Computational Physics

Physics problems

and Solutions

Fernando Barao, Phys Department IST (Lisbon)

Example: system of 1st-order ODEs

✔ Solve numerically the following system:

$$\begin{aligned} \frac{dw}{dx} &= \sin(x) + y \\ \frac{dy}{dx} &= \cos(x) - w \end{aligned}$$

✔ Initial values:

$$\begin{aligned} w(0) &= 0 \\ y(0) &= 0 \end{aligned}$$

✔ variables and functions:

$$\begin{cases} y^{(0)} = w \\ y^{(1)} = y \end{cases} \qquad \begin{cases} f^{(0)} = \sin(x) + y^{(1)} \\ f^{(1)} = \cos(x) - y^{(0)} \end{cases}$$

✔ runge-kutta 4th-order iterations:

$$\begin{cases} K_1^{(0)} = hf^{(0)}\left(x_n, y_n^{(0)}, y_n^{(1)}\right) \\ K_1^{(1)} = hf^{(1)}\left(x_n, y_n^{(0)}, y_n^{(1)}\right) \\ K_2^{(0)} = hf^{(0)}\left(x_n + \frac{h}{2}, y_n^{(0)} + \frac{K_1^{(0)}}{2}, y_n^{(1)} + \frac{K_1^{(1)}}{2}\right) \\ K_2^{(1)} = hf^{(1)}\left(x_n + \frac{h}{2}, y_n^{(0)} + \frac{K_1^{(0)}}{2}, y_n^{(1)} + \frac{K_1^{(1)}}{2}\right) \\ K_3^{(0)} = hf^{(0)}\left(x_n + \frac{h}{2}, y_n^{(0)} + \frac{K_2^{(0)}}{2}, y_n^{(1)} + \frac{K_2^{(1)}}{2}\right) \\ K_3^{(1)} = hf^{(1)}\left(x_n + \frac{h}{2}, y_n^{(0)} + \frac{K_2^{(0)}}{2}, y_n^{(1)} + \frac{K_2^{(1)}}{2}\right) \\ K_4^{(0)} = hf^{(0)}\left(x_n + h, y_n^{(0)} + K_3^{(0)}, y_n^{(1)} + K_3^{(1)}\right) \\ K_4^{(1)} = hf^{(1)}\left(x_n + h, y_n^{(0)} + K_3^{(0)}, y_n^{(1)} + K_3^{(1)}\right) \end{cases}$$

2nd order ODE: numerical solutions

- **Taylor method (2nd order) - Stormer-Verlet**

  using the numerical approximation for the 2nd-order derivative:

$$\left.\frac{d^2y}{dt^2}\right|_n \simeq \frac{y_{n+1}-2y_n+y_{n-1}}{(\delta t)^2} + O[(\delta t)^2]$$

  the differential equation becomes:

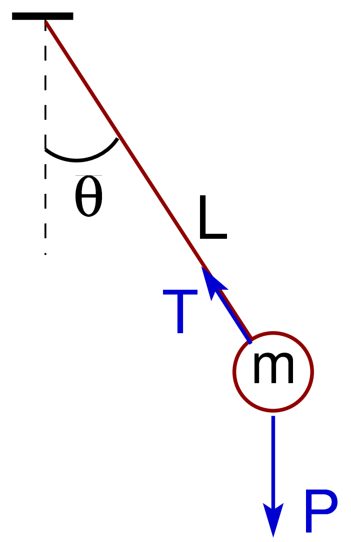
$$\boxed{\left.\frac{d^2y}{dt^2}\right|_n = f[t_n, y(t_n)] \quad \Rightarrow \quad y_{n+1} = -y_{n-1} + 2y_n + (\delta t)^2\, f[t_n, y(t_n)] + O[(\delta t)^4]}$$

