Computational Physics

Ch 06 - Solutions to Differential Equations

Korea University
Eunil Won

Numerical Solution to Differential Equations

- This is a completely new subject

: I promised to cover this topic in the beginning of the class. Let us start with an easy example.

- Radioactive decay

If $N_U(t)$ is the number of uranium nuclei at time t, the behavior of the number is governed by the differential equation

$$\frac{dN_U}{dt} = -\frac{N_U}{\tau}$$

The solution to this differential equation is well known to be

$$N_U(t) = N_U(0)e^{-t/\tau}$$

Numerical Solution to Differential Equations

- Radioactive decay

The analytic solution is known for this problem. But let's discuss how to solve this numerically.

From the definition of a derivative, we know that

$$\frac{dN_U}{dt} = \lim_{\Delta t \to 0} \frac{N_U(t+\Delta t) - N_U(t)}{\Delta t} \approx \frac{N_U(t+\Delta t) - N_U(t)}{\Delta t}$$

so $$N_U(t+\Delta t) \approx N_U(t) + \frac{dN_U}{dt}\Delta t$$

For the radioactive decay problem, using the original differential equation, we get

$$N_U(t+\Delta t) \approx N_U(t) - \frac{N_U(t)}{\tau}\Delta t$$

Therefore, we can "estimate" N_U at later time from the present time numerically.

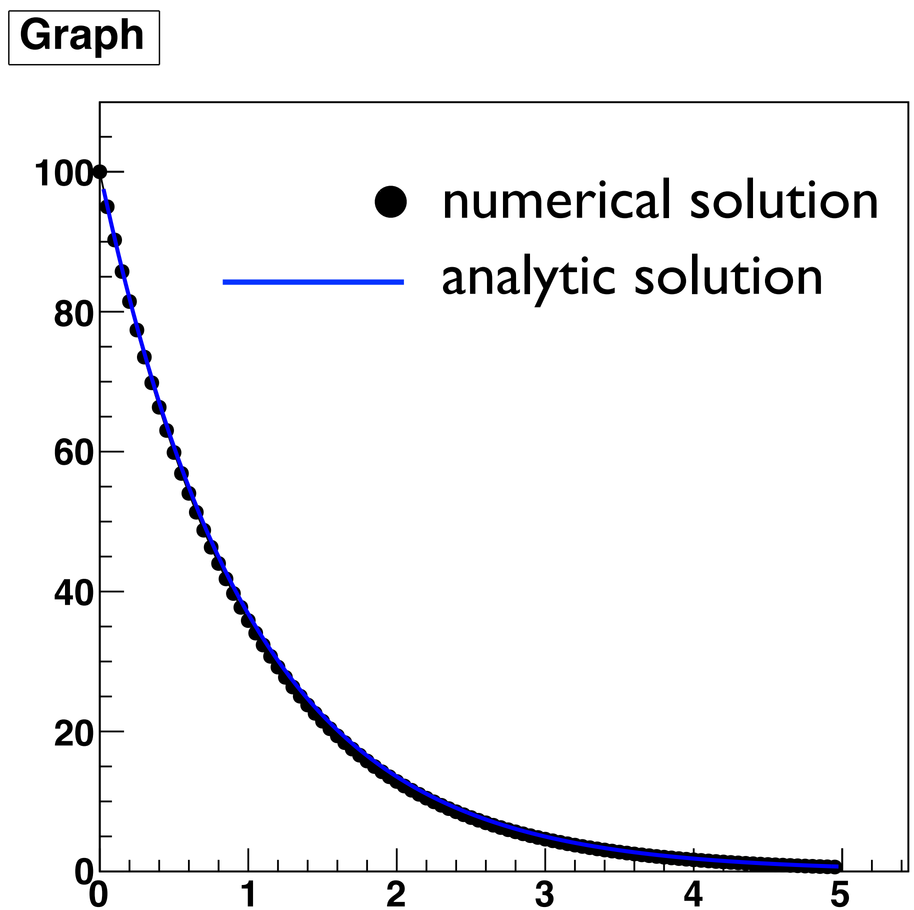
This type of calculating N_U is called *Euler method* (will discuss more)

Numerical Solution to Differential Equations

- Radioactive decay

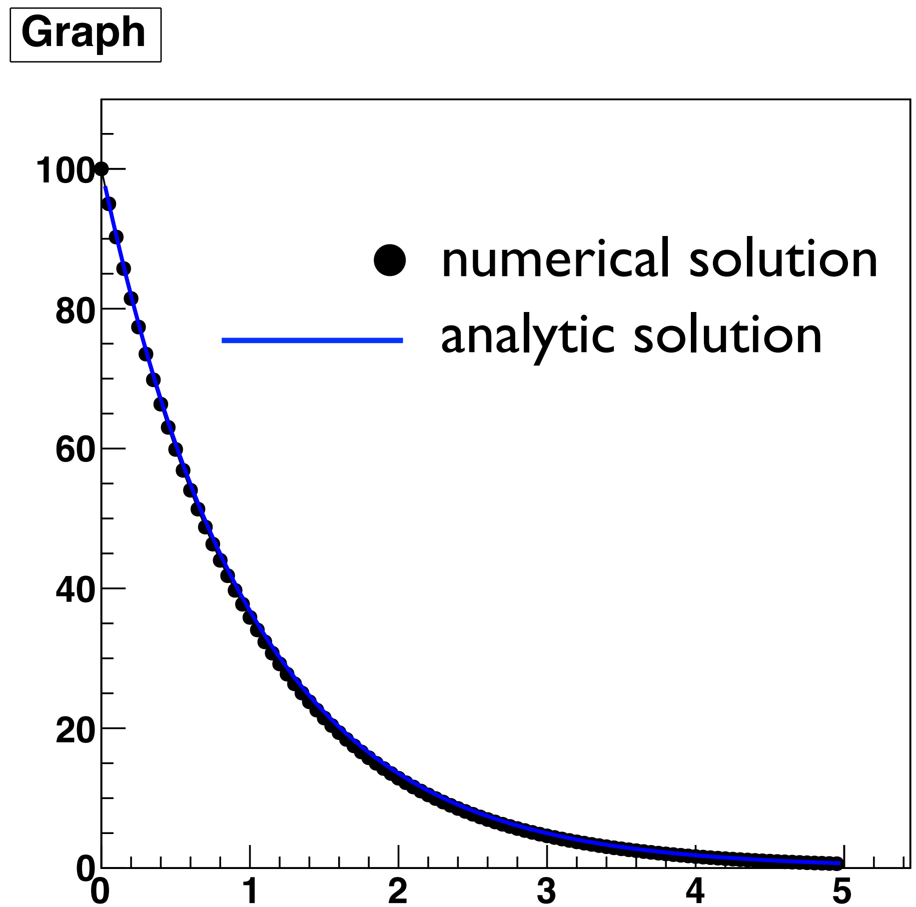
Let's compare the numerical solution and the analytic solution for this particular case.

$$
\begin{aligned}
N_{\text{spacing}} &= 100 \\
dt &= 0.05 \text{ sec} \\
\tau &= 1 \text{ sec} \\
N_U(0) &= 100
\end{aligned}
$$



```
//
// E. Won (eunil@hep.korea.ac.kr)
//

const static Int_t Nspace = 100;
#define dt      0.05   // second
#define tau     1.0    // second
#define N0      100

Double_t N[Nspace];
Double_t t[Nspace];

Double_t f1(Double_t *x, Double_t *par)
{
  return N0 * TMath::Exp(-x[0]/tau);
}
void decay()
{
  gROOT->SetStyle("Plain");
  gROOT->ForceStyle();

  N[0] = N0;
  for (Int_t i=0; i<Nspace-1; i++)
  {
    t[i+1] = t[i] + dt;
    N[i+1] = N[i] - (N[i] / tau) * dt;
  }

  TCanvas* c1 = new TCanvas("c1","",10,10,500,500);
  c1->cd();

  g1 = new TGraph(Nspace,t,N);
  g1->SetMarkerSize(1.5);
  g1->SetMarkerStyle(20);
  g1->Draw("ACP");

  TF1 *func = new TF1("func",f1,0.0,Nspace*dt,1);
  func->SetLineColor(kBlue);
  func->Draw("SAME");

}
```

