Ordindary Differential Equations

Master 2 Acoustical Engineering
Numerical Techniques for Acoustics - Session 1

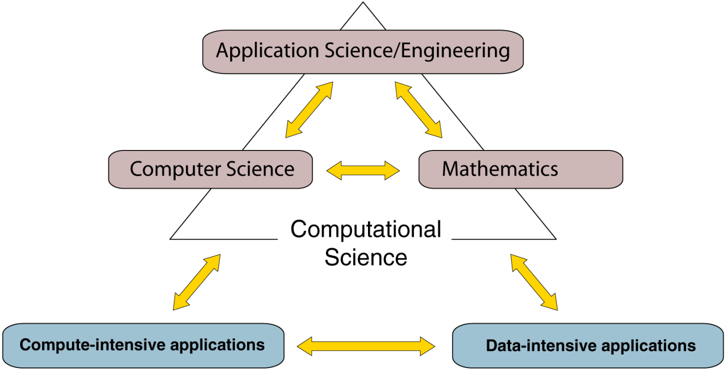
Matthieu Aussal*

*Centre de Mathématiques Appliquées de l'École Polytechnique
Route de Saclay - 91128 Palaiseau CEDEX France

Monday 23 September 2019 - ENSTA

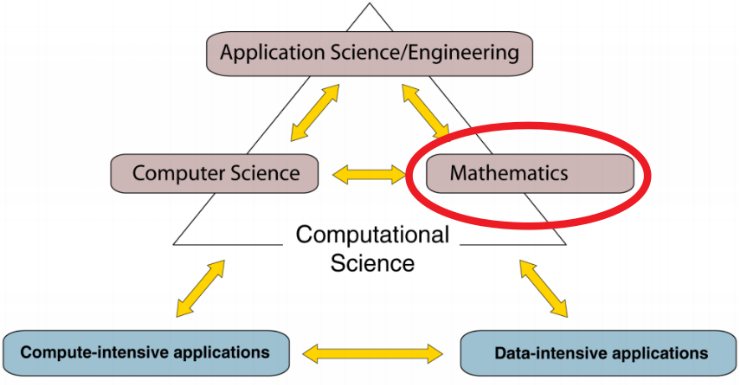
Computational Science Fuses Three Distinct Elements:

5



Computational Science Fuses Three Distinct Elements:

5



ODE / PDE ?

An Ordinary Differential Equations (ODE) is on the form :

$$\sum_i \alpha_i \frac{d^{(n_i)} U(x)}{dx^{(n_i)}} = F(x, U(x))$$

with U is an unknown **function** and x a **variable**.

A Partial Differential Equations (PDE) is on the form :

$$\sum_i \alpha_i \frac{\partial^{(n_i)} U(\mathbf{x})}{\partial x_i^{(n_i)}} = F(\mathbf{x}, U(\mathbf{x}))$$

with U is an unknown **function** and $\mathbf{x} = (x_1, \ldots, x_n)$ are **variableS**.

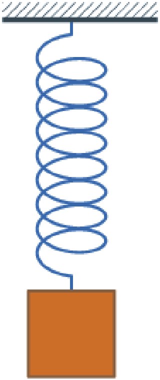
NB : These equations are always delivered with their initial conditions !

Example 1 : Mass-spring system (EDO)

Newton principle conduct to :

$$m\frac{d^2z(t)}{dt^2} = -c\frac{dz(t)}{dt} - kx(t)$$

with $z(t)$ the position of the spring, m the mass, c the damping factor and k the stiffness.



(fr) *https ://fr.wikipedia.org/wiki/Exemples_d'équations_différentielles*

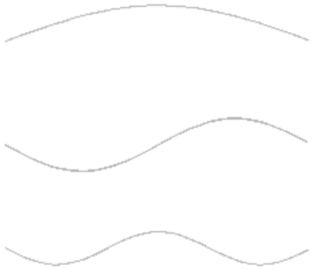
(en) *https ://en.wikipedia.org/wiki/Examples_of_differential_equations*

Examples 2 : Vibration modes of a string (EDO)

Eigen-values equation :

$$\frac{d^2u(x)}{dx^2} = -k^2u(x)$$

with $u(t)$ is the string amplitude around initial position and k is the wave number.



