Numerical Modelling

Jonathan GULA
gula@univ-brest.fr

the anatomy of an ocean model

Outline

- **Lesson 1 :** [D109]
  - Introduction
  - Equations of motions
  - *Activity 1 [run an ocean model]*
- **Lesson 2 :** [D109]
  - Subgrid-scale parameterization
  - Dynamics of the ocean gyre
  - *Activity 2 [Dynamics of an ocean gyre]*
- **Lesson 3 :** [D109]
  - Horizontal Discretization
  - Vertical coordinates
  - *Activity 2 [Dynamics of an ocean gyre]*
  - *Activity 3 [Impacts of numerics / topography]*
- **Lesson 4 :** [D109]
  - Numerical schemes
  - *Activity 3 [Impacts of numerics / topography]*
- **Lesson 5 :** [D109]
  - Presentation of the model CROCO
  - Boundary Forcings
  - *Activity 5 [Design a realistic simulation]*
- **Lesson 6 :** [D109]
  - Diagnostics and validation
  - *Activity 6 [Analyze a realistic simulation]*
- **Lesson 7 :** [D109]
  - *Project*

Presentations and material will be available at :

jgula.fr/ModNum/

#5
Solving the equations

Master's degree 2nd year Marine Physics

Jonathan GULA
gula@univ-brest.fr

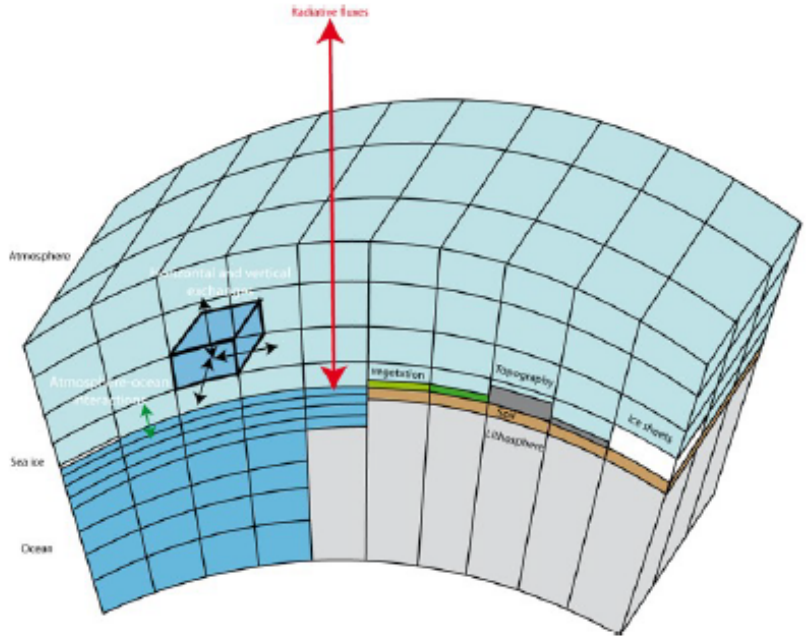
Solving the equations ?

$$\frac{\partial u}{\partial t} + \vec{u} \cdot \vec{\nabla}_H u + w \frac{\partial u}{\partial z} - fv = -\frac{\partial_x P}{\rho_0} + \mathcal{F}_u + \mathcal{D}_u$$

$$\frac{\partial v}{\partial t} + \vec{u} \cdot \vec{\nabla}_H v + w \frac{\partial v}{\partial z} + fu = -\frac{\partial_y P}{\rho_0} + \mathcal{F}_v + \mathcal{D}_v$$

Equations

Discretization



Solving the equations ?

$$\frac{\partial u}{\partial t} + \vec{u} \cdot \vec{\nabla}_H u + w \frac{\partial u}{\partial z} - fv = -\frac{\partial_x P}{\rho_0} + \mathcal{F}_u + \mathcal{D}_u$$

$$\frac{\partial v}{\partial t} + \vec{u} \cdot \vec{\nabla}_H v + w \frac{\partial v}{\partial z} + fu = -\frac{\partial_y P}{\rho_0} + \mathcal{F}_v + \mathcal{D}_v$$

