

Numerical solution of ODE

General formulation

Consider relation given a system of **first order** differential equation

Mechanical systems are often expressed as

- *scalar equation of second order in p*
- *system of first order in (p, v)*

In general, we can write

$$u'(t) = \mathcal{F}(u(t), t)$$

If $\mathcal{F}$ is an affine function in u

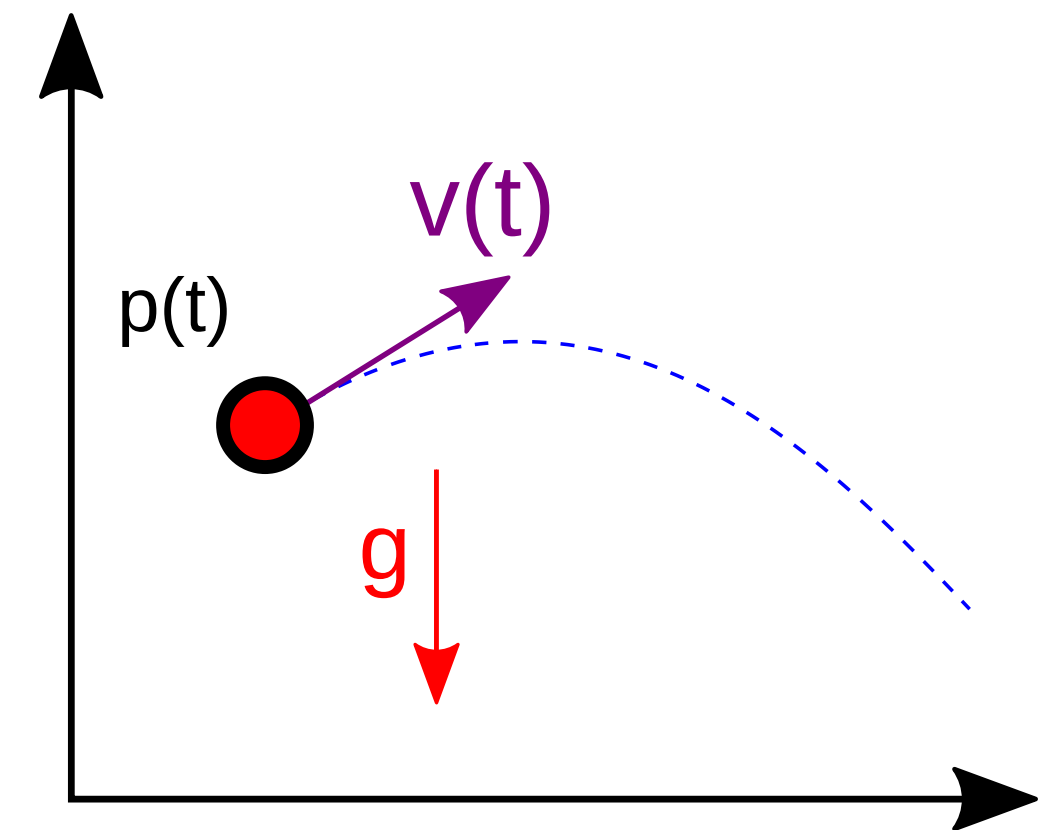
$$u'(t) = A(t) u(t) + b(t)$$

When A is constant through time

$$u'(t) = A u(t) + b(t)$$

Example: Free fall under gravity

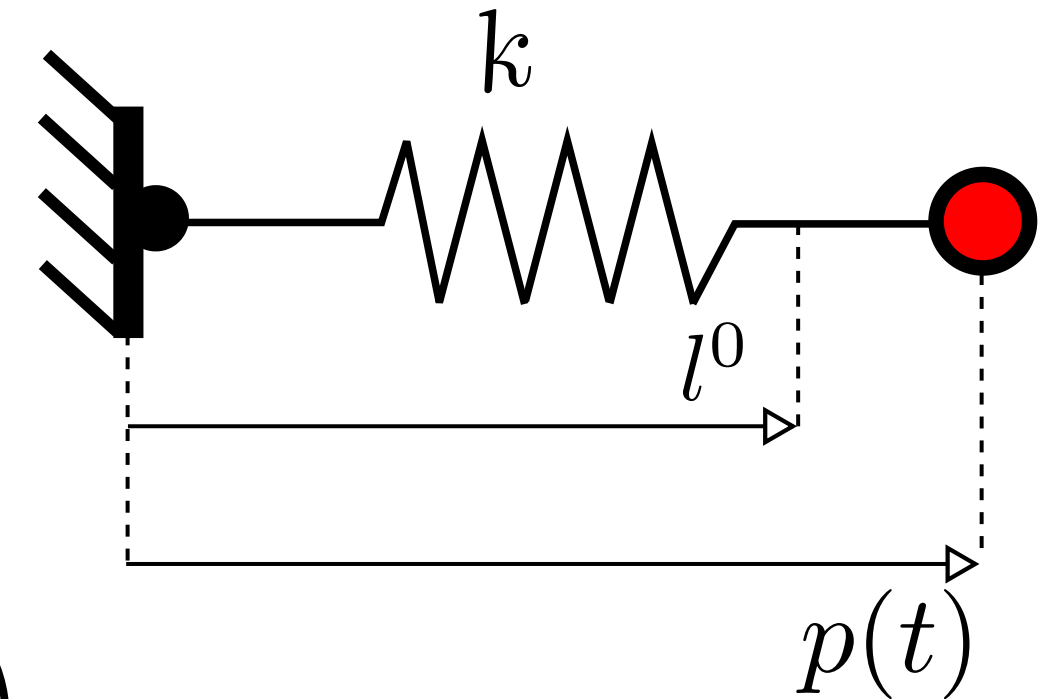
- Force $F(t) = m g$
- Second order differential equation: $p''(t) = g$
- First order system $\underbrace{\begin{pmatrix} p \\ v \end{pmatrix}}_{u'(t)}'(t) = \underbrace{\begin{pmatrix} v(t) \\ g \end{pmatrix}}_{\mathcal{F}(u,t)}$
- Linear system $\underbrace{\begin{pmatrix} p \\ v \end{pmatrix}}_{u'(t)}'(t) = \underbrace{\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}}_A \underbrace{\begin{pmatrix} p \\ v \end{pmatrix}}_{u(t)}(t) + \underbrace{\begin{pmatrix} 0 \\ g \end{pmatrix}}_{b(t)}$
- Exact solution known: $p(t) = \frac{1}{2} g t^2 + v_0 t + p_0$



Note: variables are vectors - matrix can be expressed by block, or per components.

Example: 1D spring (/Harmonic oscillator)

- Force $F(t) = -k(p(t) - l^0)$, k spring stiffness, l^0 rest length
- Second order differential equation: $m p''(t) + k p(t) = k l^0$



- First order system $\underbrace{\begin{pmatrix} p \\ v \end{pmatrix}}_{u'(t)}'(t) = \underbrace{\begin{pmatrix} v(t) \\ -k/m(p(t) - l^0) \end{pmatrix}}_{\mathcal{F}(u,t)}$
- Linear system $\underbrace{\begin{pmatrix} p \\ v \end{pmatrix}}_{u'(t)}'(t) = \underbrace{\begin{pmatrix} 0 & 1 \\ -k/m & 0 \end{pmatrix}}_A \underbrace{\begin{pmatrix} p \\ v \end{pmatrix}}_{u(t)}(t) + \underbrace{\begin{pmatrix} 0 \\ k/m l^0 \end{pmatrix}}_b$

- Exact solution known: $p(t) = A \sin(\omega t + \varphi) + l^0$

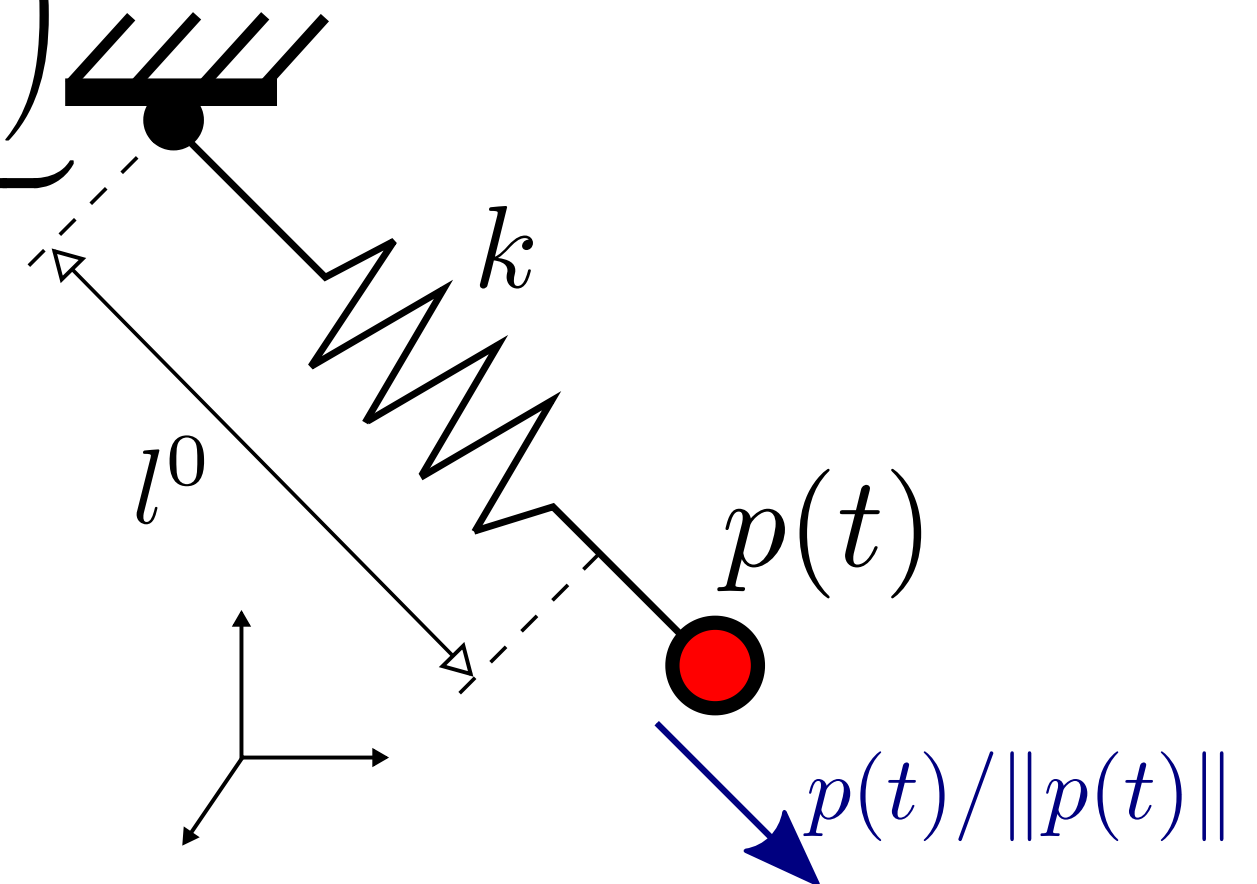
$$\omega = \sqrt{k/m}, A^2 = (p^0 - l^0)^2 + \left(\frac{v_0}{\omega}\right)^2, \tan(\varphi) = \frac{p^0 - l^0}{v^0/\omega}$$

