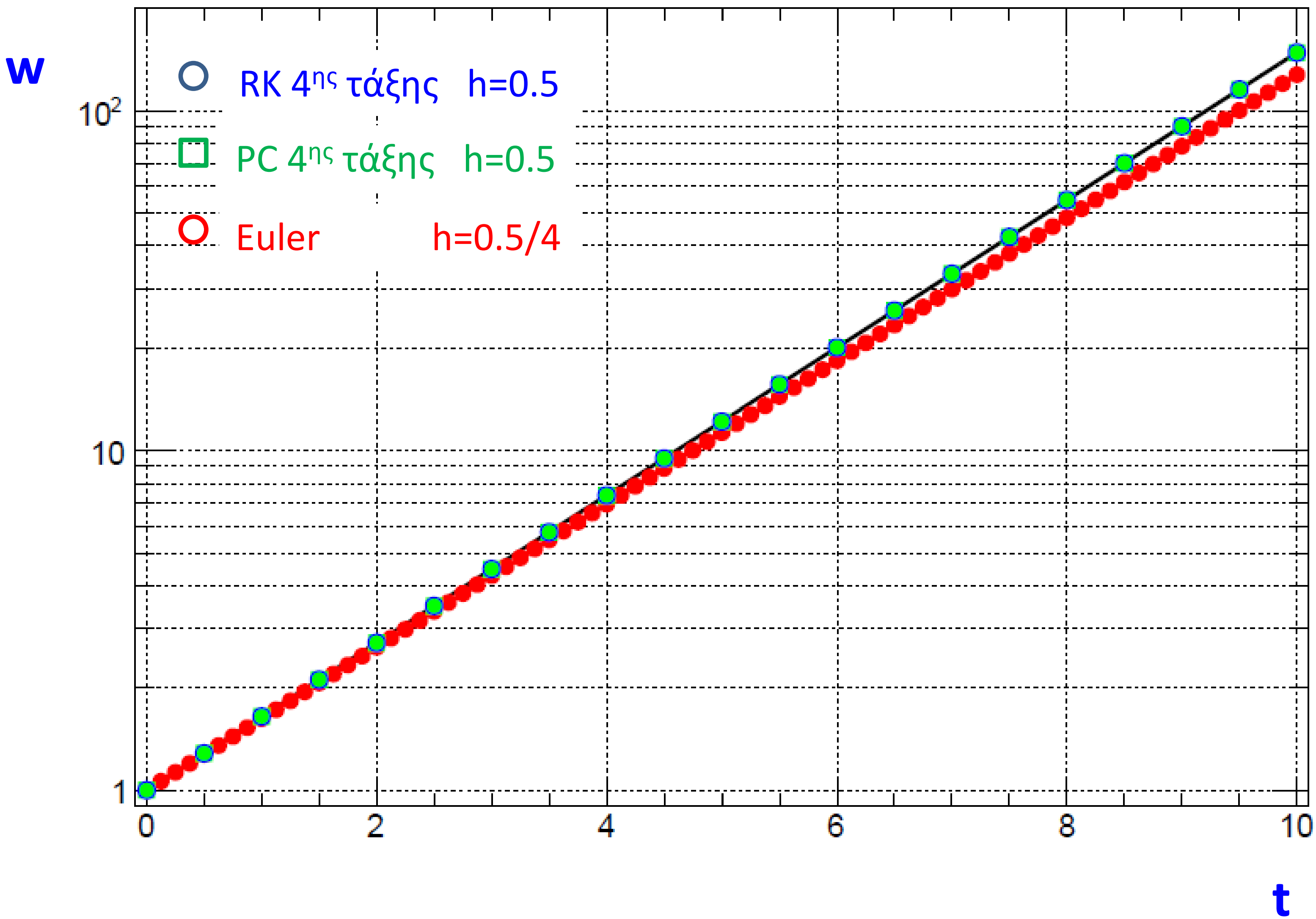
$$\begin{aligned} w_{i+1} = w_i + \frac{h}{720}[&251f(t_{i+1}, w_{i+1}) + 646f(t_i, w_i) \\ &- 264f(t_{i-1}, w_{i-1}) + 106f(t_{i-2}, w_{i-2}) - 19f(t_{i-3}, w_{i-3})], \end{aligned}$$

Εν γένει, οι έμμεσες μέθοδοι συμπεριφέρονται καλύτερα από τις άμεσες αβαφορικά με την ευστάθεια και την ακρίβεια που προσφέρουν. Χρειάζονται όμως αναλυτικές πράξεις για την επίλυση της εξίσωσης που προκύπτει για το $\boldsymbol{w_{i+1}}$.

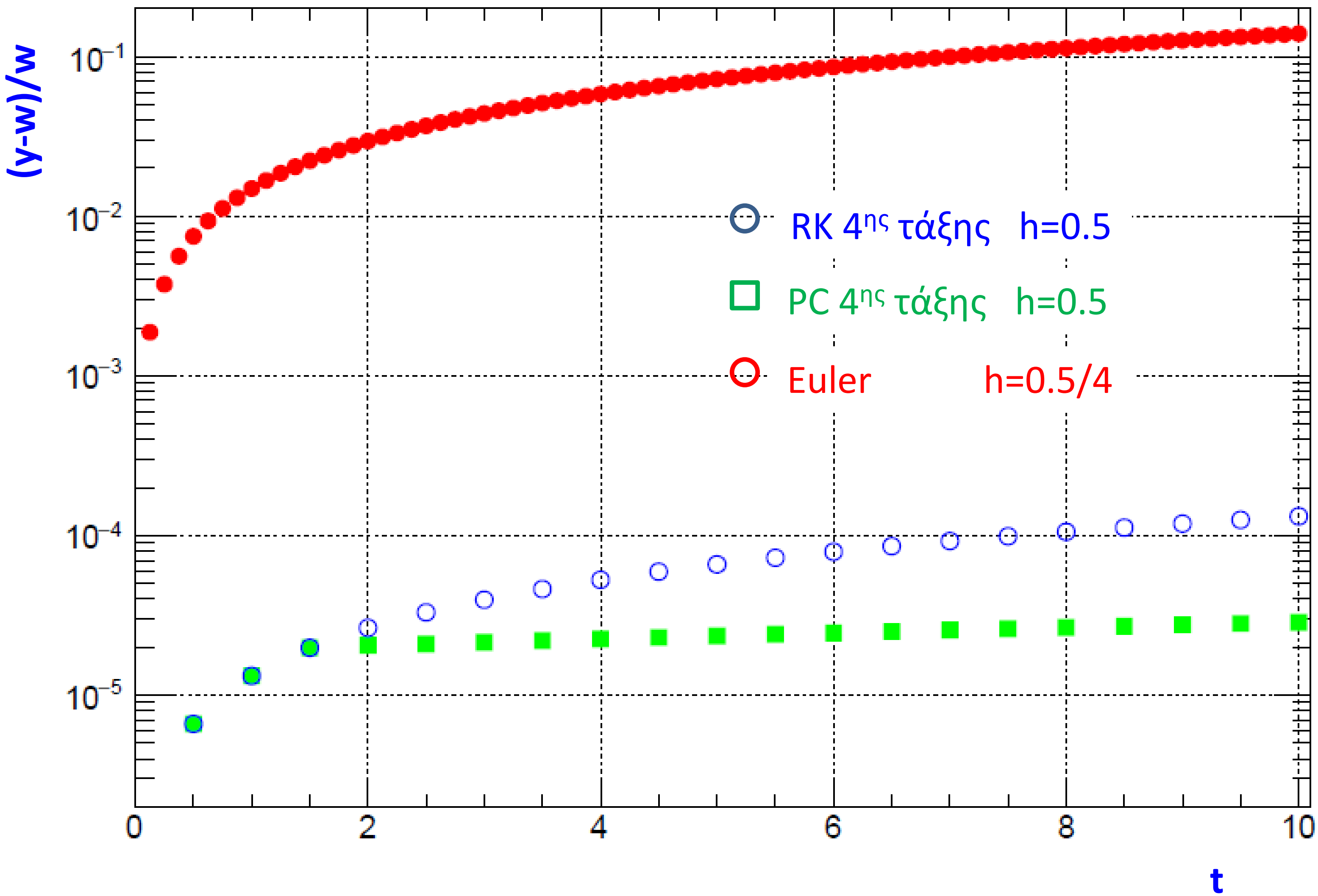
Εναλλακτικά μπορεί κανείς να χρησιμοποιήσει μια άμεση μέθοδο ως αρχική πρόβλεψη (predictor) και την έμμεση μέθοδο για τη διόρθωση της εκτίμησης (corrector).

```
void precor(double h)
{
double t[100]; double w[100];
double a=0; double b=10; int N=(b-a)/h;
t[0]=a; w[0]=1.;
for(int i=0; i<3; i++)
{ // Αρχικά βήματα με RK-4ης τάξης
double k1=h*myf(t[i], w[i]);
double k2=h*myf(t[i]+h/2, w[i]+k1/2);
double k3=h*myf(t[i]+h/2, w[i]+k2/2);
double k4=h*myf(t[i]+h, w[i]+k3);
w[i+1]= w[i]+(k1+2*k2+2*k3+k4)/6;
t[i+1]=t[i]+h;
}

for(int i=3; i<N; i++)
{
t[i+1]=t[i]+h;
//Predictor
w[i+1]= w[i]+(h/24)*(55*myf(t[i],w[i])-59*myf(t[i-1],w[i-1])+37*myf(t[i-2],w[i-2])-
9*myf(t[i-3],w[i-3]));
//Corrector
w[i+1]= w[i]+(h/24)*(9*myf(t[i+1],w[i+1])+19*myf(t[i],w[i])-5*myf(t[i-1],w[i-
1])+myf(t[i-2],w[i-2]));
//ew[i+1]= fabs(exp(0.5*t[i+1])-w[i+1]);    //απόλυτο σφάλμα
ew[i+1]= fabs(exp(0.5*t[i+1])-w[i+1])/exp(0.5*t[i+1]); //σχετικό σφάλμα
}
}
```

$$y' = \frac{1}{2}y, \quad 0 \le t \le 10, \quad y(0) = 1.$$



$$y' = \frac{1}{2} y, \quad 0 \le t \le 10, \quad y(0) = 1.$$



$$\frac{d\vec{y}}{dt} = \vec{f}(t, \vec{y}) \qquad \vec{y}(a) = \vec{c}$$

$$\frac{dy_1}{dt} = f_1(t, y_1, y_2, \ldots, y_m) \qquad y_1(a) = c_1$$

$$\frac{dy_2}{dt} = f_2(t, y_1, y_2, \ldots, y_m) \qquad y_2(a) = c_2$$

$$\ldots \qquad \ldots$$

$$\frac{dy_m}{dt} = f_m(t, y_1, y_2, \ldots, y_m) \qquad y_m(a) = c_m$$

Επιλύσιμο, αν: $$\left|\frac{\partial f(t, y_1, y_2, \ldots y_m}{\partial y_i}\right| \le L$$

Απλή γενίκευση της μονοδιάστατης περίπτωσης αν χρησιμοποιήσουμε δύο δείκτες
w_{ij} ,
όπου το $1 < i < m$ δηλώνει την συνάρτηση που προσεγγίζουμε
και το $0 < j < N$ το χρονικό σημείο στο οποίο βρισκόμαστε

π.χ. Μέθοδος Euler

$$w_{1,0}=c_1,\quad w_{2,0}=c_2, \dots,\quad w_{m,0}=c_m$$

$$w_{1,j+1}=w_{1,j}+\tau f_{1}(t_i, w_{1,j}, w_{2,j}, \dots, w_{m,j})$$

$$w_{2,j+1}=w_{2,j}+\tau f_2(t_i, w_{1,j}, w_{2,j}, \dots, w_{m,j})$$

$$\dots$$

$$w_{m,j+1}=w_{m,j}+\tau f_m(t_i, w_{1,j}, w_{2,j}, \dots, w_{m,j})$$

Επίλυση συστήματος διαφορικών εξισώσεων

Απλή γενίκευση της μονοδιάστατης περίπτωσης αν χρησιμοποιήσουμε δύο δείκτες w_{ij},