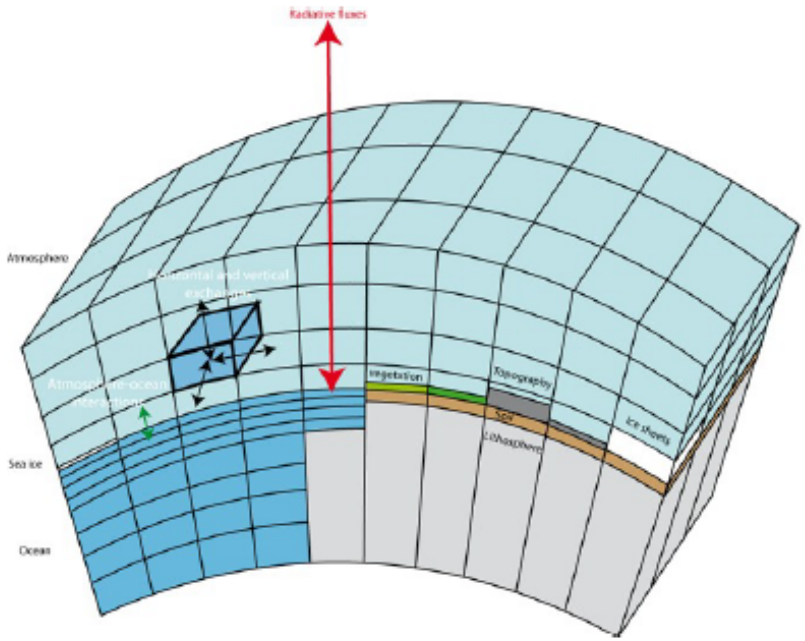
Equations

Time Stepping scheme

Horizontal Advection scheme

Vertical Advection scheme

Discretization



Finite difference schemes

- A **finite difference scheme** is produced when the partial derivatives in the partial differential equations governing a physical phenomenon are replaced by a finite difference approximation.
- The result is a system of algebraic equations which, when solved, provide an approximation to the solution of the original partial differential equations at selected points of a solution grid.
-

Taylor series expansion & truncation errors

- Definition of the derivative of a smooth function:

$$f'(x) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon) - f(x)}{\epsilon}$$

- Taylor series expansion of *f* at point *x*:

$$f(x+\epsilon) = f(x) + \frac{f'(x)}{1!}\epsilon + \frac{f^{(2)}(x)}{2!}\epsilon^2 + \ldots + \frac{f^{(n)}(x)}{n!}\epsilon^n + R_n$$

$$f(x-\epsilon) = f(x) - \frac{f'(x)}{1!}\epsilon + \frac{f^{(2)}(x)}{2!}\epsilon^2 + \ldots + (-1)^n\frac{f^{(n)}(x)}{n!}\epsilon^n + R_n$$

Taylor series expansion & truncation errors

- Constructing a difference operator using Taylor series:

$$f(x+\Delta x) = f(x) + \frac{f'(x)}{1!}\Delta x + \frac{f^{(2)}(x)}{2!}\Delta x^2 + \ldots$$

$$f(x+\Delta x) - f(x) = \frac{f'(x)}{1!}\Delta x + \frac{f^{(2)}(x)}{2!}\Delta x^2 + \ldots$$

- 1st order approx. for derivative:

$$f'(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x} - \frac{f^{(2)}(x)}{2!}\Delta x + \ldots$$

Truncation Error

$$O(\Delta x)$$

Spatial advection scheme:

First order

$$\frac{\partial u}{\partial x} = \frac{u_{i+1} - u_i}{\Delta x} + \mathcal{O}(\Delta x)$$

Downstream

$$\frac{\partial u}{\partial x} = \frac{u_i - u_{i-1}}{\Delta x} + \mathcal{O}(\Delta x)$$

Upstream

2nd order

$$\frac{\partial u}{\partial x} = \frac{u_{i+1} - u_{i-1}}{2\Delta x} + \mathcal{O}(\Delta x^2)$$