- algorithm

time:	$\delta t = (t_f - t_0)/n$
initial conditions:	$t(0) \equiv t_0$
	$y(0) \equiv y_0$
	$\left.\frac{dy}{dt}\right\|_{t=0} \equiv \left(\frac{dy}{dt}\right)_0$
first iteration (n=0):	$y_1 = 2\,y_0 - y_{-1} + (\delta t)^2\, f[t_n, y(t_n)]$
	$y_{-1} \equiv y(x-h) \simeq y(x) - h\left.\frac{dy}{dt}\right\|_x + \frac{h^2}{2}\left.\frac{d^2y}{dt^2}\right\|_x = y_0 - hv_0 + \frac{h^2}{2}\, f[t_0, y(t_0)]$
following iterations (n=1,...):	$y_{n+1} = -y_{n-1} + 2y_n + (\delta t)^2\, f(t_n, y_n)$
	$t_{n+1} = t_0 + (n+1)\delta t$

simple pendulum

pendulum motion



Lagrangian: $\mathcal{L} = \frac{1}{2}m\ell^2\dot\theta^2 + mg\ell\cos\theta$

eq. of motion: $\frac{d^2\theta}{dt^2} + \frac{g}{\ell}\sin\theta = 0$

initial conditions: $\theta(0) = \theta_0 \quad \dot\theta(0) = \omega_0$

simplify equation

characteristic time: $t_c = \sqrt{\frac{\ell}{g}}$

variable change: $\tau = \frac{t}{t_c}$

$$\boxed{\frac{d^2\theta}{d\tau^2} + \sin\theta = 0}$$

reduce to a system of 1st-order diff equations

$$\left\{\begin{array}{l} \frac{d\theta}{dt} = \omega \\ \frac{d\omega}{dt} = -\sin(\theta) \end{array}\right. \Rightarrow \left\{\begin{array}{lll} \frac{dx_1}{dt} = & x_2 = f_1(t, x_1, x_2) \\ \frac{dx_2}{dt} = & -\sin(x_1) = f_2(t, x_1, x_2) \end{array}\right.$$

Solution methods:

- Euler
- Euler-Cromer
- Euler-Verlet
- Runge-Kutta

simple pendulum: solutions

Euler method

velocity is computed at the beginning of interval $(t, t+\Delta t)$

$$\begin{cases} x_{2,n+1} = x_{2,n} + (\Delta t)\ f_2(x_{1,n}) \\ x_{1,n+1} = x_{1,n} + (\Delta t)\ f_1(x_{2,n}) \end{cases}$$

$$\boxed{\begin{aligned} \omega_{n+1} &= \omega_n + (\Delta t)\,[-\sin\theta_n] \\ \theta_{n+1} &= \theta_n + (\Delta t)\,[\omega_n] \end{aligned}}$$

Euler-Cromer method

coordinate uses velocity computed at the end of the interval $(t, t+\Delta t)$ improving behavior

$$\begin{cases} x_{2,n+1} = x_{2,n} + (\Delta t)\ f_2(x_{1,n}) \\ x_{1,n+1} = x_{1,n} + (\Delta t)\ f_1(x_{2,\mathbf{n+1}}) \end{cases}$$

$$\boxed{\begin{aligned} \omega_{n+1} &= \omega_n + (\Delta t)\,[-\sin\theta_n] \\ \theta_{n+1} &= \theta_n + (\Delta t)\,[\omega_{n+1}] \end{aligned}}$$

Euler-Verlet method

it uses the 2nd-derivative operator $\ddot{y} = (\Delta t)^{-2}\ (y_{n+1} - 2y_n + y_{n-1}) + O[(\Delta t)^2]$

$$\begin{cases} \theta_{n+1} = 2\ \theta_n - \theta_{n-1} + (\Delta t)^2\ f(\theta_n) \\ \omega_n = \frac{\theta n+1 - \theta n-1}{2\ (\Delta t)} \end{cases}$$

first iteration $n = 0$
$$\begin{cases} \theta_1 = \theta_0 - (\Delta t)\ \omega_0 + \frac{(\Delta t)^2}{2}\ f(\theta_0) \\ \omega_0 \end{cases}$$

next iterations $n = 1, \ldots, N-1$
$$\begin{cases} \theta_2 = 2\theta_1 - \theta_0 + (\Delta t)^2\ f(\theta_1) \\ \omega_1 = \frac{\theta_2 - \theta_0}{2(\Delta t)} \end{cases}$$

simple pendulum: solutions (cont.)

