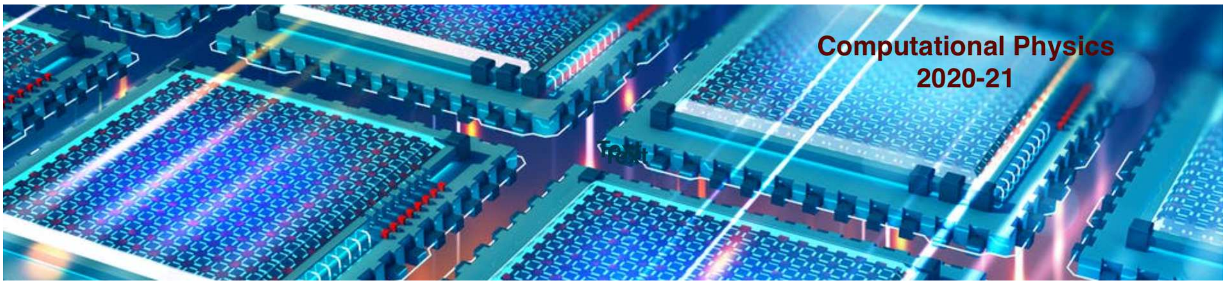
Computational Physics

numerical methods with C++ (and UNIX)
2020-21



Fernando Barao
Instituto Superior Técnico, Dep. Fisica
email: fernando.barao@tecnico.ulisboa.pt

Computational Physics

Physics problems

and Solutions

Fernando Barao, Phys Department IST (Lisbon)

Numerical methods

- Solving Ordinary Differential Equations
  - Euler method
  - Runge-Kutta method
  - examples

Ordinary Differential Equations

- Ordinary Differential Equations involve only derivatives with respect to a single variable, usually time

$\frac{dy}{dt} = f(t,y)$ Ex: $\frac{dy}{dt} + \alpha\, y = 0$ (decay equation, 1st order diff eq)

- Higher order differential equations

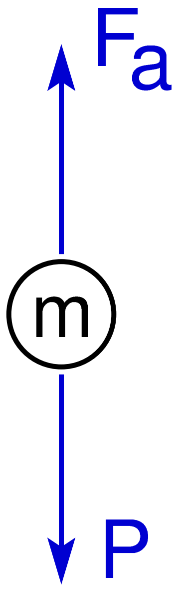
$\frac{d^2y}{dt^2} + \lambda \frac{dy}{dt} = f(t, \frac{dy}{dt}, y)$ Ex: $\frac{d^2y}{dt^2} + \frac{\lambda}{m}\frac{dy}{dt} + \frac{k}{m} y = 0$ (damped harmonic osc)

- Can be reduced to first-order by redefining dependent variables

$$\begin{cases} \frac{dy}{dt} = v \equiv y^{(1)} \\ y \equiv y^{(0)} \end{cases} \Rightarrow \begin{cases} \frac{dy^{(0)}}{dt} = y^{(1)}(t) \\ \frac{dy^{(1)}}{dt} = f(t, y^{(0)}, y^{(1)}) \end{cases} \Rightarrow \boxed{\frac{d\vec{y}}{d} = f(t,\vec{y})}$$

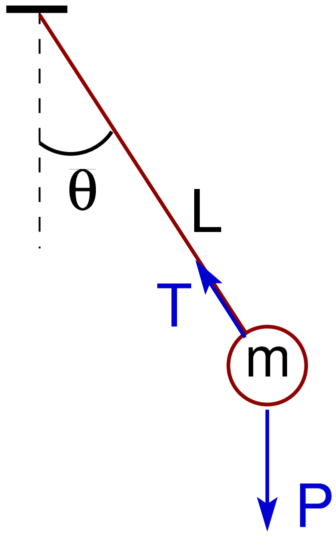
Examples

free fall with friction



$$\boxed{m\frac{d^2x}{dt^2} = mg - k\frac{dx}{dt}}$$

pendulum motion



$$\boxed{\frac{d^2\theta}{dt^2} + \frac{g}{\ell}\sin\theta = 0}$$

1st order ODE: numerical solutions

- ✔ **Solution:**

  $\frac{d\vec{y}}{dt} = f(t,\vec{y}) \quad \Rightarrow \quad \vec{y}(t) = \vec{y}(t_0) + \int_{t_0}^{t} f[t', \vec{y}(t')]\, dt'$

- ✔ **Euler method (1st order accurate)**

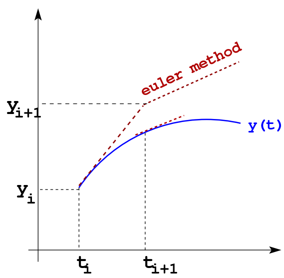
  $y(t+\delta t) \equiv y_{n+1} = y_n + \delta t\, \left.\frac{dy}{dt}\right|_n + \mathcal{O}[(\delta t)^2]\cdots$

  using the forward difference approximation for the derivative:

  $\left.\frac{dy}{dt}\right|_n \simeq \frac{y_{n+1}-y_n}{\delta t}$

  the differential equation becomes:

  $$\boxed{\frac{dy}{dt} = f[t, y(t)] \quad \Rightarrow \quad y_{n+1} = y_n + (\delta t)\ f[t_n, y(t_n)] + \mathcal{O}((\delta t)^2)}$$

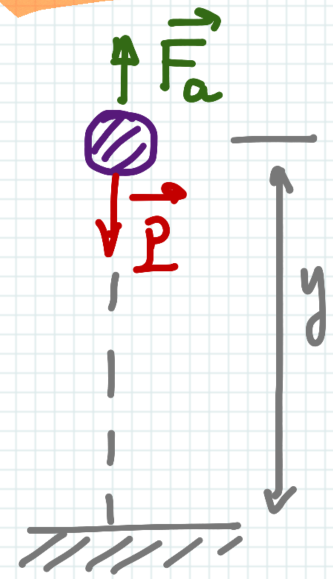


- ✔ Stability

  Suppose an error is introduced in the iteration value (δy) - like a round-off for instance - causing therefore a progressive deviation from the nominal numerical value