(fr) *https ://fr.wikipedia.org/wiki/Onde_sur_une_corde_vibrante*

(en) *https ://en.wikipedia.org/wiki/String_vibration*

Examples 3 : Predator-Prey equations (EDO)

Lotka-Volterra equations :

$$\begin{aligned} \frac{dx(t)}{dt} &= \alpha x(t) - \beta x(t) y(t) \\ \frac{dy(t)}{dt} &= \delta x(t) y(t) - \gamma y(t) \end{aligned}$$

with $x(t)$ are the prey and $y(t)$ the predator.

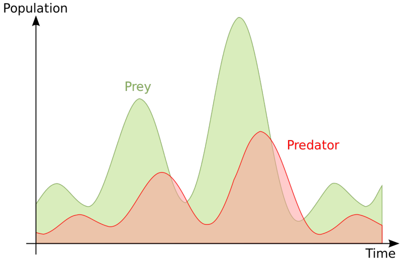
- Prey have an exponential growth : $\alpha x(t)$
- Predator have a natural death : $\gamma y(t)$
- Predator-prey interaction : $x(t)y(t)$

(fr) *https ://fr.wikipedia.org/wiki/équations_de_prédation_de_Lotka-Volterra*
(en) *https ://en.wikipedia.org/wiki/Lotka-Volterra_equations*

Examples 3 : Predator-Prey equations (EDO)

Lotka-Volterra equations :

$$\begin{aligned} \frac{dx(t)}{dt} &= \alpha x(t) - \beta x(t) y(t) \\ \frac{dy(t)}{dt} &= \delta x(t) y(t) - \gamma y(t) \end{aligned}$$



with $x(t)$ are the prey and $y(t)$ the predator.

(fr) *https ://fr.wikipedia.org/wiki/équations_de_prédation_de_Lotka-Volterra*

(en) *https ://en.wikipedia.org/wiki/Lotka-Volterra_equations*

Examples 4 : Laplace equation (EDP)

The best known equation :

$$\begin{aligned} \frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} + \frac{\partial^2 \Phi}{\partial z^2} &= 0 \\ \Longleftrightarrow \ div(\mathbf{grad}(\Phi)) &= 0 \\ \Longleftrightarrow \ \nabla \cdot (\nabla \Phi) &= 0 \\ \Longleftrightarrow \ \nabla^2(\Phi) &= 0 \\ \Longleftrightarrow \ \Delta \Phi &= 0 \end{aligned}$$

with $\Phi(x, y, z)$ is a function that we find in astronomy, electro-static, fluid mechanics, heat propagation, quantum mechanic, and so on.

(fr) *https ://fr.wikipedia.org/wiki/Equation_de_Laplace*

(en) *https ://en.wikipedia.org/wiki/Laplace's_equation*

Examples 5 : Convection diffusion (EDP)

The transport equation with a diffuse term :

$$\frac{\partial u}{\partial t} = \nabla \cdot (D \nabla u) - \nabla \cdot (\mathbf{c} u) + R$$

with $u(x, y, z)$ is the function of interest (mass, temperature, etc.), D is the diffusion coefficient, $\mathbf{c}$ is the motion celerity and R a source.