όπου το $1 < i < m$ δηλώνει την συνάρτηση που προσεγγίζουμε

και το $0 < j < N$ το χρονικό σημείο στο οποίο βρισκόμαστε

π.χ. Μέθοδος Runge-Kutta 4 ης τάξης

$$w_{1,0}=c_1,\quad w_{2,0}=c_2,\ldots,\quad w_{m,0}=c_m$$

$$1<i<m,\quad k_{1,i}=\tau f_i(t_j, w_{1,j}, w_{2,j},\ldots, w_{m,j})$$

$$1<i<m,\quad k_{2,i}=\tau f_i(t_j+\tau/2, w_{1,j}+k_{1,1}/2, w_{2,j}+k_{1,2}/2,\ldots, w_{m,j}+k_{1,m}/2)$$

$$1<i<m,\quad k_{3,i}=\tau f_i(t_j+\tau/2, w_{1,j}+k_{2,1}/2, w_{2,j}+k_{2,2}/2,\ldots, w_{m,j}+k_{2,m}/2)$$

$$1<i<m,\quad k_{4,i}=\tau f_i(t_j+\tau, w_{1,j}+k_{3,1}, w_{2,j}+k_{3,2},\ldots, w_{m,j}+k_{3,m})$$

$$1<i<m,\quad w_{i,j+1}=w_{i,j}+1/6(k_{1,i}+2k_{2,i}+2k_{3,i}+k_{4,i})$$

Χρειάζεται λίγη προσοχή στη σειρά υπολογισμών των ***k***

Επίλυση συστήματος διαφορικών εξισώσεων – Παράδειγμα I

Διάσπαση δύο διαδοχικών ραδιενεργών υλικών

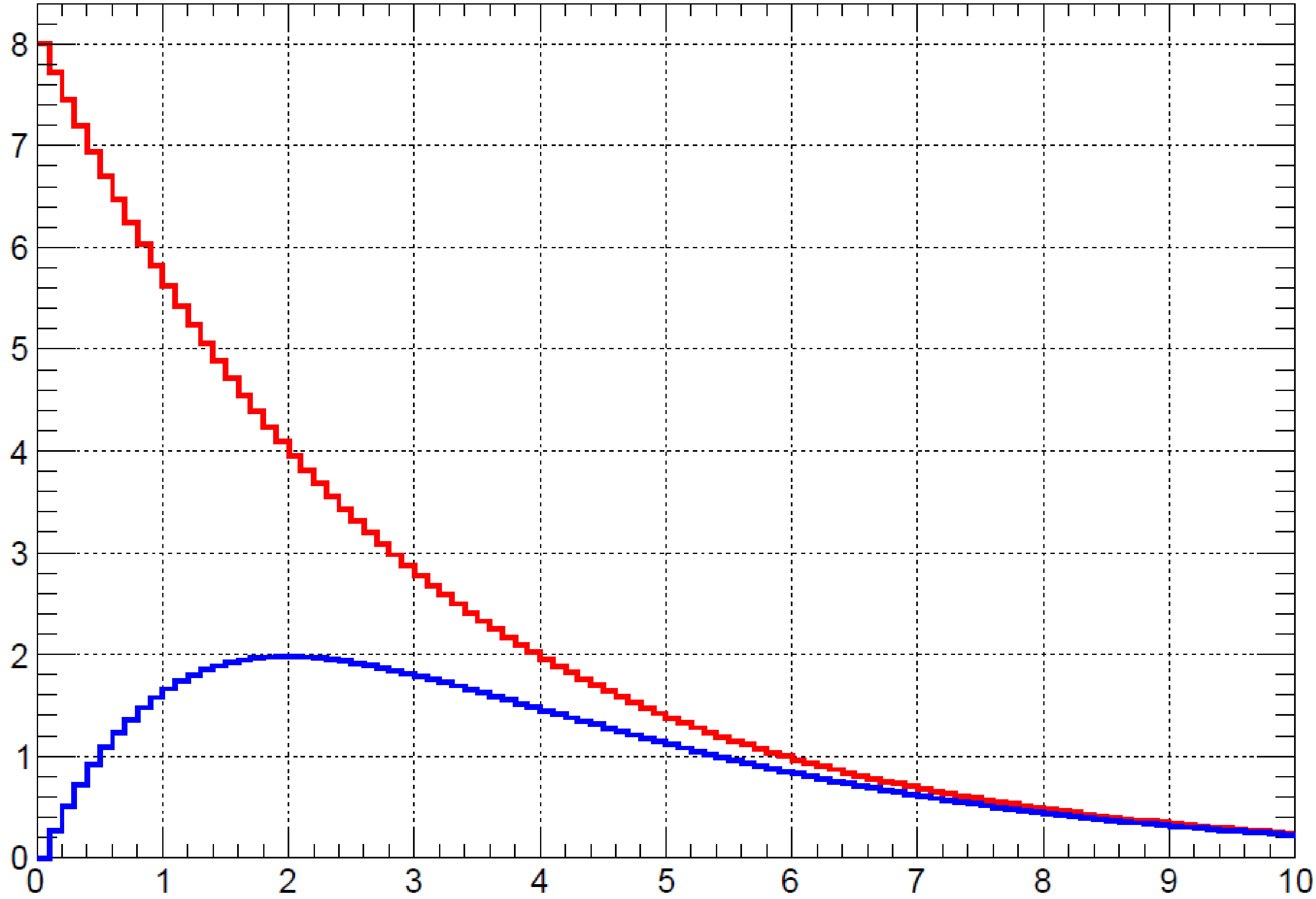
$$\frac{dA_1}{dt} = -\lambda_1 A_1 \qquad \frac{dA_2}{dt} = +\lambda_1 A_1 - \lambda_2 A_2 \qquad \lambda_1 = \frac{\ln 2}{T^1_{1/2}}, \lambda_2 = \frac{\ln 2}{T^2_{1/2}}$$

Επίλυση συστήματος διαφορικών εξισώσεων – Παράδειγμα Ι

```
double myf1(double t, double y1,double y2, double tl1, double tl2){
    double val=-y1*log(2.)/tl1;  return val;
}
double myf2(double t, double y1,double y2, double tl1, double tl2){
    double val= -y2*log(2.)/tl2+y1*log(2.)/tl1; return val;
}
void decay_euler(double tl1, double tl2)
{
  TH1F *h1=new TH1F("h_1"," ",100, 0., 10.);
  TH1F *h2=new TH1F("h_2"," ",100, 0., 10.);
  double t=0., w1=8, w2=0;
  int N=100;  double h=10./N;
  h1->SetBinContent(1,w1);
  h2->SetBinContent(1,w2);
  for(int i=1;i<=N;i++)
    {
     w1=w1+h*myf1(t,w1,w2, tl1,tl2);
     w2=w2+h*myf2(t,w1,w2, tl1,tl2);
     t+=h;
     h1->SetBinContent(i+1,w1);
     h2->SetBinContent(i+1,w2);
    }
  h1->SetLineWidth(2.);
  h1->SetLineColor(kRed);
  h2->SetLineWidth(2.);
  h2->SetLineColor(kBlue);
  h1->Draw("");
  h2->Draw("same");
}
```

* Σχετίζονται με την παρουσίαση των αποτελεσμάτων σε ιστογράμματα στο root

Επίλυση συστήματος διαφορικών εξισώσεων – Παράδειγμα I



Ανώτερης τάξης διαφορικές εξισώσεις:

$$\frac{dy^{(m)}}{dt} = f(t, y, y', y'', \ldots, y^{(m-1)}) \qquad y^{(i)}(a) = c_i$$

Επιλύουμε μετατρέποντας σε σύστημα θέτοντας:

$$u_1(t) = y(t), u_2(t) = y'(t), u_3(t) = y''(t), \ldots, u_m(t) = y^{(m-1)}(t)$$

Οπότε καταλήγουμε στο σύστημα ΔΕ:

$$\frac{du_1}{dt} = \frac{dy}{dt} = u_2$$

$$\frac{du_2}{dt} = \frac{dy'}{dt} = u_3$$

$$\ldots$$

$$\frac{du_m}{dt} = \frac{dy^{(m-1)}}{dt} = y^{(m)} = f(t, y, y', y'', \ldots, y^{(m-1)}) = f(t, u_1, u_2, \ldots, u_m)$$

$$u_1(a) = y(a) = c_1, \;\; u_2(a) = y'(a) = c_2, \; \ldots \; , u_m(a) = y^{(m-1)}(a) = c_m$$

Επίλυση ανώτερης τάξης διαφορικών εξισώσεων - Παράδειγμα 1

Ελεύθερη πτώση

$$\frac{d^2y}{dt^2} = -g \qquad 0 \le t \le t_1 \qquad y(0) = h,\ \ y'(0) = 0$$

$$\boxed{\begin{aligned} y &= y(0) + v(0)t - \frac{1}{2}gt^2 \\ v &= v(0) - gt \end{aligned}}$$

Επίλυση ανώτερης τάξης διαφορικών εξισώσεων - Παράδειγμα 1

Ελεύθερη πτώση $\dfrac{d^2y}{dt^2} = -g \qquad 0 \le t \le t_1 \quad y(0) = h, \ y'(0) = 0$

$$\boxed{\begin{aligned} y &= \quad y(0) + v(0)t - \frac{1}{2}gt^2 \\ v &= \quad v(0) - gt \end{aligned}}$$

$$\frac{dy}{dt} = v$$

$$\frac{dv}{dt} = -g$$

Euler ▬ ▶

$$y(t+\delta t) = \quad y(t) + v(t)\delta t$$

$$v(t+\delta t) = \quad v(t) - g\delta t$$

Επίλυση ανώτερης τάξης διαφορικών εξισώσεων - Παράδειγμα 1

Ελεύθερη πτώση $\frac{d^2y}{dt^2} = -g \qquad 0 \le t \le t_1 \qquad y(0) = h,\ y'(0) = 0$

$$\begin{aligned} y &= y(0) + v(0)t - \frac{1}{2}gt^2 \\ v &= v(0) - gt \end{aligned}$$

$$\frac{dy}{dt} = v$$

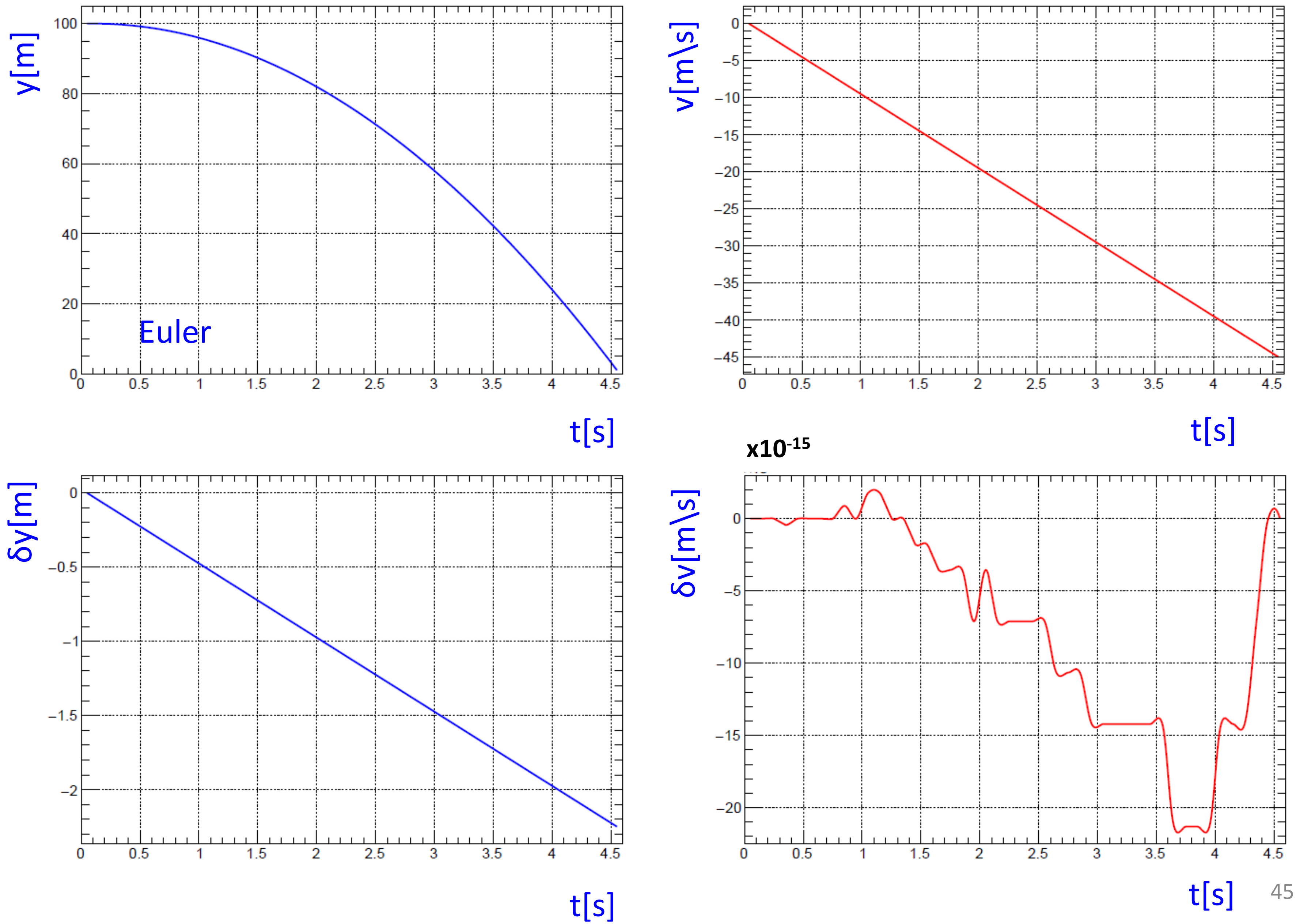
$$\frac{dv}{dt} = -g$$

Euler ▶

$$\begin{aligned} y(t+\tau) &= y(t) + \tau v(t) \\ v(t+\tau) &= v(t) - \tau g \end{aligned}$$

```c
#include <stdio.h>
#include <math.h>
void elptosi()
{
    double y=100;   //initial conditions
    double v=0;
    double t=0;
    double g=10.;  // rest of constants
    double h=0.1;  // time step
    printf("%lf %lf %lf \n", t,y,v);
    while(y>=0)
       {
         y=y+h*v;
         v=v-h*g;
         t=t+h;
         printf("%lf %lf %lf \n", t,y,v);
       }
}
```