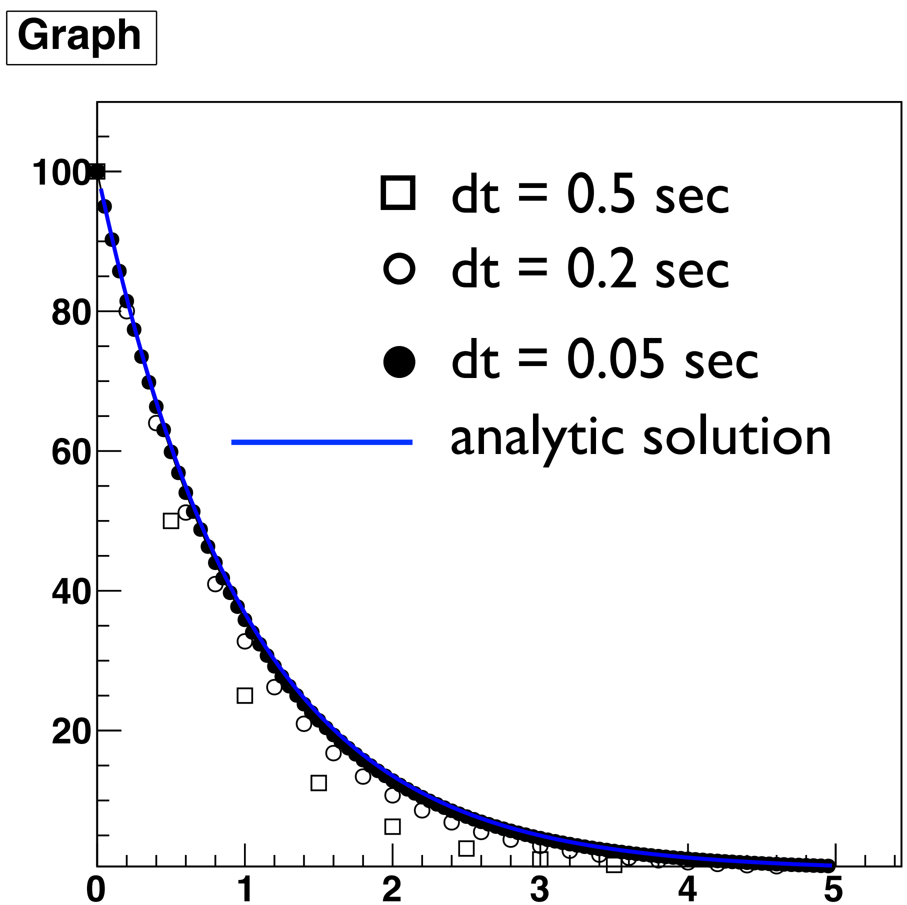
- Radioactive decay: source code

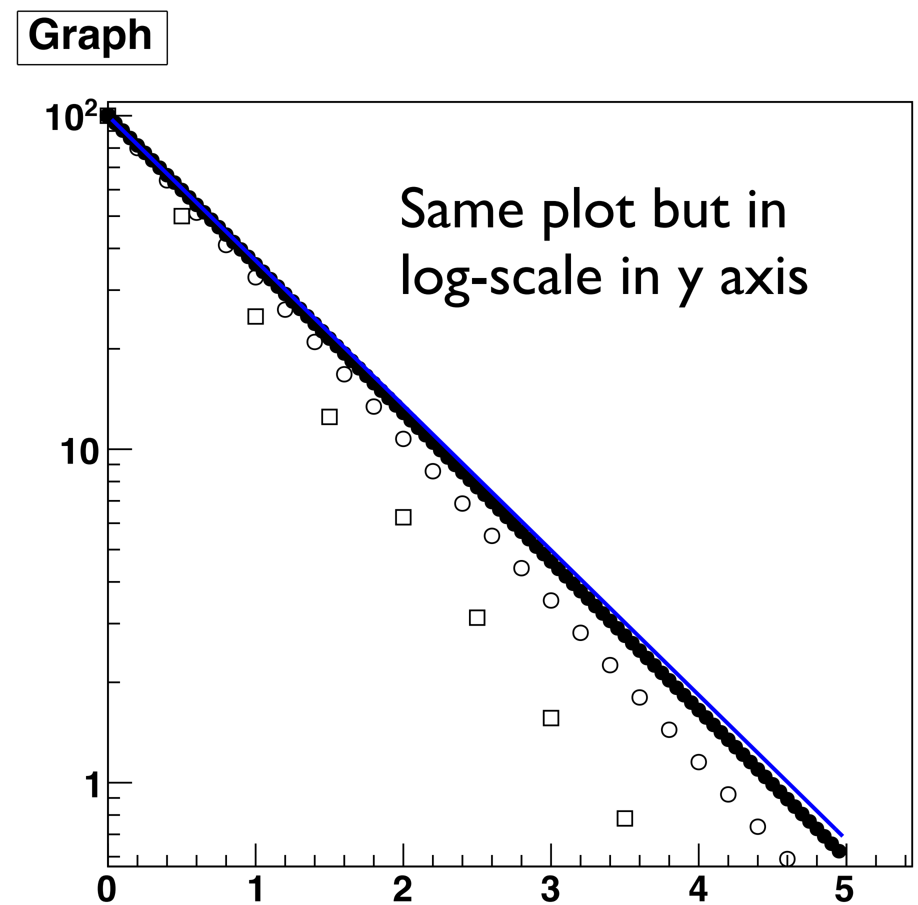


Numerical Solution to Differential Equations

- Radioactive decay

What happens if you try with dt = 0.2 or 0.5 sec?





First-order, ordinary differential equations

- Let's consider the following the 1st order differential equation of the form

$$\frac{d}{dt}x(t) = f(x,t) \quad \text{with the initial condition that} \quad x(0) = x_0$$

The Taylor series expansion of *x* around *t* gives

$$x(t+\Delta t) = x(t) + \frac{dx}{dt}\Delta t + \frac{1}{2}\frac{d^2x}{dt^2}(\Delta t)^2 + \dots$$

Thus, assuming a reasonably smooth function of *x(t)* and a small interval Δ*t*, we can propagate *x(t)* to x(t+Δt) to any accuracy desired as long as we know the derivatives of *x(t)*.

The Euler method introduced before corresponds to dropping terms of order $(\Delta t)^2$ and higher. If we are somehow given $x(t_i)$, the exact value of *x* at time $t_i = i\Delta t$. Propagation forward with the Euler method gives

$$x(t_{i+1}) \approx x(t_i) + f(x(t_i), t_i)\Delta t$$

If we want to calculate *x(b)* for some fixed value of *t=b*, starting from *t=0*, the Euler propagation step must be repeated $N = b/\Delta t$ times.

First-order, ordinary differential equations

Since the error is of the order of $(\Delta t)^2$ at each step, the final estimate for $x(b)$ will have an error of order $N(\Delta t)^2 \approx (b/(\Delta t))(\Delta t)^2 \propto \Delta t$

The Euler method is said to be an approximation of zeroth order globally. One can reduce the actual error by reducing Δt, the error at $t=b$ is only reduced linearly as Δt (and computing requirements go up linearly)

The choice of Δt is very important
- smaller values of Δt will give smaller errors with greater computational cost.
- too large a value of Δt can make approximation unstable or meaningless.

ex) Nuclear decay example: in this case, the first Euler propagation step gives

$$x(\Delta t) \approx x_0 - (\Delta t/\tau)x_0$$

If one takes $\Delta t=\tau$, this approximation becomes $x(t) = 0$, and even worse, if $\Delta t>\tau$ is taken, the Euler result for $x(t)$ oscillates in sign.

First-order, ordinary differential equations

How can we improve situation?

- according to the mean value theorem, there is a value t_m in the interval $[t, t+\Delta t]$ such that the *exact* solution can be gotten while stopping at first order in Δt

$$x(t+\Delta t) = x(t) + \left.\frac{dx}{dt}\right|_{t_m} \Delta t \quad \text{(this relation is exact)}$$

- so, by estimating t_m and $dx/dt|_{t_m}$ intelligently, we can effectively construct approximations of higher order than the Euler method: a popular example is called Runge-Kutta methods. The most common second-order Runge-Kutta approximation is constructed by

$$x(t+\Delta t) = x(t) + f(x', t')\Delta t$$

where

$$x' = x(t) + \frac{1}{2} f(x(t), t)\Delta t$$

$$t' = t + \frac{1}{2}\Delta t$$

First-order, ordinary differential equations

In other words, the slope $dx/dt|_{t_m}$ is estimated as the value $f(x', t')$ where is the midpoint of the interval and x' is the Euler approximated value of x at t'.

It is easy to show that this approximation is indeed locally of second order and of first order globally. This means that the error in *x(b)* for fixed *b* can be reduced quadratically by reducing Δ*t*.

Can we improve further?

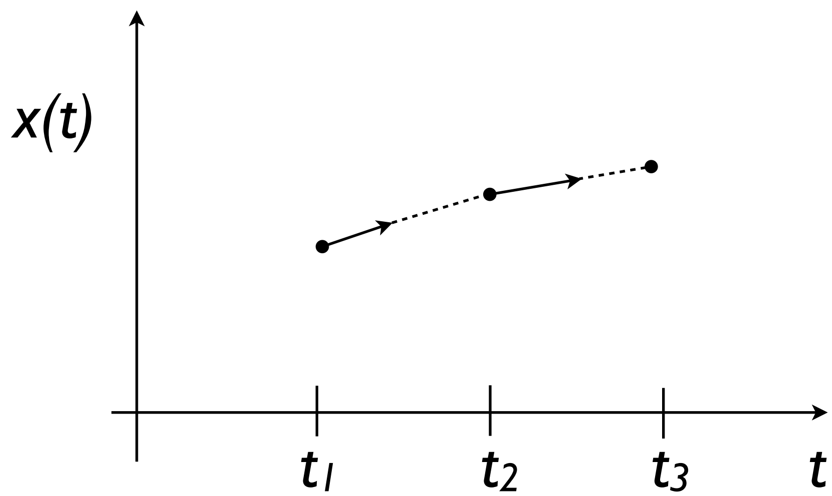
- yes, there are higher order Runge-Kutta approximations. Here we show the fourth-order Runge-Kutta method defined by

$$x(t+\Delta t) \equiv x(t) + \frac{1}{6}[f(x_1', t_1') + 2f(x_2', t_2') + 2f(x_3', t_3') + f(x_4', t_4')]\Delta t$$

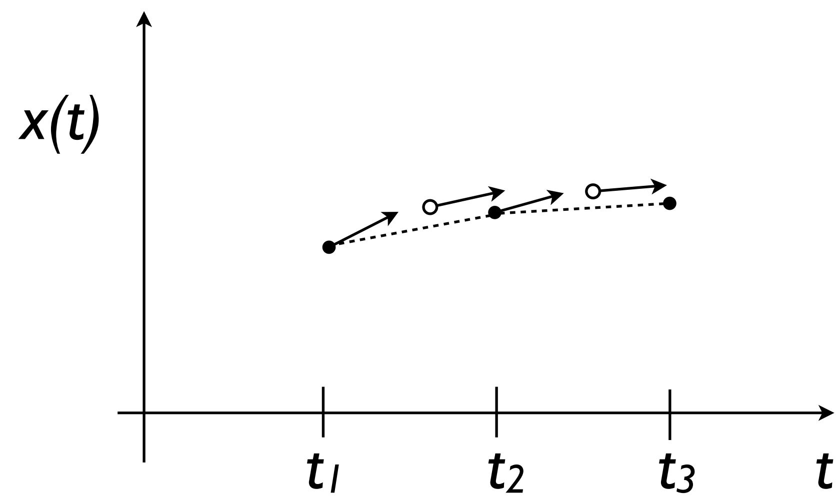
where

$$\begin{aligned}
x_1' &= x(t), & t_1' &= t \\
x_2' &= x(t) + \frac{1}{2}f(x_1', t_1')\Delta t, & t_2' &= t + \frac{1}{2}\Delta t \\
x_3' &= x(t) + \frac{1}{2}f(x_2', t_2')\Delta t, & t_3' &= t + \frac{1}{2}\Delta t \\
x_4' &= x(t) + f(x_3', t_3')\Delta t, & t_4' &= t + \Delta t
\end{aligned}$$