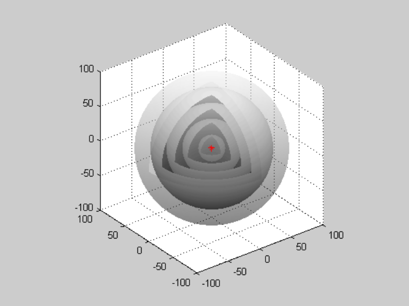
https ://en.wikipedia.org/wiki/Convection-diffusion_equation

Examples 6 : Wave equation (EDP)

The wave equation :

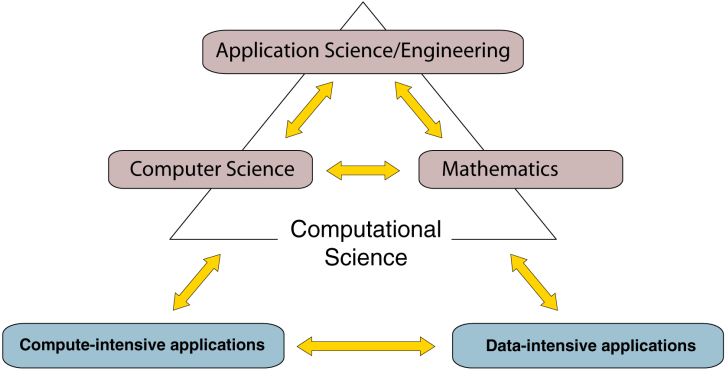
$$\frac{\partial^2 u}{\partial^2 t} = c^2 \nabla^2 u$$

with $u(x, y, z)$ is the function of interest (pressure, amplitude, etc.) and **c** is the celerity (scalar, fixed).



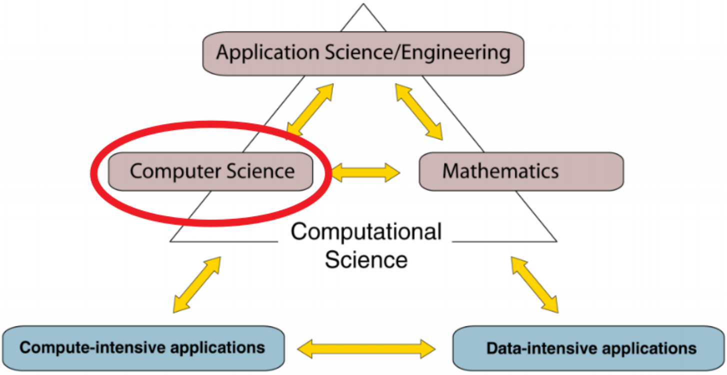
https ://en.wikipedia.org/wiki/Wave_equation

Computational Science Fuses Three Distinct Elements:



Computational Science Fuses Three Distinct Elements:

5



Gerald Recktenwald - Portland State University

Numerical Integration of Ordinary Differential Equations for Initial Value Problems

Gerald Recktenwald
Portland State University
Department of Mechanical Engineering
gerry@me.pdx.edu

Numerical Integration

Numerical Integration of First Order ODEs (1)

The generic form of a first order ODE is

$$\frac{dy}{dt} = f(t, y); \qquad y(0) = y_0$$

where the right hand side $f(t, y)$ is any single-valued function of t and y.

The approximate numerical solution is obtained at discrete values of t

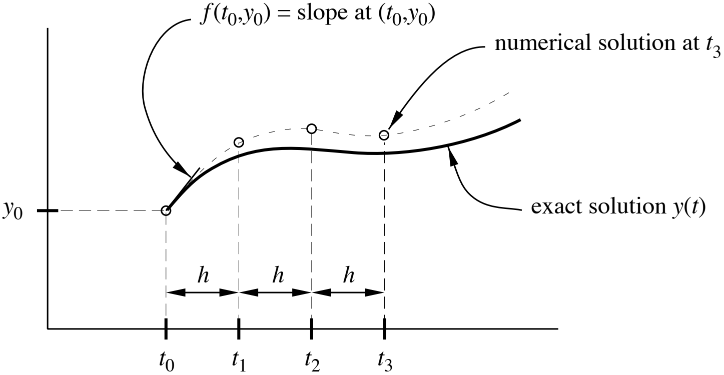
$$t_j = t_0 + jh$$

where h is the "stepsize"

Numerical Integration

Numerical Integration of ODEs (2)

Graphical Interpretation



Forward Euler scheme

Euler's Method (1)