Centered

4th order

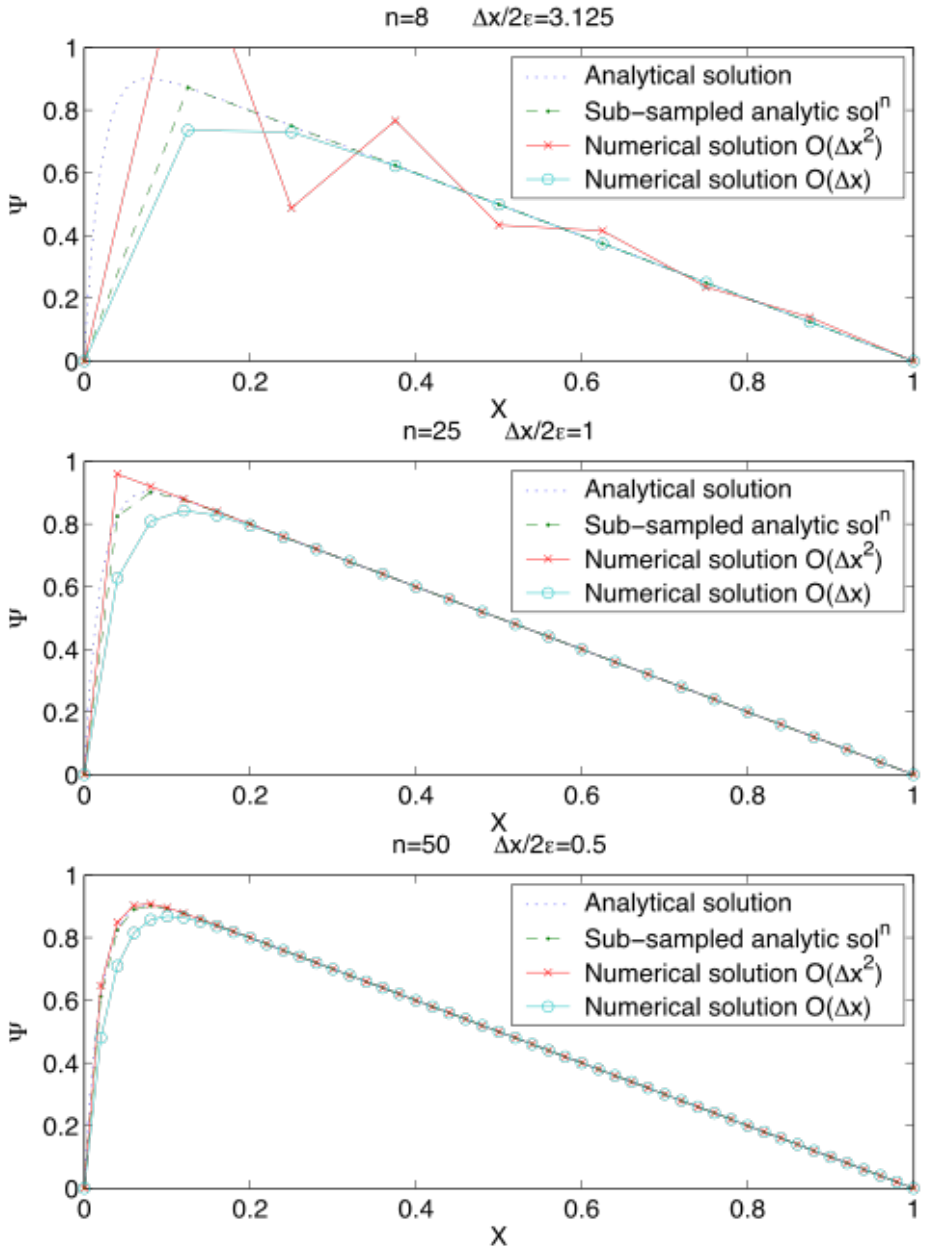
$$\frac{\partial u}{\partial x} = \frac{4}{3}\left(\frac{u_{i+1} - u_{i-1}}{2\Delta x}\right) - \frac{1}{3}\left(\frac{u_{i+2} - u_{i-2}}{2\Delta x}\right) + \mathcal{O}(\Delta x^4)$$

Spatial advection scheme :

- Ex: Stommel equation in 1D

$$\epsilon \partial_{xx}\psi + \partial_x \psi = -1$$

$$\psi = C\left(e^{-\frac{x}{\epsilon}} - 1\right) - x \qquad \text{with} \quad C^{-1} = e^{-\frac{1}{\epsilon}} - 1.$$



Solving the equations ?

$$\frac{\partial u}{\partial t} + \vec{u} \cdot \vec{\nabla}_H u + w \frac{\partial u}{\partial z} - fv = -\frac{\partial_x P}{\rho_0} + \mathcal{F}_u + \mathcal{D}_u$$

$$\frac{\partial v}{\partial t} + \vec{u} \cdot \vec{\nabla}_H v + w \frac{\partial v}{\partial z} + fu = -\frac{\partial_y P}{\rho_0} + \mathcal{F}_v + \mathcal{D}_v$$