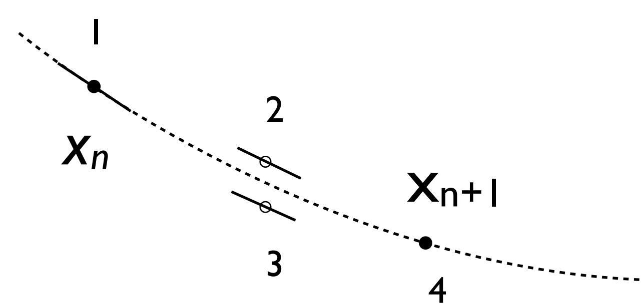
Graphical Interpretation of Euler’s method and RK



Euler’s method: the simplest (and least accurate) method



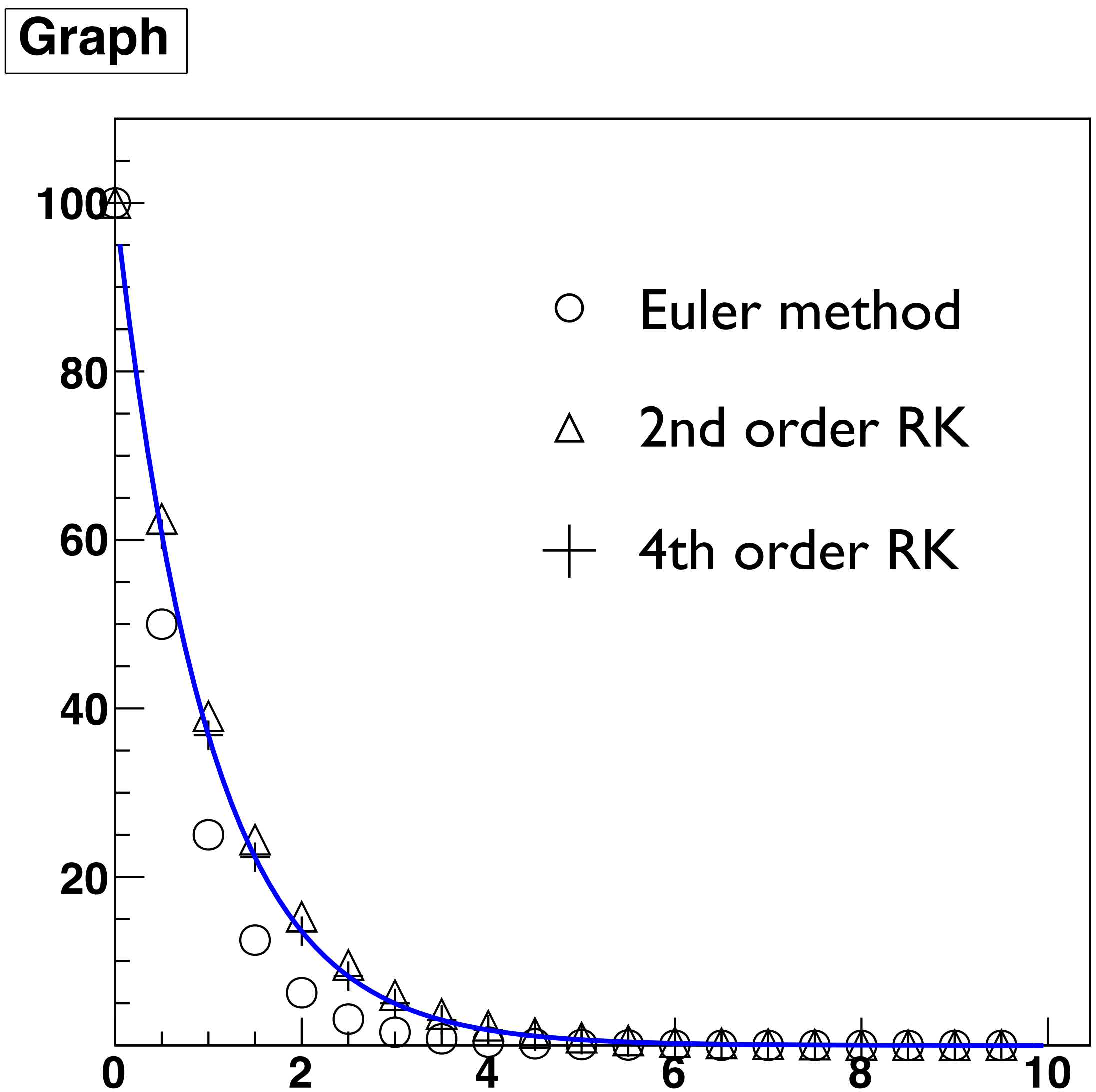
2nd order RK (or sometimes called midpoint method) by using the initial derivate at each step to find halfway across the interval.



4th order RK. In each step, the derivative is evaluated four times: once at the initial point, twice at trial midpoints, at once at a trial endpoint. From these derivatives the final function value is calculated. (Think about he equation carefully.)

First-order, ordinary differential equations

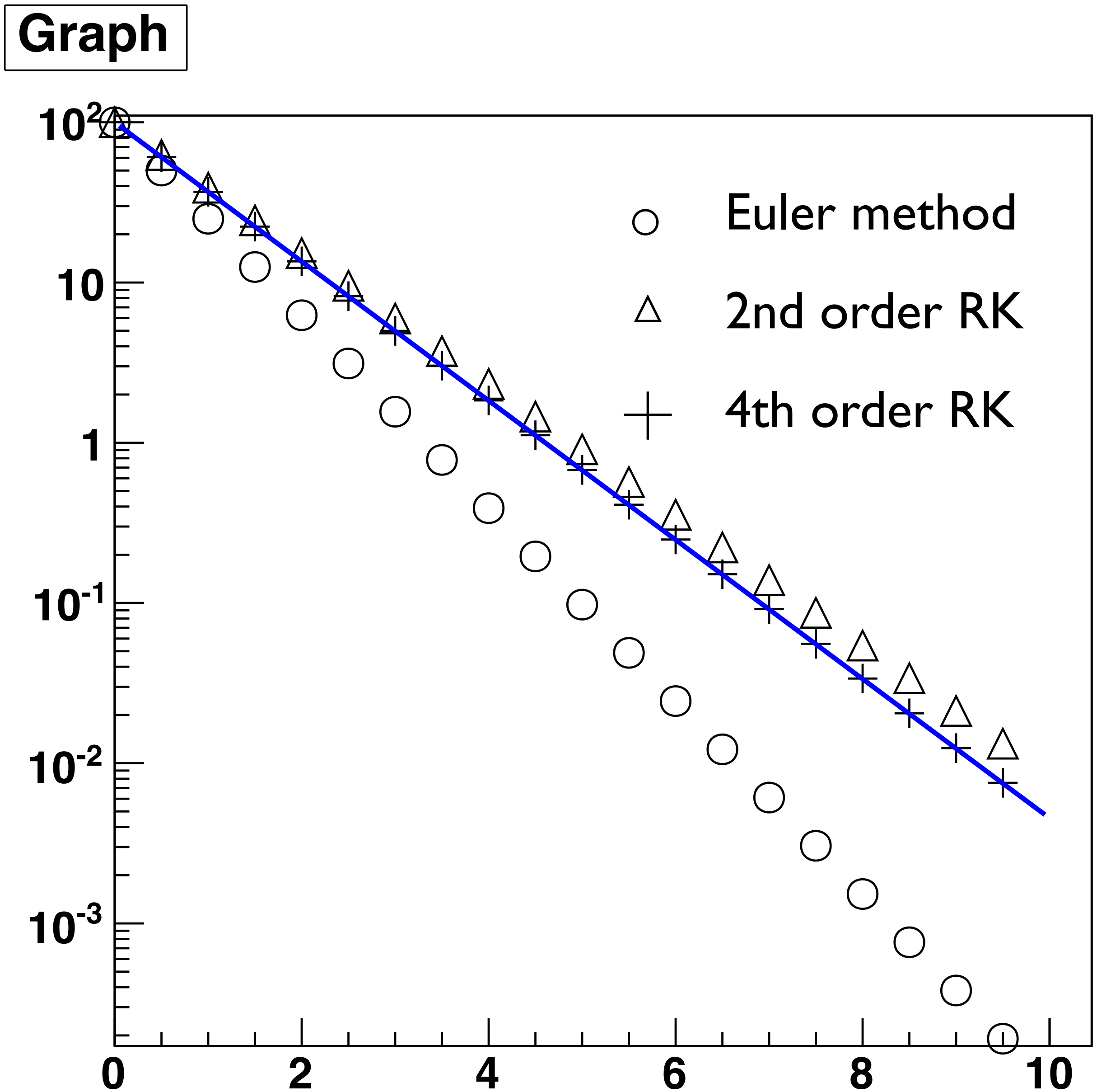
- Let us come back to our radioactive decay example: we want to compare Euler method, 2nd and 4th order Runge-Kutta (Δt = 0.5 s)



As we know, Euler method does not work with Δt = 0.5 s step.
- 2nd order is better than Euler method
- 4th order is better indeed than the 2nd order?