Runge-Kutta method (4th order)

$$\begin{aligned} K_{1_1} &= f_1(t_n,\ x_{1,n},\ x_{2,n}) \\ K_{1_2} &= f_2(t_n,\ x_{1,n},\ x_{2,n}) \end{aligned}$$

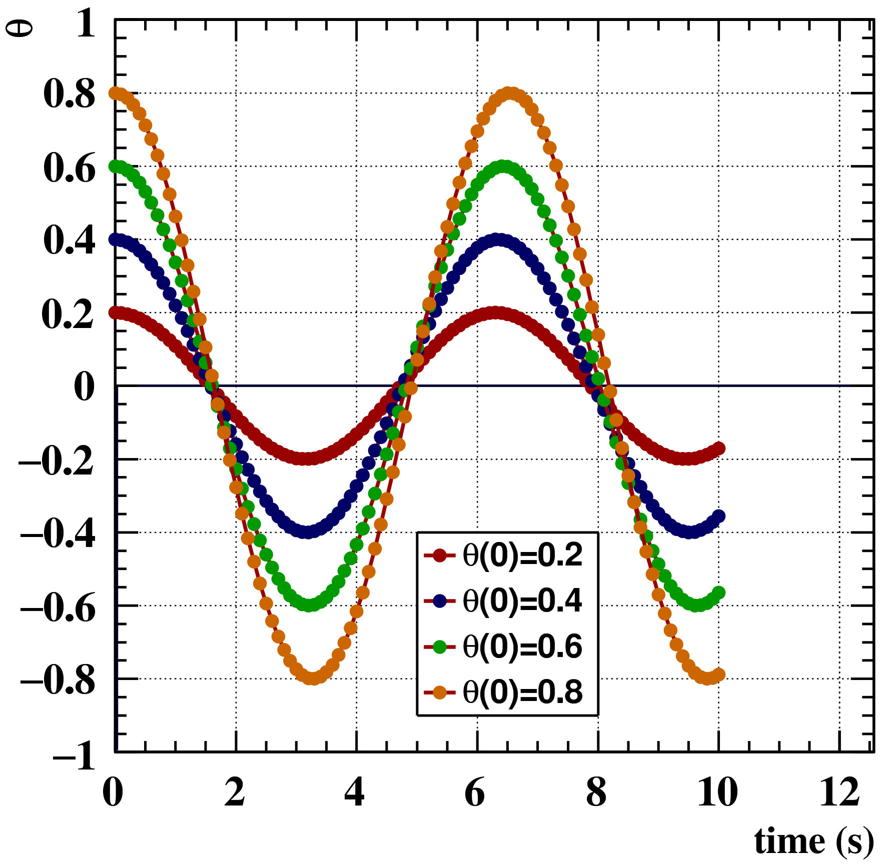
$$\begin{aligned} K_{2_1} &= f_1\left(t_n + \tfrac{h}{2},\ x_{1,n} + \tfrac{h}{2}K_{11},\ x_{2,n} + \tfrac{h}{2}K_{12}\right) \\ K_{2_2} &= f_2\left(t_n + \tfrac{h}{2},\ x_{1,n} + \tfrac{h}{2}K_{11},\ x_{2,n} + \tfrac{h}{2}K_{12}\right) \end{aligned}$$

$$\begin{aligned} K_{3_1} &= f_1\left(t_n + \tfrac{h}{2},\ x_{1,n} + \tfrac{h}{2}K_{21},\ x_{2,n} + \tfrac{h}{2}K_{22}\right) \\ K_{3_2} &= f_2\left(t_n + \tfrac{h}{2},\ x_{1,n} + \tfrac{h}{2}K_{21},\ x_{2,n} + \tfrac{h}{2}K_{22}\right) \end{aligned}$$