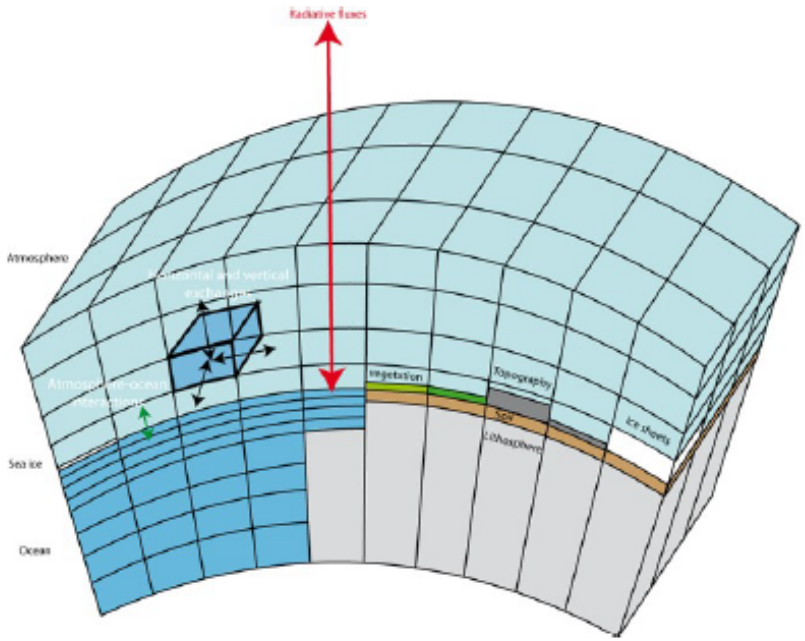
Equations

Time Stepping scheme

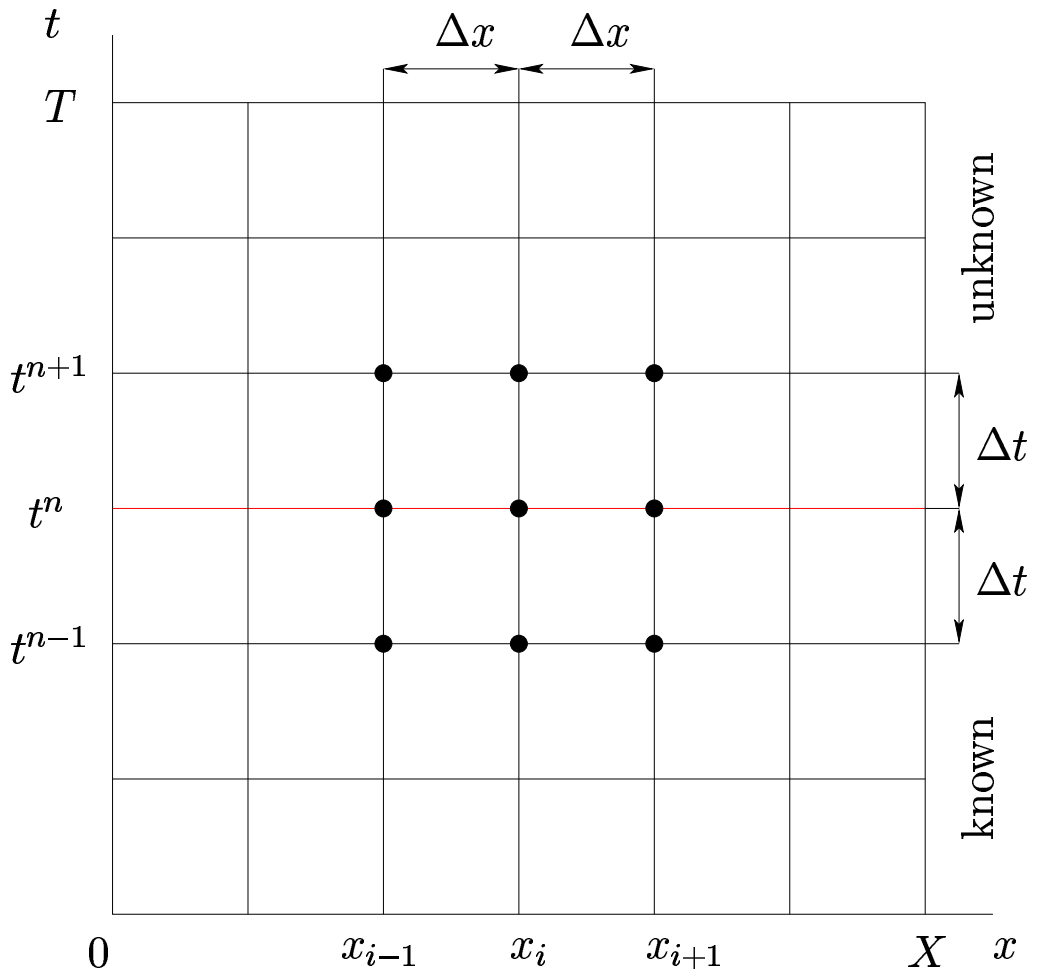
Horizontal Advection scheme

Vertical Advection scheme

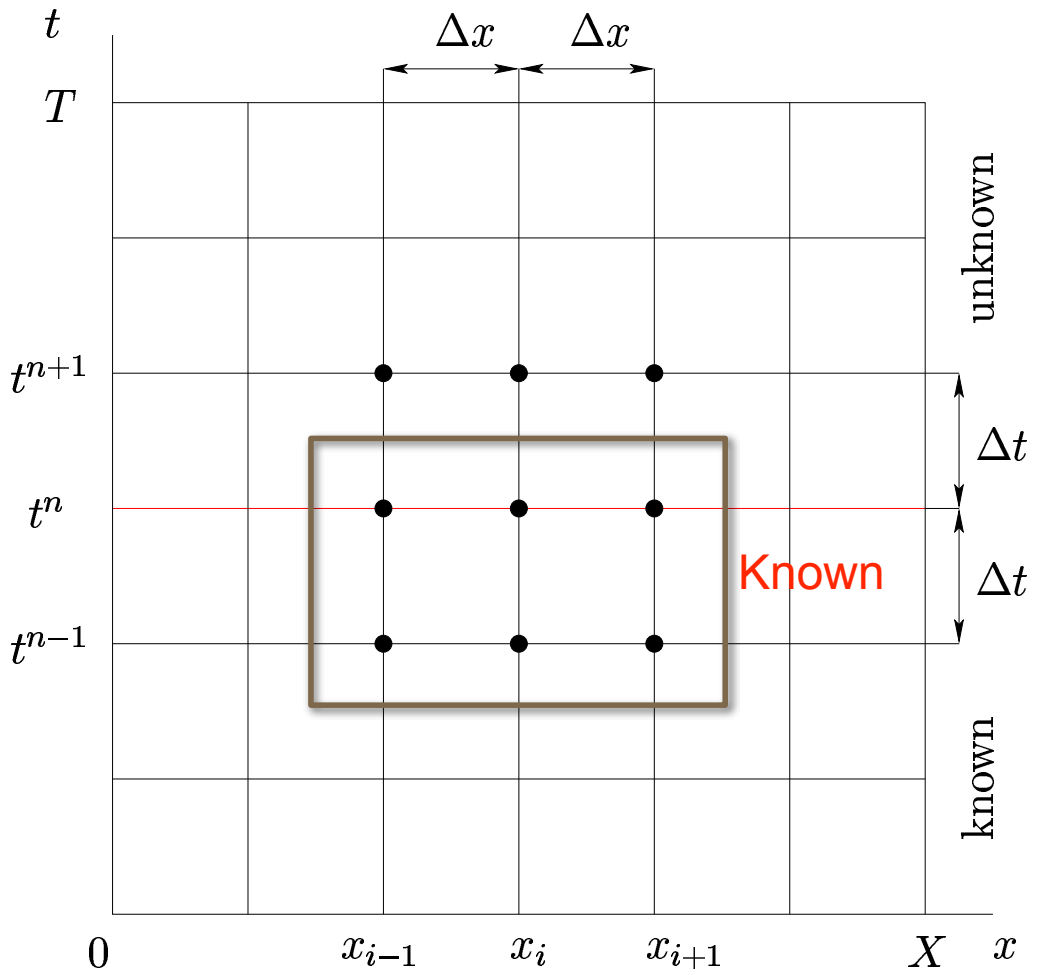
Discretization