Επίλυση ανώτερης τάξης διαφορικών εξισώσεων - Παράδειγμα 1



Επίλυση ανώτερης τάξης διαφορικών εξισώσεων - Παράδειγμα 1

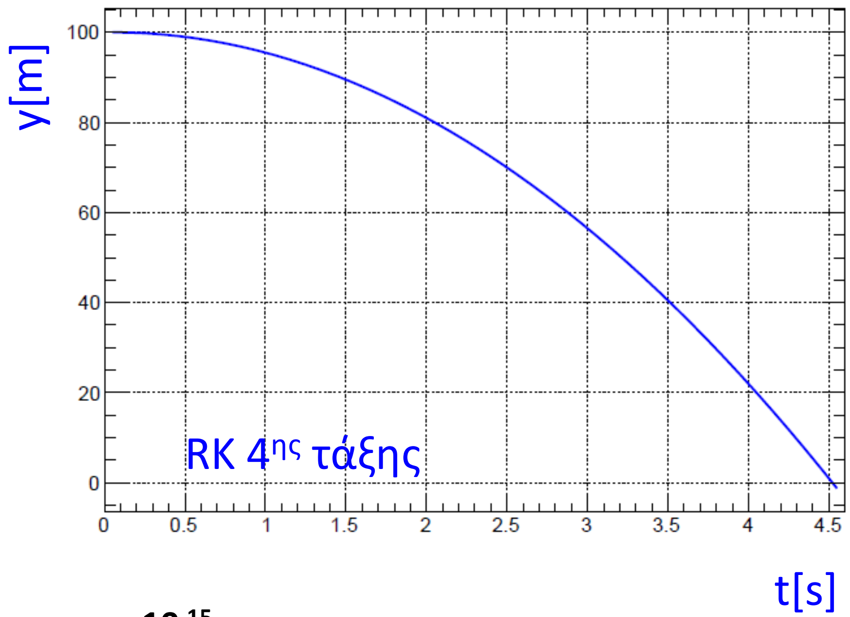
Runge-Kutta 4ης τάξης

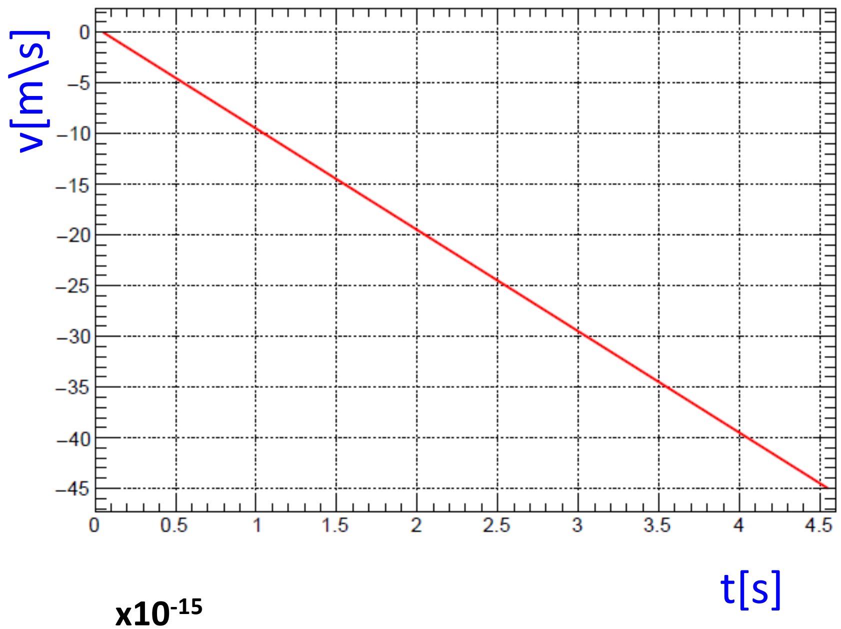
$$\frac{dy}{dt} = v$$

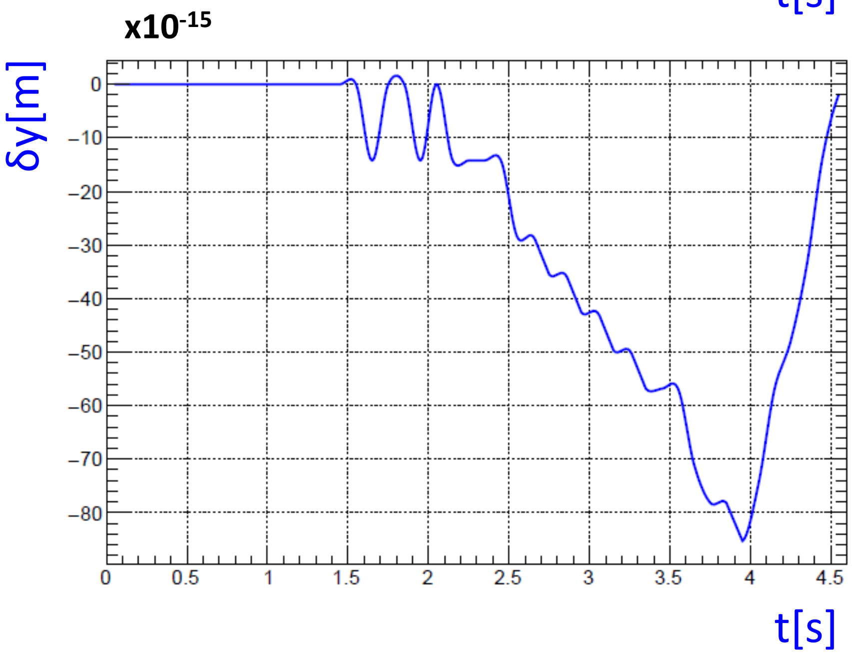
$$\frac{dv}{dt} = -g$$

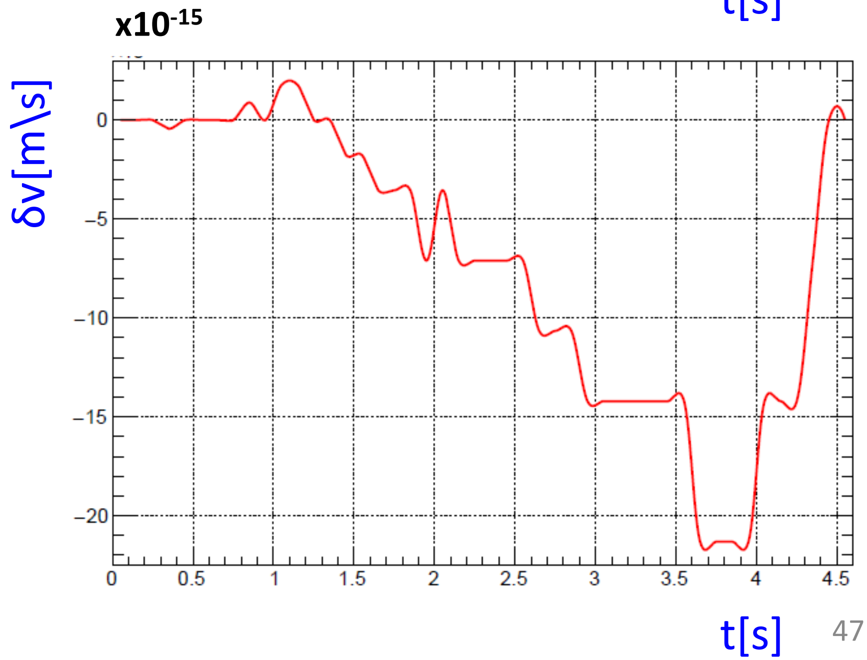
```
#include <stdio.h>
#include <math.h>
double myy(double t, double y, double v){
  return v;
}
double myv(double t, double y, double v){
  double g =10.;
  return -g;
}
void elptosi()
{
    double y=100;  //initial conditions
    double v=0;
    double t=0;
    double g=10.;  // rest of constants
    double h=0.1;  // time step
    printf("%lf %lf %lf \n", t,y,v);
    while(y>=0)
       {
          double k11=h*myy(t,y,v);
          double k12=h*myv(t,y,v);
          double k21=h*myy(t+h/2, y+k11/2, v+k12/2);
          double k22=h*myv(t+h/2, y+k11/2, v+k12/2);
          double k31=h*myy(t+h/2, y+k21/2, v+k22/2);
          double k32=h*myv(t+h/2, y+k21/2, v+k22/2);
          double k41=h*myy(t+h, y+k31, v+k32);
          double k42=h*myv(t+h, y+k31, v+k32);
          y=y+(k11+2.*k21+2*k31+k41)/6.;
          v=v+(k12+2.*k22+2*k32+k42)/6.;
          t=t+h;
          printf("%lf %lf %lf \n", t,y,v);
       }
}
```

Επίλυση ανώτερης τάξης διαφορικών εξισώσεων - Παράδειγμα 1









Επίλυση ανώτερης τάξης διαφορικών εξισώσεων – Παράδειγμα 1 συνέχεια

Αλλά στην πραγματικότητα έχουμε και την αντίσταση του αέρα

$$\frac{d^2y}{dt^2} = -g - c\frac{v}{|v|}|v|^k$$

Επίλυση ανώτερης τάξης διαφορικών εξισώσεων – Παράδειγμα 1 συνέχεια

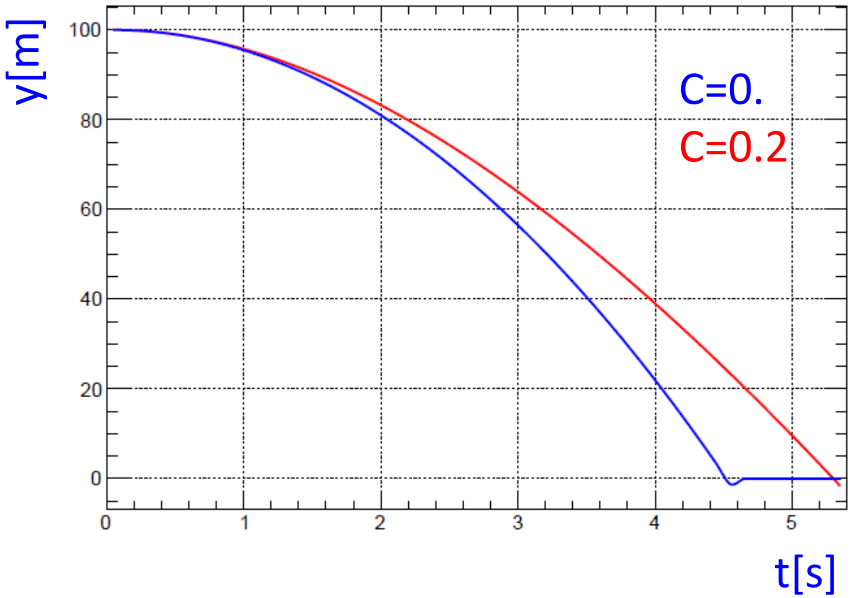
Αλλά στην πραγματικότητα έχουμε και την αντίσταση του αέρα

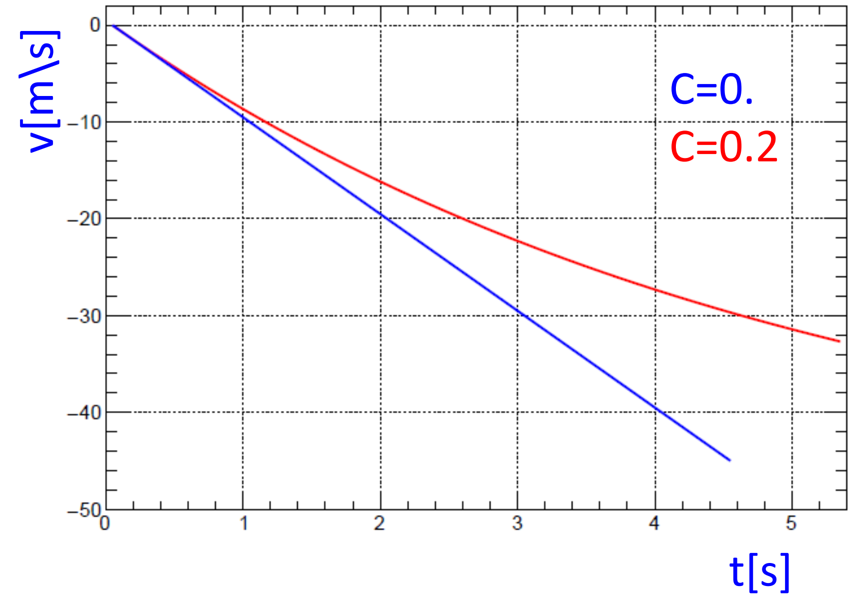
$$\frac{d^2y}{dt^2} = -g - c\frac{v}{|v|}|v|^k$$