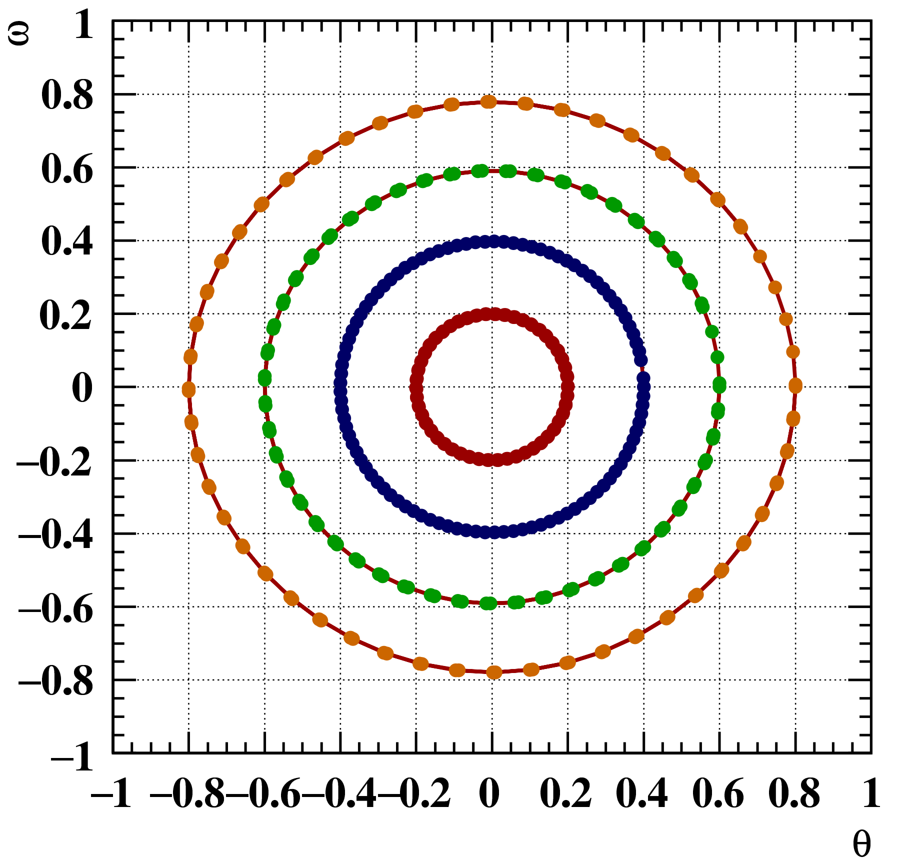
$$\begin{aligned} K_{4_1} &= f_1\,(t_n + h,\ x_{1,n} + hK_{31},\ x_{2,n} + hK_{32}) \\ K_{4_2} &= f_2\,(t_n + h,\ x_{1,n} + hK_{31},\ x_{2,n} + hK_{32}) \end{aligned}$$

$$\boxed{\begin{aligned} x_{1,n+1} &= x_{1,n} + \frac{h}{6}\left(K_{1_1} + 2K_{2_1} + 2K_{3_1} + K_{4_1}\right) \\ x_{2,n+1} &= x_{2,n} + \frac{h}{6}\left(K_{1_2} + 2K_{2_2} + 2K_{3_2} + K_{4_2}\right) \end{aligned}}$$