Consider a Taylor series expansion in the neighborhood of t_0

$$y(t) = y(t_0) + (t - t_0) \left.\frac{dy}{dt}\right|_{t_0} + \frac{(t - t_0)^2}{2} \left.\frac{d^2y}{dt^2}\right|_{t_0} + \ldots$$

Retain only first derivative term and define

$$f(t_0, y_0) \equiv \left.\frac{dy}{dt}\right|_{t_0}$$

to get

$$y(t) \approx y(t_0) + (t - t_0) f(t_0, y_0)$$

Forward Euler scheme

Euler's Method (2)

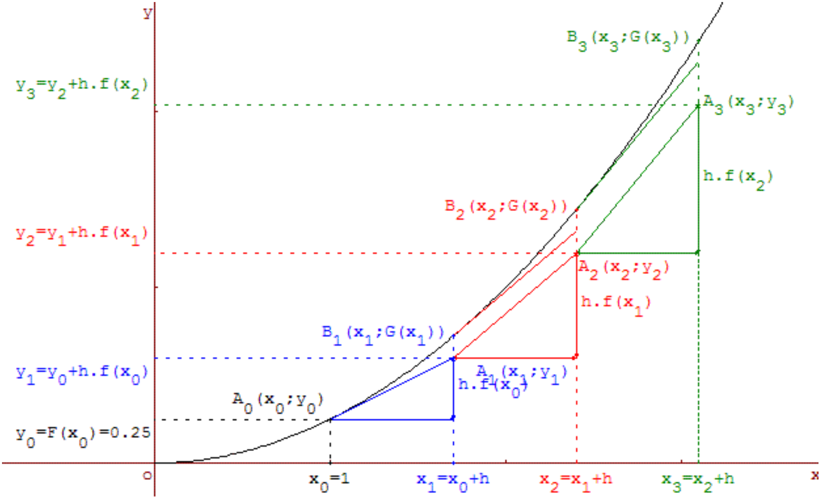
Given $h = t_1 - t_0$ and initial condition, $y = y(t_0)$, compute

$$\begin{aligned} y_1 &= y_0 + h\, f(t_0, y_0) \\ y_2 &= y_1 + h\, f(t_1, y_1) \\ &\vdots \\ y_{j+1} &= y_j + h\, f(t_j, y_j) \end{aligned}$$

or

$$\boxed{y_j = y_{j-1} + h\, f(t_{j-1}, y_{j-1})}$$

Forward Euler scheme



Numerical example

Example: Euler's Method

Use Euler's method to integrate

$$\frac{dy}{dt} = t - 2y \qquad y(0) = 1$$

The exact solution is

$$y = \frac{1}{4}\left[2t - 1 + 5e^{-2t}\right]$$

j	t_j	$f(t_{j-1}, y_{j-1})$	Euler $y_j = y_{j-1} + h\, f(t_{j-1}, y_{j-1})$	Exact $y(t_j)$	Error $y_j - y(t_j)$
0	0.0	NA	(initial condition) 1.0000	1.0000	0
1	0.2	$0 - (2)(1) = -2.000$	$1.0 + (0.2)(-2.0) = 0.6000$	0.6879	-0.0879
2	0.4	$0.2 - (2)(0.6) = -1.000$	$0.6 + (0.2)(-1.0) = 0.4000$	0.5117	-0.1117
3	0.6	$0.4 - (2)(0.4) = -0.400$	$0.4 + (0.2)(-0.4) = 0.3200$	0.4265	-0.1065

Exercice 1.a : Recompute numerically the four error values.

Numerical example

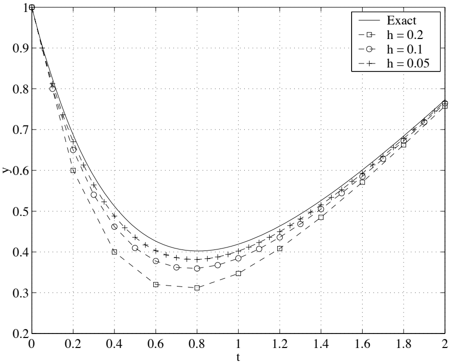
Reducing Stepsize Improves Accuracy (1)

Use Euler's method to integrate

$$\frac{dy}{dt} = t - 2y; \qquad y(0) = 1$$

for a sequence of smaller h (see `demoEuler`).