First-order, ordinary differential equations

- Let us take a look at the result again in log-scale



Now, it is clearer what is happening at later time
- 4th order is the best approximation

• Highly optimized code example of RK method (Numerical Recipes in C++, 2nd ed. by W. Press *et al.*, Cambridge)

The user needs to provide `derivs` externally
- f'(x,t) in our discussion

For my programming of the radioactive decay, I did not use this but wrote my own simpler one...

```
void rk4(VecDoub_I &y, VecDoub_I &dydx, const Doub x, const Doub h,
  VecDoub_O &yout, void derivs(const Doub, VecDoub_I &, VecDoub_O &))
{
  Int n=y.size();
  VecDoub dym(n),dyt(n),yt(n);
  Doub hh=h*0.5;
  Doub h6=h/6.0;
  Doub xh=x+hh;
  for (Int i=0;i<n;i++) yt[i]=y[i]+hh*dydx[i];
  derivs(xh,yt,dyt);
  for (Int i=0;i<n;i++) yt[i]=y[i]+hh*dyt[i];
  derivs(xh,yt,dym);
  for (Int i=0;i<n;i++) {
    yt[i]=y[i]+h*dym[i];
    dym[i] += dyt[i];
  }
  derivs(x+h,yt,dyt);
  for (Int i=0;i<n;i++)
    yout[i]=y[i]+h6*(dydx[i]+dyt[i]+2.0*dym[i]);
}
```

$$x(t+\Delta t) \equiv x(t) + \frac{1}{6}[f(x'_1,t'_1) + 2f(x'_2,t'_2) + 2f(x'_3,t'_3) + f(x'_4,t'_4)]\Delta t$$

$$x'_1 = x(t), \qquad t'_1 = t$$

$$x'_2 = x(t) + \frac{1}{2}f(x'_1,t'_1)\Delta t, \qquad t'_2 = t + \frac{1}{2}\Delta t$$

$$x'_3 = x(t) + \frac{1}{2}f(x'_2,t'_2)\Delta t, \qquad t'_3 = t + \frac{1}{2}\Delta t$$

$$x'_4 = x(t) + f(x'_3,t'_3)\Delta t, \qquad t'_4 = t + \Delta t$$

Second-order, ordinary differential equations

Most of all the differential equations that appear in our textbooks are indeed second order, ordinary (or partial) differential differential equations.

$$\vec{F} = m\frac{d^2\vec{r}}{dt^2}$$

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0} \qquad \nabla \cdot \mathbf{B} = 0$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \qquad \nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \mu_0 \epsilon_0 \frac{\partial \mathbf{E}}{\partial t}$$

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r}, t) = -\frac{\hbar^2}{2m}\nabla^2\Psi(\mathbf{r}, t) + V(\mathbf{r})\Psi(\mathbf{r}, t)$$

Second-order, ordinary differential equations

So, let us discuss how to compute numerical solutions of second order differential equations. For the following 2nd order ordinary differential equation (ODE)

$$\frac{d^2y}{dx^2} + q(x)\frac{dy}{dx} = r(x)$$

one can always write as two first-order equations with the new variable *z*

$$\begin{aligned} \frac{dy}{dx} &= z(x) \\ \frac{dz}{dx} &= r(x) - q(x)z(x) \end{aligned}$$

so the general problem in ordinary differential equations is reduced to the study of a set of *N* coupled first-order differential equations for the functions *y, i=0, 1,..., N-1*

$$\frac{dy_i(x)}{dx} = f_i(x, y_0, \ldots, y_{N-1}), \qquad i = 0, \ldots, N-1$$

where the functions f_i on the right-hand side are known.

One can use RK method for 2nd order ODEs!

Second-order, ordinary differential equations

- Let's consider the following differential equation (2nd order)

$$\frac{d^2y}{dt^2} + Ay = 0$$

This is a simple harmonic oscillator problem (mass is set to unity, and *A* is a constant). Now from the previous argument, we can do something like

$$\frac{dy}{dt} = v$$
$$\frac{dv}{dt} = -Ay$$

Euler method gives

$$y(t_{i+1}) \approx y(t_i) + v(t_i)\Delta t$$
$$v(t_{i+1}) \approx v(t_i) - Ay(t_i)\Delta t$$

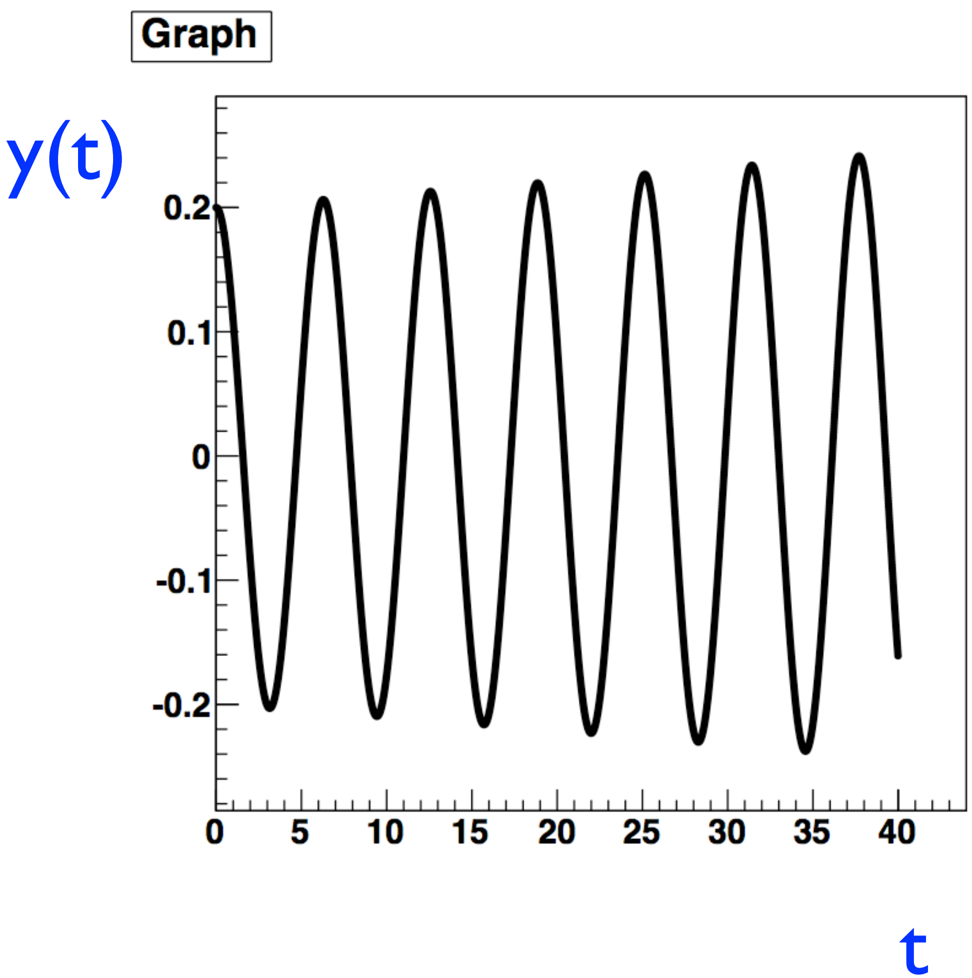
So the simple harmonic oscillator problem becomes a two first-order differential equations, as we pointed out earlier. We can also calculate the "energy" term like:

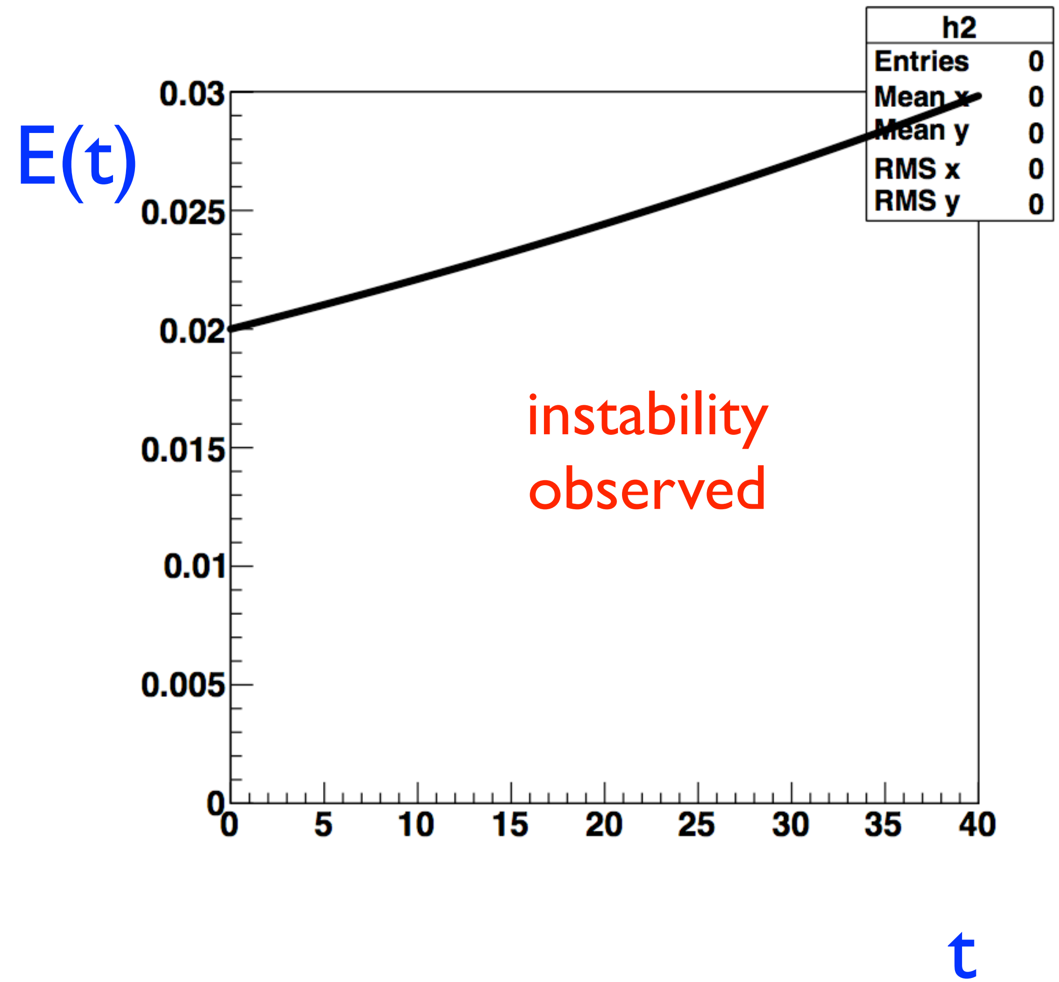
$$E = \frac{1}{2}v^2 + \frac{1}{2}Ay^2$$

Later we can check
if this "energy" term
is stable or not

Second-order, ordinary differential equations

- Let's solve the simple harmonic oscillator problem with Δt=0.01, A=1, y(0)=0.2 and v(0) = 0





The oscillation amplitude is unstable and the energy is increasing as a function of time due to imperfect numerical computation (Euler method)