Example: 3D mass spring

- Force $F(t) = m g - k (\|p(t)\| - l^0) \frac{p(t)}{\|p(t)\|}$, k spring stiffness, l^0 rest length
- Second order differential equation: $m p''(t) = m g - k (\|p(t)\| - l^0) \frac{p(t)}{\|p(t)\|}$

- First order system $\underbrace{\begin{pmatrix} p \\ v \end{pmatrix}}_{u'(t)}'(t) = \underbrace{\begin{pmatrix} v(t) \\ g - k/m (\|p(t)\| - l^0) \frac{p(t)}{\|p(t)\|} \end{pmatrix}}_{\mathcal{F}(u,t)}$

- Not linear
- No simple explicit solution



Existence of solution

Solving the physical system = Solving the **Initial Value Problem** (IVP)

- Find the solution of $u'(t) = \mathcal{F}(u, t), t \geq 0$
- With initial condition $u(t = 0) = u_0$

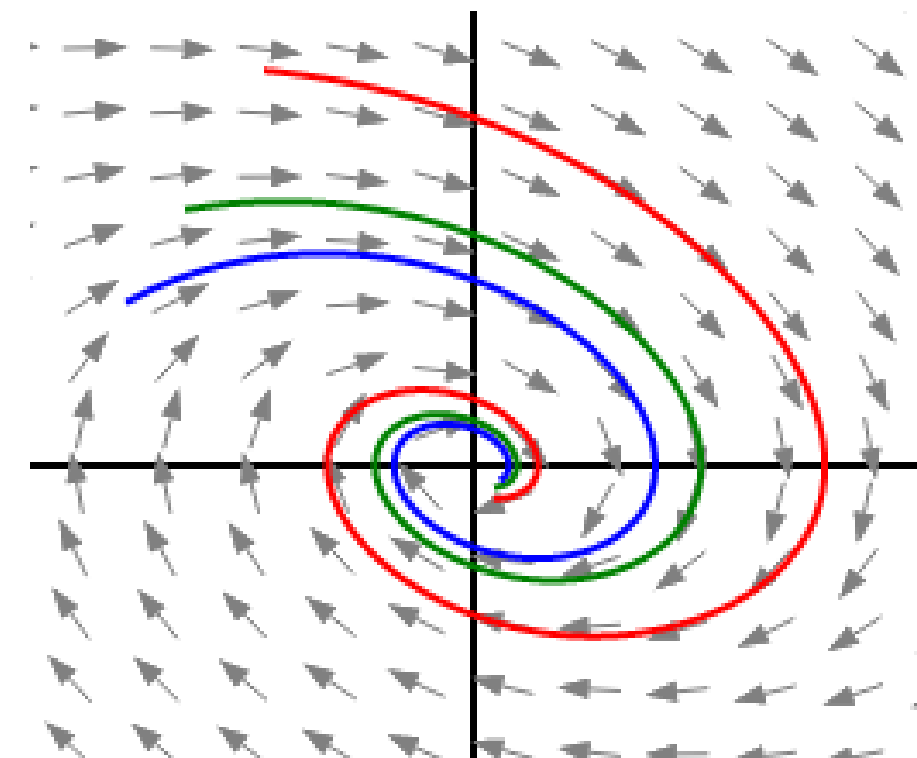
Cauchy-Lipschitz (/Picard-Lindelof) theorem states that

If $\mathcal{F}$ is Lipschitz with respect to u and continuous with respect to t

Then there exists a unique solution $u(t)$.

The solution is called the **integral curve** of the IVP.

- $\mathcal{F}$ can be seen as a vector field
(Vector field in 6D for $p(t) \in \mathbb{R}^3, v(t) \in \mathbb{R}^3$)
- Solution is a path along this vector field passing by u_0



Solving the ODE exactly

Equation is linear

Homogeneous, constant coefficients

$$u'(t) = A u(t), u(0) = u_0 \\ \Rightarrow u(t) = u_0 \exp(A t)$$

Non-Homogeneous, constant coefficients

$$u'(t) = A u(t) + b(t), u(0) = u_0 \\ \Rightarrow u(t) = u_0 \exp(A t) + \exp(A t) \int_{t'=0}^t b(t') \exp(-A t') dt'$$

Homogeneous, variable coefficients

$$u'(t) = A(t) u(t), u(0) = u_0 \\ \Rightarrow \text{No closed-form solution in the general case} \\ \text{- Rem. Unfortunately, in general, } u(t) \text{ is } \mathbf{not} u_0 \exp\left(\int_0^t A(t') dt'\right)$$

Equation is non linear

No general method

$\Rightarrow$ Numerical approaches are required most of the time

Numerical solution

1st order Explicit Euler

Naive numerical scheme: Approximation of the derivative

$$\frac{u^{k+1} - u^k}{h} = \mathcal{F}(u^k, t^k)$$

$$\Rightarrow u^{k+1} = u^k + h \mathcal{F}(u^k, t^k)$$

In the linear case

$$u^{k+1} = (\mathbf{I} + h \mathbf{A}) u^k + h b^k$$

Pro : very easy to implement

Is u^k a good approximation of the true solution $\tilde{u}(t^k)$?

Explicit Euler - study case

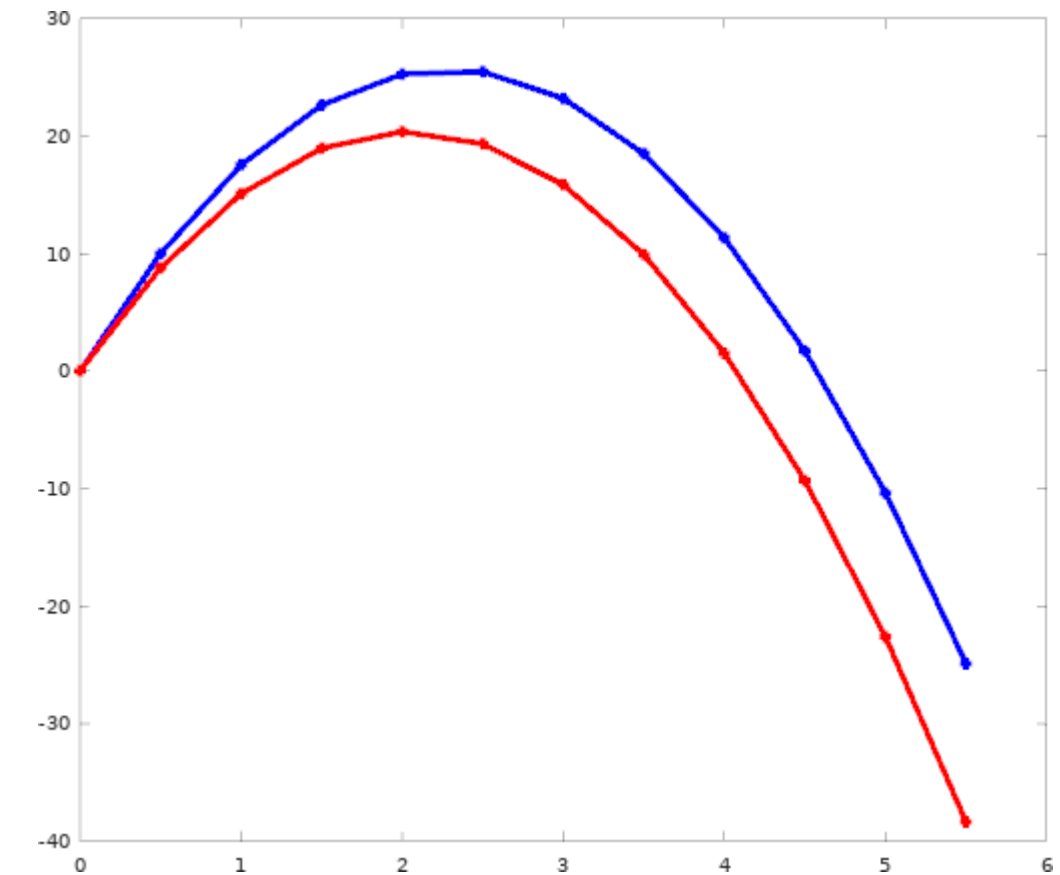
Free fall under gravity

- True solution $\tilde{u}(k h) = p_0 + (k h)v_0 + \frac{(k h)^2}{2} g$

- Numerical scheme: $\begin{pmatrix} p \\ v \end{pmatrix}^{k+1} = \begin{pmatrix} 1 & h \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p \\ v \end{pmatrix}^k + \begin{pmatrix} 0 \\ h g \end{pmatrix}$

- Numerical solution: $p^k = p_0 + k h v_0 + \frac{k(k-1)}{2} h^2 g$

$\Rightarrow$ **Not exact** : Error $e^k = |u^k - \tilde{u}(k h)| = \frac{g}{2} k h^2$