Time-stepping schemes



Time-stepping schemes



Time-stepping scheme

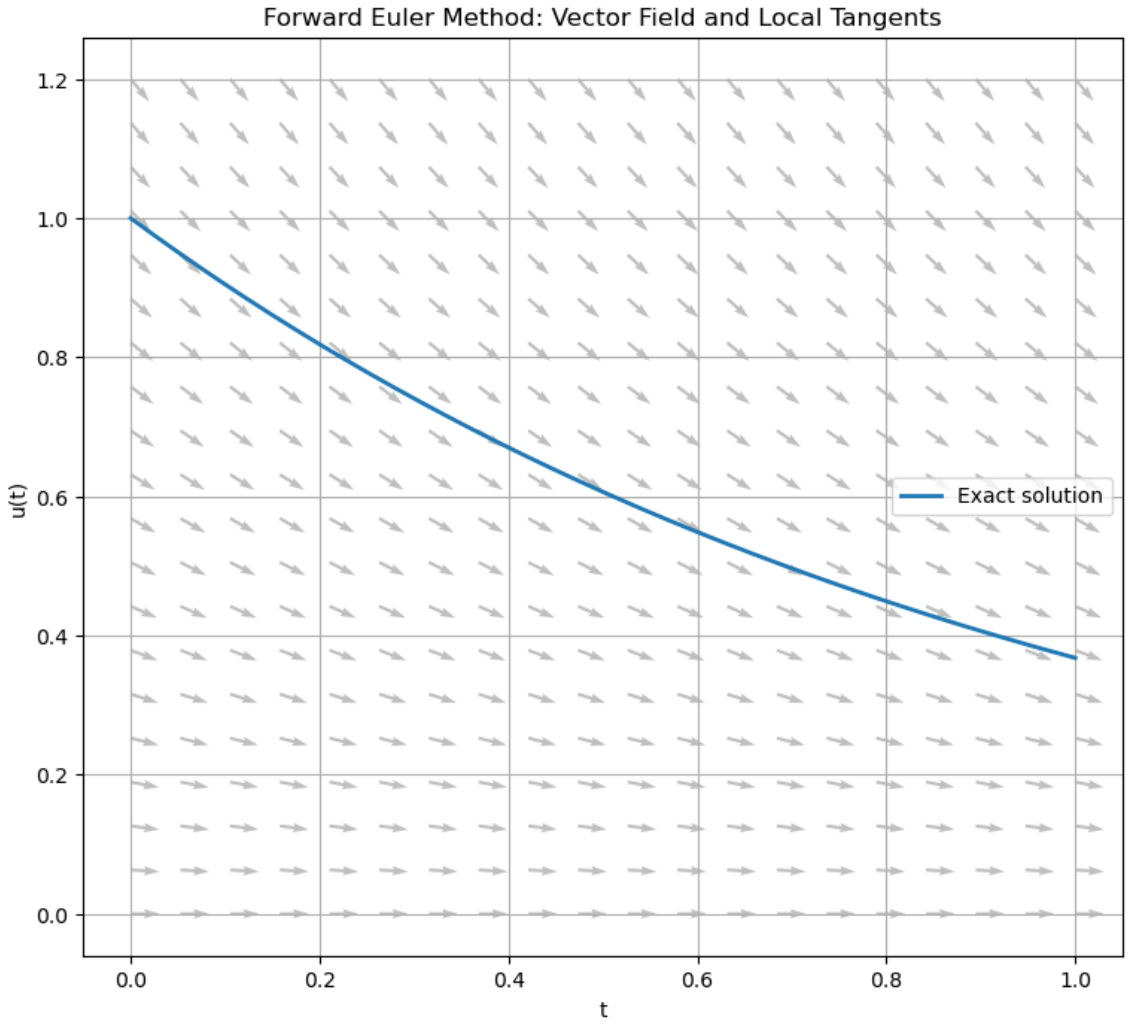
First-order ODE system

$$\begin{cases} \dfrac{du}{dt} = f(u,t), & \text{for } t \in (t^n, t^{n+1}), \\ u(t^n) = u^n, & \forall n = 0,1,\ldots,M-1. \end{cases}$$

Time-stepping scheme

First-order ODE system

$$\begin{cases} \dfrac{du}{dt} = f(u,t), & \text{for } t \in (t^n, t^{n+1}), \\ u(t^n) = u^n, & \forall n = 0,1,\ldots,M-1. \end{cases}$$