For a given h, the largest error in the numerical solution is the *Global Discretization Error* or *GDE*.



Exercice 1.b : Reproduce this figure.

Numerical example

Reducing Stepsize Improves Accuracy (2)

Local error at any time step is

$$e_j = y_j - y(t_j)$$

where $y(t_j)$ is the exact solution evaluated at t_j.

$$\mathrm{GDE} = \max(e_j), \qquad j = 1, \ldots$$

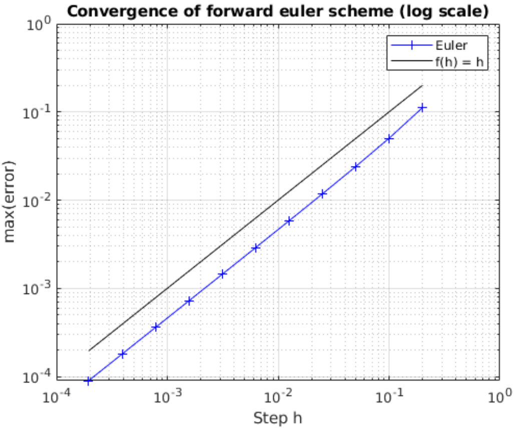
For Euler's method, GDE decreases linearly with h.

Here are results for the sample problem plotted on previous slide:

$$dy/dt = t - 2y; \quad y(0) = 1$$

h	$\max(e_j)$
0.200	0.1117
0.100	0.0502
0.050	0.0240
0.025	0.0117

Scheme order for Euler method



First order scheme : $e_j = y_j - y(t_j) \propto h$

Exercice 1.c : Reproduce this figure.

Midpoint scheme

Midpoint Method (1)

Increase accuracy by evaluating slope twice in each step of size h

$$k_1 = f(t_j, y_j)$$

Compute a tentative value of y at the midpoint

$$y_{j+1/2} = y_j + \frac{h}{2} f(t_j, y_j)$$

re-evaluate the slope

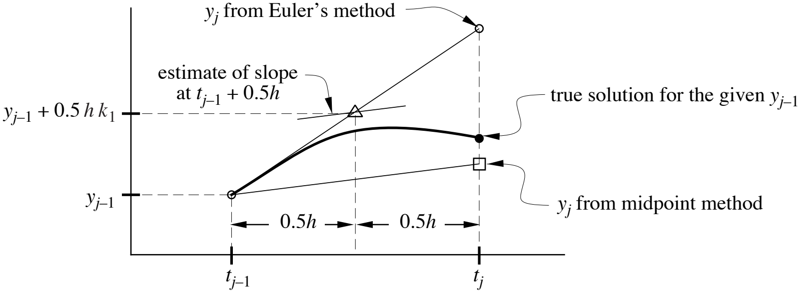
$$k_2 = f(t_j + \frac{h}{2}, y_j + \frac{h}{2} k_1)$$

Compute final value of y at the end of the full interval

$$y_{j+1} = y_j + h k_2$$

Midpoint scheme

Midpoint Method (2)



Heun's scheme

Heun's Method (1)

Compute the slope at the starting point

$$k_1 = f(t_j, y_j)$$

Compute a tentative value of y at the endpoint

$$y_j^* = y_j + hf(t_j, y_j)$$

re-evaluate the slope

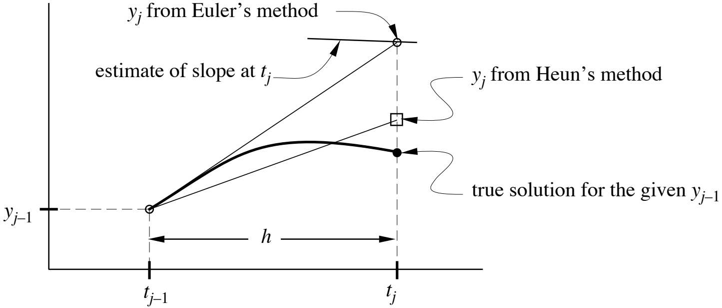
$$k_2 = f(t_j + h, y_j^*) = f(t_j + j, y_j + hk_1)$$