red: true solution

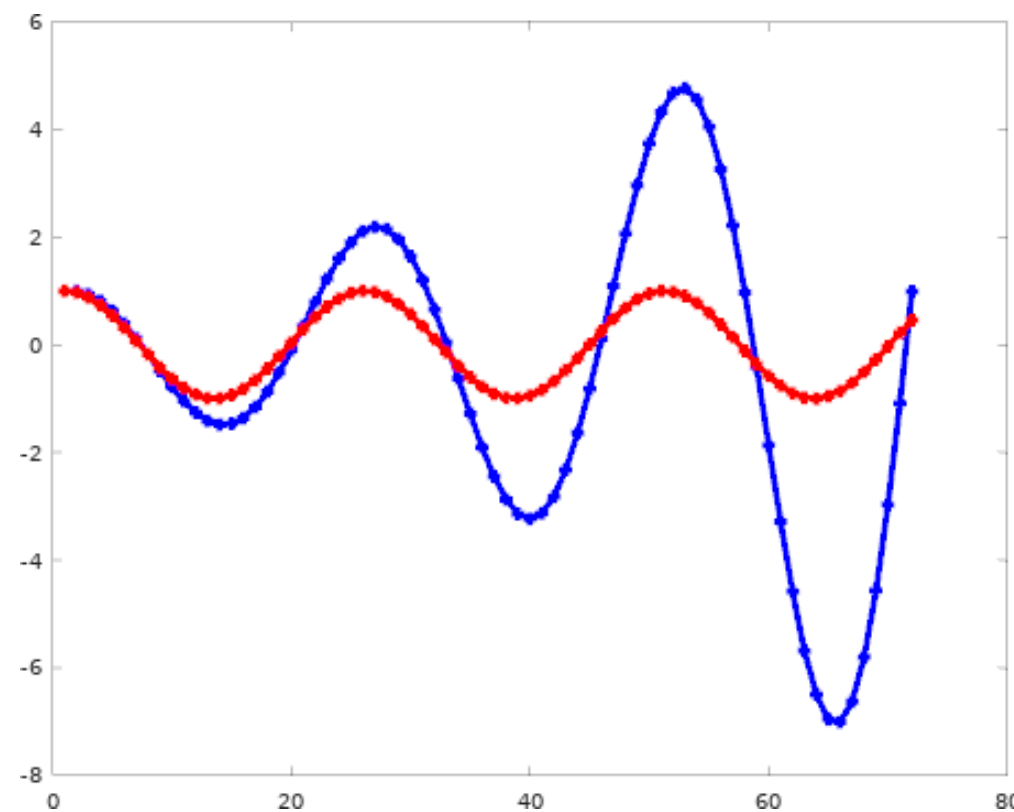
blue: numerical solution

Explicit Euler - study case

1D spring

- True solution: permanent oscillation

- Numerical scheme:
$$\begin{pmatrix} p \\ v \end{pmatrix}^{k+1} = \begin{pmatrix} 1 & h \\ -K/m h & 1 \end{pmatrix} \begin{pmatrix} p \\ v \end{pmatrix}^k + \begin{pmatrix} 0 \\ K/m h l^0 \end{pmatrix}$$



red: true solution

red: numerical solution

- Numerical solution diverge to ∞
- Worse than bad accuracy for Graphics

Explicit Euler - study case

1D spring: Analysis of the system energy

- Energy $E = \frac{1}{2}mv^2 + \frac{K}{2}(p - l^0)^2$

$$E^{k+1} = \frac{1}{2}m \left(-\frac{K}{m}\Delta t (p^k - l^0) + v^k \right)^2 + \frac{1}{2}K(p^k + \Delta t v^k - l^0)^2$$

$$E^{k+1} = \underbrace{\frac{1}{2}m(v^k)^2 + \frac{1}{2}K(p^k - l^0)^2}_{E^k} + \frac{1}{2} \left[\underbrace{\frac{K^2}{m}(\Delta t)^2(p^k - l^0)^2}_{>0} - \underbrace{2K\Delta t(p^k - l^0)v^k}_{=0} + \underbrace{K(\Delta t)^2(v^k)^2}_{>0} \right]$$

$$E^{k+1} = E^k + \epsilon (\Delta t)^2, \epsilon > 0$$

$\Rightarrow$ gain of energy

$\Rightarrow$ divergence

Accuracy of a numerical method - general definition

- Define the **local truncation error** τ :

Error accumulated during one step, assuming perfect knowledge of the true solution

$$\tau_k = \|\tilde{u}(t^k) - u^k\|, \text{ assuming } u^{k-1} = \tilde{u}(t^{k-1})$$

- A numerical scheme is said to be accurate of order k , if its local truncation error is in $\mathcal{O}(h^{k+1})$.

Explicit Euler is of order 1, at every step we add an error in $\mathcal{O}(h^2)$.

Stability of a numerical method - general definition

- Classical stability of a method studied on $u'(t) = \lambda u(t)$, $\lambda \in \mathbb{C}$.
 - The true solution $\tilde{u}(t) = \exp(\lambda t)$ converge if $\Re_e(\lambda) \leq 0$.
 - For linear system, λ refers to eigenvalues of A .
 - For non linear system, λ refers to eigenvalues of the Jacobian of $\mathcal{F}$
- A numerical method is *unconditionnaly stable* if $\Re_e(\lambda) \leq 0 \Rightarrow$ stable discrete solution.
- Otherwise, it is conditionnally stable/unstable.
- Region of stability: Set of conditions on λ such that the discrete solution doesn't diverge.

Rem.

- A numerical solution can diverge even when the true ODE solution converge when using unstable numerical method.
- Converseley, a numerical solution can converge even when the true ODE solution diverge when using stable numerical method.
- Stability ! = Accuracy.

Stability analysis of explicit Euler

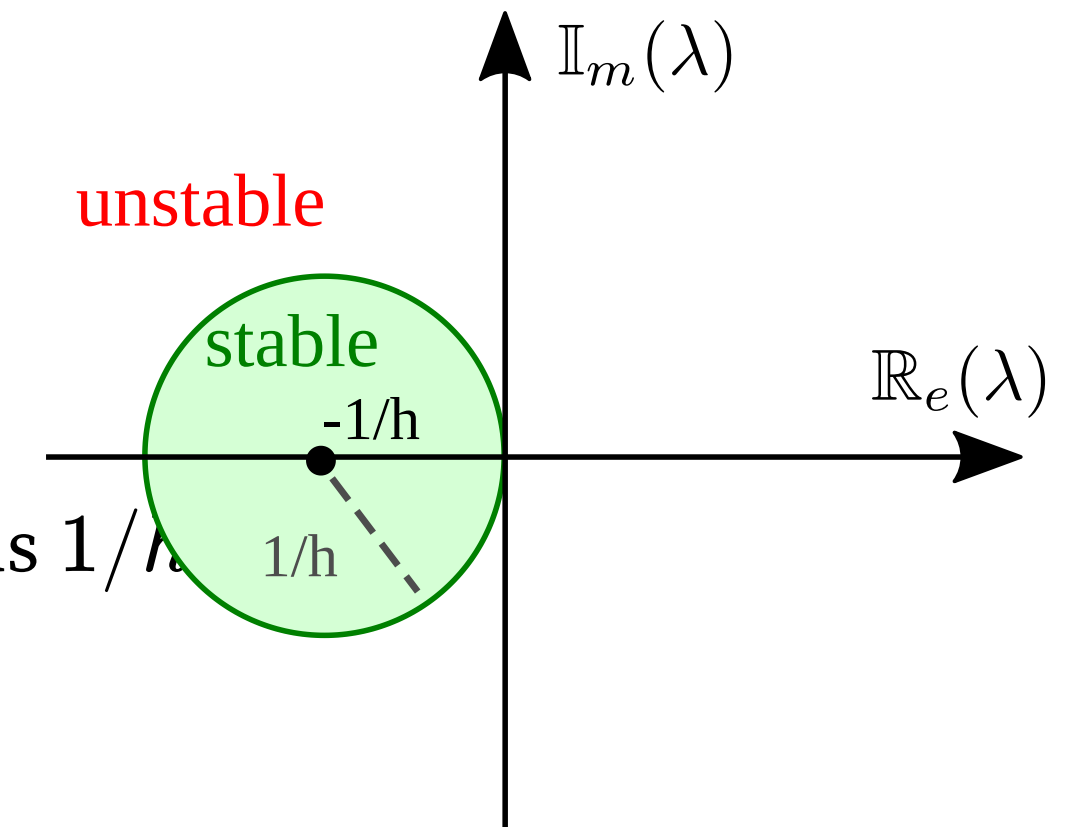
$$u'(t) = \lambda u(t)$$

$$\Rightarrow u^{k+1} = u^k + \lambda u^k \text{ using explicit Euler}$$

$$\Rightarrow u^{k+1} = (1 + \lambda h) u^k$$

$$\Rightarrow \text{Stable if } |1 + \lambda h| \leq 1 \text{ (conditionnal stability)}$$

$$|1/h + \lambda| \leq 1/h: \text{ Interior of a disc centered on } (-1/h, 0) \text{ with radius } 1/h$$



Rem. For 1D elastic spring

$$\lambda = \pm i \sqrt{K/m}$$

$$\Rightarrow |1 + i \sqrt{K/m} h| = \sqrt{1 + K/m h^2} > 1$$

$\Rightarrow$ Explicit euler is always unstable on the elastic spring problem.

Other approach: Implicit method

Other approach: Implicit Euler

Explicit Euler

$$u^{k+1} = u^k + h \mathcal{F}(u^k, t^k)$$

u^k is known to compute u^{k+1}

- In the linear case

$$u^{k+1} = (\mathbf{I} + h \mathbf{A}) u^k + h b(t^k)$$

Implicit Euler

$$u^{k+1} = u^k + h \mathcal{F}(u^{k+1}, t^{k+1})$$

u^{k+1} appears in RHS

$\Rightarrow$ Need to solve $u^{k+1} - h \mathcal{F}(u^{k+1}, t^{k+1}) = u^k$

- In the linear case : solve a linear system

$$u^{k+1} = u^k + h (\mathbf{A} u^{k+1} + b(t^{k+1}))$$

$$\Rightarrow u^{k+1} = (\mathbf{I} - h \mathbf{A})^{-1} (u^k + h b(t^{k+1}))$$

Implicit Euler - study case

Free fall under gravity

- True solution $\tilde{u}(k \Delta t) = p_0 + (k \Delta t)v_0 + \frac{(k \Delta t)^2}{2} g$

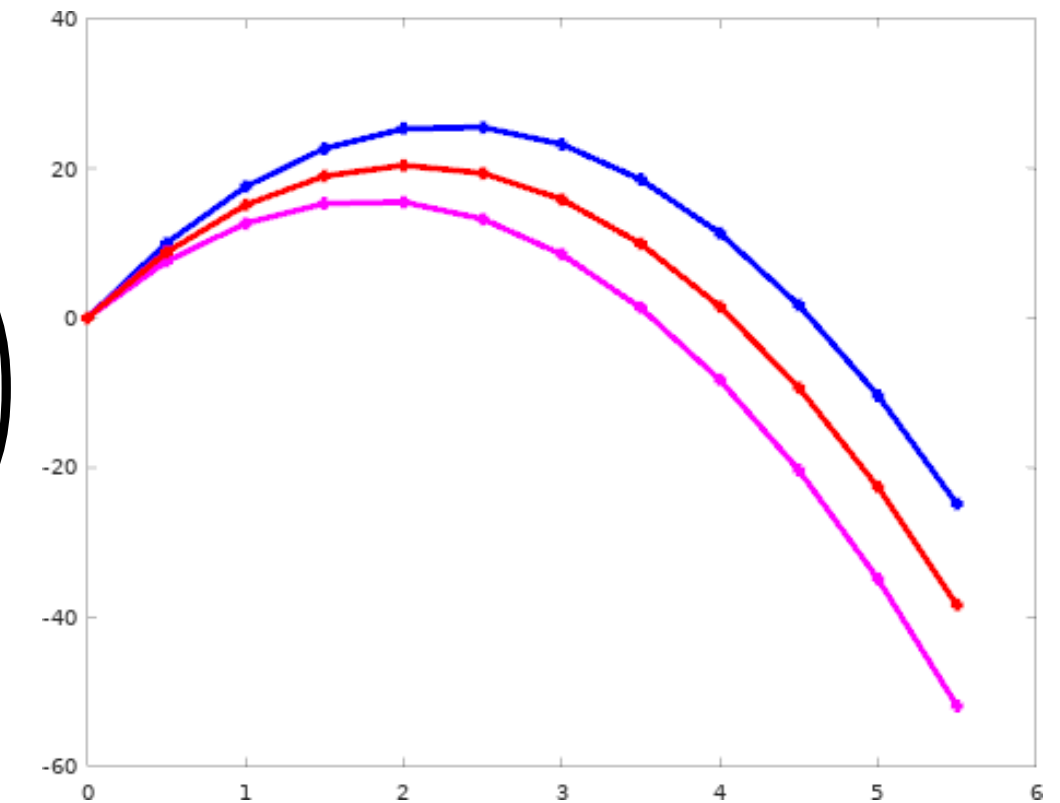
- Numerical scheme: $\begin{pmatrix} p \\ v \end{pmatrix}^{k+1} = \begin{pmatrix} 1 & -h \\ 0 & 1 \end{pmatrix}^{-1} \left(\begin{pmatrix} p \\ v \end{pmatrix}^k + \begin{pmatrix} 0 \\ g \end{pmatrix} \right)$