Second-order, ordinary differential equations

- The 2nd order RK method for the simple harmonic oscillator problem should be

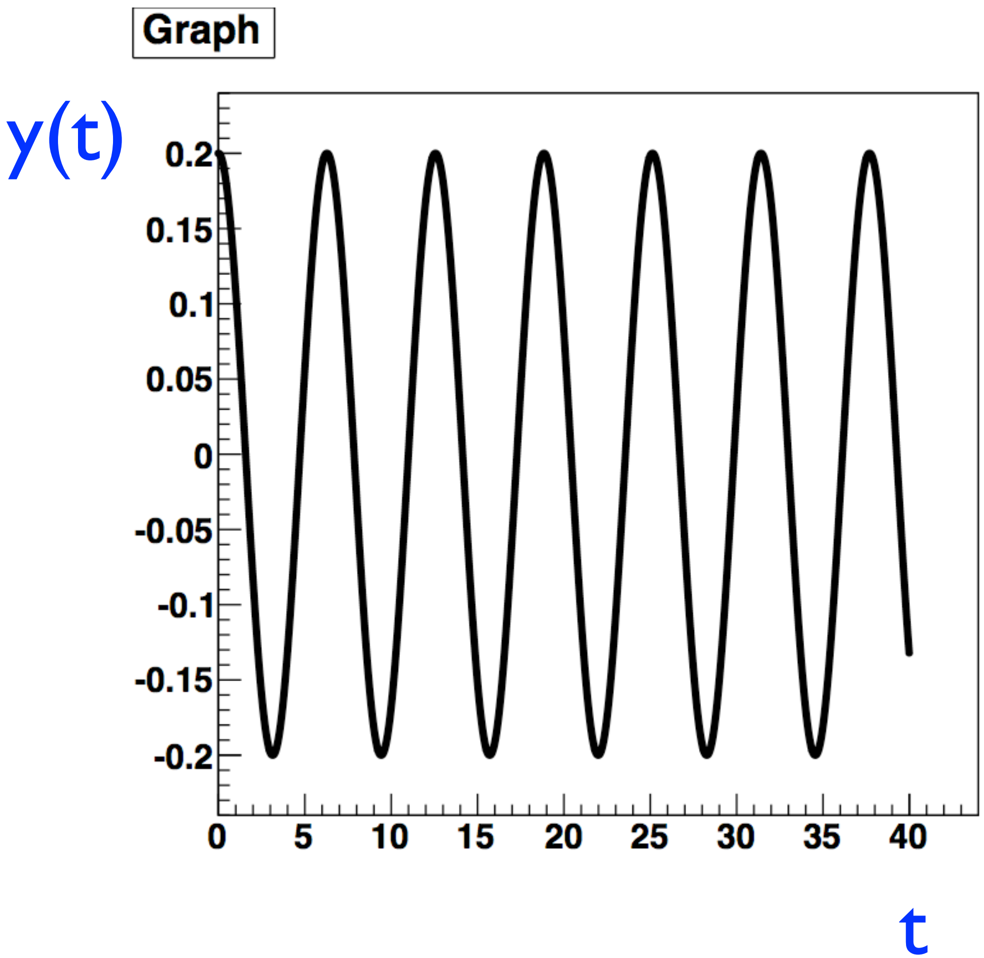
$$y_{i+1} = y_i + v'\Delta t$$
$$v_{i+1} = v_i - Ay'\Delta t$$

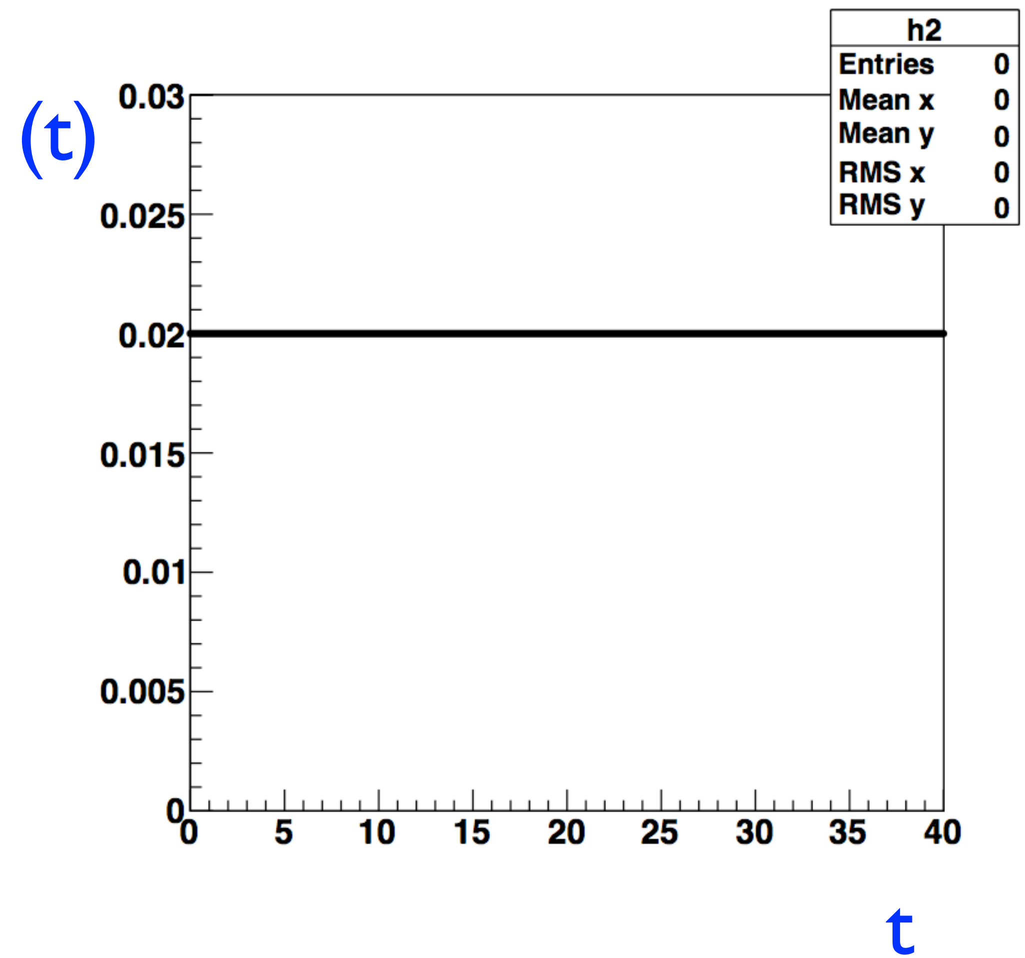
where

$$y' = y_i + \frac{1}{2}v_i\Delta t$$
$$v' = v_i - \frac{1}{2}Ay_i\Delta t$$
$$t' = t_i + \frac{1}{2}\Delta t$$

Second-order, ordinary differential equations

- Let's solve the simple harmonic oscillator problem with Δt=0.01, A=1, y(0)=0.2 and v (0) = 0: now this time 2nd order RK method



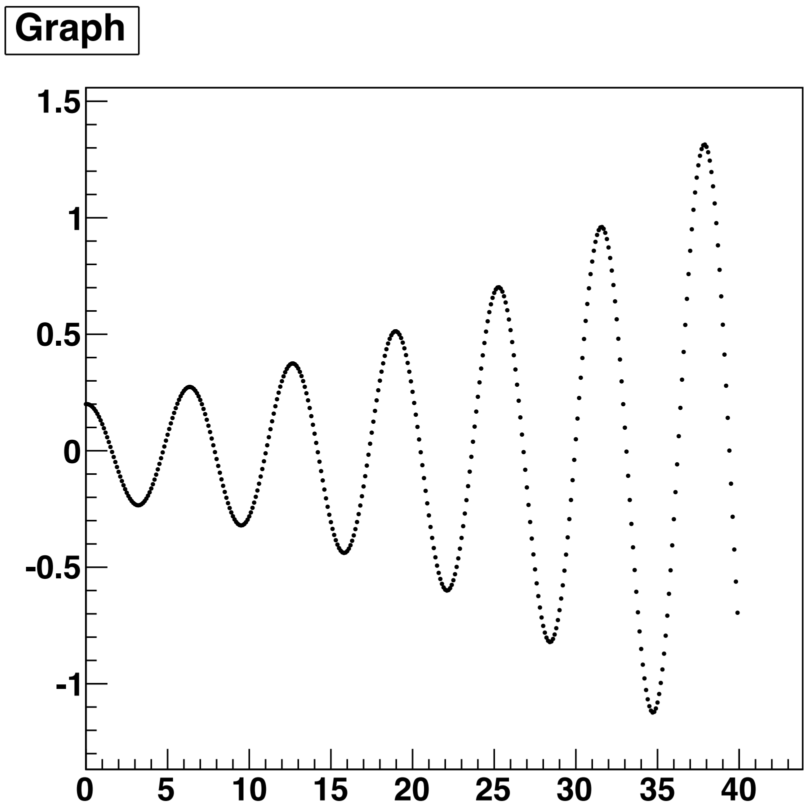


The oscillation amplitude is now stable.