  $y_{n+1} + \delta y_{n+1} = y_n + \delta y_n - \delta t\left[f\,(t_n, y(t_n)) + \left.\frac{\partial f}{\partial t}\right|_n\ \delta y_n\right] \quad \Rightarrow \quad \boxed{\delta y_{n+1} = \delta y_n\left[1 - \delta t\ \left.\frac{\partial f}{\partial t}\right|_n\right]}$

Free-fall



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

Two 1st order eqs:

$$\frac{d\vec{v}}{dt} = \vec{F}/m$$

$$\frac{d\vec{r}}{dt} = \vec{v}$$

problem variables: $t, \vec{r}, \vec{v}$

analytical solution: $\vec{F}_a = 0$

$$\frac{d\vec{v}}{dt} = -g\vec{e}_y \qquad \vec{v} = \vec{v}_0 - gt\,\vec{e}_y \qquad v(0)=0$$

$$\frac{d\vec{r}}{dt} = \vec{v} \qquad y = y_0 - \frac{1}{2}gt^2 \qquad r(0) = y_0$$

Euler

$$\boxed{\begin{aligned} v_{n+1} &= v_n + h\,a_n \\ r_{n+1} &= r_n + h\,v_n \end{aligned}}$$

$$f(x+h) = f(x) + h f'(x) + O(h^2)$$

$$f'(x) = \frac{f(x+h) - f(x)}{h} - O(h)$$

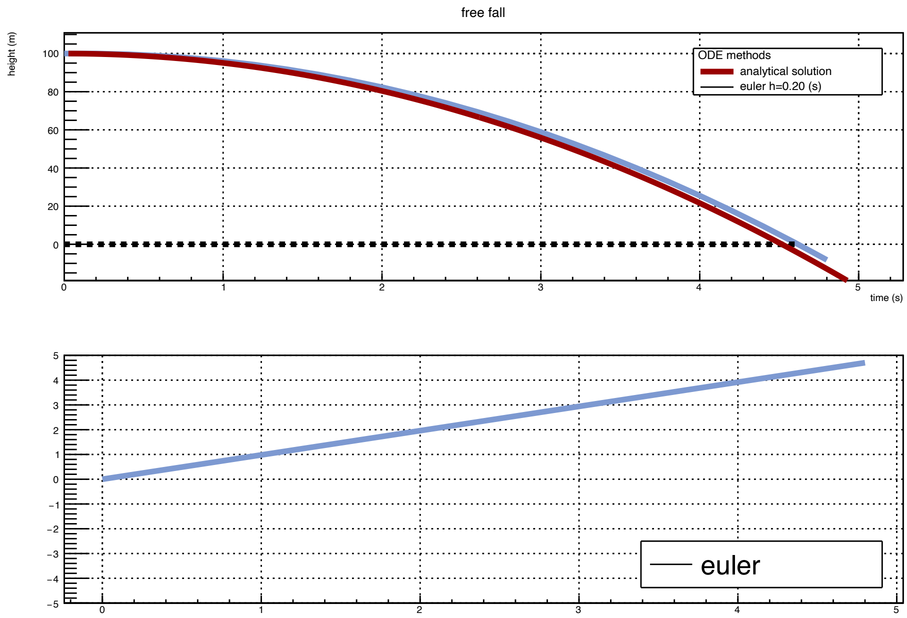
$$\left[\frac{v(t+h) - v(t)}{h} - O(h)\right] = -g$$

$$v(t+h) = v(t) - gh + O(h^2)$$

$$v_{n+1} \simeq v_n + a_n h$$

Numerical scheme:

- specify initial conditions
- choose time step, h
- calculate acceleration, $a(r_n, v_n)$
- compute v_{n+1}, r_{n+1}



1st order ODE: numerical solutions

✔ Solution of the 1st-order equation

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

✔ **Predictor-Corrector (Crank-Nicolson)**
using the average of the two slopes at **i** and **i+1**:

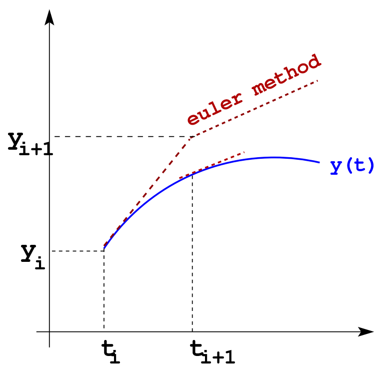
$$y(t_{i+1}) = y(t_i) + (\delta t)\frac{1}{2}\left[\left.\frac{dy}{dt}\right|_i + \left.\frac{dy}{dt}\right|_{i+1}\right]$$

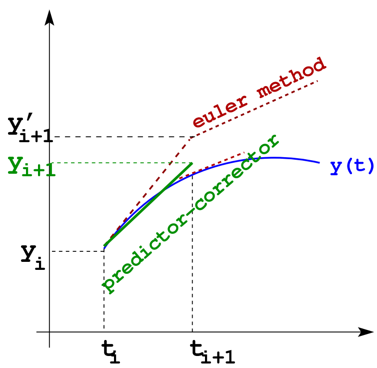
✔ Accuracy

$O((\delta t)^3)$ $\Rightarrow$ second-order accurate

✔ algorithm

- compute the slope at t_i : $f(t_i, y_i)$
- user Euler approach to make a prediction for next slope value:
  $y_{t+i} = y(t_i) + (\delta t) f(t_i, y_i) \quad \Rightarrow \quad f(t_{i+1}, y_{i+1})$
- average slopes and get next iteration:
  $y_{i+1} = y_i + \frac{\delta t}{2}\left[f(t_i, y_i) + f(t_{i+1}, y_{i+1})\right]$