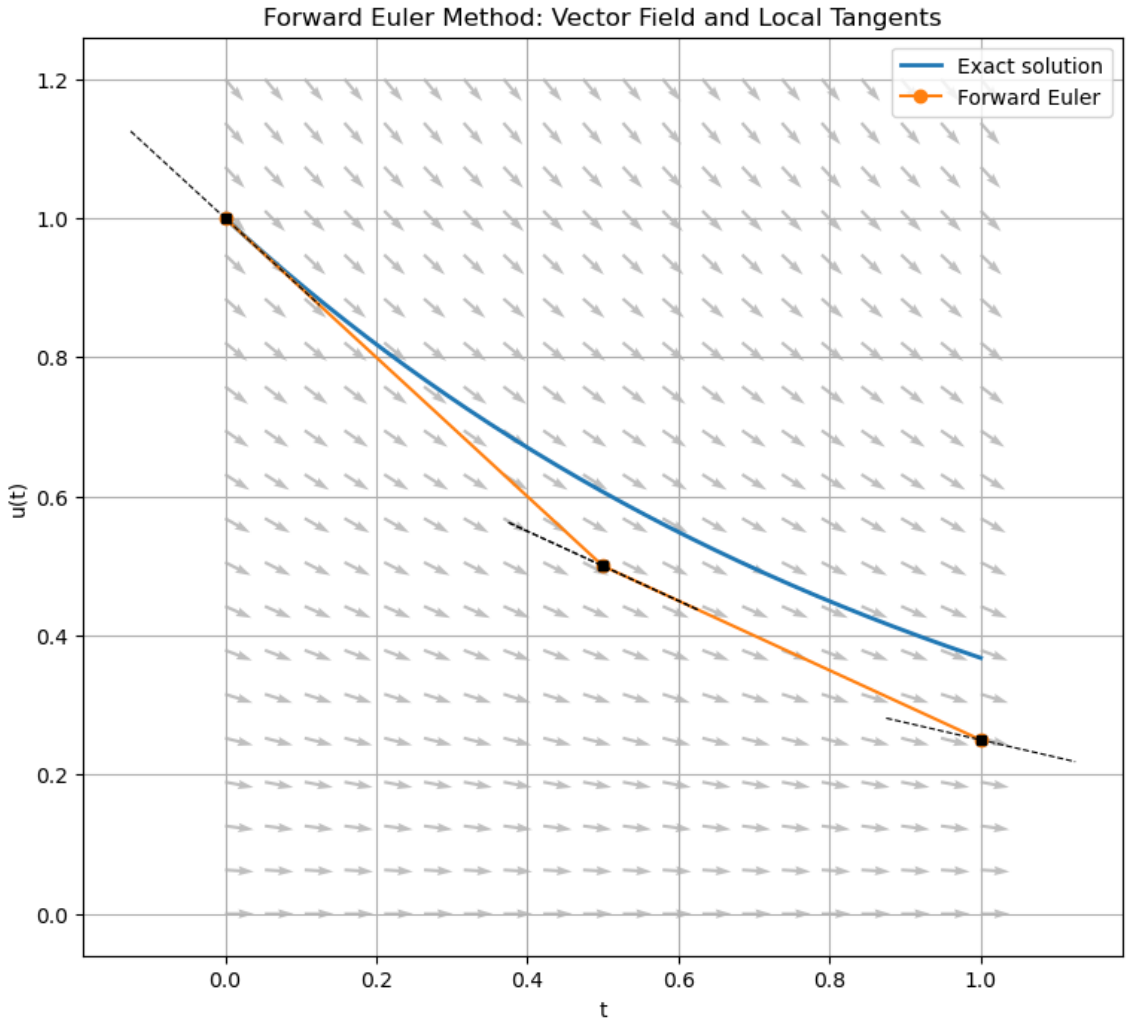


- Ex: $\dfrac{du}{dt} = -u$

$$u(t) = e^{-t}$$

Time-stepping scheme

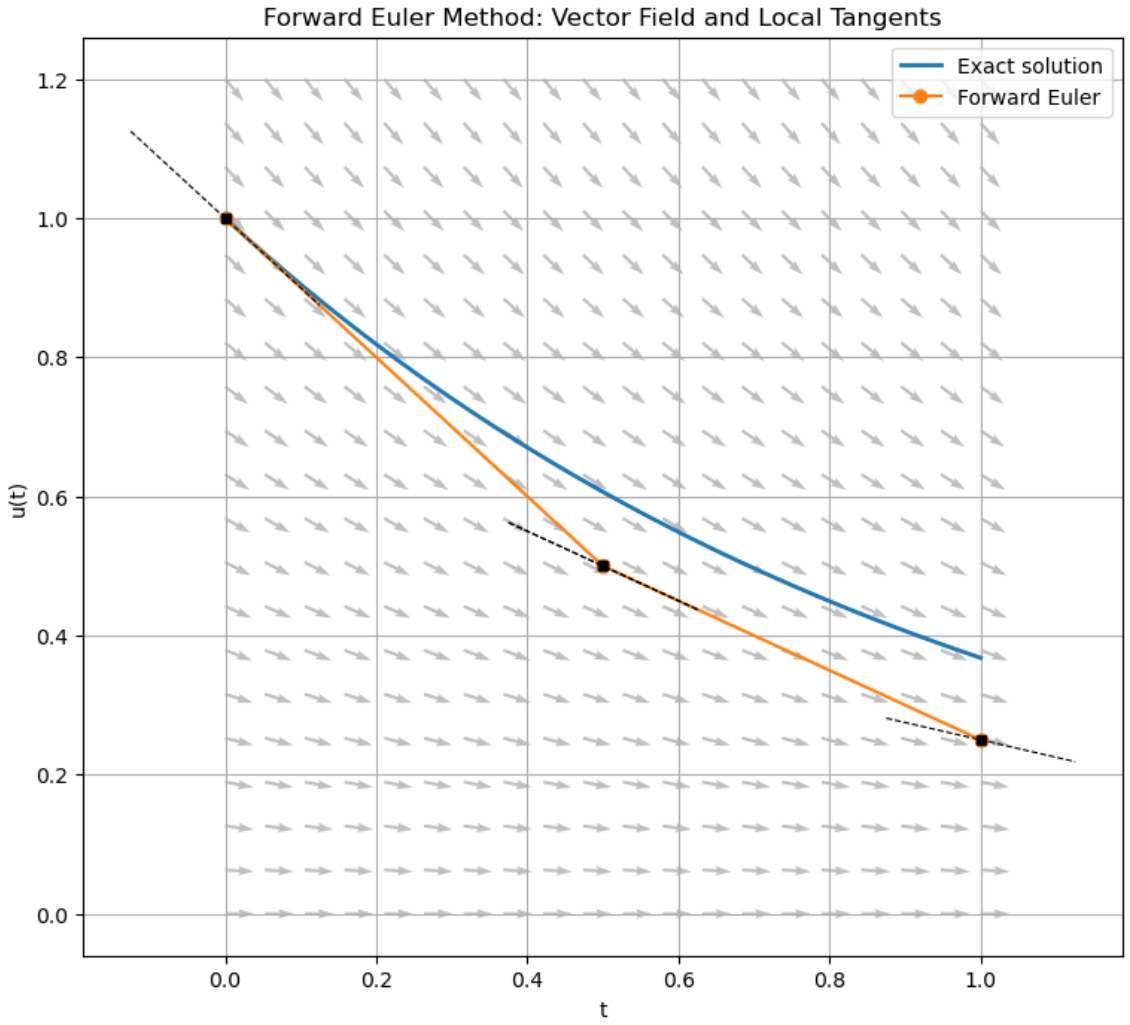
- Forward (Explicit) Euler: $$\frac{u^{n+1} - u^n}{\Delta t} = f\left(u^n, t^n\right)$$