$$\Rightarrow \begin{pmatrix} p \\ v \end{pmatrix}^{k+1} = \begin{pmatrix} 1 & h \\ 0 & 1 \end{pmatrix} \begin{pmatrix} p \\ v \end{pmatrix}^k + \begin{pmatrix} h^2 g \\ h g \end{pmatrix}$$

- Numerical solution: $p^k = p_0 + (k \Delta t)v_0 + \frac{k(k+1)}{2} (\Delta t)^2 g$

$\Rightarrow$ Not exact : Error $e^k = |u^k - \tilde{u}(k \Delta t)| = \frac{k}{2} (\Delta t)^2 g$

Same error magnitude than explicit Euler



- red: True solution
- magenta: Implicit Euler
- blue: Explicit Euler

Implicit Euler - study case

1D spring

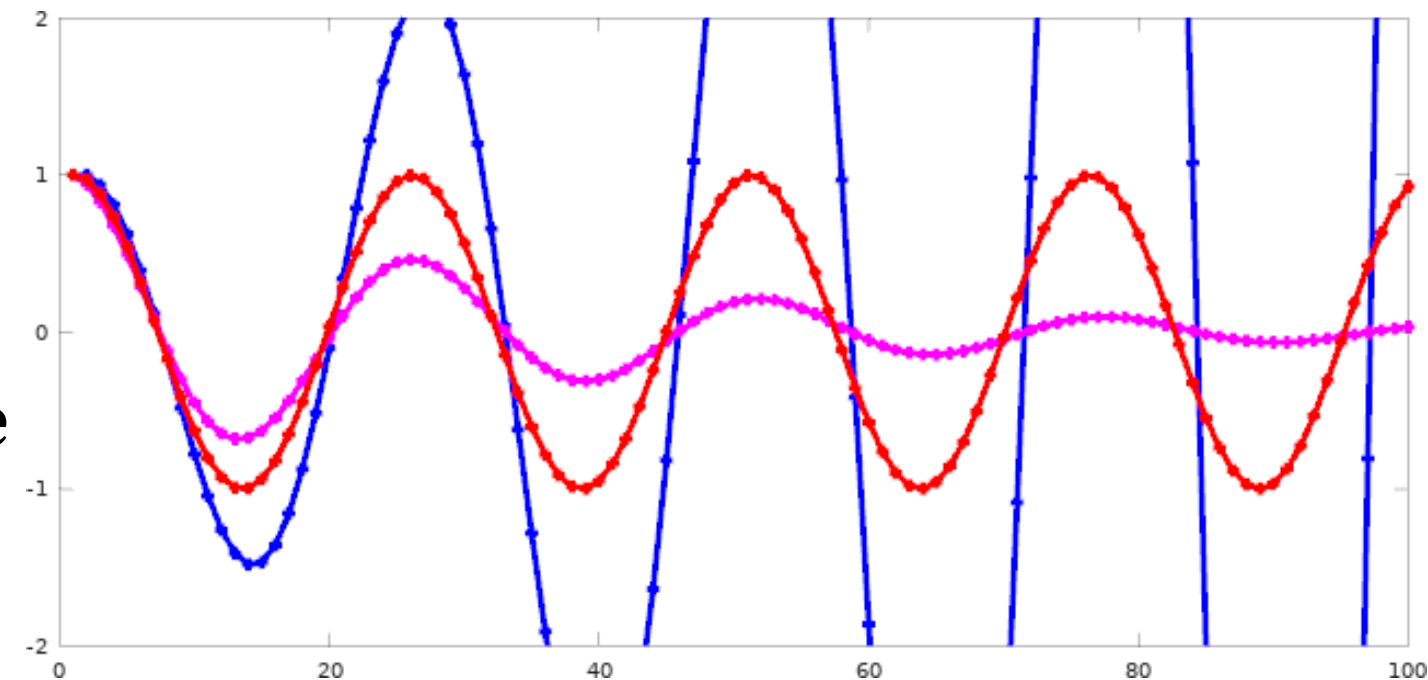
- True solution: permanent oscillations

- Numerical scheme:
$$\begin{pmatrix} p \\ v \end{pmatrix}^{k+1} = \begin{pmatrix} 1 & -h \\ K/m h & 1 \end{pmatrix}^{-1} \left(\begin{pmatrix} p \\ v \end{pmatrix}^k + \begin{pmatrix} 0 \\ K/m h l^0 \end{pmatrix} \right)$$
$$\Rightarrow \begin{pmatrix} p \\ v \end{pmatrix}^{k+1} = \frac{1}{1+h^2 K/m} \left(\begin{pmatrix} 1 & h \\ -K/m h & 1 \end{pmatrix} \begin{pmatrix} p \\ v \end{pmatrix}^k + \begin{pmatrix} K/m h^2 l^0 \\ K/m h l^0 \end{pmatrix} \right)$$

Eigenvalues of $(I - Ah)^{-1}$ are $\frac{1 \pm i \sqrt{\frac{K}{m}} h}{1 + \frac{K}{m} h^2}$

$$\Rightarrow \left| \frac{1 \pm i \sqrt{\frac{K}{m}} h}{1 + \frac{K}{m} h^2} \right| = \frac{1}{\sqrt{1 + \frac{K}{m} h^2}} < 1 \Rightarrow \text{always converge}$$

Even if the true solution oscillates



Stability analysis of Implicit Euler

What are the general conditions for which $u'(t) = \lambda u(t)$ converge using Implicit Euler ?

$$u^{k+1} = u^k + h \lambda u^{k+1}$$

$$\Rightarrow (1 - h \lambda) u^{k+1} = u^k$$

$$\Rightarrow u^{k+1} = \frac{1}{1 - h \lambda} u^k$$

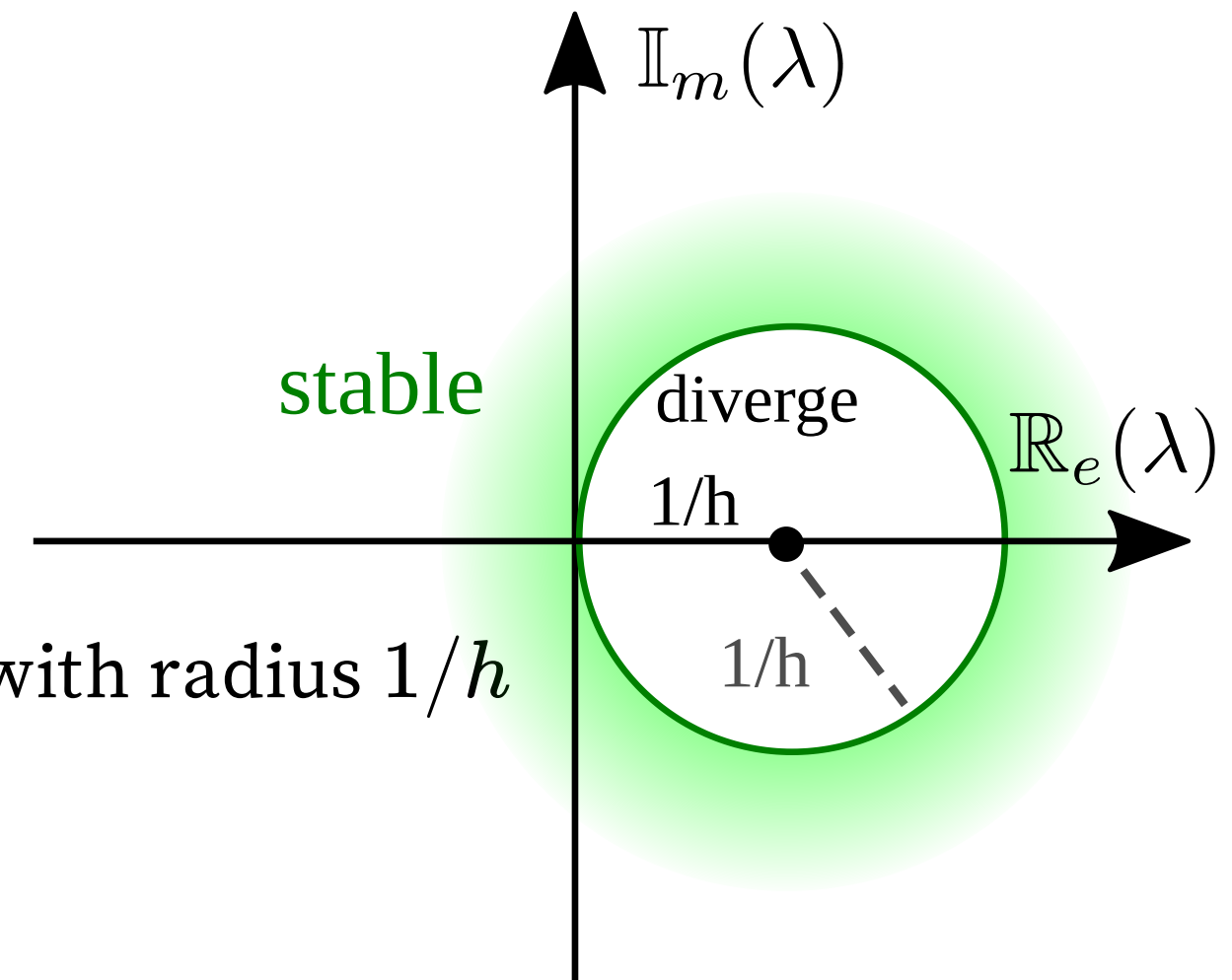
$$\text{Stability condition: } \left| \frac{1}{1 - h \lambda} \right| \leq 1 \Rightarrow |1 - h \lambda| \geq 1$$

$|1/h - \lambda| \geq 1/h$: exterior of a disc centered on $(1/h, 0)$ with radius $1/h$

$\Rightarrow$ Fully enclose $\mathbb{R}_e(\lambda) \leq 0$

$\Rightarrow$ Implicit euler is unconditionally stable

Rem. May converges even when the true solution does not.