Second-order, ordinary differential equations

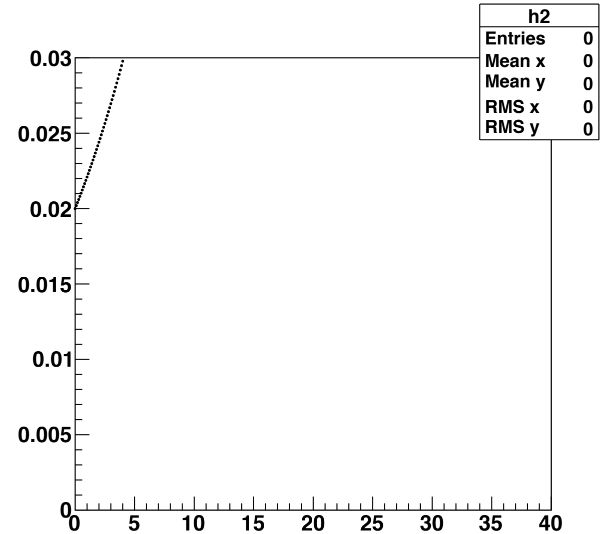
- What happens if we use the simple harmonic oscillator problem with $\Delta t=0.1$, $A=1$, $y(0)=0.2$ and $v(0) = 0$: this time Euler method

y(t)

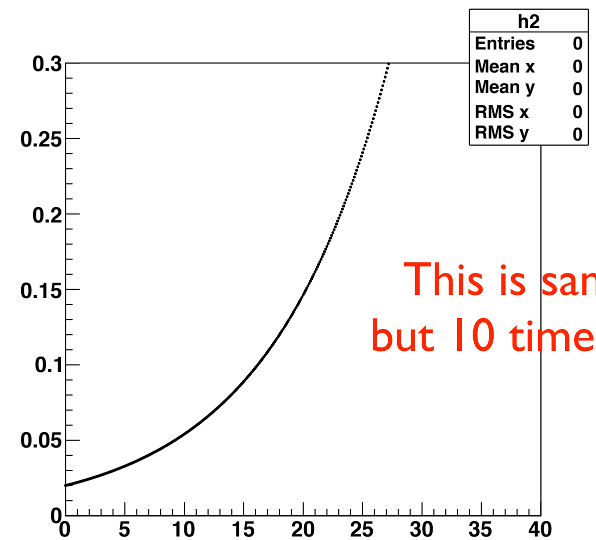


t

E(t)



E(t)



This is same as the top but 10 times larger y scale

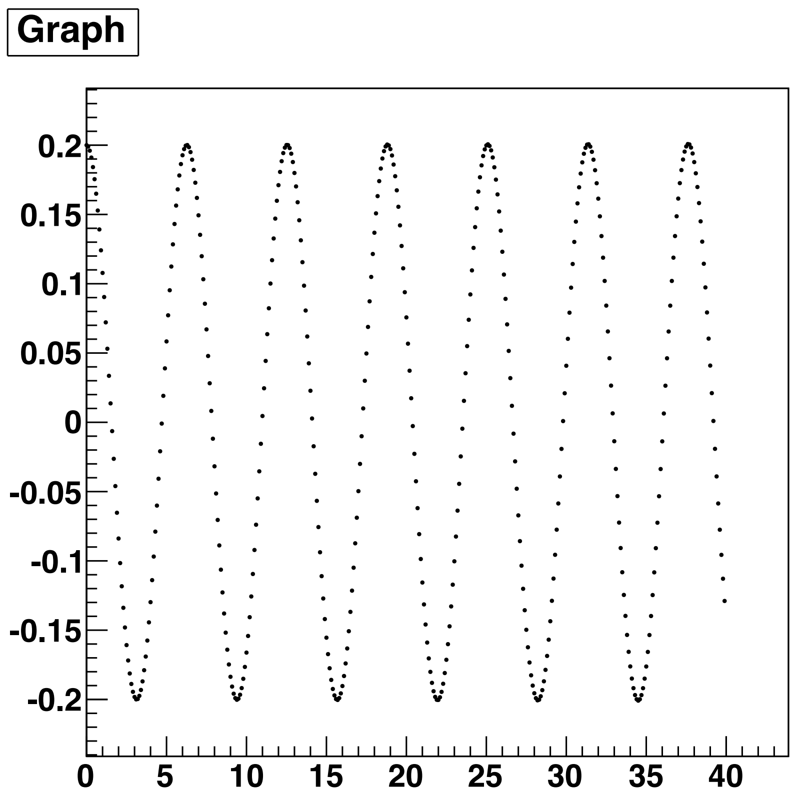
t

The oscillation amplitude is now even worse (quite unstable).

Second-order, ordinary differential equations

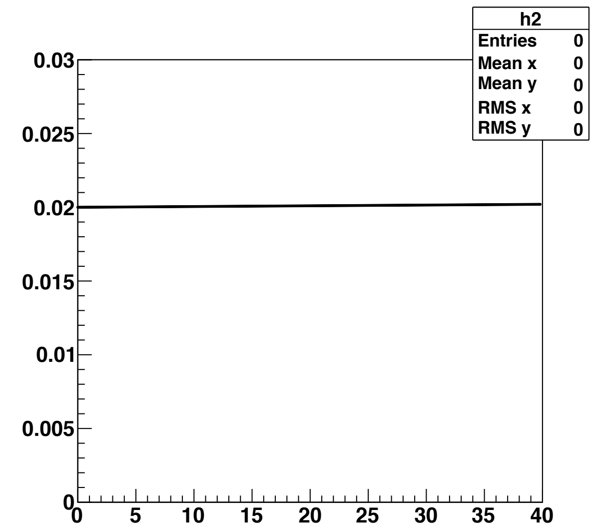
- What happens if we use the simple harmonic oscillator problem with Δt=0.1, A=1, y(0)=0.2 and v(0) = 0: this time 2nd order RK method

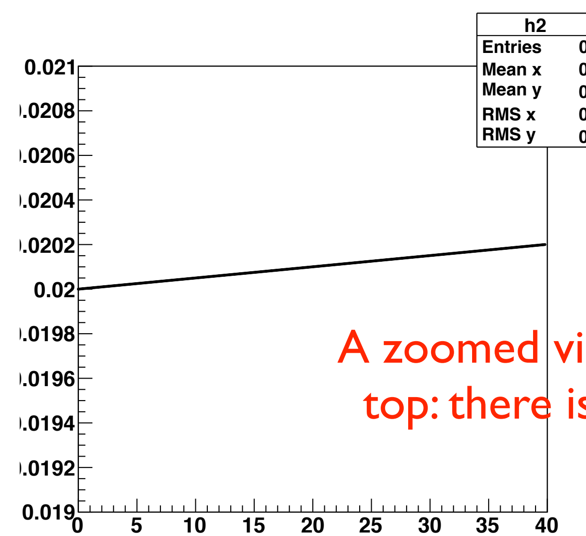
y(t)



t

E(t)





A zoomed view of the top: there is a slope

t

The oscillation amplitude is still unstable but much better than before.

Ordinary differential equations

- So one should really careful in choosing Δt and other parameters: always test your codes on known problems first and then apply to the real-world problem.

Bicycle racing: with and without air resistance

• Let's consider an example of a bicycle racing. We begin by ignoring friction. From the Newton's second law, we know that

$$\frac{dv}{dt} = \frac{F}{m}$$

assuming *v* is the velocity, *m* is the mass of the racer + bicycle system, and *F* is the force generated by the racer.

$$\frac{dE}{dt} = P$$

Using the definition of the power, we can express the power generated by the racer *P* in terms of the energy (*E*):

For a flat load, the energy is all kinetic so $E=1/2\ mv^2$ and $dE/dt = mv(dv/dt)$. Therefore

$$\frac{dv}{dt} = \frac{P}{mv}$$

Finally let us assume that P is a constant over a long time:

$$\int_{v_0}^{v} v'\ dv' = \int_0^t \frac{P}{m}\ dt'$$

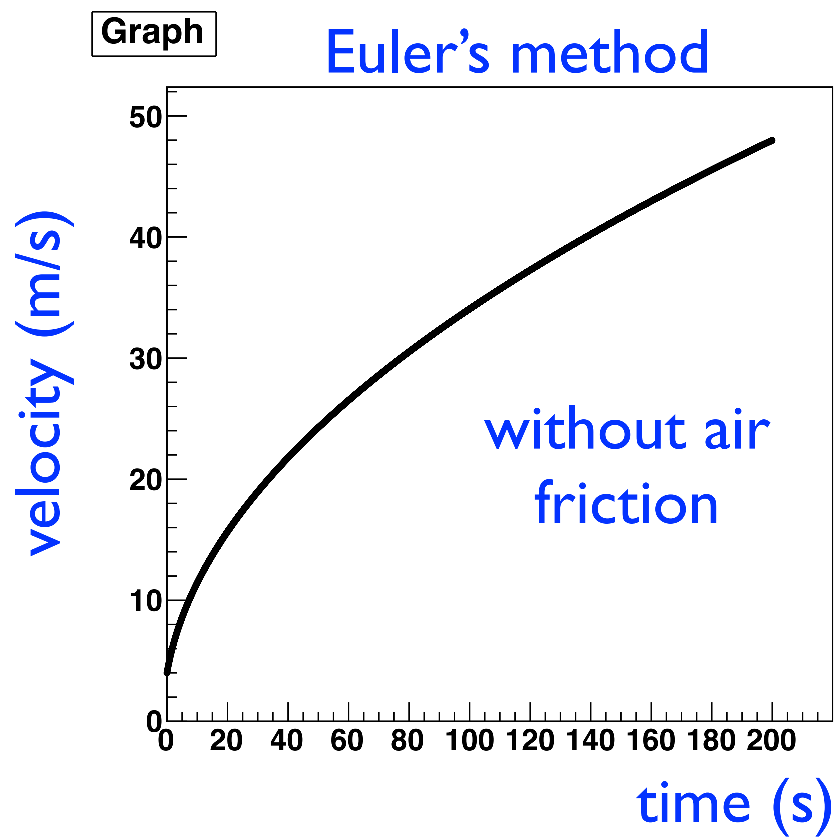
This can be analytically solved to be: $v = \sqrt{v_0^2 + 2Pt/m}$

Bicycle racing: with and without air resistance

- Let's solve this problem numerically with the Euler's method. We need to decide several numerical constants first.

Physiological studies of elite racing bicyclists have shown that athletes are able to produce a power output of 400 watts over extended period of time (~1 hour): P = 400 watts

Let's put the initial velocity to be v_0 = 4 m/s, time step is 0.1 s, and m=70 kg



So, without the friction, the velocity increases forever as time goes on: unphysical?

Bicycle racing: with and without air resistance

- Let's include the air resistance term, by adding the following drag force

$$F_{\text{drag}} \approx -\frac{1}{2} C\rho A v^2$$

where *C* is the drag coefficient, *A* is the front area, and ρ is the air density.