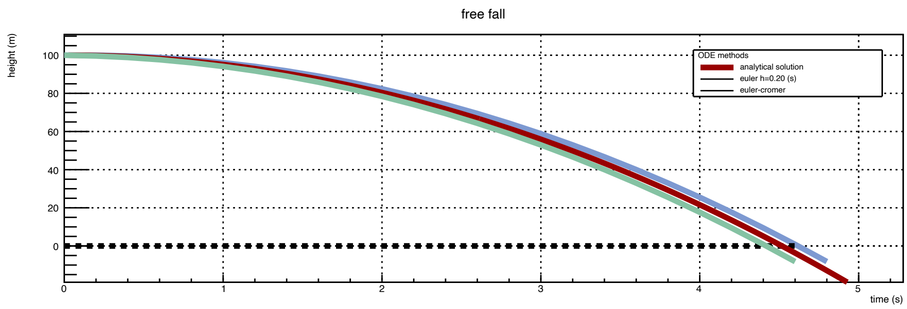


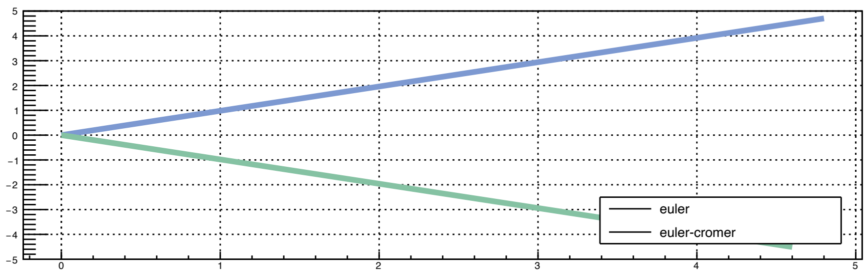
Euler-Cromer

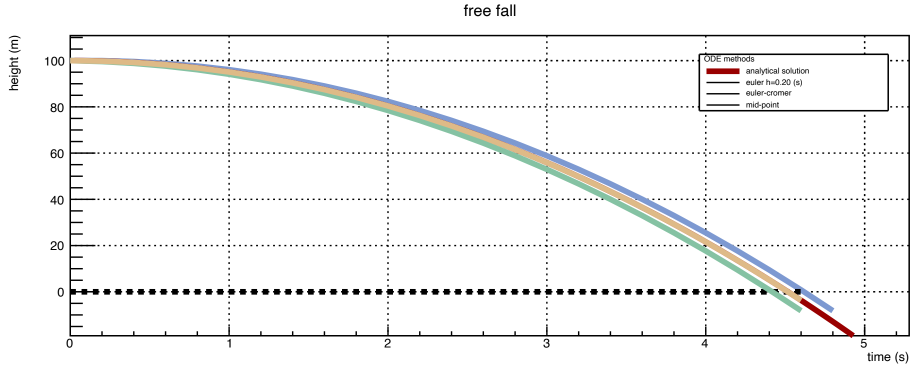
$$v_{n+1} = v_n + h\, a_n$$
$$x_{n+1} = x_n + h\, v_{n+1}$$

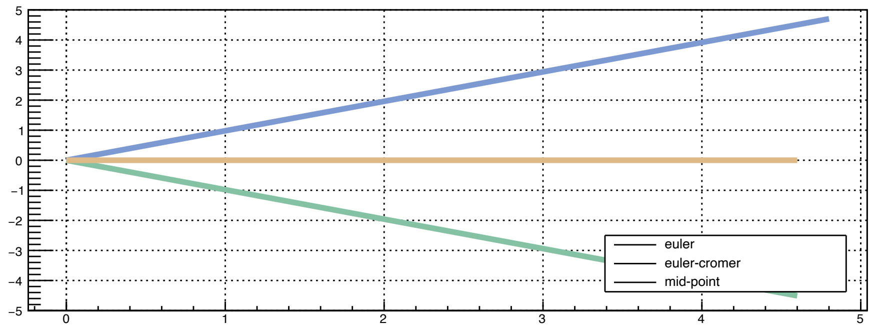
mid-point method (predictor-corrector)

$$v_{n+1} = v_n + h\, a_n$$
$$x_{n+1} = x_n + h\, \frac{v_{n+1} + v_n}{2}$$









1st order ODE: numerical solutions

✔ **Leap-Frog method (Stormer-Verlet)**

using the centered difference approximation for the derivative:

$$\left.\frac{dy}{dt}\right|_n \simeq \frac{y_{n+1}-y_{n-1}}{2\delta t}$$

the differential equation becomes:

$$\boxed{\left.\frac{dy}{dt}\right|_n = f[t_n, y(t_n)] \quad \Rightarrow \quad y_{n+1} = y_{n-1} + 2\,(\delta t)\; f[t_n, y(t_n)]}$$

✔ Accuracy

$O((\delta t)^3)$ $\Rightarrow$ second-order accurate

✔ Stability

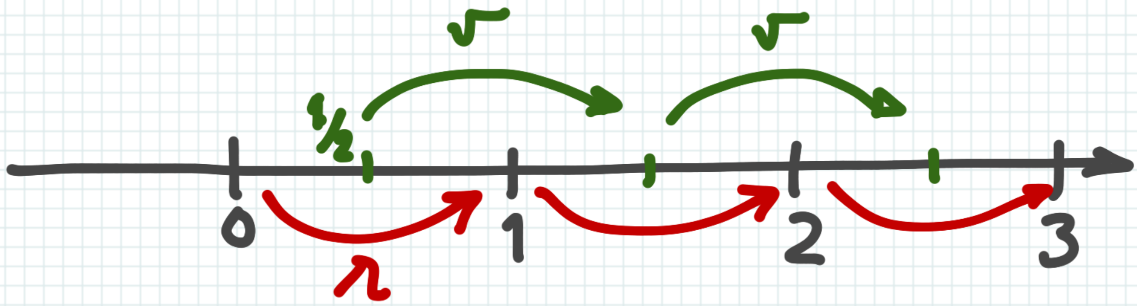
$$\delta y_{n+1} = \delta y_{n-1} - 2\;\delta t\;\left.\frac{\partial y}{\partial t}\right|_n\;\delta y_n$$