Compute final value of y with an average of the two slopes

$$y_{j+1} = y_j + h\,\frac{k_1 + k_2}{2}$$

Heun's scheme

Heun's Method (2)



Runge and Kutta 4 scheme

Runge-Kutta Methods

Generalize the idea embodied in Heun's method. Use a *weighted average of the slope* evaluated at multiple in the step

$$y_{j+1} = y_j + h \sum \gamma_m k_m$$

where γ_m are weighting coefficients and k_m are slopes evaluated at points in the interval $t_j \leq t \leq t_{j+1}$

In general,

$$\sum \gamma_m = 1$$

Runge and Kutta 4 scheme

Fourth Order Runge-Kutta

Compute slope at four places within each step

$$k_1 = f(t_j, y_j)$$

$$k_2 = f(t_j + \frac{h}{2}, y_j + \frac{h}{2}k_1)$$

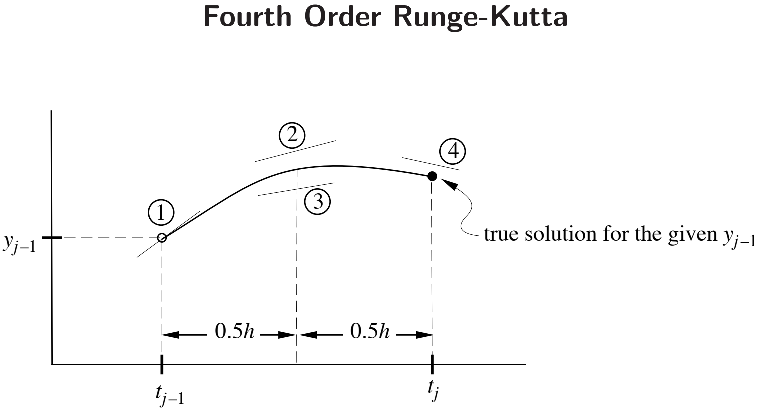
$$k_3 = f(t_j + \frac{h}{2}, y_j + \frac{h}{2}k_2)$$

$$k_4 = f(t_j + h, y_j + hk_3)$$

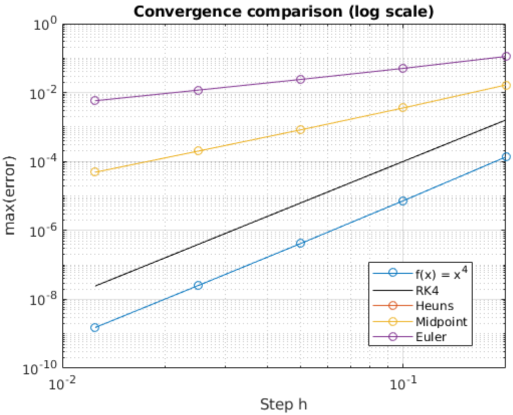
Use weighted average of slopes to obtain y_{j+1}

$$y_{j+1} = y_j + h\left(\frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}\right)$$

Runge and Kutta 4 scheme



Comparison of schemes order



First to fourth order schemes : $RK4 \propto h^4$

Exercice 1.d : Reproduce this figure.