So the Euler's method is altered to be

$$v_{i+1} = v_i + \frac{P}{m v_i}\Delta t - \frac{C\rho A v_i^2}{2m}\Delta t$$

For numerical values:

C = *0.5* (dimensionless)
A = 0.33 m^2
ρ = 1.204 kg/m^3

http:://en.wikipedia.org/wiki/Density_of_air

Therefore:

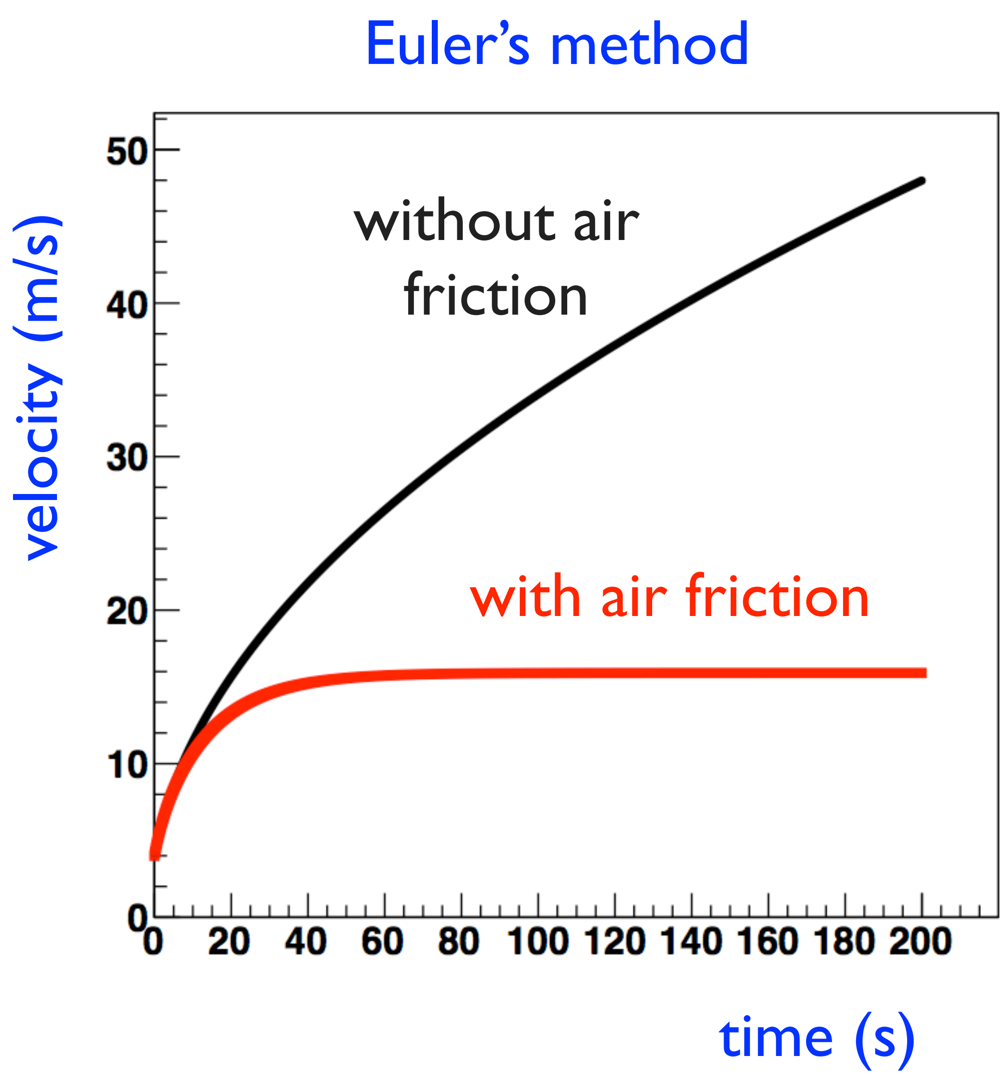
- At IUPAC standard temperature and pressure (0 °C and 100 kPa), dry air has a density of 1.2754 kg/m^3.
- At 20 °C and 101.325 Pa, dry air has a density of 1.2041 kg/m^3.
- At 70 °F and 14.696 psia, dry air has a density of 0.074887 lb_m/ft^3.

The following table illustrates the air density - temperature relationship:

Effect of temperature			
Temperature	Speed of sound	Density of air	Acoustic impedance
ϑ in °C	c in $m \cdot s^{-1}$	ρ in $kg \cdot m^{-3}$	Z in $N \cdot s \cdot m^{-3}$
−25	315.8	1.423	449.4
−20	318.9	1.395	444.9
−15	322.1	1.368	440.6
−10	325.2	1.342	436.1
−5	328.3	1.317	432.0
0	331.3	1.292	428.4
+5	334.3	1.269	424.3
+10	337.3	1.247	420.6
+15	340.3	1.225	416.8
+20	343.2	$\boxed{1.204}$	413.2
+25	346.1	1.184	409.8
+30	349.0	1.164	406.2
+35	351.9	1.146	403.3

Bicycle racing: with and without air resistance

- Numerical result with the air resistance using the Euler's method



Now with the air friction, there is a finite terminal velocity of the bicycle, which makes more sense.

Simple Harmonic Oscillator + damping term

- Let us consider SHO when a damping force is present

$$\frac{d^2\theta}{dt^2} = -\frac{g}{\ell}\theta - q\frac{d\theta}{dt}$$

This is the usual damping term : you should review your classical mechanics if you are not sure!

Analytic solutions:

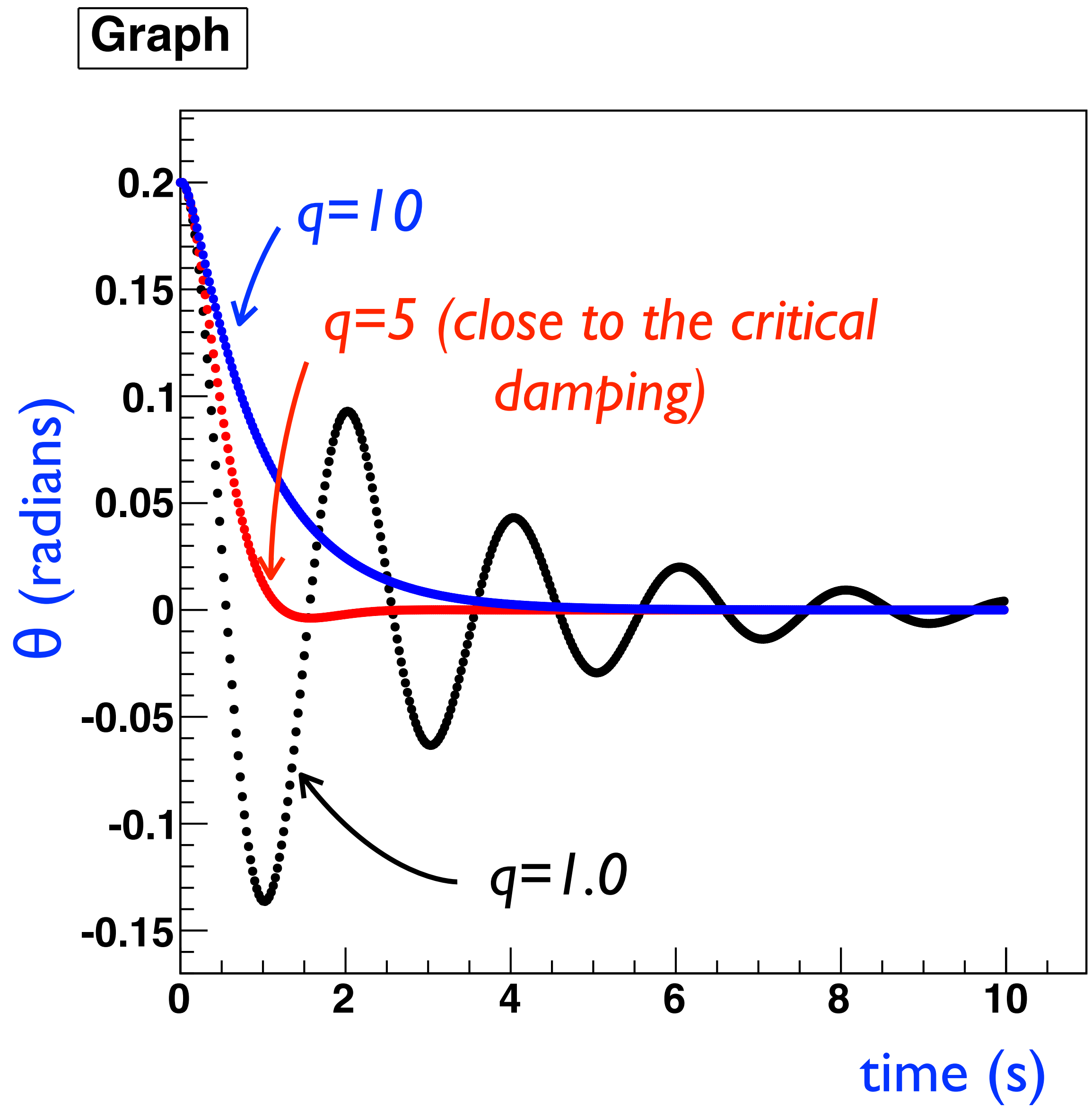
underdamping $$\theta(t) = \theta_0 e^{-qt/2} \sin\left(\sqrt{\omega^2 - q^2/4}\,t + \phi\right)$$

overdamping $$\theta(t) = \theta_0 e^{-(q/2 \pm \sqrt{q^2/4 - \omega^2})t}$$

critical damping $$\theta(t) = (\theta_0 + Ct)e^{-qt/2}$$

Simple Harmonic Oscillator + damping term

- Numerical solutions with Euler method



Used parameters
- q = 1.0,5,10
- g = 9.8 m/s
- l = 1.0 m

Note: critical damping is the case that you want to have your door behave like ...

Coupled differential equations

- Coupled differential equation: supposed we are interested in solving a coupled first order differential equations such as

$$\dot{y}_i(x) = f_i(\dot{y}_1, \ldots, \dot{y}_N, y_1, \ldots, y_N, x)$$

numerically. Can we use RK method for this? Yes, we know the answer from the discussion of 2nd order ODE already.