✔ algorithm

time: $\delta t = (t_f - t_0)/n$
first iteration: $y_1 = y_0 + \delta t\; f(t_0, y_0) \quad ; \quad t_1 = t_0 + \delta t$
following iterations (i=1,n-1): $y_{i+1} = y_{i-1} + 2\delta t\; f(t_i, y_i) \quad ; \quad t_{i+1} = t_0 + (i+1)\delta t$

Leap-frog

$$\frac{df}{dt} = \frac{f(x+h) - f(x-h)}{2h} - \mathcal{O}(h^2)$$



$$v_{n+1/2} = v_{n-1/2} + h a_n \quad + \mathcal{O}(h^3)$$

$$x_{n+1} = x_n + h v_{n+1/2} \quad + \mathcal{O}(h^3)$$

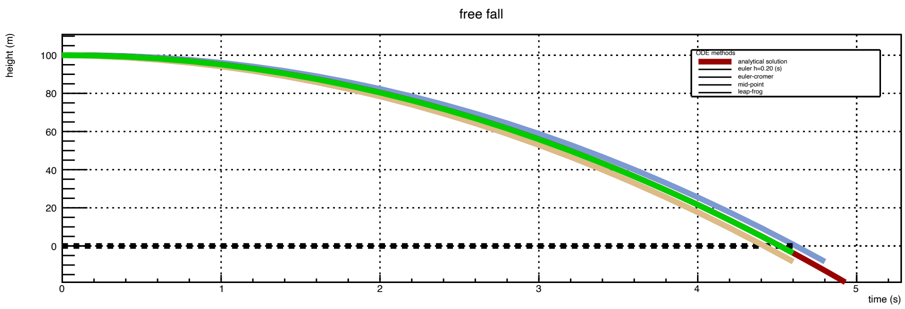
$n = 0, 1, 2, 3, \ldots$

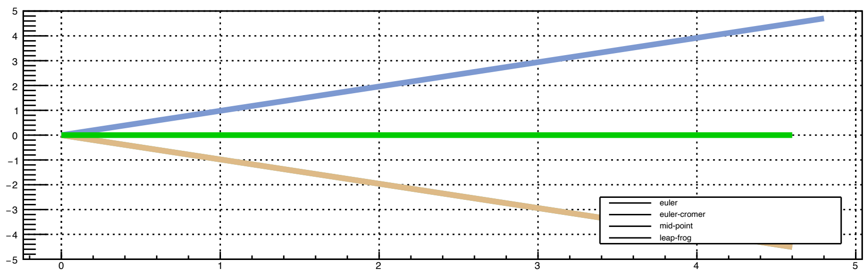
$n=0 \;\Big|\; \begin{aligned} v_{1/2} &= v_{-1/2} + h a_0(x_0) \\ x_1 &= x_0 + h v_{1/2} \end{aligned}$