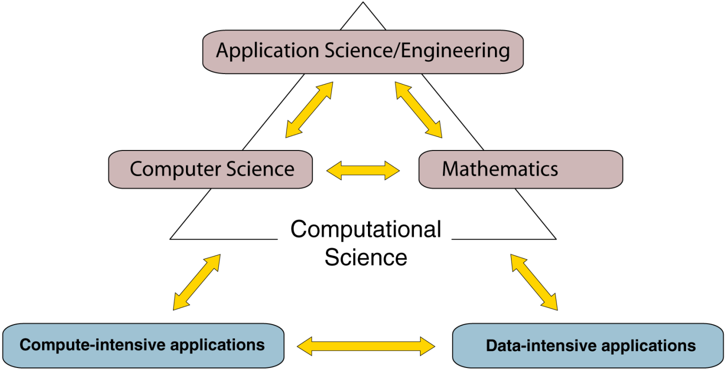
Native ODE set for matlab

MATLAB's Built-in ODE Routines

Function	Description
`ode113`	Variable order solution to nonstiff systems of ODEs. `ode113` uses an explicit predictor-corrector method with variable order from 1 to 13.
`ode15s`	Variable order, multistep method for solution to stiff systems of ODEs. `ode15s` uses an implicit multistep method with variable order from 1 to 5.
`ode23`	Lower order adaptive stepsize routine for non-stiff systems of ODEs. `ode23` uses Runge-Kutta schemes of order 2 and 3.
`ode23s`	Lower order adaptive stepsize routine for moderately stiff systems of ODEs. `ode23` uses Runge-Kutta schemes of order 2 and 3.
`ode45`	Higher order adaptive stepsize routine for non-stiff systems of ODEs. `ode45` uses Runge-Kutta schemes of order 4 and 5.

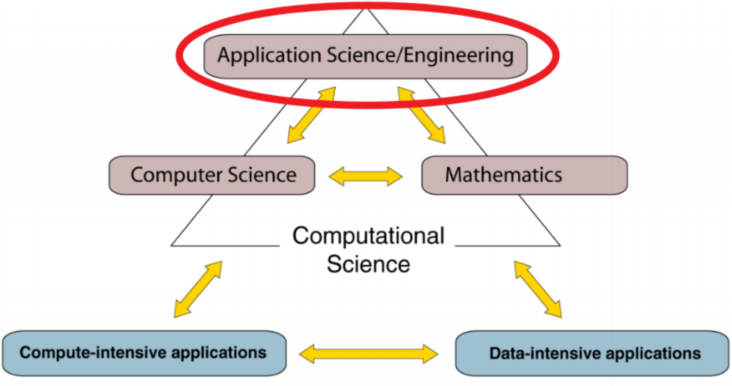
Computational Science Fuses Three Distinct Elements:

5



Computational Science Fuses Three Distinct Elements:

5



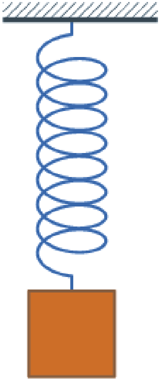
Mass-spring

Newton principle conduct to :

$$m\frac{d^2z(t)}{dt^2} = -c\frac{dz(t)}{dt} - kx(t)$$

with $z(t)$ the position of the spring, m the mass, c the damping factor and k the stiffness. Initial condition :

- $z(t=0) = z_0$,
- $v = \frac{dz(t)}{dt} = 0$.



(fr) *https ://fr.wikipedia.org/wiki/Exemples_d'équations_différentielles*
(en) *https ://en.wikipedia.org/wiki/Examples_of_differential_equations*

Mass-spring linearization

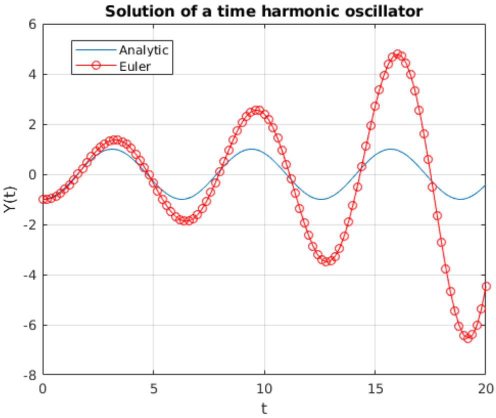
Considering $c = 0$ (no damping), we get a two equation system :

$$\begin{aligned} \frac{dv(t)}{dt} &= -\frac{k}{m} z(t) \\ \frac{dz(t)}{dt} &= v(t) \end{aligned}$$