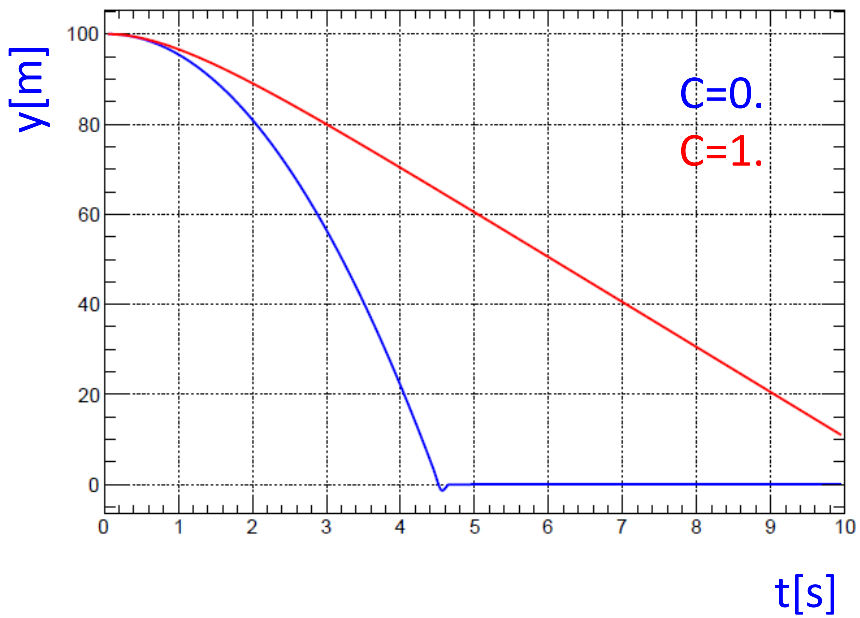
Το μόνο που χρειάζεται να αλλάξει στο πρόγραμμά μας είναι:

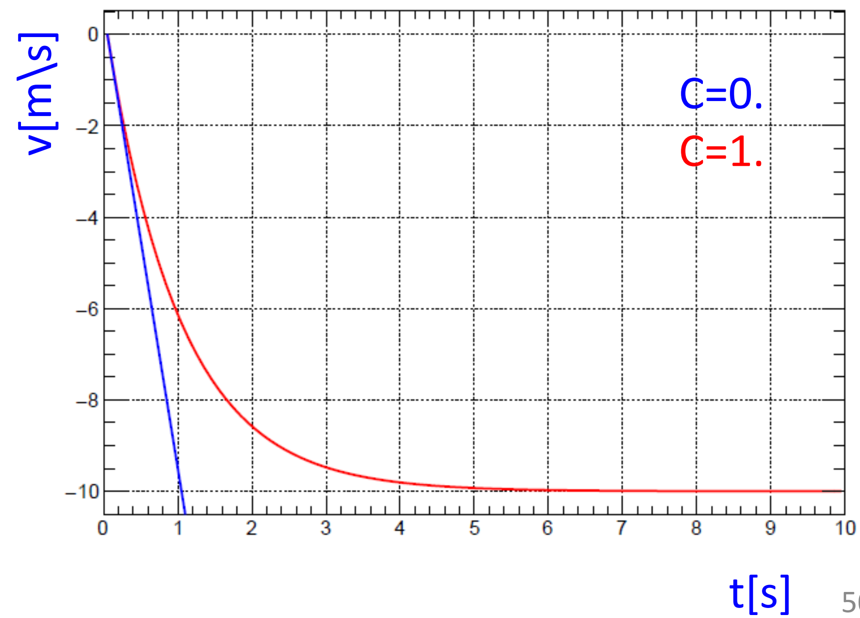
```
double myv(double t, double y, double v){
  double g =10.;
  return -g-0.2*v;
}
```

Επίλυση ανώτερης τάξης διαφορικών εξισώσεων - Παράδειγμα 1 συνέχεια









Μονοδιάστατη ταλάντωση

$$\frac{d^2y}{dt^2} = -Ky \qquad 0 \le t \le t_1 \qquad y(0) = h,\ \ y'(0) = 0$$

$$\frac{dy}{dt} = v$$

$$\frac{dv}{dt} = -Ky$$

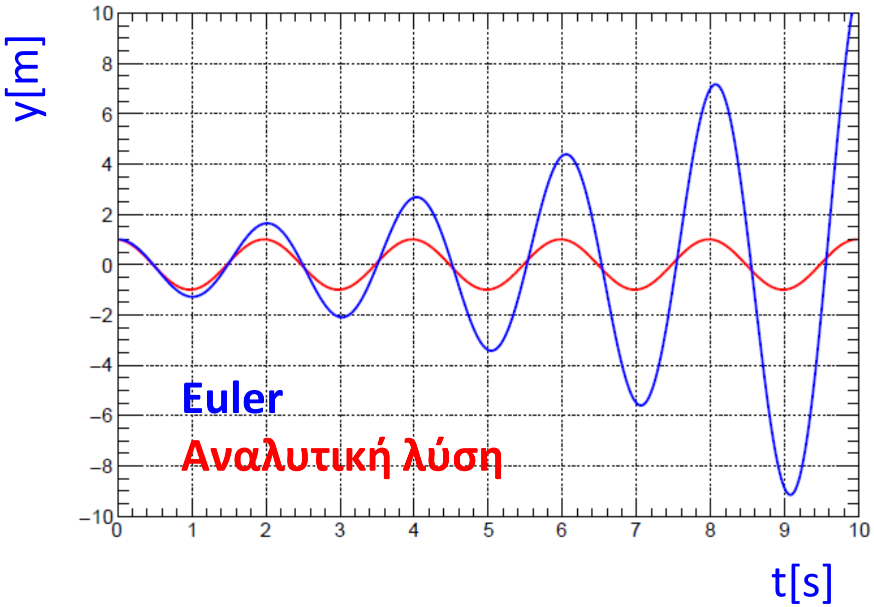
Euler ▶

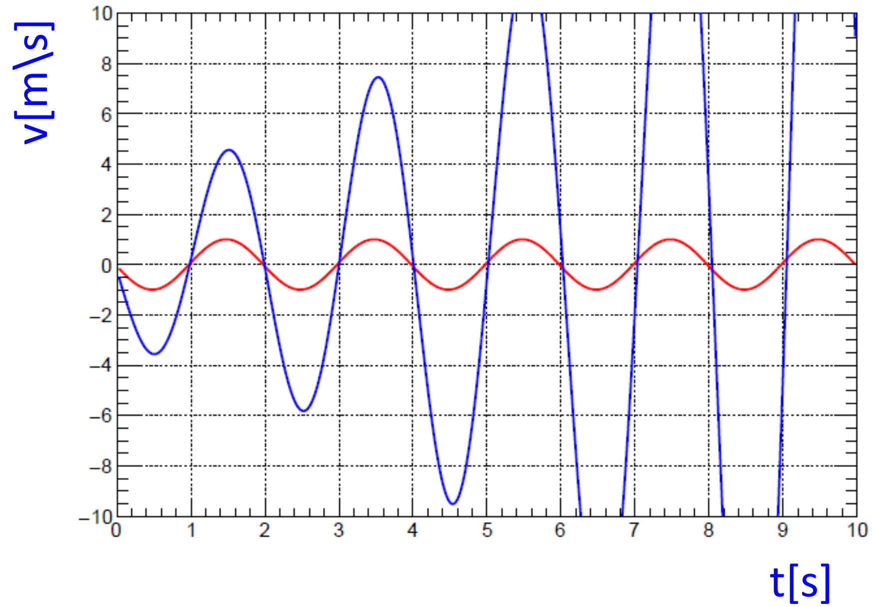
$$y(t+\tau) = y(t) + \tau v(t)$$

$$v(t+\tau) = v(t) - \tau K y(t)$$

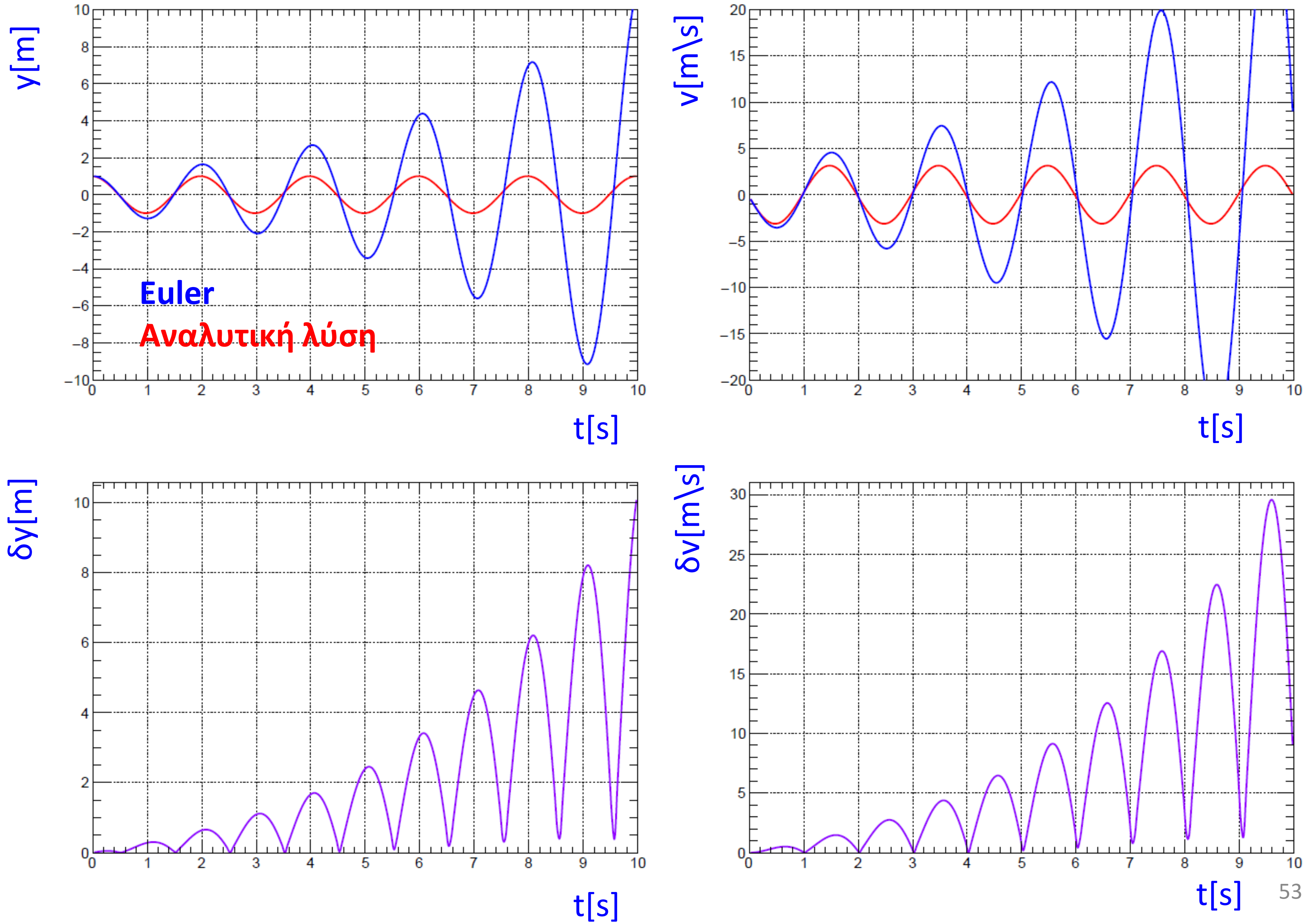
```
#include <stdio.h>
#include <math.h>
void elptosi()
{
    double y=1.;       //initial conditions
    double v=0;
    double t=0;
    double pi=acos(-1.);
    double K=pi*pi;    // rest of constants
    double h=0.05;     // time step
    printf("%lf %lf %lf \n", t,y,v);
    for(int i=1;i<200;i++)
       {
         double yo=y; double vo=v;
         y=yo+h*vo;
         v=vo-h*K*yo;
         t=t+h;
         printf("%lf %lf %lf \n", t,y,v);
       }
}
```