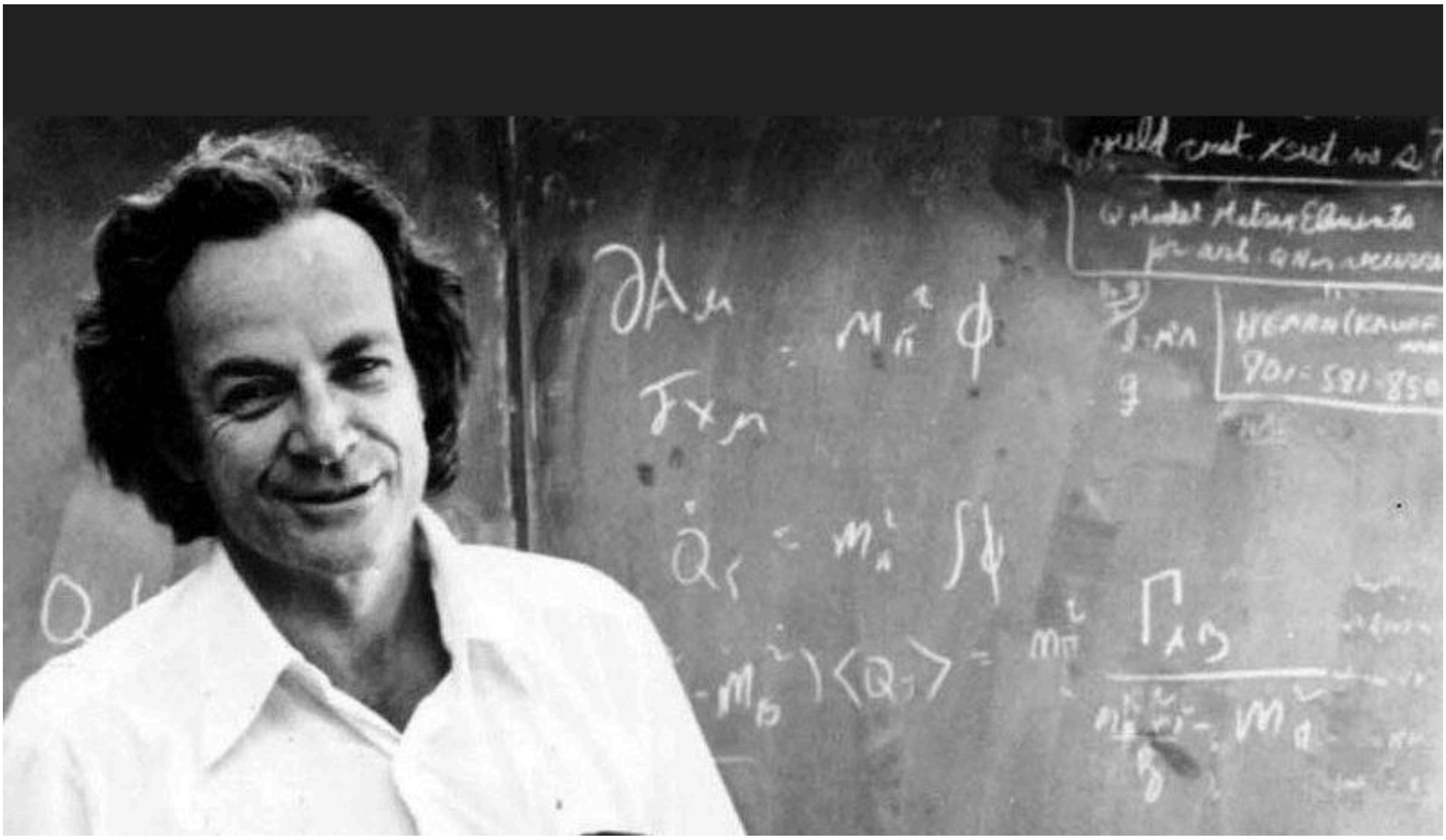
$v_{-1/2} = v(x-h) = ?$

$v(x-h) = v(x) - h \dfrac{dv}{dt}(x)$

$v_{-1/2} = v_0 - \frac{h}{2} a_0$







ODE: numerical solution improvement?

Can we improve the numerical solution of $\frac{dy}{dt} = f(t,y)$?

- ✔ **Use more terms in the Taylor expansion of** y_{n+1}

$$
\begin{aligned}
y_{n+1} &= y_n + (\delta t)\ \left.\frac{dy}{dt}\right|_n + \frac{(\delta t)^2}{2}\ \left.\frac{d^2y}{dt^2}\right|_n + O(h^3) \\
&= y_n + (\delta t)\ f(t_n, y_n) + \frac{(\delta t)^2}{2}\ \frac{d}{dt}\left[f(t_n, y_n)\right] \\
&= y_n + (\delta t)\ f(t_n, y_n) + \frac{(\delta t)^2}{2}\ \left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial t}\right)
\end{aligned}
$$

  Interesting if analytic differentiation possible! Otherwise numerical derivatives...(errors)

- ✔ **Use intermediate points within one time step (Runge-Kutta methods)**
  We have seen that the general solution for the 1st order differential equation was:

$$
\frac{dy}{dt} = f(t,y) \quad \Rightarrow \quad y(t) = y(t_0) + \int_{t_0}^{t} f(t', y(t'))\ dt'
$$

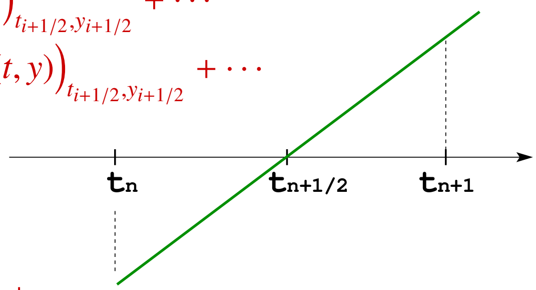
  Considering a small interval $\delta t = t_{n+1} - t_n$, the solution comes:

$$
\boxed{y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f\left[t', y(t')\right]\ dt'}
$$

Runge-Kutta of second order (RK2)

- ✔ Let's use for the integrand $f(t,y)$ a Taylor expansion at 1st-order around an intermediate abcissa $t_{i+\frac{1}{2}} \equiv t_i + h/2$

$$\begin{aligned}
f(t,y) &= f(t_{i+1/2}, y_{i+1/2}) + (t - t_{i+1/2} \left(\frac{df}{dt}\right)_{t_{i+1/2}, y_{i+1/2}} + \cdots \\
&= f(t_{i+1/2}, y_{i+1/2}) + (t - t_{i+1/2}) \left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial y}\frac{dy}{dt}\right)_{t_{i+1/2}, y_{i+1/2}} + \cdots \\
&= f(t_{i+1/2}, y_{i+1/2}) + (t - t_{i+1/2}) \left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial y} f(t,y)\right)_{t_{i+1/2}, y_{i+1/2}} + \cdots
\end{aligned}$$



- ✔ The integration in the step interval:

$$\begin{aligned}
\int_{t_n}^{t_{n+1}} f\left[t', y(t')\right] dt' &= f(t_{i+1/2}, y_{i+1/2}) \int_{t_n}^{t_{n+1}} dt' + \\
&\quad \left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial y} f(t,y)\right)_{t_{i+1/2}, y_{i+1/2}} \int_{t_n}^{t_{n+1}} (t' - t_{i+1/2})\, dt' \\
&= h\ f(t_{i+1/2}, y_{i+1/2}) + O(h^3)
\end{aligned}$$