Comparison Explicit Euler / Implicit Euler

Explicit Euler

- $u^k = u^k + h \mathcal{F}(u^k, t^k)$
- $u^k = (\mathbf{I} + h \mathbf{A}) u^k + h b(t^k)$
- Accuracy: 1st Order
- Stability:
 - Conditional $|1 + h\lambda| \geq 1$
 - Solution to spring model diverges

Implicit Euler

- $u^{k+1} = u^k + h \mathcal{F}(u^{k+1}, t^{k+1})$
- $u^k = (\mathbf{I} - h \mathbf{A})^{-1} (u^k + h b(t^{k+1}))$
- Accuracy: 1st Order
- Stability:
 - Unconditionnaly stable
 - Solution to spring model converge to l^0

Higher order method

Higher accuracy can be achieved using Taylor expansion.

$$\text{ex. } u^{k+1} = u^k + h \mathcal{F}(u^k, t^k) + \frac{h^2}{2} \frac{d\mathcal{F}}{dt}(u^k, t^k)$$

- Second order accurate: **2nd order explicit Euler**
 - Can achieve arbitrary accuracy.
 - In practice: computing derivatives of $\mathcal{F}$ is complex.
- $\Rightarrow$ Not often used

Runge-Kutta methods

Runge-Kutta numerical methods allow to

- Reach higher order accuracy
- Only involve knowledge of $\mathcal{F}$, without its derivatives
- Involves several evaluation of $\mathcal{F}$ at various points

2nd order Runge-Kutta: RK2

$$u^{k+1} = u^k + \frac{1}{2} (k_1 + k_2)$$

$$k_1 = h \mathcal{F}(u^k, t^k)$$

$$k_2 = h \mathcal{F}(u^k + k_1, t^k + h)$$

Accuracy, stability ?

Computing Accuracy

Reminder: Local truncation error

$$\tau_k = \|\tilde{u}(t^k + h) - u(t^k + h)\|$$

Assuming true solution at time t_k : $\tilde{u}(t^k) = u(t^k)$ (and all its derivatives).

General methodology to compute accuracy

- (1) Compute Taylor expansion of the numerical scheme
 - Express $u(t^k + h)$ at position (u^k, t^k)
- (2) Compute Taylor expansion of the exact differential equation
 - Express $\tilde{u}(t^k + h)$ at position (u^k, t^k) , given $\tilde{u}'(t^k) = \mathcal{F}(u^k, t^k)$
- (3) Compare both expressions

RK2 - Accuracy

(1) Compute Taylor expansion of the numerical scheme

$$\text{RK2 definition: } u(t^k + h) = u^k + \frac{1}{2}(k_1 + k_2)$$

$$k_1 = h \mathcal{F}(u^k, t^k)$$

$$k_2 = h \mathcal{F}(u^k + k_1, t^k + h)$$

$$\Rightarrow k_2 = h \left(\mathcal{F}(u^k, t^k) + k_1 \frac{\partial \mathcal{F}}{\partial u}(u^k, t^k) + h \frac{\partial \mathcal{F}}{\partial t}(u^k, t^k) + \mathcal{O}(h^2) \right) \quad (\text{differential on } k_1 \text{ and on } h)$$

$$\Rightarrow k_2 = h\mathcal{F} + h^2 \mathcal{F} \frac{\partial \mathcal{F}}{\partial u} + h^2 \frac{\partial \mathcal{F}}{\partial t} + \mathcal{O}(h^3) \quad (\text{dropping parameters } (u^k, t^k))$$

$$u(t^k + h) = u^k + \frac{1}{2} \left[h\mathcal{F} + \left(h\mathcal{F} + h^2 \frac{\partial \mathcal{F}}{\partial u} \mathcal{F} + h^2 \frac{\partial \mathcal{F}}{\partial t} + \mathcal{O}(h^3) \right) \right]$$

$$\Rightarrow u(t^k + h) = u^k + h\mathcal{F} + \frac{h^2}{2} \left(\frac{\partial \mathcal{F}}{\partial u} \mathcal{F} + \frac{\partial \mathcal{F}}{\partial t} \right) + \mathcal{O}(h^3)$$

RK2 - Accuracy

(2) Compute Taylor expansion of the exact differential equation

$$\tilde{u}(t^k + h) = \tilde{u}(t^k) + h \tilde{u}'(t^k) + \frac{h^2}{2} \tilde{u}''(t^k) + \mathcal{O}(h^3)$$

Assuming exact solution at t^k : $\tilde{u}^{(n)}(t^k) = u^{(n)}(t^k)$ + Injecting the ODE relation $u'(t^k) = \mathcal{F}(u^k, t^k)$

$$\Rightarrow \tilde{u}(t^k + h) = u(t^k) + h \mathcal{F}(u^k, t^k) + \frac{h^2}{2} \frac{d\mathcal{F}}{dt}(u^k, t^k) + \mathcal{O}(h^3)$$

$$\Rightarrow \tilde{u}(t^k + h) = u(t^k) + h \mathcal{F}(u^k, t^k) + \frac{h^2}{2} \left(\frac{\partial \mathcal{F}}{\partial u}(u^k, t^k) u'(t^k) + \frac{\partial \mathcal{F}}{\partial t}(u^k, t^k) \right) + \mathcal{O}(h^3)$$

$$\text{Drop parameter } (u^k, t^k) \Rightarrow \tilde{u}(t^k + h) = u^k + h \mathcal{F} + \frac{h^2}{2} \left(\frac{\partial \mathcal{F}}{\partial u} \mathcal{F} + \frac{\partial \mathcal{F}}{\partial t} \right) + \mathcal{O}(h^3)$$

(3) Compare both expressions

$$\tau_k = \|\tilde{u}(t^k + h) - u(t^k + h)\| = \mathcal{O}(h^3)$$

$\Rightarrow$ **RK2 is 2nd order accurate**

RK2 - Stability

Study $u'(t) = \lambda u(t) \Rightarrow \mathcal{F}(u, t) = \lambda u(t)$

$$- k_1 = h \mathcal{F}(u^k, t^k) = h \lambda u^k$$

$$- k_2 = h \mathcal{F}(u^k + k_1, t^k + h) = h \lambda (u^k + h \lambda u^k) = (h \lambda + h^2 \lambda^2) u^k$$

$$\Rightarrow u^{k+1} = u^k + \frac{1}{2} (k_1 + k_2)$$

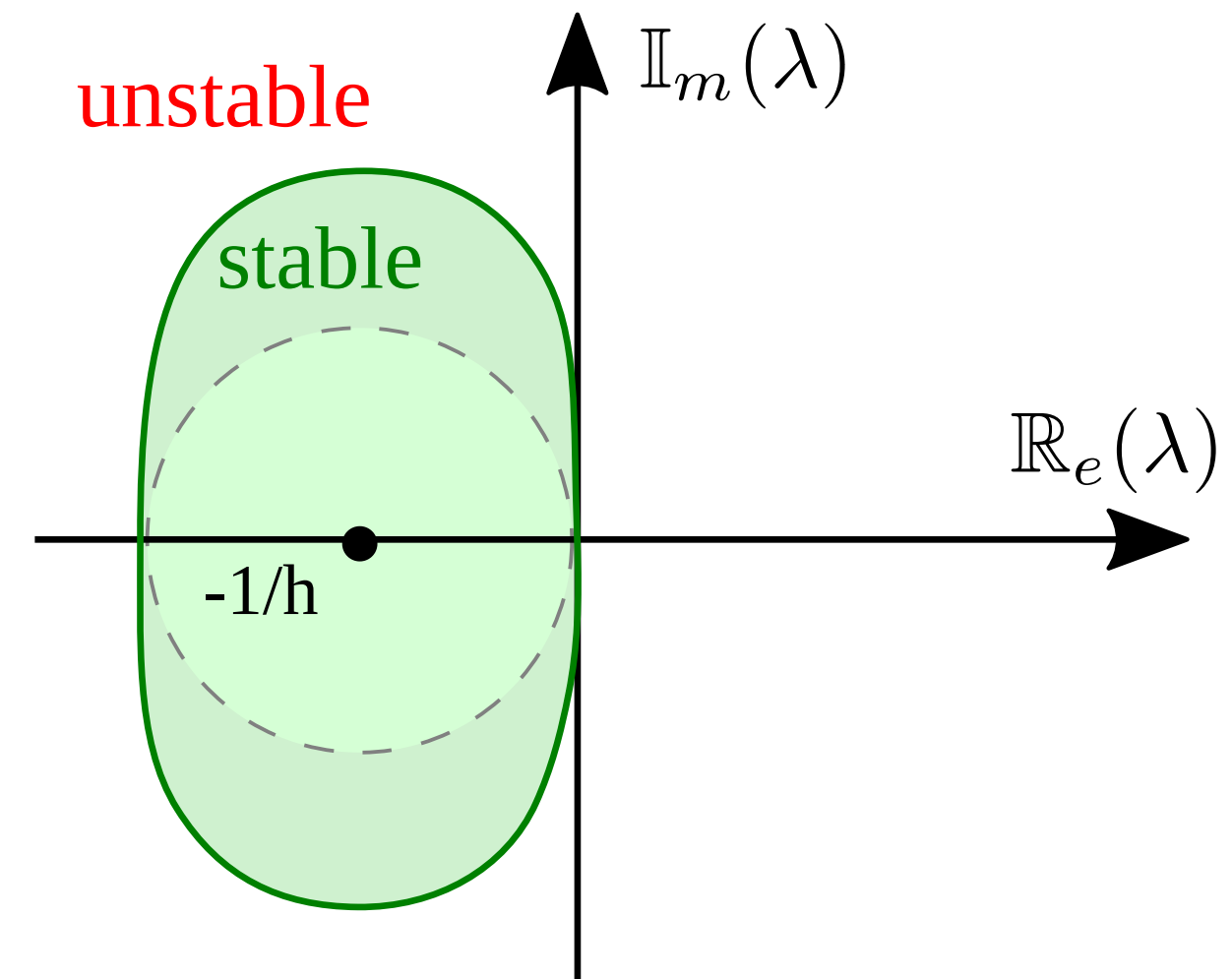
$$\Rightarrow u^{k+1} = \left(1 + h \lambda + \frac{h^2}{2} \lambda^2\right) u^k$$

$$\text{Stability condition } \left|1 + h \lambda + \frac{h^2}{2} \lambda^2\right| \leq 1$$