Επίλυση ανώτερης τάξης διαφορικών εξισώσεων - Παράδειγμα 2



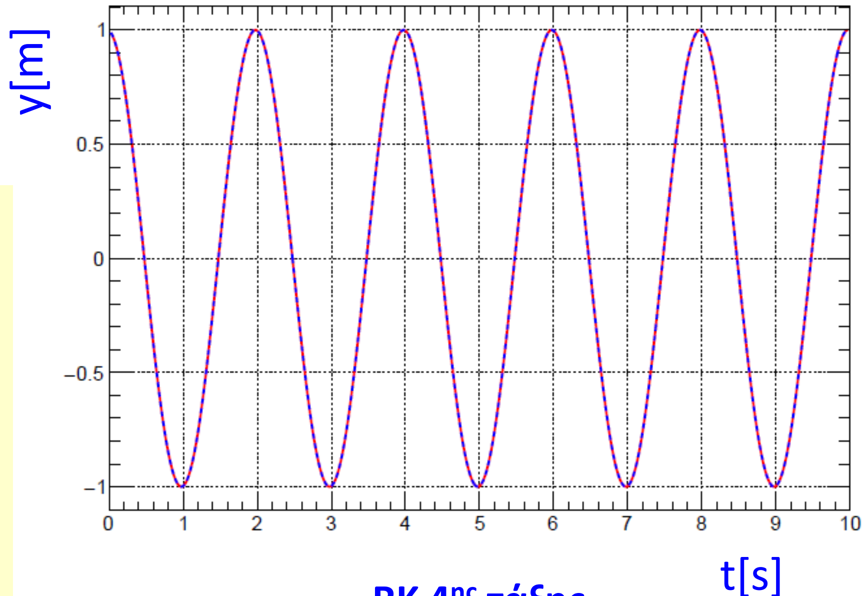


Επίλυση ανώτερης τάξης διαφορικών εξισώσεων - Παράδειγμα 2

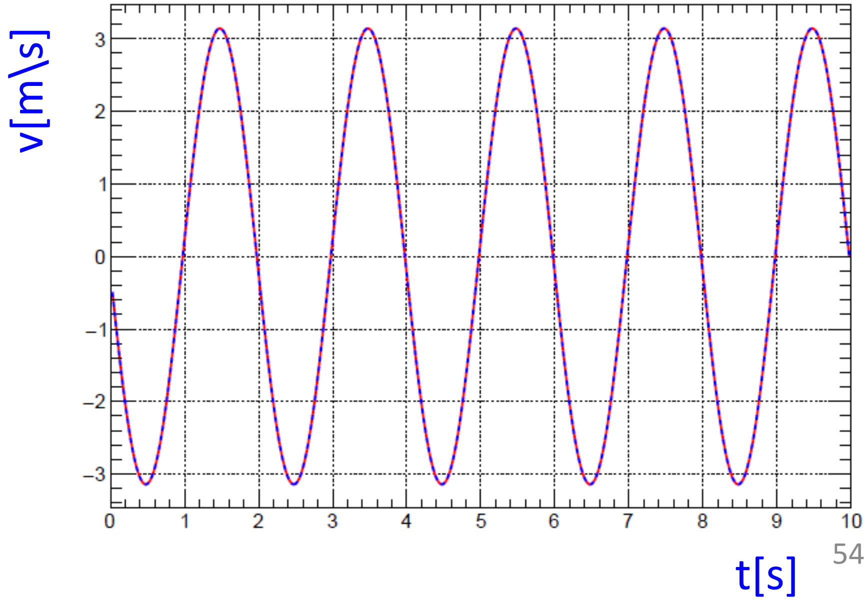


RK 4ης τάξης

```
double myy(double t, double y, double v)
{
  return v;
}
double myv(double t, double y, double v)
{
  double pi=acos(-1.);
  double K=pi*pi;
  return -K*y;
}
```



RK 4ης τάξης
Αναλυτική λύση



Παραλλαγές του προβλήματος:
π.χ. απόσβεση, εξωτερική δύναμη, μη γραμμικότητα απλά χρειάζονται την αλλαγή μιας γραμμής του αλγορίθμου

Τι συμβαίνει με την Euler σε προβλήματα ταλάντωσης;

$d^2x/dt^2 + \omega^2 x = 0$

Η εφαρμογή της μεθόδου Euler:

$v_{i+1} = v_i - \tau\omega^2 x_i$

$x_{i+1} = x_i + \tau v_i$

οπότε:

Τι συμβαίνει με την Euler σε προβλήματα ταλάντωσης;

$d^2x/dt^2 + \omega^2 x = 0$

Η εφαρμογή της μεθόδου Euler:

$v_{i+1} = v_i - \tau\omega^2 x_i$

$x_{i+1} = x_i + \tau v_i$

οπότε:

$$
\begin{aligned}
E_{i+1} = \quad & \frac{1}{2} m v_{i+1}^2 + \frac{1}{2} m\omega^2 x_{i+1}^2 \\
& \frac{1}{2} m v_i^2 + \frac{1}{2} m\omega^2 x_i^2 + \frac{1}{2} m\omega^4 x_i^2 \tau^2 + \frac{1}{2} m\omega^2 v_i^2 \tau^2 \\
& E_i + \frac{1}{2} m\omega^2 \tau^2 (\omega^2 x_i^2 + v_i^2)
\end{aligned}
$$

Τι συμβαίνει με την Euler σε προβλήματα ταλάντωσης;

$d^2x/dt^2 + \omega^2 x = 0$

Η εφαρμογή της μεθόδου Euler:

$v_{i+1} = v_i - \tau\omega^2 x_i$

$x_{i+1} = x_i + \tau v_i$

οπότε:

$$
\begin{aligned}
E_{i+1} = \quad & \frac{1}{2} m v_{i+1}^2 + \frac{1}{2} m\omega^2 x_{i+1}^2 \\
& \frac{1}{2} m v_i^2 + \frac{1}{2} m\omega^2 x_i^2 + \frac{1}{2} m\omega^4 x_i^2 \tau^2 + \frac{1}{2} m\omega^2 v_i^2 \tau^2 \\
& E_i + \frac{1}{2} m\omega^2 \tau^2 (\omega^2 x_i^2 + v_i^2) \longrightarrow \mathbf{>0}
\end{aligned}
$$

Μια μικρή δόση «μαγείας»!

Μονοδιάστατη ταλάντωση

$$\frac{dy}{dt} = v$$

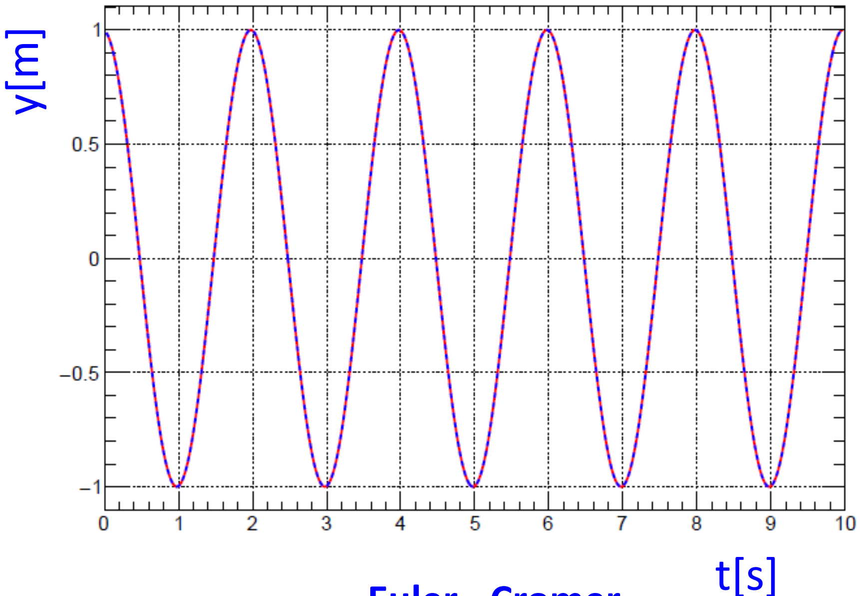
$$\frac{dv}{dt} = -Ky$$

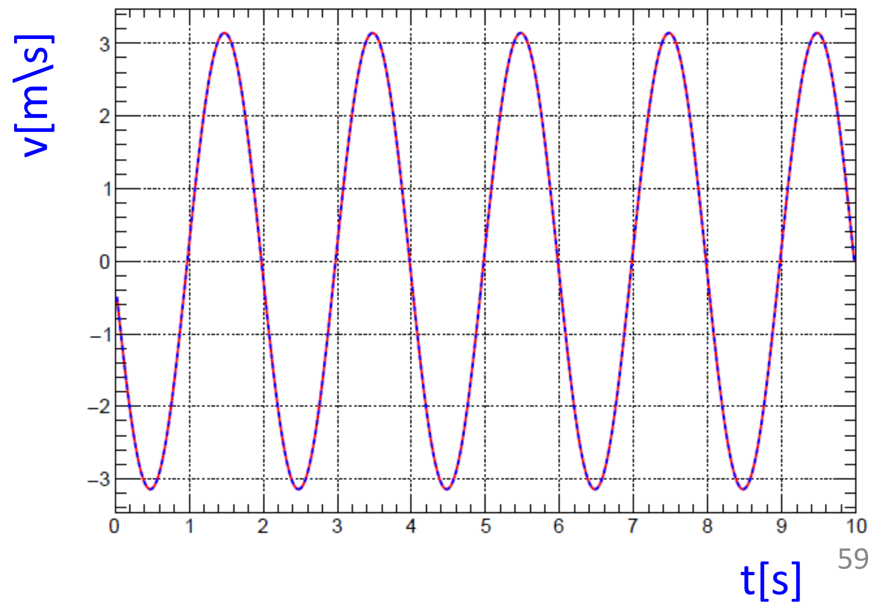
```
#include <stdio.h>
#include <math.h>
void elptosi()
{
    double y=1.;        //initial conditions
    double v=0;
    double t=0;
    double pi=acos(-1.);
    double K=pi*pi;     // rest of constants
    double h=0.05;      // time step
    printf("%lf %lf %lf \n", t,y,v);
    for(int i=1;i<200;i++)
       {
         y = y+h*v;
         v = v-h*K*y;
         t=t+h;
         printf("%lf %lf %lf \n", t,y,v);
       }
}
```

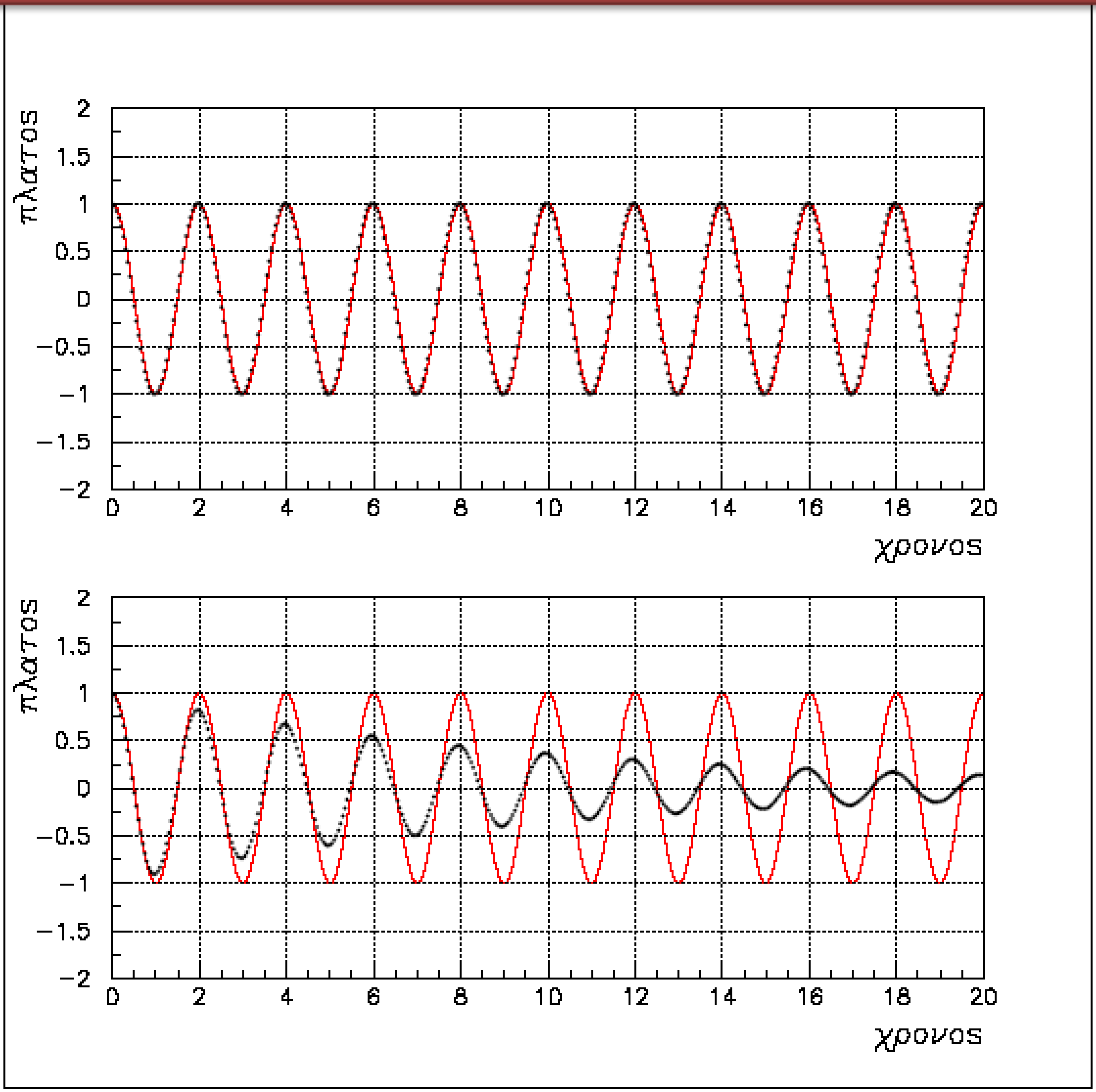
… όχι πια Euler

Μέθοδος Euler - Cromer

```
#include <stdio.h>
#include <math.h>
void elptosi()
{
    double y=1.;
    double v=0;
    double t=0;
    double pi=acos(-1.);
    double K=pi*pi;
    double h=0.05;

    for(int i=1;i<200;i++)
       {
         y = y+h*v;
         v = v-h*K*y;
         t=t+dt;

       }
}
```