$$\boxed{y_{i+1} = y_i + h\ f(t_{i+1/2}, y_{i+1/2}) + O(h^3)}$$

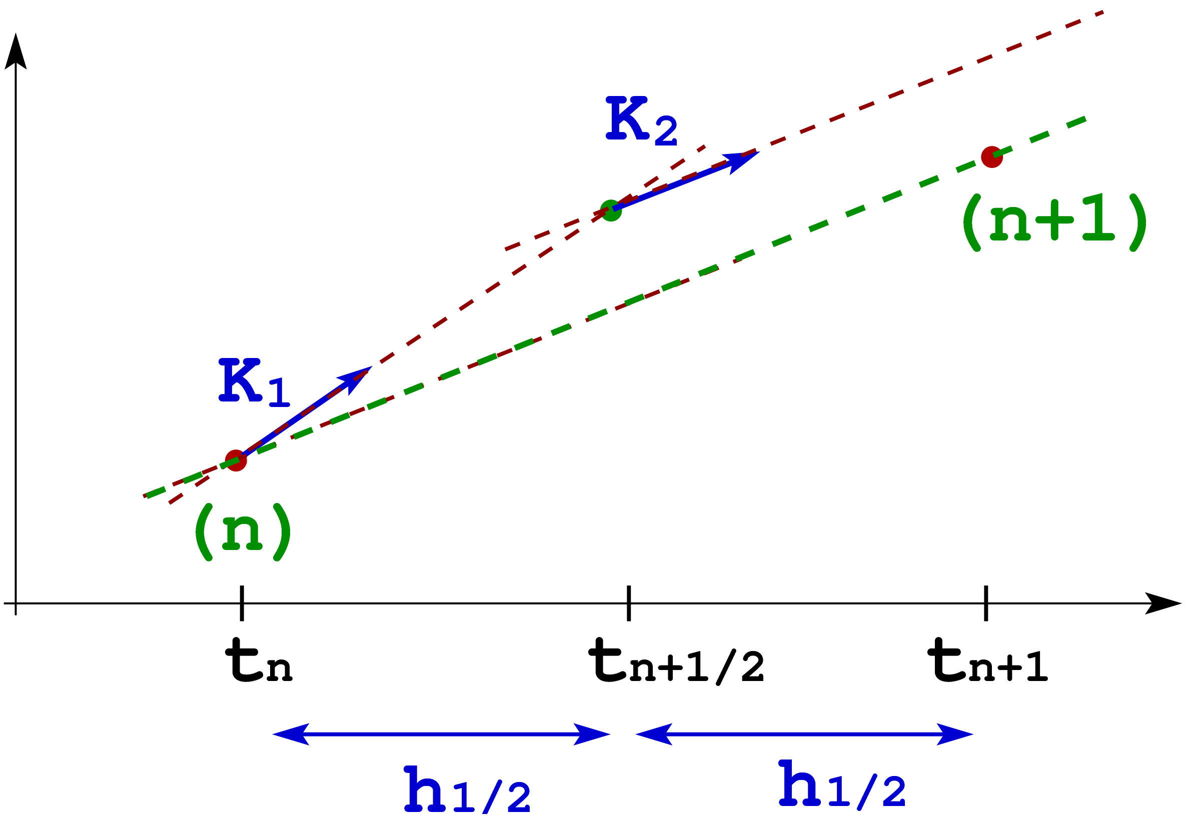
RK2 (cont.)

- ✔ **algorithm**
  - ► the derivative $f(t_{i+1/2}, y_{i+1/2})$ is computed using the Euler relation

$$\begin{aligned}
t_{i+1/2} &= t_i + \frac{h}{2} \\
y_{i+1/2} &= y_i + \frac{h}{2}\ f(t_i, y_i) \quad \text{(euler relation)} \\
y_{i+1} &= y_i + h\ f(t_{i+1/2}, y_{i+1/2})
\end{aligned}$$

$$\begin{aligned}
K_1 &= h\ f(t_i, y_i) \\
K_2 &= h\ f\left(t_i + \frac{h}{2}, y_i + \frac{K_1}{2}\right) \\
y_{i+1} &= y_i + K_2
\end{aligned}$$

RK2 (cont.)



RK2 algorithm

RK2 algorithm

```
equation: dy/dt = f(t,y)
solution points: (t,y)    (t,y)     (t,y)    ...
                      0        1         2
It's useful to define class ODEpoint to store iterations

h = (tf-ti)/n; //time step
t=ti;
y=y0;
for (int i=0; i<n; i++) {
  K1 = h*f(t,y);
  K2 = h*f(t+h/2,y+K1/2);
  //next point
  y += K2;
  t += h;
}
```

Runge-Kutta of 4th-order (RK4)

- ✔ Instead of approximating the integral with the midpoint rule we can now use the Simpson rule (2nd-deg polynomial) for the integration in the step interval:

  $[t_i,\ t_{i+1/2},\ t_{i+1}]$

$$\int_{t_n}^{t_{n+1}} f\left[t', y(t')\right]\, dt' = \frac{h}{6}\left[f(t_i, y_i) + 4\, f(t_{i+1/2}, y_{i+1/2}) + f(t_{i+1}, y_{i+1})\right] + O(h^5)$$

- ✔ the fourth-order runge-kutta method splits the mid-point evaluation in two steps

$$y_{i+1} = y_i + \frac{h}{6}\left[f(t_i, y_i) + 2\, f(t_{i+1/2}, y_{i+1/2}) + 2\, f(t_{i+1/2}, y_{i+1/2}) + f(t_{i+1}, y_{i+1})\right]$$

Runge-Kutta of 4th-order (RK4)

- ✔ compute slope at $\mathbf{t_i}$:

  $\boxed{K_1 = hf(t_i, y_i)}$

- ✔ compute slope at $\mathbf{t_{i+1/2}}$ by using euler approx. for computing $\mathbf{y_{i+1/2}}$:

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

  $\boxed{K_2 = hf(t_{i+1/2}, y_{i+1/2}) \equiv hf(t_i + h/2, y_i + K_1/2)}$

- ✔ use previous slope to compute third slope, also located at $t_{i+1/2}$:

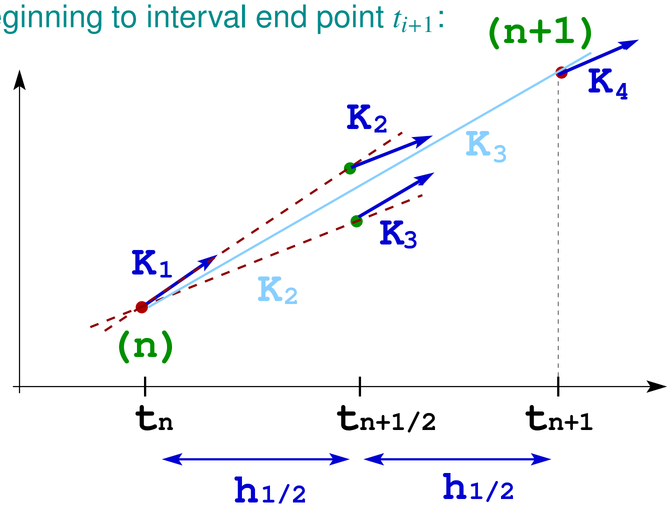
  $\boxed{K_3 = hf(t_{i+1/2}, y_{i+1/2}) \equiv hf(t_i + h/2, y_i + K_2/2)}$

- ✔ use third slope to linearly extrapolate from beginning to interval end point t_{i+1}:

  $y_{i+1} = y_i + K_3$

  calculate slope at end point:

  $\boxed{K_4 = hf(t_{i+1}, y_{i+1}) \equiv hf(t_i + h, y_i + K_3)}$



We use now all four slopes K_1, K_2, K_3, K_4 as estimators for computing the function in the interval

RK4 (cont.)

✔ **algorithm**

$$
\begin{aligned}
K_1 &= h\ f(t_i, y_i) \\
K_2 &= h\ f\left(t_i + \frac{h}{2}, y_i + \frac{K_1}{2}\right) \\
K_3 &= h\ f\left(t_i + \frac{h}{2}, y_i + \frac{K_2}{2}\right) \\
K_4 &= h\ f\left(t_i + h, y_i + K_3\right) \\
y_{i+1} &= y_i + \frac{1}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right)
\end{aligned}
$$

Example: ODE 1st-order

✔ Let's solve numerically the 1st-order differential equation:

$\frac{dy}{dx} = 3x - y + 8$

$x \in [0.0, 0.5]$

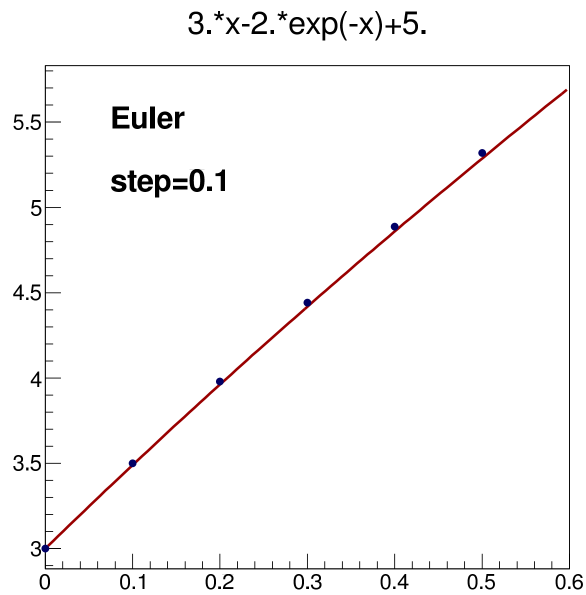
$y(0) = 3.$

✔ analytical solution:

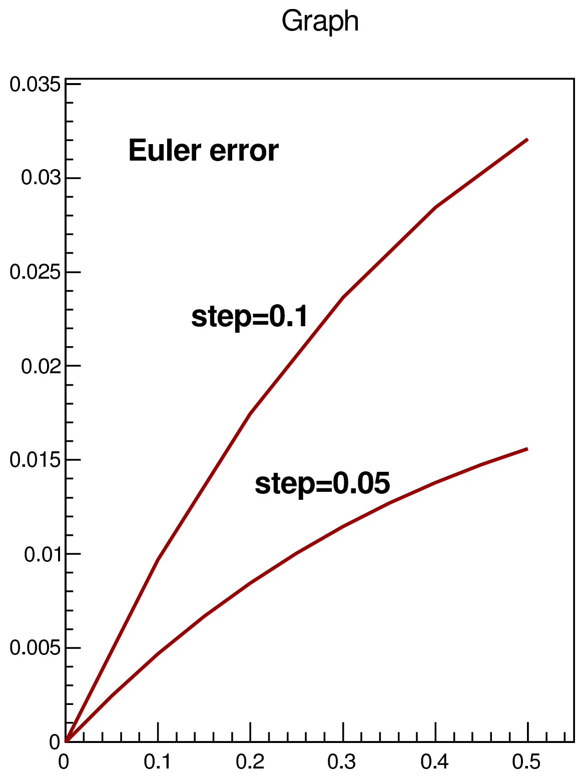
$y(x) = 3x - 2e^{-x} + 5$

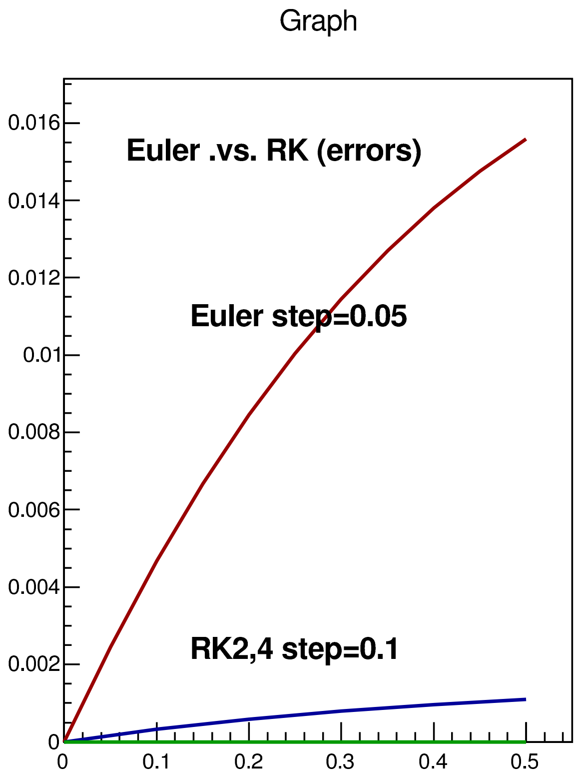
✔ Solving numerically with Euler method:

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



Example: ODE 1st-order (cont.)





Example: High-order equations

- High-order differential equations can be reduced to a systems of 1st-order equations
- These equations are very common in physics (dynamics):

$$m\frac{d^2\vec{r}}{dt^2} = \vec{F}(\vec{r}, t)$$

- The presence of frictional forces (or electromagnetic ones) introduce also a velocity dependence,

$$m\frac{d^2\vec{r}}{dt^2} = \vec{F}(\vec{r}, \dot{\vec{r}}, t)$$

- We can transform this equation into a set of 1st-order differential equations, in terms of variables $\vec{r}$ and $\vec{p}$:

$$\begin{cases} \frac{d\vec{r}}{dt} & = & \frac{\vec{p}(t)}{m} \\ \frac{d\vec{p}}{dt} & = & \vec{F}(t, \vec{r}, \dot{\vec{r}}) \end{cases} \Rightarrow \begin{bmatrix} \dot{r}_x \\ \dot{r}_y \\ \dot{r}_z \\ \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{m} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{m} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{m} \\ \cdots & & & & & \\ \cdots & & & & & \\ \cdots & & & & & \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_z \\ p_x \\ p_y \\ p_z \end{bmatrix}$$

Runge-Kutta 4

✔ a set of 1st order equations

$$\frac{d\vec{r}}{dt} = \vec{v}$$

$$\frac{d\vec{v}}{dt} = \frac{\vec{F}(t,\vec{r},\vec{v})}{m}$$

✔ in terms of scalar variables

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

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

$$\frac{dr_z}{dt} = v_z$$

$$\frac{dv_x}{dt} = \frac{F_x(t,\vec{r},\vec{v})}{m}$$

$$\frac{dv_y}{dt} = \frac{F_y(t,\vec{r},\vec{v})}{m}$$

$$\frac{dv_z}{dt} = \frac{F_z(t,\vec{r},\vec{v})}{m}$$

gen variables: $(r_x, r_y, r_z, v_x, v_y, v_z) \to (y^{(0)}, y^{(1)}, y^{(2)}, y^{(3)}, y^{(4)}, y^{(5)})$

$$\frac{dy^{(0)}}{dt} = y^{(3)} = f^{(0)}(t,\vec{y}) \qquad \frac{dy^{(3)}}{dt} = \frac{F_x(t,\vec{r},\vec{v})}{m} = f^{(3)}(t,\vec{y})$$

$$\frac{dy^{(1)}}{dt} = y^{(4)} = f^{(1)}(t,\vec{y}) \qquad \frac{dy^{(4)}}{dt} = \frac{F_y(t,\vec{r},\vec{v})}{m} = f^{(4)}(t,\vec{y})$$

$$\frac{dy^{(2)}}{dt} = y^{(5)} = f^{(2)}(t,\vec{y}) \qquad \frac{dy^{(5)}}{dt} = \frac{F_z(t,\vec{r},\vec{v})}{m} = f^{(5)}(t,\vec{y})$$

Iteration step: $n \to (n+1)$

$$\begin{pmatrix} y^{(0)} \\ y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ y^{(4)} \\ y^{(5)} \end{pmatrix}_{n+1} = \begin{pmatrix} y^{(0)} \\ y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ y^{(4)} \\ y^{(5)} \end{pmatrix}_{n} + \frac{1}{6}\left(\begin{pmatrix} K_1^{(0)} \\ K_1^{(1)} \\ K_1^{(2)} \\ K_1^{(3)} \\ K_1^{(4)} \\ K_1^{(5)} \end{pmatrix} + 2\begin{pmatrix} K_2^{(0)} \\ K_2^{(1)} \\ K_2^{(2)} \\ K_2^{(3)} \\ K_2^{(4)} \\ K_2^{(5)} \end{pmatrix} + 2\begin{pmatrix} K_3^{(0)} \\ K_3^{(1)} \\ K_3^{(2)} \\ K_3^{(3)} \\ K_3^{(4)} \\ K_3^{(5)} \end{pmatrix} + \begin{pmatrix} K_4^{(0)} \\ K_4^{(1)} \\ K_4^{(2)} \\ K_4^{(3)} \\ K_4^{(4)} \\ K_4^{(5)} \end{pmatrix} \right)$$

$$\left| \begin{array}{lcl} K_1^{(i)} & = & h\, f^{(i)}\left(t_n, y_n^{(i)}\right) \\ K_2^{(i)} & = & h\, f^{(i)}\left(t_n + \frac{h}{2}, y_n^{(i)} + \frac{1}{2}K_1^{(i)}\right) \\ K_3^{(i)} & = & h\, f^{(i)}\left(t_n + \frac{h}{2}, y_n^{(i)} + \frac{1}{2}K_2^{(i)}\right) \\ K_4^{(i)} & = & h\, f^{(i)}\left(t_n + h, y_n^{(i)} + K_3^{(i)}\right) \end{array} \right.$$

i variables: $0, \cdots, 5$