$\Rightarrow$ RK2 is conditionnaly stable (explicit method)

More stable than explicit Euler

(Still unstable for oscillatory 1D spring)



Other Runge-Kutta methods

Midpoint method (2nd order accuracy)

$$k_1 = h \mathcal{F}(u^k, t^k)$$

$$k_2 = h \mathcal{F}(u^k + k_1/2, t^k + h/2)$$

$$u^{k+1} = u^k + h k_2$$

RK4 (Classical Runge-Kutta)

$$k_1 = h \mathcal{F}(u^k, t^k)$$

$$k_2 = h \mathcal{F}\left(u^k + \frac{k_1}{2}, t^k + \frac{h}{2}\right)$$

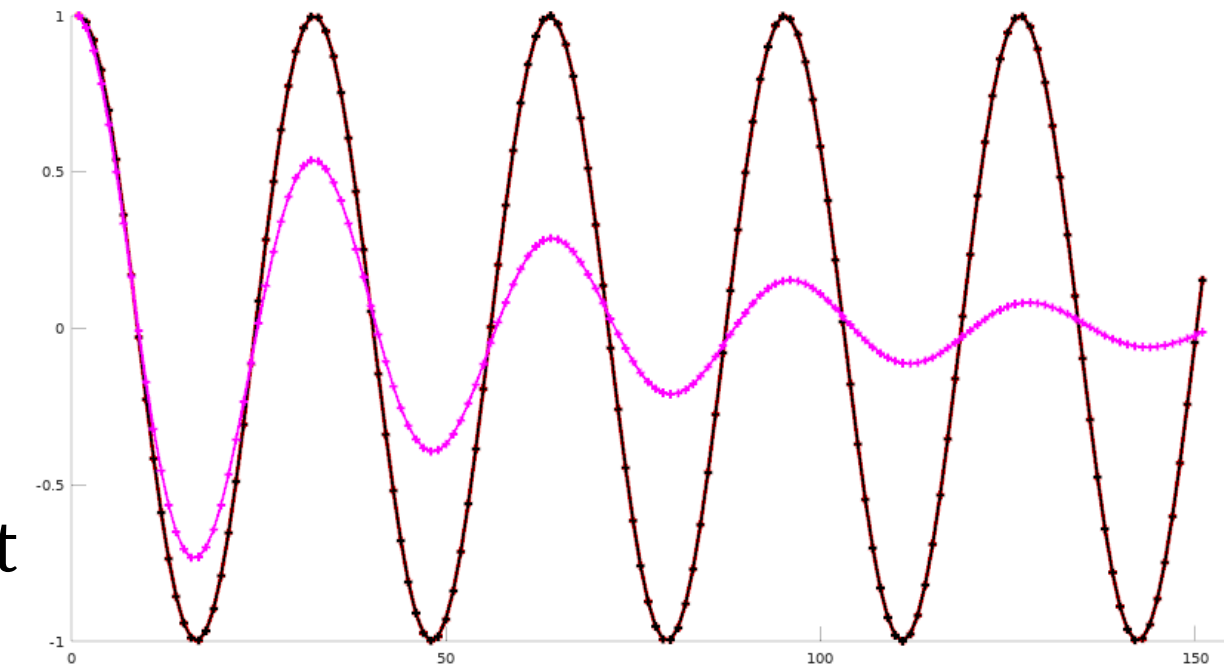
$$k_3 = h \mathcal{F}\left(u^k + \frac{k_2}{2}, t^k + \frac{h}{2}\right)$$

$$k_4 = h \mathcal{F}(u^k + k_3, t^k + h)$$

$$u^{k+1} = u^k + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

RK4 - Case study

- Rk4 conditionally stable for oscillating spring
 - Not permanent oscillations: Slight decrease of magnitude through time*
- Large improvement of accuracy compared to implicit Euler



$h = 0.2$, error rk4 $\simeq 10^{-4}$

- red: true solution

- black: rk4

- magenta: implicit Euler

$$\text{Stability region: } \left| 1 + h\lambda + \frac{h^2}{2}\lambda^2 + \frac{h^3}{6}\lambda^3 + \frac{1}{24}h^4\lambda^4 \right| \leq 1$$

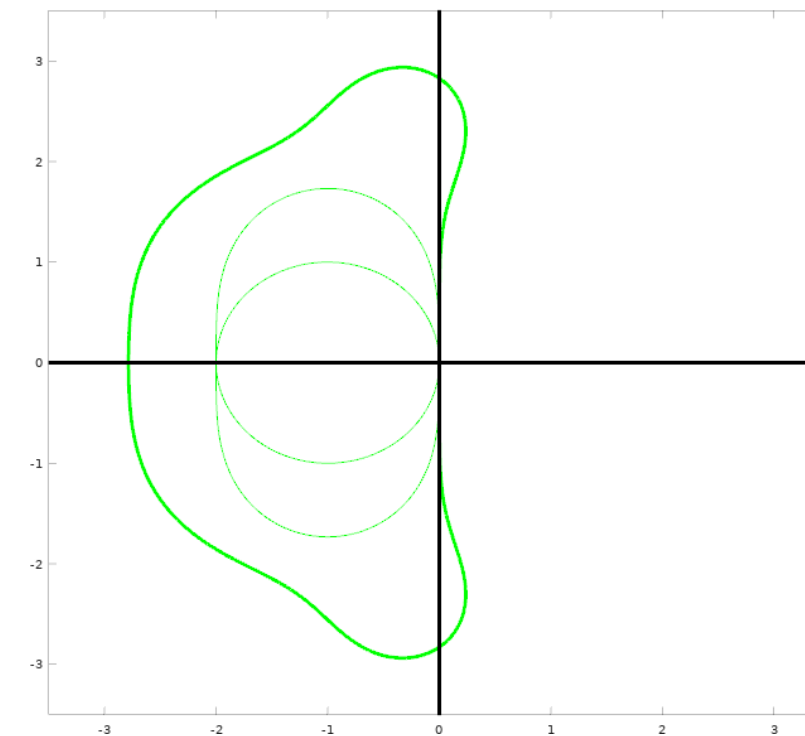
In the case of oscillating spring: $\lambda = i\sqrt{K/m} = i\omega$

$$\text{Stable if } \left| 1 + ih\omega - \frac{h^2}{2}\omega^2 - i\frac{h^3}{6}\omega^3 + \frac{h^4}{24}\omega^4 \right| \leq 1$$

$$\Rightarrow 1 - \frac{h^6\omega^6}{72} + \frac{h^8\omega^8}{576} \leq 1$$

$$\Rightarrow h^6\omega^6(h^2\omega^2 - 8)/576 \leq 0$$

$$\Rightarrow h \leq \frac{2^{3/2}}{\omega} = \frac{2^{3/2}}{\sqrt{K/m}} \simeq \frac{2.8}{\sqrt{K/m}}$$



Dormand Prince (ode45)

$$k_1 = h \mathcal{F}(u^k, t^k)$$

$$k_2 = h \mathcal{F}\left(u^k + \frac{1}{5}k_1, t^k + \frac{1}{5}h\right)$$

$$u_5^{k+1} = u^k + h \left(\frac{35}{384}k_1 + \frac{500}{1113}k_3 + \frac{125}{192}k_4 - \frac{2187}{6784}k_5 + \frac{11}{84}k_6 \right)$$

$$k_3 = h \mathcal{F}\left(u^k + \frac{3}{40}k_1 + \frac{9}{40}k_2, t^k + \frac{3}{10}h\right)$$

$$u_4^{k+1} = u^k + h \left(\frac{5179}{57600}k_1 + \frac{7571}{16695}k_3 + \frac{393}{640}k_4 - \frac{92097}{339200}k_5 + \frac{187}{2100}k_6 + \frac{1}{40}k_7 \right)$$

$$k_4 = h \mathcal{F}\left(u^k + \frac{44}{45}k_1 - \frac{56}{15}k_2 + \frac{32}{9}k_3, t^k + \frac{4}{5}h\right)$$

$$k_5 = h \mathcal{F}\left(u^k + \frac{19372}{6561}k_1 - \frac{25360}{2187}k_2 + \frac{64448}{6561}k_3 - \frac{212}{729}k_4, t^k + \frac{8}{9}h\right)$$

$$k_6 = h \mathcal{F}\left(u^k + \frac{9017}{3168}k_1 - \frac{355}{33}k_2 + \frac{46732}{5247}k_3 + \frac{49}{176}k_4 - \frac{5103}{18656}k_5, t^k + h\right)$$

$$k_7 = h \mathcal{F}\left(u^k + \frac{35}{384}k_1 + \frac{500}{1113}k_3 + \frac{125}{192}k_4 - \frac{2187}{6784}k_5 + \frac{11}{84}k_6, t^k + h\right)$$

u_5^{k+1} is 5th order, u_4^{k+1} is 4th order $\Rightarrow$ Automatic error estimation for adaptive time steps.

Embedded method

[J. Dormand, P. Prince. A family of embedded Runge-Kutta formulae. J. of Computation and Applied Mathematics, 1980].

Symplectic methods / semi-implicit

Introduction to symplectic methods

Standard approaches trade-off

- Explicit methods: (+) Simple to compute, (-) limited stability
- Implicit methods: (-) Hard to compute (especially on non linear functions), (+) very stable
- Oscillatory systems are not easy to model
 - (-) Numerical solution either diverge or converge.

Symplectic approach

- *Remark:* Mechanical systems have position and speed variables
 - Derivative of position is linear w/r speed
 - Derivative of speed is more complex (forces - non linear)
- ⇒ *General idea:* separate treatment of speed and position

Semi-implicit:

- Implicit scheme for position p^{k+1} (linear part)
 - Explicit scheme for speed v^{k+1} (non linear part)
- ⇒ In practice: use speed v^{k+1} to evaluate p^{k+1} .

Pro

- (+) As simple as explicit method to implement
- (+) Improved stability
- (+) Well adapted to oscillatory systems

Semi implicit method

Simplest semi-implicit method: **Semi-implicit Euler** / **Verlet**

General case

$$v^{k+1} = v^k + h \mathcal{F}_v(p^k, v^k, t^k)$$

$$p^{k+1} = p^k + h \mathcal{F}_p(p^k, v^{k+1}, t^k)$$

Application to classical mechanical cases

$$p'(t) = v(t), m v'(t) = F(p, v, t)$$

$$\Rightarrow \begin{cases} v^{k+1} = v^k + h F(p^k, v^k, t^k)/m \\ p^{k+1} = p^k + h v^{k+1} \end{cases}$$

(+) *Trivially easy to convert explicit Euler to semi-implicit Euler*

Expressed using positions only

$$p^{k+1} = p^k + h(v^k + h F(p^k, v^k, t^k))$$

$$p^{k+1} = p^k + h \left(\frac{p^k - p^{k-1}}{h} + h F(p^k, v^k, t^k)/m \right)$$

$$\Rightarrow p^{k+1} = 2 p^k - p^{k-1} + h^2 F(p, v, t)/m$$

1st order accurate (like explicit/implicit Euler) in position and speed.