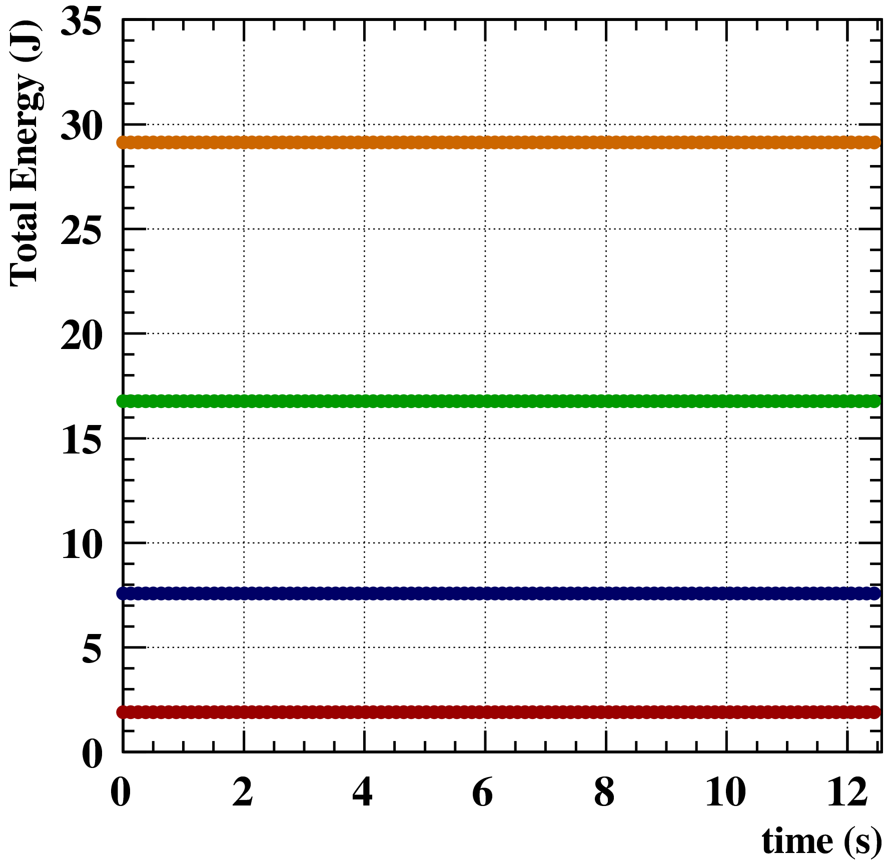
simple pendulum: RK4 solution



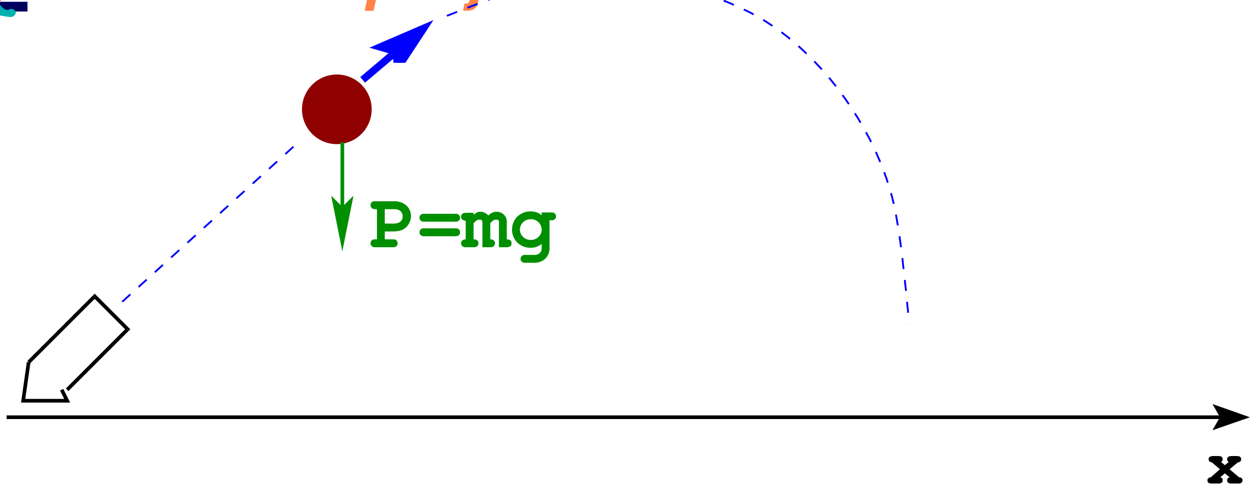
simple pendulum: RK4 solution



simple pendulum: RK4 solution



projectile motion



- ✔ Forces:
  - gravitic: $\vec{P} = m\,\vec{g}$
  - friction: $\vec{F}_a = c\,v^2\,\left(\frac{\vec{v}}{v}\right)$
- ✔ Initial conditions
  - position: $(x, y) = (0, 0)$
  - velocity: $\vec{v}_0$

projectile motion: multi-dimensions

- ✔ equation of motion

$$\frac{d^2\vec{r}}{dt^2} = -mg\vec{e}_y$$

- ✔ in terms of coordinates

$$\begin{aligned} \frac{d^2x}{dt^2} &= 0 \\ \frac{d^2y}{dt^2} &= -mg \end{aligned}$$

- ✔ system of 1st-order differential equations

$$\begin{aligned} \frac{dx}{dt} &= v_x \\ \frac{dy}{dt} &= v_y \\ \frac{dv_x}{dt} &= 0 \\ \frac{dv_y}{dt} &= -mg \end{aligned}$$

renaming variables

$x, y, v_x, v_y \;\rightarrow\; x_1, x_2, x_3, x_4$

$$\begin{aligned} \frac{dx_1}{dt} &= x_3 = & f_1(t, x_1, x_2, x_3, x_4) \\ \frac{dx_2}{dt} &= x_4 = & f_2(t, x_1, x_2, x_3, x_4) \\ \frac{dx_3}{dt} &= 0 = & f_3(t, x_1, x_2, x_3, x_4) \\ \frac{dx_4}{dt} &= -mg = & f_4(t, x_1, x_2, x_3, x_4) \end{aligned}$$

ODEsolver

- Runge–Kutta
- Euler
- Euler–Verlet

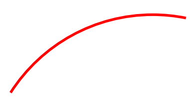
needs

- number variables
- functions

Library?