Αρμονικός Ταλαντωτής

$d^2x/dt^2 = -\omega^2 x$

Όχι πια Euler

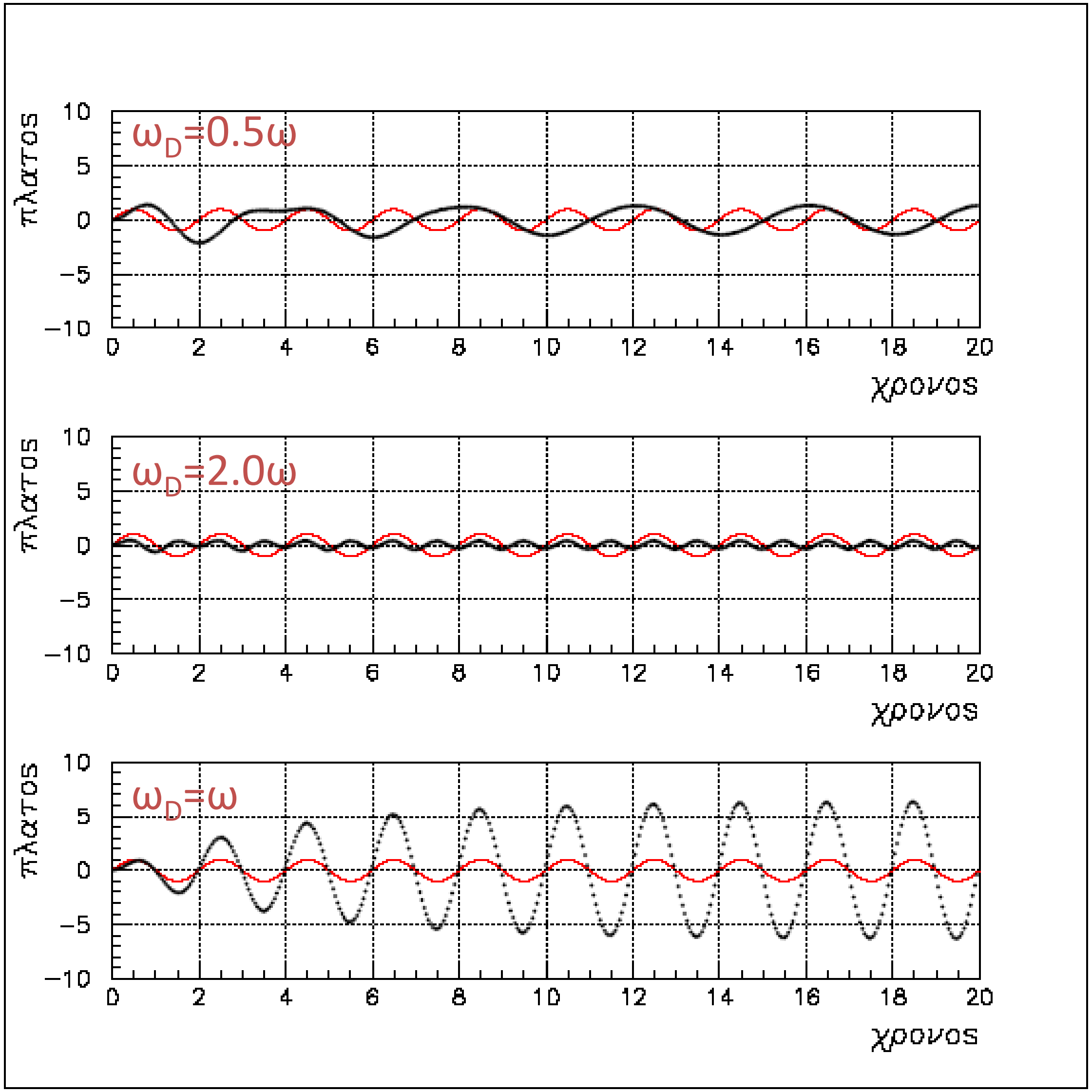
```
v[ih]= v[ih-1]-pi*pi*dt*x[ih-1];
x[ih]= x[ih-1]+dt*v[ih];
t[ih] = t[ih-1]+dt;
```

Αρμονικός Ταλαντωτής
με απόσβεση

$d^2x/dt^2 = -\omega^2 x - C dx/dt$

```
v[ih]= v[ih-1]-pi*pi*dt*x[ih-1]
        - C*dt*v[ih-1];
x[ih]= x[ih-1]+dt*v[ih];
t[ih] = t[ih-1]+dt;
```

$$x(t) = x_0 e^{-Ct/2} \sin\left(\sqrt{\omega^2 - C^2/4}\, t + \varphi \right)$$



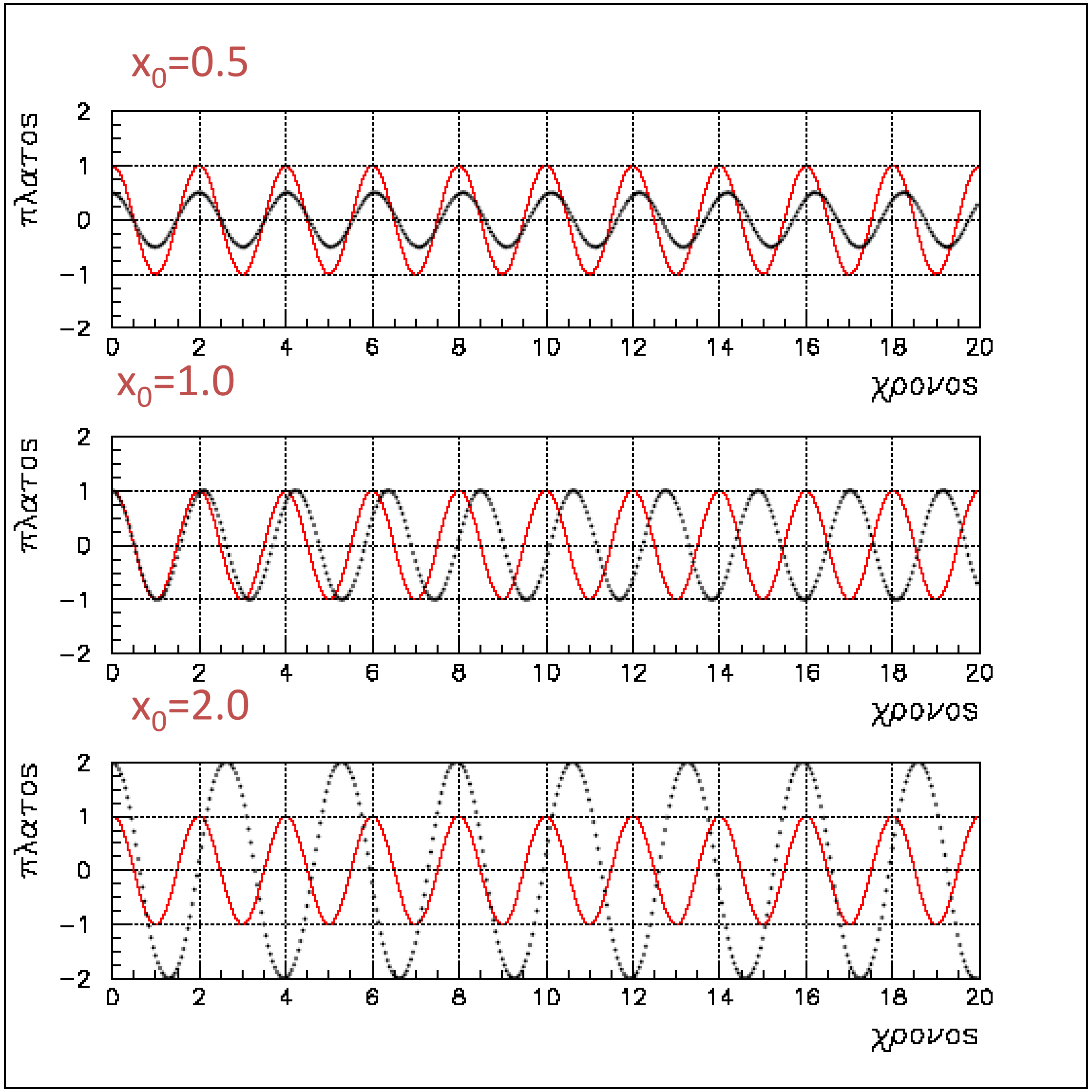
Αρμονικός Ταλαντωτής με απόσβεση και εξωτερική δύναμη

$d^2x/dt^2 = -\omega^2 x - C dx/dt + D\cos(\omega_D t)$

```
v[ih]= v[ih-1]-pi*pi*h*x[ih-1]
          - C*dt*v[ih-1]
          +D*dt*cos(ωD*t[ih-1]);
x[ih]= x[ih-1]+dt*v[ih];
t[ih] = t[ih-1]+dt;
```

$$x(t) = x_0 \sin(\omega_D + \varphi)$$

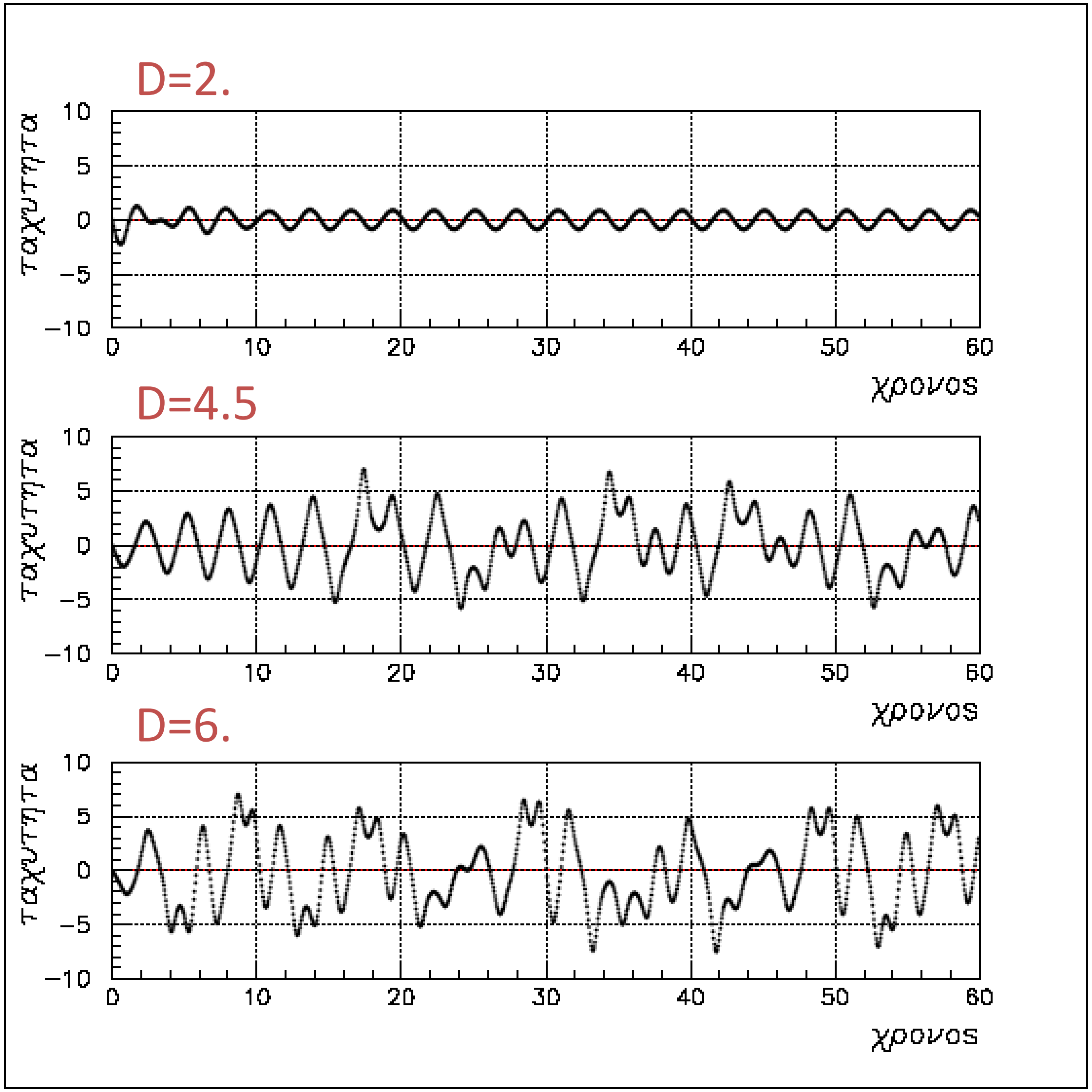
$$x_0 = \frac{D}{\sqrt{(\omega^2 - \omega_D^2)^2 + (C\omega_D)^2}}$$