Stability on oscillatory system

1D spring system: $F(p^k, v^k, t^k) = -K p^k$

$$p^{k+1} = 2p^k - p^{k-1} - h^2 K/m p^k$$

$$\Rightarrow p^{k+1} = (2 - h^2 K/m) p^k - p^{k-1}$$

Stable and permanent oscillation when $h < \frac{2}{\sqrt{K/m}}$

Demonstration

- Study the roots of the associated characteristic equation

- $x^2 + (y - 2)x + 1 = 0$, with $y = h^2 K/m$

- Discriminant $\Delta = y(y - 4)$, with roots $r = \frac{(2-y) \pm \sqrt{y(y-4)}}{2}$

- Two cases:

$\Delta < 0 \Rightarrow y \in]0, 4[: r = \frac{(2-y) \pm i\sqrt{y(4-y)}}{2} \Rightarrow |r| = 1/2 \sqrt{(2-y)^2 + y(4-y)} = 1 \Rightarrow$ Permanent oscillations (stable)

$\Delta \geq 0 \Rightarrow y \leq 0$, or $y \geq 4 : |r| > 1$ Unstable solution

- Finally, stable for $y \in]0, 4[\Rightarrow h^2 K/m < 4 \Rightarrow h < \frac{2}{\sqrt{K/m}}$, and system oscillate permanently

Use of semi-implicit

Ex. Elastic Spring: $F(p) = -K(p - L_0)$

Explicit Euler

```
for(int k=0; k<N; ++k) {  
    p = p + h*v;  
    v = v + h * F/m;  
}
```

(-) Always diverges for elastic problem

Semi-Implicit Euler

```
for(int k=0; k<N; ++k) {  
    v = v + h * F/m;  
    p = p + h*v;  
}
```

(+) Permanent oscillation for sufficiently small h .

⇒ Extremely simple change !

Big gain in stability

Higher order symplectic

2nd order Velocity Verlet

$$\begin{cases} v^{k+1/2} &= v^k + \frac{h}{2} F(p^k, v^k, t^k)/m \\ p^{k+1} &= p^k + h v^{k+1/2} \\ v^{k+1} &= v^{k+1/2} + \frac{h}{2} F(p^{k+1}, v^{k+1/2}, t^{k+1})/m \end{cases}$$

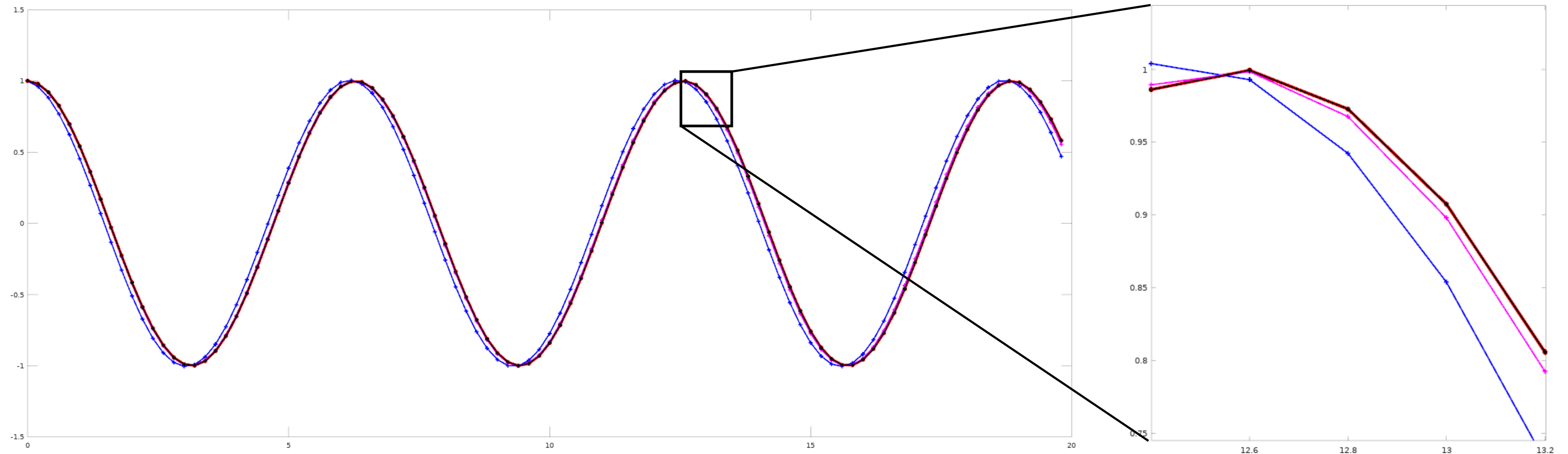
- Remains stable with permanent oscillation for $h \leq \frac{2}{\sqrt{K/m}}$
- Still simple to implement

Example for 1D elastic spring

```
for(int k=0; k<N; ++k) {  
    v = v + h/2 * (-K/m*p);  
    p = p + h*v;  
    v = v + h/2 * (-K/m*p);  
}
```

Comparison between approaches

Small $h = 0.2/\omega$

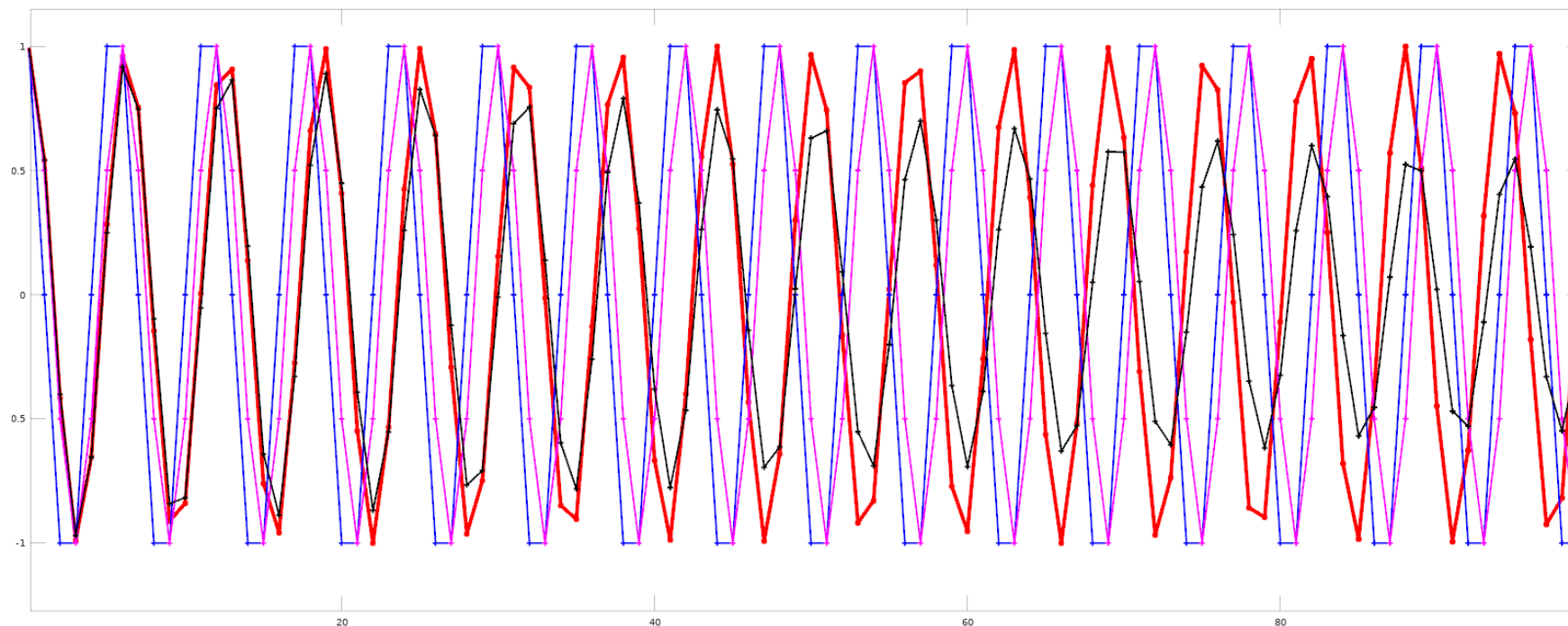


True solution , *Semi-implicit Euler* , *Velocity Verlet* , *Runge-Kutta RK4*

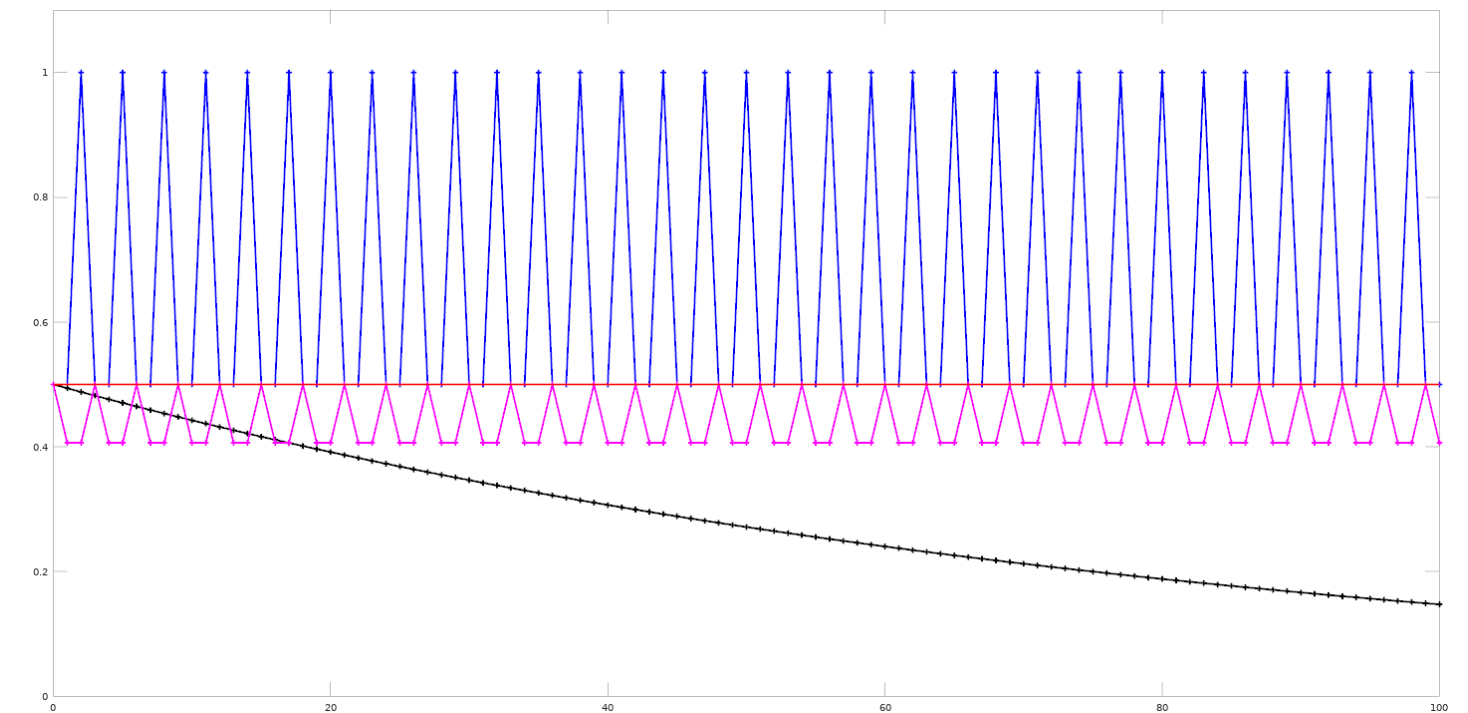
- RK4 - best behavior (undistinguishable from true solution)