Physics Problem

eqs of motion



user problem

ODEsolver class

ODEsolver class

constructor

```
ODEsolver(const vector<TF1>&);
```

methods

```
void SetODEfunctions(const vector<TF1>&);
vector<ODEpoint> RK4(ODEpoint i,
                     double step,
                     double time);
```

ODEpoint class

OPEpoint class

```
class ODEpoint {
 public:
  ODEpoint(int fndim = 0, const double* x = nullptr);
  ODEpoint(const ODEpoint&);
  ~ODEpoint();

  int Ndim() const {return ndim;};
  double T() const {return x[0];};
  double X(int i) const {return x[i+1];};
  double* GetArray() { return x;}

  ODEpoint operator*(double) const;
  ODEpoint operator+(const ODEpoint&) const;
  ODEpoint operator-(const ODEpoint&) const;

  void operator=(const ODEpoint&);

  const double& operator[] (int) const;
  double& operator[] (int);

private:
  int ndim; //nb of dependent variables
  double* x; //independent variable + dependent variables (ndim+1)
};
```


ODEsolver class

OPEpoint class

```
class ODEsolver {
 public:
  ODEsolver(const vector<TF1>&);
  ~ODEsolver();

  SetODEfunc(const vector<TF1>&);

  vector<ODEpoint> Euler(ODEpoint i, double step, double T);
  vector<ODEpoint> PredictorCorrector(ODEpoint i, double step, double T);
  vector<ODEpoint> LeapFrog(ODEpoint i, double step, double T);

  vector<ODEpoint> RK2(ODEpoint i, double step, double T);
  vector<ODEpoint> RK4(ODEpoint i, double step, double T);

  (...)

private:
  vector<TF1> F;
  (...)
};
```

Runge-Kutta: adaptive step

✔ How to find the most appropriate step size?

- a step $\mathbf{h}$ too large leads to a big truncation error
- a step $\mathbf{h}$ too small leads to waste of computer resources

✔ We need an adaptive method to have an estimate of the truncation error and adjust the step size

- two embedded integration formulas of order $\mathbf{m}$ and $\mathbf{m}+\mathbf{1}$ will be used to evaluate error

$$\boxed{E^{(i)}(h) = y^{(i)}_{m+1}(x+h) - y^{(i)}_{m}(x+h)}$$

Runge-Kutta-Fehlberg formulas of orders $\mathbf{5}$ and $\mathbf{4}$

a set of 1st-order equations to solve $\frac{dy^{(i)}}{dt} = f^{(i)}(t, y^{(i)})$ [$\mathbf{i}$ dependent variables]

$$K^{(i)}_1 = h\ f^{(i)}\left(x, y^{(i)}\right)$$

$$K^{(i)}_{u+1} = h\ f^{(i)}\left(x + h\,A_u, y^{(i)} + \sum_{\ell=0}^{u-1} B_{u\ell} K_{\ell+1}\right)$$

$[u = 1, 2, 3, 4, 5]$

$$y^{(i)}_5(x+h) = y^{(i)}(x) + \sum_{u=0}^{5} C_u K^{(i)}_{u+1} \quad \text{(5th-order)}$$

$$y^{(i)}_4(x+h) = y^{(i)}(x) + \sum_{u=0}^{5} D_u K^{(i)}_{u+1} \quad \text{(4th-order)}$$