Μη αρμονικός ταλαντωτής

$d^2x/dt^2 = -\omega^2 \sin(x)$

Η περίοδος εξαρτάται από το πλάτος



Μη αρμονικός ταλαντωτής με απόσβεση και εξωτερική δύναμη

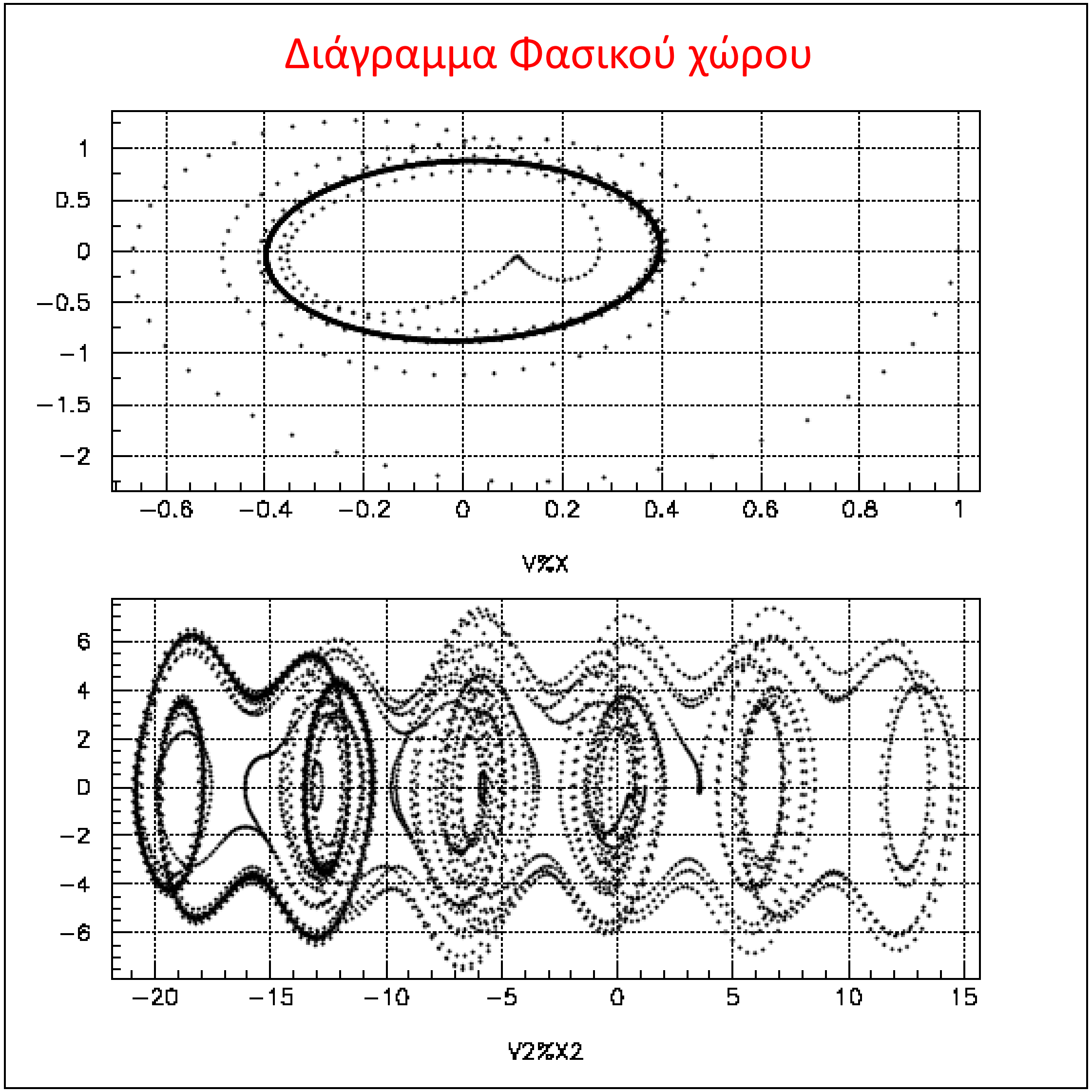
$d^2x/dt^2 = -\omega^2\sin(x) - Cdx/dt + D\cos(\omega_D t)$

$\omega = \pi$

$C = 0.5$

$\omega_D = 0.7\pi$

Επίλυση ανώτερης τάξης διαφορικών εξισώσεων - Παράδειγμα 2 συνέχεια



D=2.0

D=6.0

Το ηλιακό σύστημα

$$\frac{d^2x}{dt^2} = \frac{F_{Gx}}{m} = -\frac{GM}{r^2}\cos\theta$$

$$\frac{d^2y}{dt^2} = \frac{F_{Gy}}{m} = -\frac{GM}{r^2}\sin\theta$$

$$\frac{dv_x}{dt} = -\frac{GMx}{r^3}$$

$$\frac{dx}{dt} = v_x$$

$$\frac{dv_y}{dt} = -\frac{GMy}{r^3}$$

$$\frac{dy}{dt} = v_y$$

r (AU)
t (yr)

v = 2π
GM=v^2r
=$4\pi^2$ (AU^3/yr^2)

$$v_x^{i+1} = v_x^i - \frac{4\pi^2 x^i}{\left(r^i\right)^3}\delta t$$

$$x^{i+1} = x^i + v_x^{i+1}\delta t$$

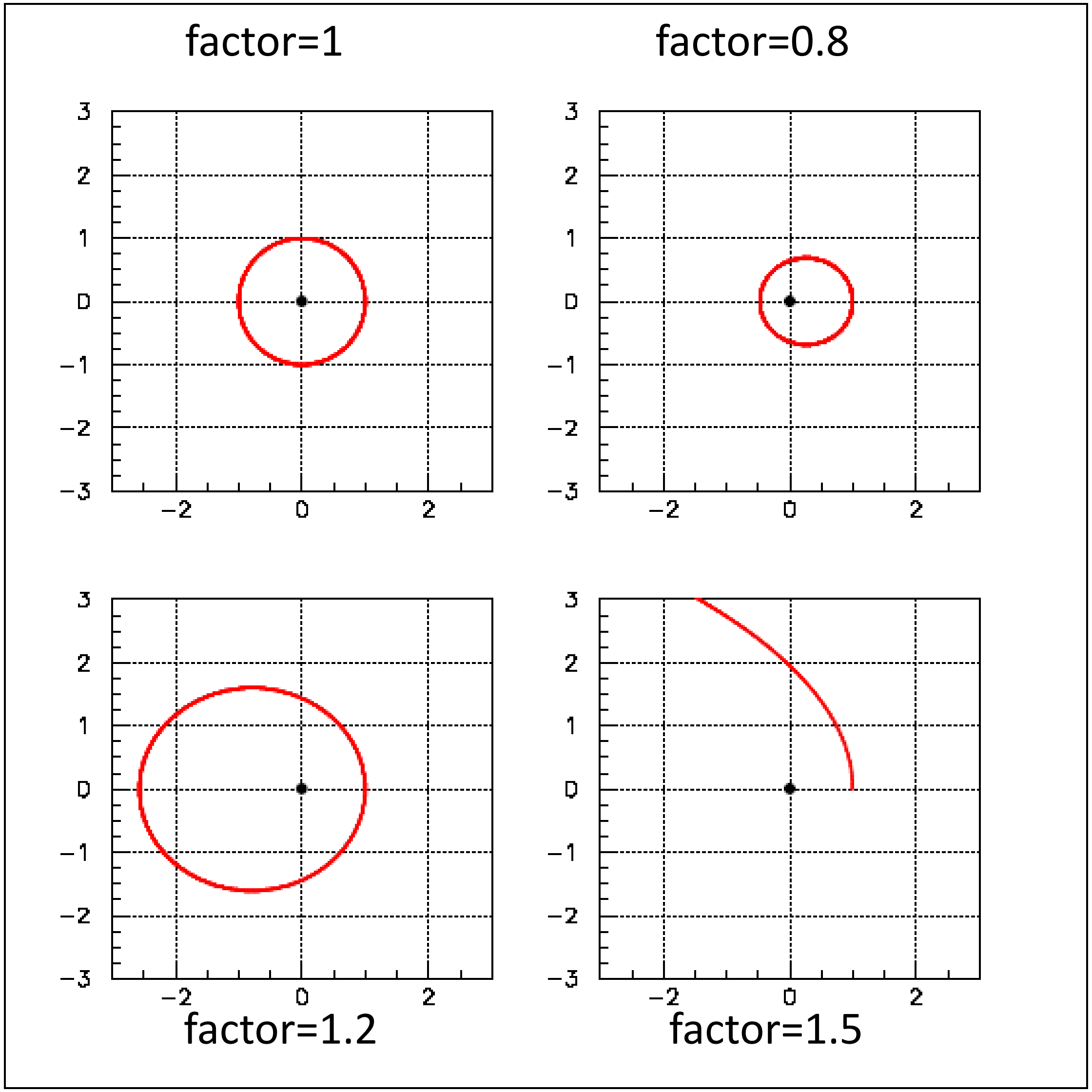
$$v_y^{i+1} = v_y^i - \frac{4\pi^2 y^i}{\left(r^i\right)^3}\delta t$$

$$y^{i+1} = y^i + v_y^{i+1}\delta t$$

Επίλυση ανώτερης τάξης διαφορικών εξισώσεων - Παράδειγμα 3

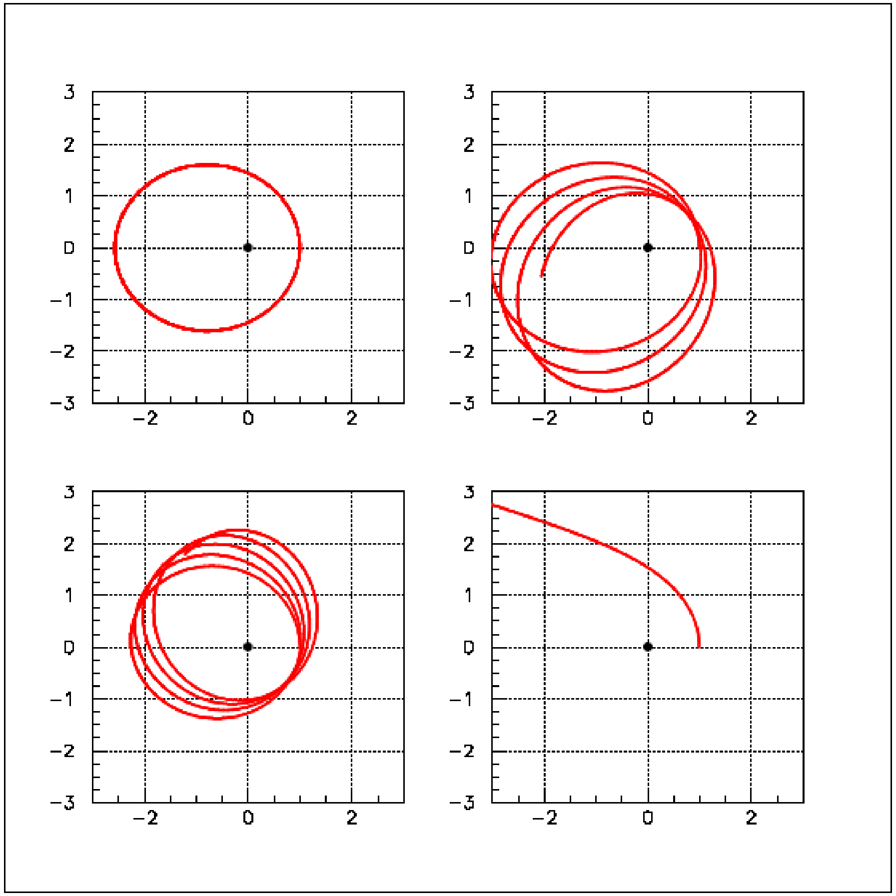
```
    double vini = factor*2*pi;
        double rini  = 1.;
        double ran  = 1.;
        rx[0] = rini*ran;
        ry[0] = rini*sqrt(1-ran*ran);
        vx[0] = vini*sqrt(1-ran*ran);
        vy[0] = vini*ran;
        t[0]=0.;
        for(int i=1; i<N;i++)
            {
               double rr=sqrt(rx[i-1]*rx[i-1]+ry[i-1]*ry[i-1]);
               vx[i]=vx[i-1]-4*pi*pi*rx[i-1]*dt/(rr*rr*rr);
               vy[i]=vy[i-1]-4*pi*pi*ry[i-1]*dt/(rr*rr*rr);
               rx[i]=rx[i-1]+dt*vx[i];
               ry[i]=ry[i-1]+dt*vy[i];
               t[i]=t[i-1]+dt;
             }
```

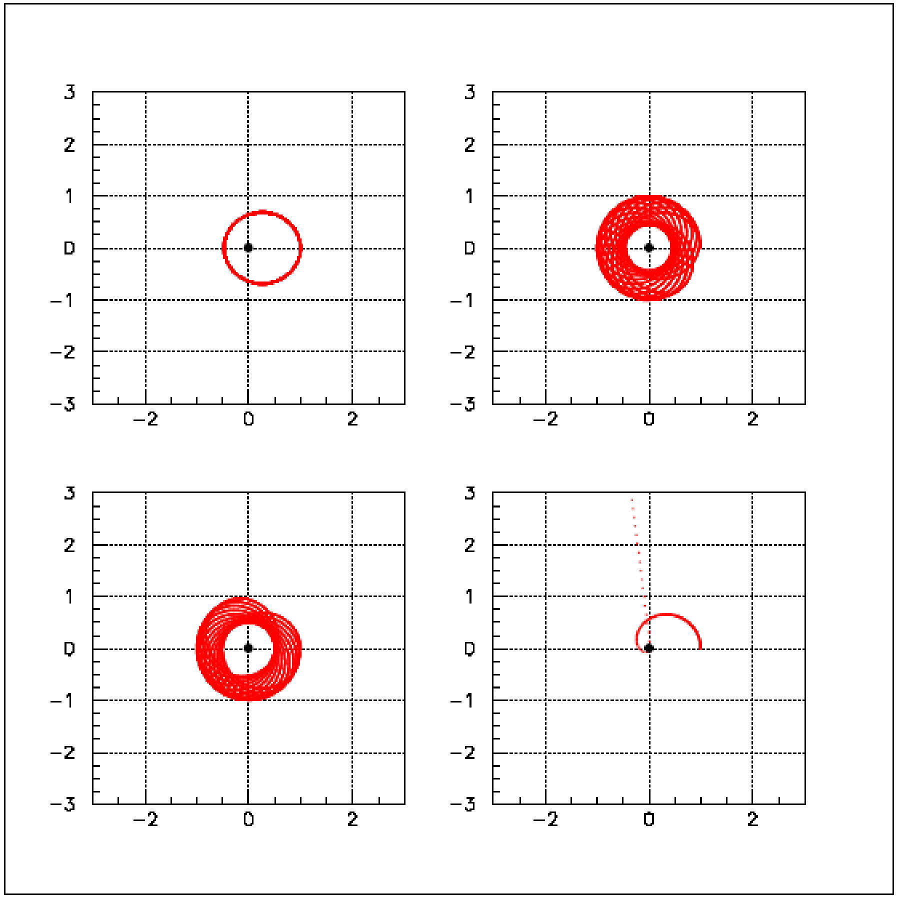
Επίλυση ανώτερης τάξης διαφορικών εξισώσεων - Παράδειγμα 3