Wich can be reformulated using matrix representation.
Anaytical solution is given by

$$z(t) = z_0 cos(\frac{k}{m} t)$$

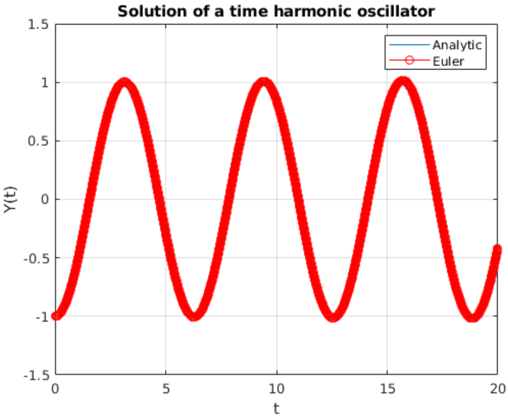
Mass-spring solutions



Euler 100 steps with $k = 1$, $m = 1$, $z0 = -1$.

Exercice 2.a : Reproduce this figure.

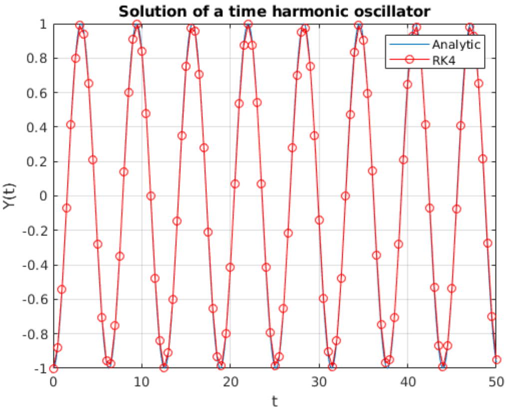
Mass-spring solutions



Euler 10000 steps with $k = 1$, $m = 1$, $z0 = -1$.

Exercice 2.b : Reproduce this figure.

Mass-spring solutions



RK4 100 steps with $k = 1$, $m = 1$, $z0 = -1$.

Exercice 2.c : Reproduce this figure.

Predator-prey

Lotka-Volterra equations :

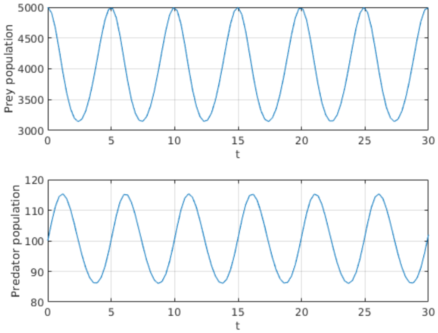
$$\begin{aligned}\frac{dx(t)}{dt} &= \alpha x(t) - \beta x(t)y(t)\\ \frac{dy(t)}{dt} &= \delta x(t)y(t) - \gamma y(t)\end{aligned}$$

with $x(t)$ are the prey and $y(t)$ the predator.

- Prey have an exponential growth : $\alpha x(t)$
- Predator have a natural death : $\gamma y(t)$
- Predator-prey interaction : $x(t)y(t)$

(fr) *https ://fr.wikipedia.org/wiki/équations_de_prédation_de_Lotka-Volterra*
(en) *https ://en.wikipedia.org/wiki/Lotka-Volterra_equations*

Predator-prey solutions



Predator-prey solution for $\alpha = 2$, $\beta = 0.02$, $\delta = 0.0002$ and $\gamma = 0.8$. Initial populations are 5000 prey and 100 predators. Computation wih RK4 scheme, 100 steps.

Exercice 3 : Reproduce this figure.

Furthers exercices (optional)

- **Mass-spring system :** What's happen with $c > 0$? Compare with analytical solutions.
- **Other's schemes :** Implement others shemes as Backward euler, variable steps schemes, multi-steps, RKC, etc.
- **String modelization :** Solve EDO from N-coupled oscillators *https ://en.wikipedia.org/wiki/Normal_mode*,