Cash and Karp coefficients

$$\boxed{E^{(i)}(h) = y^{(i)}_5(x+h) - y^{(i)}_4(x+h) = \sum_{u=0}^{5} (C_u - D_u)\, K^{(i)}_{u+1}}$$

errors on dependent variables $y^{(i)}$

error estimate

u	A_u	$B_{u\ell}$					C_u	D_u
0	-	-	-	-	-	-	$\frac{37}{378}$	$\frac{2825}{27648}$
1	$\frac{1}{5}$	$\frac{1}{5}$	-	-	-	-	0	0
2	$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$	-	-	-	$\frac{250}{621}$	$\frac{18575}{48384}$
3	$\frac{3}{5}$	$\frac{3}{10}$	$-\frac{9}{10}$	$\frac{6}{5}$	-	-	$\frac{125}{594}$	$\frac{13525}{55296}$
4	1	$-\frac{11}{54}$	$\frac{5}{2}$	$-\frac{70}{27}$	$\frac{35}{27}$	-	0	$\frac{277}{14336}$
5	$\frac{7}{8}$	$\frac{1631}{55296}$	$\frac{175}{512}$	$\frac{575}{13824}$	$\frac{44275}{110592}$	$\frac{253}{4096}$	$\frac{512}{1771}$	$\frac{1}{4}$

adaptive step

- suppose a system of $\mathbf{k}$ 1st-order equations to solve corresponding to $\mathbf{k}$ dependent variables $\mathbf{y}^{(\mathbf{i})}$
- an error estimate at step $\mathbf{h}$ can be provided by,

$$\varepsilon(h) = \sqrt{\frac{1}{k}\sum_{i=0}^{k-1} E_i^2(h)}$$

- Given that the error corresponds to the fourth-order formula, the ratio between errors with different step sizes,

$$\frac{\varepsilon(h_1)}{\varepsilon(h_2)} = \left(\frac{h_1}{h_2}\right)^5$$

- Last step $\mathbf{h_n}$ provided us with an error estimate of $\varepsilon(h_n)$
- an envisaged error tolerance of δ is required
  for instance $\delta = 10^{-6}$
- For reaching that tolerance error
  $\varepsilon(h_{n+1}) = \delta$
  we need for the next step a size of,

$$\boxed{h_{n+1} = \alpha\ h_n \left[\frac{\delta}{\varepsilon(h_n)}\right]^{1/5}}$$

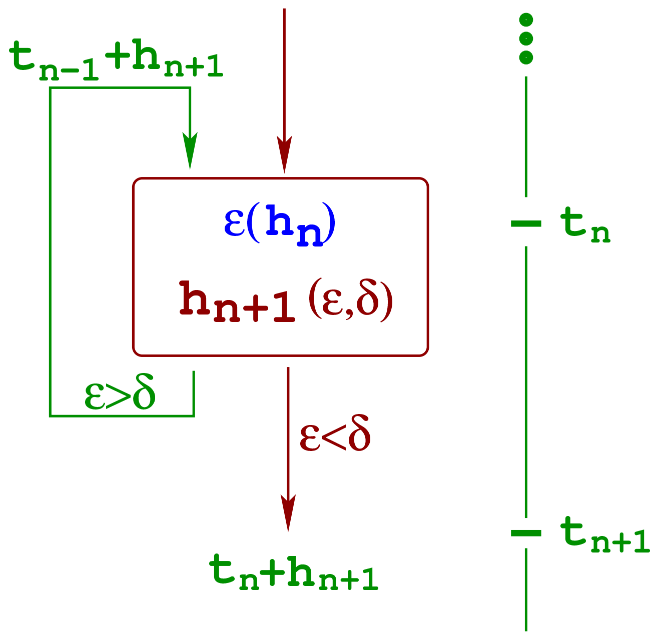
α, safety margin (0.9)

adaptive step algorithm

- At a given grid point, iterate dependent variables $\mathbf{y}^{(\mathbf{i})}$ using 4th and 5th order Runge-Kutta-Fehlberg algorithms
  Compute the error on current iteration by using the 5th order as a reference

$$\boxed{\varepsilon(h) = \sqrt{\frac{1}{m}\sum_{i=0}^{m-1} E_i^2(h)}}$$

- Compute next step size taking into account error tolerance
  $\mathbf{h_0}$ has to be provided at beginning



1. If the error estimate $\varepsilon(h)$ exceeds the prescribed tolerance δ,
   - a smaller step size h_{n+1} shall be assigned,
   - a new grid point $t_n = t_{n-1} + h_{n+1}$ computed and
   - the preceding iteration repeated
2. If the error estimate $\varepsilon(h)$ is smaller than the prescribed tolerance δ, the next grid point $t_{n+1} = t_n + h_{n+1}$ is defined and a new iteration made

adimensional equations

- In a differential equation involving observables like masses, velocities and time for instance, the arbitrary choice of the units can cause problems for numerical calculations
- We shall write the equations invariants to unit changes, i.e. written as function of adimensional observables

  find the characteristic scales of the problem like for length L_c and velocity V_c and define the reduced variables:

  $v = V_c\, v_0$ (v_0, adimensional velocity)

  $\ell = L_c\, \ell_0$
- radioactive decay example:

  $dN = -p\, dt\, N = -\frac{dt}{\tau} N \quad \Rightarrow \quad \frac{dN}{dt} = -\frac{N}{\tau}$

  logistic population model:

  $dN = +\kappa\, dt\, N - \frac{\kappa dt}{a^2} N^2 \quad \Rightarrow \quad \frac{dN}{dt} = +\kappa\, N - \frac{\kappa}{a^2} N^2$

ODE's : numerical solution optimization

Nondimensionalization

Decay equation

radioactive element : τ, decay time

$$\boxed{\frac{dN}{dt} = -\frac{N}{\tau}}$$

scaling variables: $t = t_c\, \hat{t}$, $N = N_c\, \hat{N}$

var. scales $\rightarrow N_c$

adimensional $\rightarrow \hat{N}$

$$\frac{N_c}{t_c}\frac{d\hat{N}}{d\hat{t}} = -\frac{N_c\,\hat{N}}{\tau}$$

$$\frac{\tau}{t_c}\frac{d\hat{N}}{d\hat{t}} = -\hat{N}$$

initial condition: $N(0) = N_0$

$\hat{N}(0) = \frac{N_0}{N_c}$

Scaling equations:

$$\Pi_1 = \frac{\tau}{t_c} = 1 \qquad \Pi_2 = \frac{N_0}{N_c} = 1$$

$$\boxed{\begin{aligned} t_c &= \tau \\ N_c &= N_0 \end{aligned}}$$ scales

Numerical equation to solve:

$$\boxed{\frac{d\hat{N}}{d\hat{t}} = -\hat{N}}$$

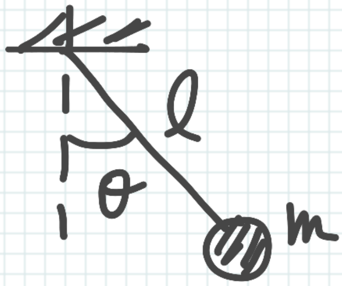
$$\hat{t}_n = \hat{t}_0 + (n+1)\,\Delta\hat{t}$$
$$t_n = t_c\,\hat{t}_m$$

$$\hat{N}_m$$
$$N_M = N_c\,\hat{N}_m$$

F. Barao

Pendulum

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



scaling variables:

$$t = t_c\,\hat{t}$$
$$\theta = \theta_c\,\hat{\theta}$$

$$\Rightarrow \frac{\theta_c}{t_c^2}\frac{d^2\hat{\theta}}{d\hat{t}^2} = -\frac{g}{\ell}\sin(\theta_c\,\hat{\theta})$$

$$\Rightarrow \left[\frac{\ell}{g}\frac{\theta_c}{t_c^2}\right]\frac{d^2\hat{\theta}}{d\hat{t}^2} = -\sin(\theta_c\,\hat{\theta})$$

initial conditions:

$$\theta(0) = \theta_0 \Rightarrow \hat{\theta}(0) = \frac{\theta_0}{\theta_c}$$

caracteristic time

$$\boxed{\begin{aligned} \theta_c &= \theta_0 \\ t_c &= \sqrt{\theta_0\,\ell/g} \end{aligned}}$$

Numerical equation to solve :

$$\boxed{\frac{d^2\hat{\theta}}{d\hat{t}^2} = -\sin(\theta_0\,\hat{\theta})}$$

$\hat{t}_n = \hat{t}_0 + (n+1)\,\Delta\hat{t}$

$t_n = \sqrt{\theta_0 \frac{\ell}{g}}\; t_n$

$\hat{\theta}_n$

$\theta_n = \theta_0\,\hat{\theta}_n$

velocity: $\omega = \frac{d\theta}{dt} = \frac{\theta_c}{t_c}\frac{d\hat{\theta}}{d\hat{t}}$

$\omega_n = \sqrt{\theta_0\, g/\ell}\;\hat{\omega}_n$