Επίλυση ανώτερης τάξης διαφορικών εξισώσεων - Παράδειγμα 3

Αν δεν ίσχυε F~1/r^2





Προσέγγιση 1ης παραγώγου

$$f'(x_0) = \lim_{h\to 0} \frac{f(x_0+\tau)-f(x_0)}{\tau} - \frac{\tau}{2} f''(\xi)$$

Forward time

$$f'(x_0) = \lim_{h\to 0} \frac{f(x_0+\tau)-f(x_0-\tau)}{2\tau} - \frac{\tau^2}{6} f'''(\xi)$$

Centered time

Προσέγγιση 2ης παραγώγου

$$f''(x_0) = \lim_{h\to 0} \frac{f(x_0+\tau)+f(x_0-\tau)-2f(x_0)}{\tau^2} - \frac{\tau^2}{12} f^{(4)}(\xi)$$

Επίλυση ανώτερης τάξης διαφορικών εξισώσεων – Μέθοδος leap-frog

Μέθοδος Euler

$$f'(x_0) = \lim_{h\to 0} \frac{f(x_0+\tau) - f(x_0)}{\tau} - \frac{\tau}{2} f''(\xi)$$

$$\begin{aligned} y(t+\tau) &= \quad y(t) + \tau v(t) + \propto(\tau^2) \\ v(t+\tau) &= \quad v(t) + \tau f(t, y, v) + \propto(\tau^2) \end{aligned}$$

Μέθοδος leap-frog

$$f'(x_0) = \lim_{h\to 0} \frac{f(x_0+\tau) - f(x_0-\tau)}{2\tau} - \frac{\tau^2}{6} f'''(\xi)$$

$$\boxed{\begin{aligned} y(t+2\tau) &= \quad y(t) + 2\tau \cdot v(t+\tau) + \propto(\tau^3) \\ v(t+\tau) &= \quad v(t-\tau) + 2\tau \cdot f(t, y, v) + \propto(\tau^3) \end{aligned}}$$

Μειονέκτημα:
Χρειάζεται μια επιπλέον αρχική τιμή *v(t-τ)*

Μέθοδος Euler

$$f'(x_0) = \lim_{h\to 0} \frac{f(x_0+\tau) - f(x_0)}{\tau} - \frac{\tau}{2} f''(\xi)$$

$$\begin{aligned} y(t+\tau) &= y(t) + \tau v(t) + \propto(\tau^2) \\ v(t+\tau) &= v(t) + \tau f(t,y,v) + \propto(\tau^2) \end{aligned}$$

Μέθοδος Verlet

$$f''(x_0) = \lim_{h\to 0} \frac{f(x_0+\tau) + f(x_0-\tau) - 2f(x_0)}{\tau^2} - \frac{\tau^2}{12} f^{(4)}(\xi)$$

$$\boxed{\begin{aligned} y(t+\tau) &= 2y(t) - y(t-\tau) + \tau^2 \cdot f(t,y,v) + \propto(\tau^4) \\ v(t) &= \frac{y(t+\tau) - y(t-\tau)}{2\tau} + \propto(\tau^2) \end{aligned}}$$

Μειονέκτημα:
Χρειάζεται μια επιπλέον αρχική τιμή *y(t-τ)*

Για τη μέθοδο Euler:

$w^{n+1} = w^n + f(w^n,t^n)\tau$

Εισάγοντας τοπικό σφάλμα:

$w^{n+1} + \varepsilon^{n+1} = w^n + \varepsilon^n + \tau f(w^n+\varepsilon^n,t^n)$

Κάνοντας το ανάπτυγμα Taylor:

$f(w^n+\varepsilon^n,t^n) = f(w^n,t^n) + (\partial f/\partial w)\varepsilon^n + \propto(\varepsilon^{2n})$

Προκύπτει:

$\varepsilon^{n+1} = \varepsilon^n + (\partial f/\partial w)\tau\varepsilon^n$

Παράγοντας Ενίσχυσης

$$\boldsymbol{\xi = 1 + (\partial f/\partial w)\tau}$$

Για τη μέθοδο Euler:

$w^{n+1} = w^n + f(w^n,t^n)\tau$

Εισάγοντας τοπικό σφάλμα:

$w^{n+1} + \varepsilon^{n+1} = w^n + \varepsilon^n + \tau f(w^n+\varepsilon^n,t^n)$

Κάνοντας το ανάπτυγμα Taylor:

$f(w^n+\varepsilon^n,t^n) = f(w^n,t^n) + (\partial f/\partial w)\varepsilon^n + \propto(\varepsilon^{2n})$

Προκύπτει:

$\varepsilon^{n+1} = \varepsilon^n + (\partial f/\partial w)\tau\varepsilon^n$

Παράγοντας Ενίσχυσης

$\boldsymbol{\xi=1+ (\partial f/\partial w)\tau}$ ⇒

Ευστάθεια :

$\boldsymbol{|\xi|\leq 1}$

Έστω για παράδειγμα:

$dy/dt = -y/t_0 \Rightarrow \partial f/\partial w = -1/t_0 \Rightarrow$ $\boldsymbol{\xi = 1 - \tau / t_0}$

Η ευστάθεια επιτυγχάνεται λοιπόν για : $\boldsymbol{\tau \leq 2t_0}$

$$w_{i+1} = w_i - \frac{\tau}{t_0} w_i$$

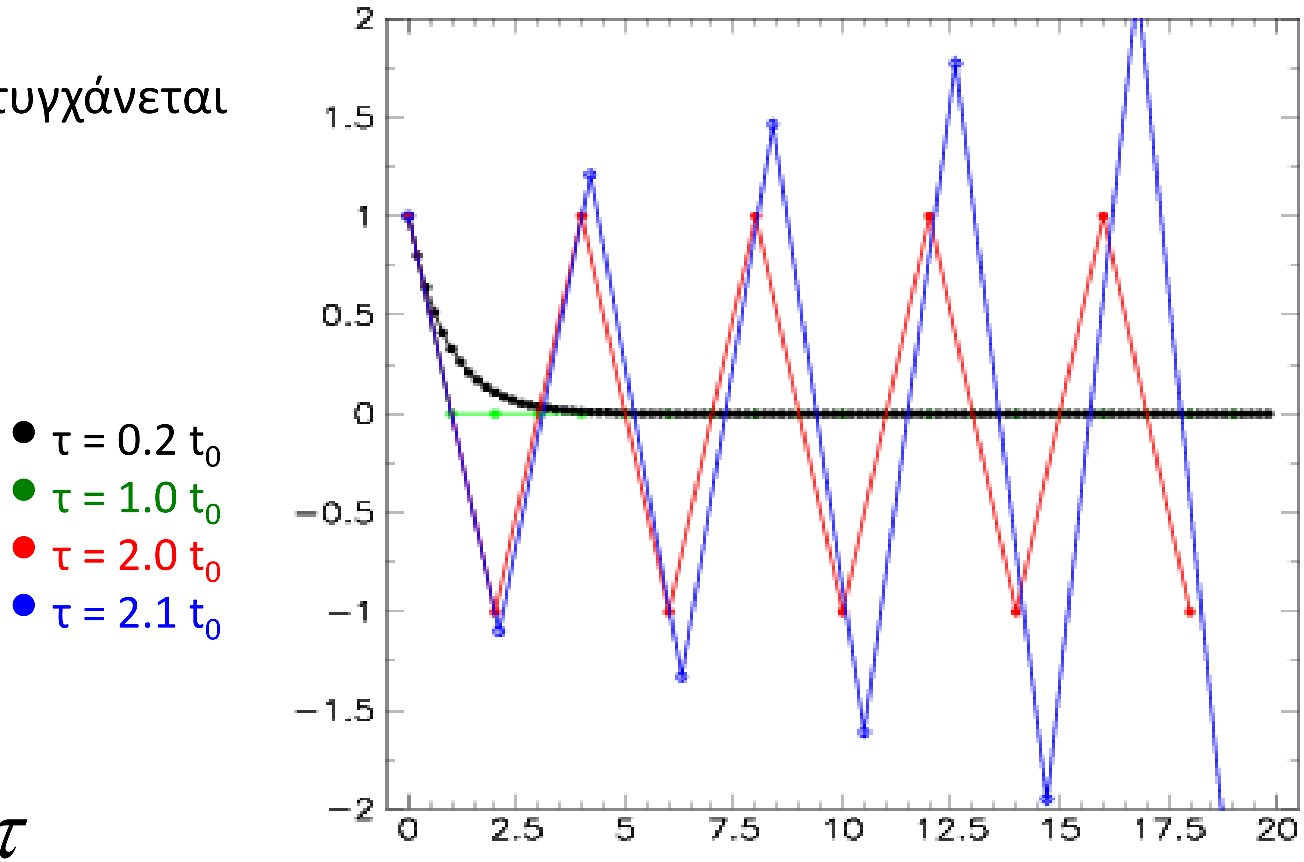
74

Έστω για παράδειγμα:

$dy/dt = -y/t_0 \Rightarrow \partial f/\partial w = -1/t_0 \Rightarrow \quad \xi=1-\tau/\, t_0$

Η ευστάθεια επιτυγχάνεται

για : $\tau \leq 2t_0$



$$w_{i+1} = w_i - \frac{\tau}{t_0} w_i$$

Επίλυση διαφορικών εξισώσεων – Ευστάθεια

Αρμονική κίνηση:

$d^2x/dt^2 + \omega^2 x = 0 \Rightarrow$ $\quad dx/dt - \omega v = 0$

$dv/dt + \omega x = 0$

Θέτοντας $u = x + iv \Rightarrow du/dt + i\omega u = 0 \quad \Rightarrow \quad \boldsymbol{\xi = 1 - i\omega\tau}$

$|\xi|^2 = \xi\xi^* = 1 + \omega^2\tau^2$

Η μέθοδος Euler δεν μπορεί να είναι ευσταθής σε ταλαντωτικές Δ.Ε. για **καμιά** τιμή χρονικού βήματος

Α Σειρά ασκήσεων στην Υπολογιστική Φυσική

Α1.

Κατασκευάστε υπολογιστικούς κώδικες που να υπολογίζουν ολοκληρώματα σε 3 διαστάσεις σε (i) καρτεσιανές και (ii) σε πολικές συντεταγμένες. Χρησιμοποιείστε μια μέθοδο Gauss.

(α) Υπολογίστε τη μάζα και το κέντρο βάρους κύβου μοναδιαίας ακμής και πυκνότητας $\rho(x,y,z)= 1+x^2+2y^2+3z^2$ με ακρίβεια 1‰. Θεωρήστε ότι η αρχή των συντεταγμένων είναι το σημείο (0,0,0).

(β) Υπολογίστε με ακρίβεια 1‰, τη ροπή αδράνειας σφαίρας μοναδιαίας ακτίνας και πυκνότητας $\rho=1+\exp(r)$ ως προς άξονα που περνά από το κέντρο της σφαίρας.

Α2.

Αποδείξτε υπολογιστικά την ισχύ των τριών νόμων του Kepler.

(α) Κάθε πλανήτης διαγράφει ελλειπτική τροχιά με τον ήλιο να βρίσκεται σε μια από τις εστίες.

(β) Η ευθεία, που ενώνει κάθε πλανήτη με τον ήλιο, σαρώνει ίσες επιφάνειες σε ίσους χρόνους.

(γ) Αν T είναι η περίοδος και A ο μεγάλος ημιάξονας της έλλειψης τότε η ποσότητα T^2/A^3 είναι σταθερά.

(Αγνοήστε την βαρυτική αλληλεπίδραση μεταξύ των πλανητών)

Α3.

Μελετείστε την κίνηση ενός φορτισμένου σωματιδίου με αρχική ορμή p_x

(α) Σε ομογενές μαγνητικό πεδίο B_z κάθετο στην ορμή του σωματιδίου

(β) Σε συνδυασμό ομογενούς μαγνητικού B_z και ομογενούς ηλεκτρικού E_x πεδίου.

(γ) Σε ομογενές μαγνητικό πεδίο B_z και αρχική ορμή του σωματιδίου (p_x,p_y,p_z)

Α4.

Επιλύστε τη διαφορική εξίσωση ***y’’-y’-2y=0*** στο χρονικό διάστημα [0.,10.] με αρχικές τιμές

(α) y(0)=1., y’(0)=-1.

(β) y(0)=1., y’(0)=-0.999 και σχολιάστε το αποτέλεσμα