Comparison between approaches

Larger $h = 1.0/\omega$



Temporal evolution of p^k



Temporal evolution of energy

$$E = 1/2m (v^k)^2 + 1/2K(p^k)^2$$

True solution , *Semi-implicit Euler* , *Velocity Verlet* , *Runge-Kutta RK4*

- RK4 loose energy and p^k converge toward l^0
- Symplectic integrator keep oscillating

Summary and extensions

Numerical integration of ODE

General formulation: $u'(t) = \mathcal{F}(u, t)$, $u(t) = (p(t), v(t))$.

Explicit Euler

$$u^{k+1} = u^k + \Delta t \mathcal{F}(u^k, t^k)$$

(+) Easy to implement

(-) Worst scheme in all cases (divergence, low accuracy)

Explicit Runge-Kutta

$$u^{k+1} = u^k + \Delta t \sum_j \alpha_j k_j$$

(+) Good **accuracy**

(+) Efficient to apply

(+/-) Stability OK for non-stiff problem, diverge on stiff problem

(-) Artificial damping for constant energy system

Implicit methods

$$u^{k+1} = u^k + \Delta t \mathcal{F}(u^{k+1}, t^{k+1})$$

(+) Good to deal with **stiff problem** - very stable

(-) Add numerical damping (converge even if solution oscillates)

(-) Hard/computationally costly to apply on non linear problem

Symplectic integrator

$$v^{k+1} = v^k + \Delta t F^k / m$$

$$p^{k+1} = p^k + \Delta t v^{k+1}$$

(+) Handle well constant energy system, preserves energy (Hamiltonian systems)

(+) Simple and efficient to implement

(-) Less accurate than RK

(-) Diverge on stiff problem

Extension - Automatic time stepping

Adjust h to limit errors at each step

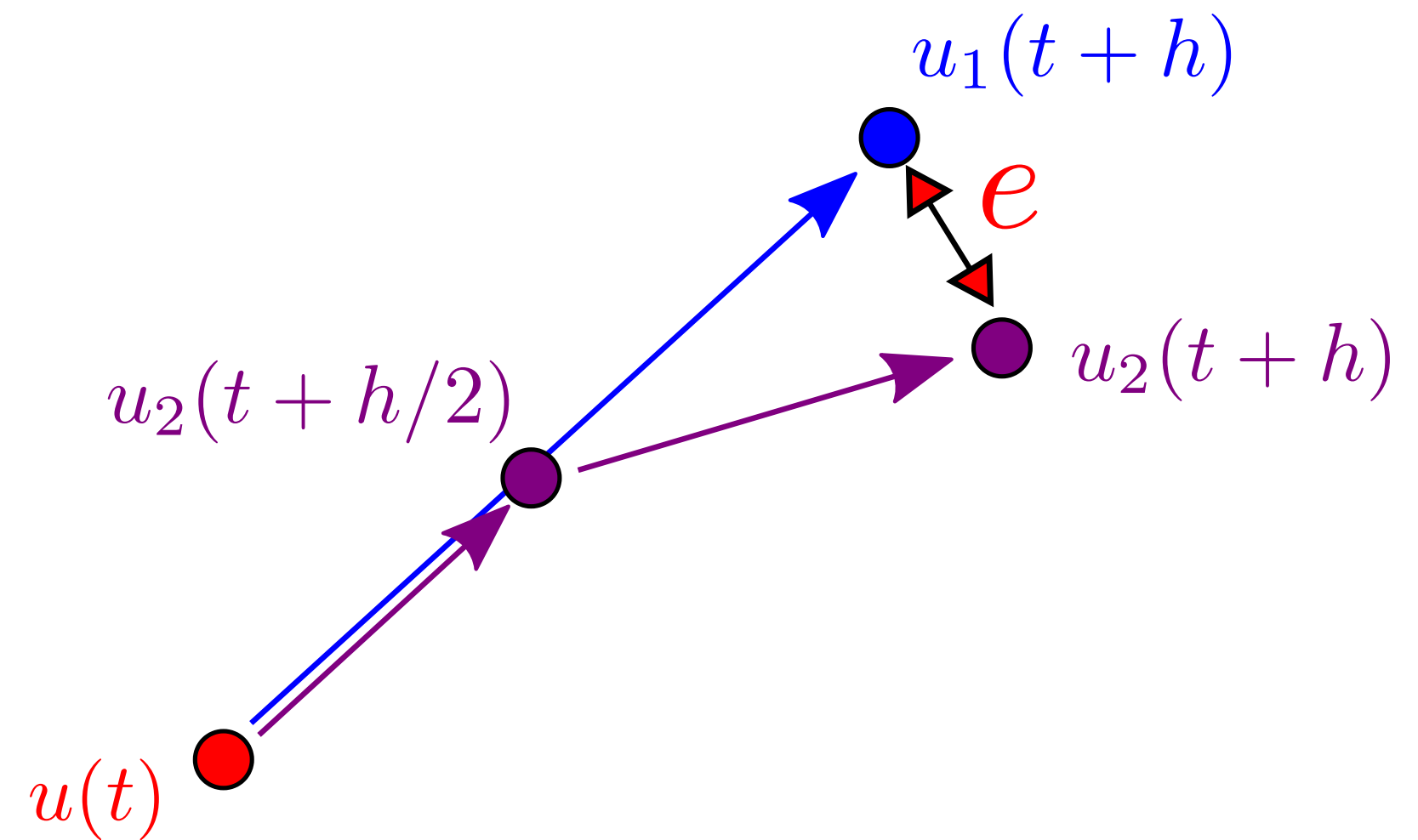
(1) Evaluate error in comparing:

- u_1^k computed using one step h
- u_2^k computed using two steps $h/2$.
- $e^k = \|u_1^k - u_2^k\|$

(2) Using embedded method $e^k = \|u_1^k - u_2^k\|$

Update

- $h^{new} = A h$ when $e^k < C_1$, ($A > 1$)
- $h^{new} = h/A$ when $e^k > C_2$



Implicit Euler on non-linear problem

$$u^{k+1} = u^k + h \mathcal{F}(u^{k+1}, t^{k+1})$$

- For high dimensional $\mathcal{F}$ - no explicit known inverse function
- Idea: Linearize $\mathcal{F}$ at u^k , assuming $\|u^{k+1} - u^k\|$ small.

$$\begin{aligned} u^{k+1} &= u^k + h \left(\mathcal{F}(u^k, t^{k+1}) + (u^{k+1} - u^k) \frac{\partial \mathcal{F}}{\partial u}(u^k, t^{k+1}) \right) \\ \Rightarrow \left(\mathbf{I} - h \frac{\partial \mathcal{F}}{\partial u}(u^k, t^{k+1}) \right) u^{k+1} &= \left(\mathbf{I} - h \frac{\partial \mathcal{F}}{\partial u}(u^k, t^{k+1}) \right) u^k + h \mathcal{F}(u^k, t^{k+1}) \\ u^{k+1} &= u^k + \left(\mathbf{I} - h \frac{\partial \mathcal{F}}{\partial u}(u^k, t^{k+1}) \right)^{-1} \mathcal{F}(u^k, t^{k+1}) \end{aligned}$$

- In theory only conditionally stable; In practice: very stable
 - Requires the expression of Jacobian matrix $\frac{\partial \mathcal{F}}{\partial u}$ - large sparse matrix
 - Requires inversion of a linear system at every steps
- $\Rightarrow$ Used for stable simulation with large time step

Implicit Euler on non-linear problem - particle system

For standard particles systems

- $p^{k+1} = p^k + h v^k$
- $v^{k+1} = v^k + h M^{-1} F(p^{k+1}, v^{k+1})$

With M the diagonal matrix of particles masses, assuming force F doesn't depends on t .

Linear approximation on force F

$$M v^{k+1} = M v^k + h \left(F(p^k, v^k) + \underbrace{(p^{k+1} - p^k)}_{h v^{k+1}} \frac{\partial F}{\partial p}(p^k, v^k) + (v^{k+1} - v^k) \frac{\partial F}{\partial v}(p^k, v^k) \right)$$

$$\Rightarrow \left(M - h \frac{\partial F}{\partial v} - h^2 \frac{\partial F}{\partial p} \right) v^{k+1} = \left(M - h \frac{\partial F}{\partial v} \right) v^k + h F(p^k, v^k)$$

[D. Baraff, A. Witkin. Large steps in Cloth Simulation. ACM SIGGRAPH 98]

Position based dynamics (PBD)

PBD - Use symplectic integrator expressed **using position only**

- Speed is computed implicitly as $(p^{k+1} - p^k)/h$
- Handle constraints using explicit projection of positions.
- Interesting to handle non-linear/stiff constraints

PBD algorithm

For **all** k

Integrate position (without constraints) $p[k+1] = \text{integrator}(p[k], v[k], t)$

For **all** constraints

Project $p[k+1]$ on constraints

Compute new speed $v[k+1] = (p[k+1] - p[k])/h$

(+) Unconditionally stable even for stiff constraints

(+) Simple to implement

(-) Low accuracy, no energy preserving

Very popular in Computer Graphics

[M. Muller et al. Position Based Dynamics. VRIPHYS 2006]

Extensions: XPBD, Projective dynamics, ...