- Let's make an example. If we are given with

$$\dot{y}_1(x) = -y_2(x)$$
$$\dot{y}_2(x) = -y_1(x)$$

with boundary conditions of $y_1(0) = 0, \quad y_2(0) = 1$

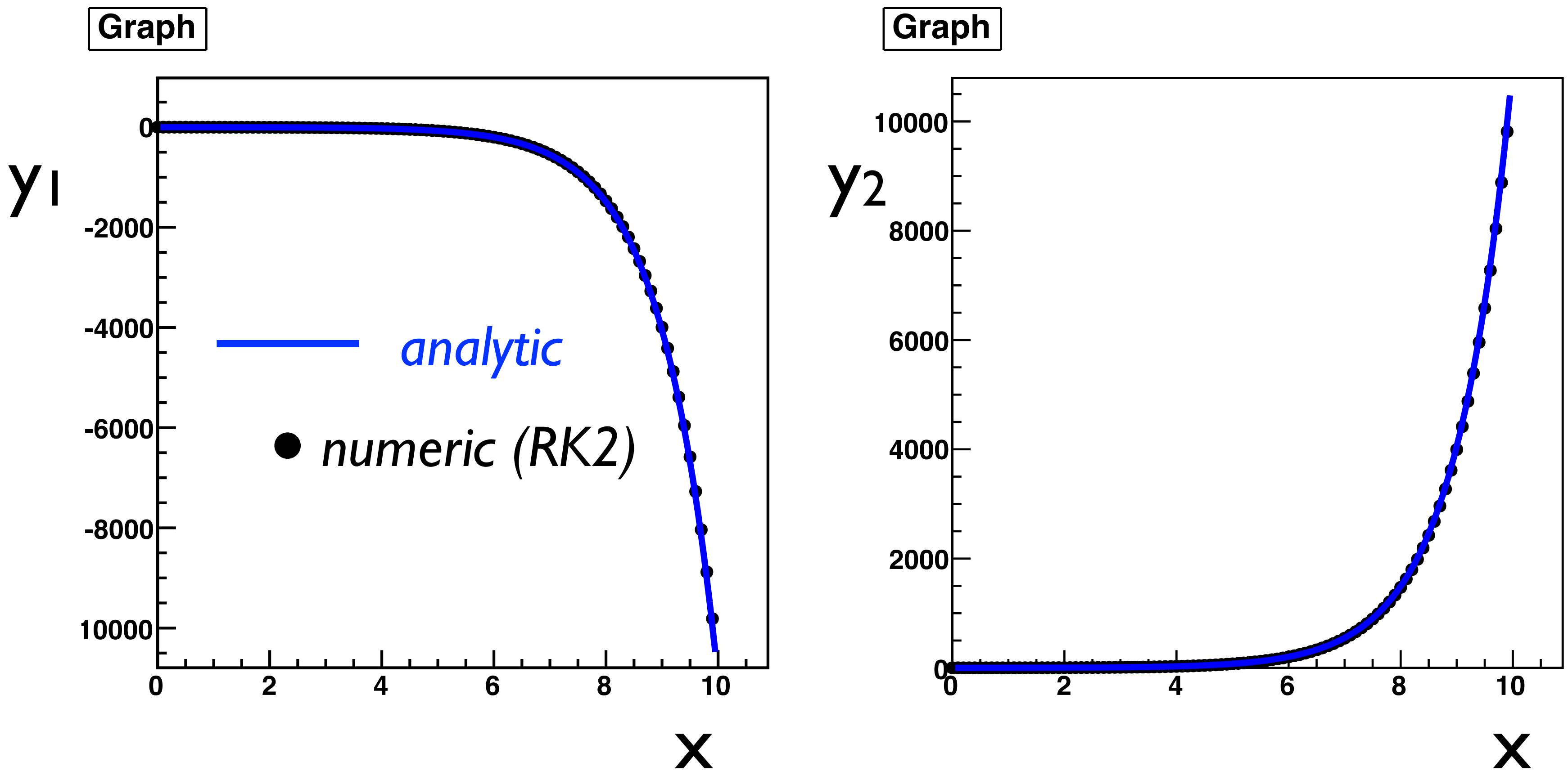
The analytic solutions become

$$y_1(x) = -\frac{1}{2}e^{-x}(-1 + e^{2x})$$
$$y_2(x) = \frac{1}{2}e^{-x}(1 + e^{2x})$$

Can you prove these are indeed solutions?

Coupled differential equations

- Numerical solutions



The numeric solutions are in good agreement with the analytic solutions.

Numerical solution of retarded term

- Let us consider the following 2nd order differential ODE:

$$\ddot{y}(t) - \dot{y}(t) + 5y(t) = 4 + 22\theta(t-2)e^{(4-2t)}$$

$$y(0) = 2, \quad \dot{y}(0) = -1 \qquad \theta(t-t_0) = \begin{cases} 1 & \text{if } t > t_0, \\ 0 & \text{otherwise} \end{cases}$$

Q: What kind of physical system is governed by such ODE?
Can you solve analytically?

A: The exact physical system is complicated, but...

$$\underbrace{\ddot{y}(t) - \dot{y}(t) + 5y(t)}_{\text{damped harmonic oscillator}} = \underbrace{4 + 22\theta(t-2)e^{(4-2t)}}_{\text{Some sort of forced, external term with a step-wise shape}}$$

One example would be a harmonic passive mode-locking of a single-frequency semiconductor laser system (feedback with a retarded term)

Numerical solution of retarded term

- Let us consider the following 2nd order differential ODE:

$$\ddot{y}(t) - \dot{y}(t) + 5y(t) = 4 + 22\theta(t-2)e^{(4-2t)}$$

Q: Can you solve analytically?

A: Yes, you can.

First Laplace transform the above ODE

$$s^2 f(s) - sy(0) - \dot{y}(0) - (sf(s) - y(0)) + 5f(s) = \frac{4}{s} + 22\frac{e^{-2s}}{s+2}$$

$$f(s) = \frac{2s^2 - 3s + 4}{s(s^2 - s + 5)} + 22e^{-2s}\frac{s+2}{s^2 - s + 5}$$

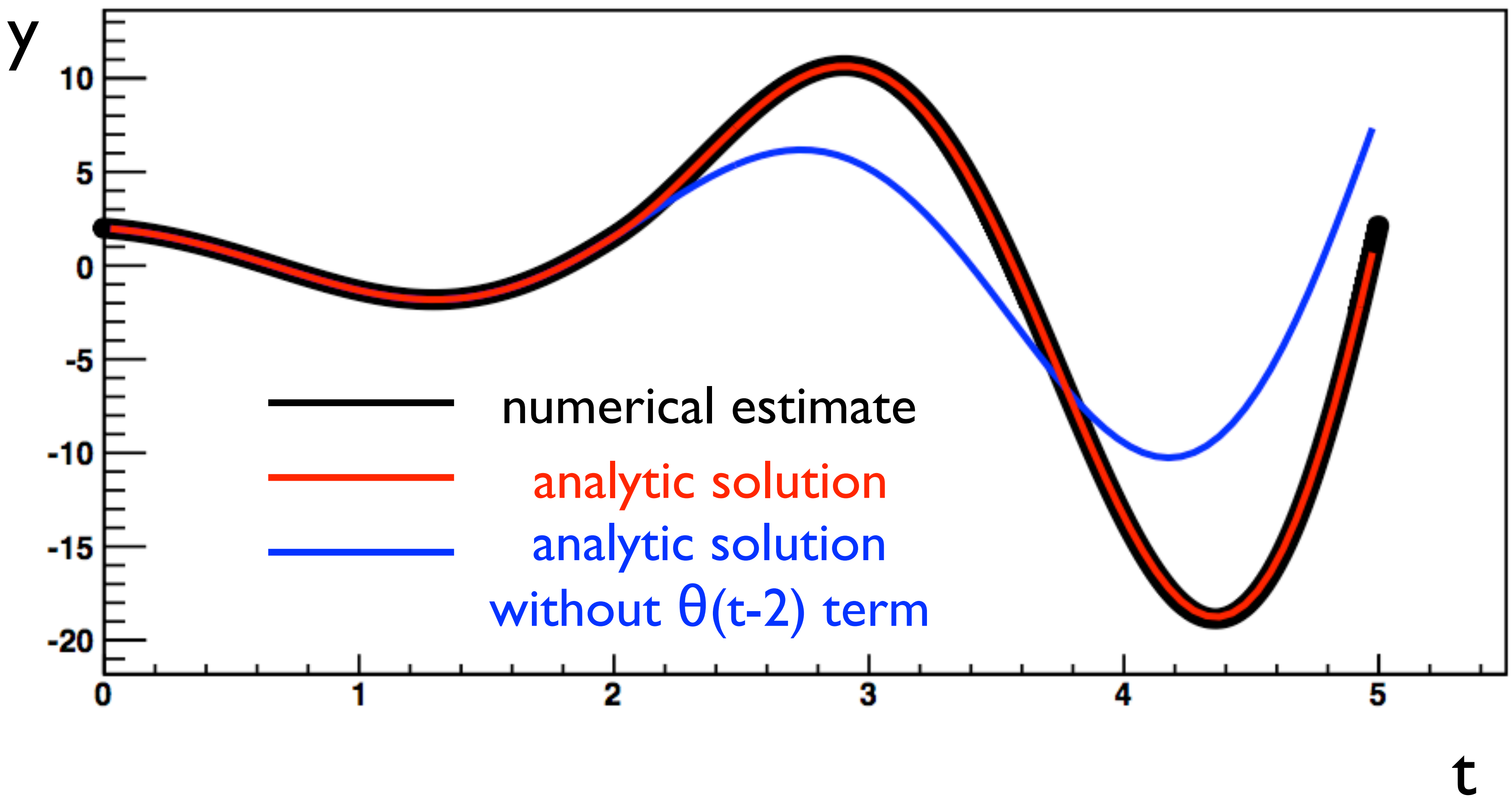
And now inverse Laplace transform gives

Can you prove these are indeed solutions?

$$y(t) = \frac{1}{5}\left(4 + 6e^{t/2}\cos\left[\frac{\sqrt{19}}{2}t\right] - \frac{16}{\sqrt{19}}\sin\left[\frac{\sqrt{19}}{2}t\right]\right)$$
$$+ 2\left(e^{-2(t-2)} - e^{(t-2)/2}\cos\left[\frac{\sqrt{19}}{2}(t-2)\right] + \frac{5}{\sqrt{19}}e^{(t-2)/2}\sin\left[\frac{\sqrt{19}}{2}(t-2)\right]\right)\theta(t-2)$$

Numerical solution of retarded term

Numerical solution:



Can you get a feeling on the solution and why does it behave like this?