$$u^{n+1} = u^n + \Delta t\, f\left(u^n, t^n\right)$$

Time-stepping scheme

- Forward (Explicit) Euler: $$\frac{u^{n+1} - u^n}{\Delta t} = f\left(u^n, t^n\right)$$



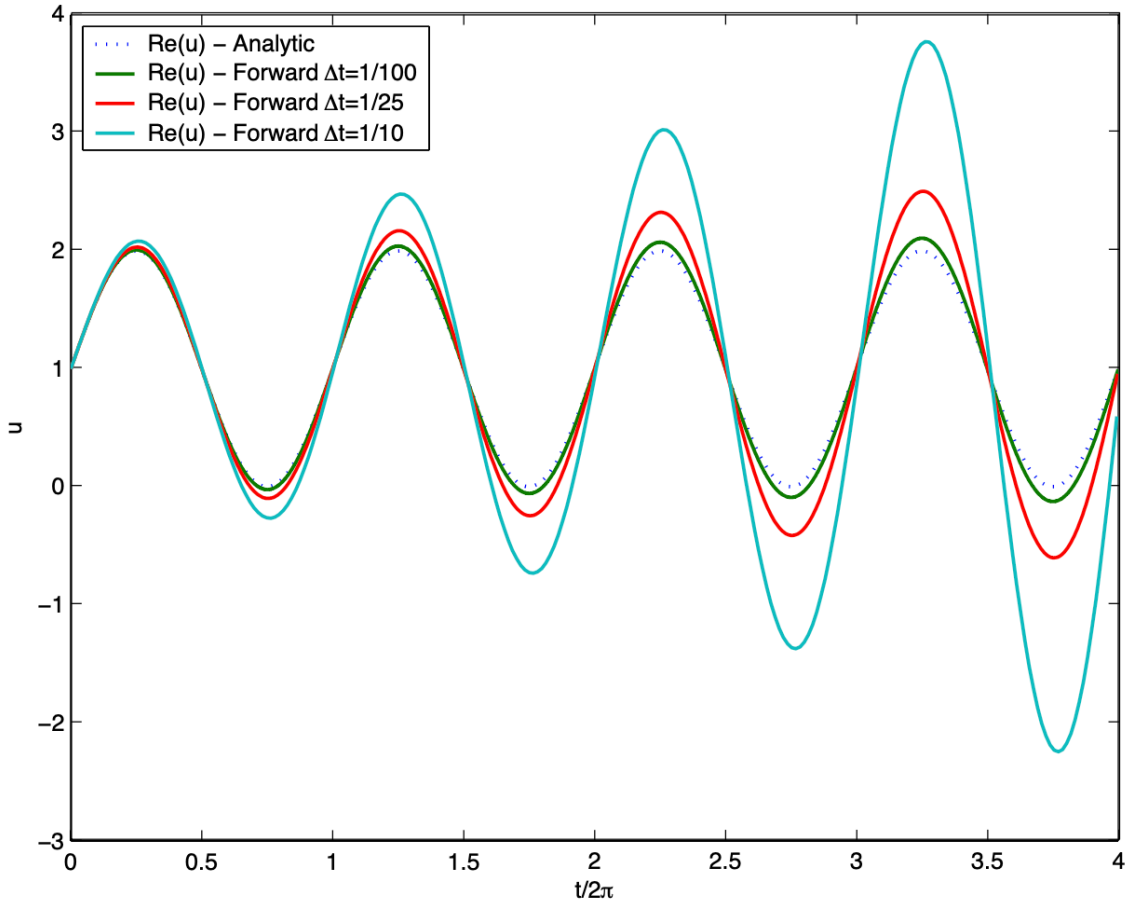
$$u^{n+1} = u^n + \Delta t\, f\left(u^n, t^n\right)$$

Explicit method

Time-stepping scheme

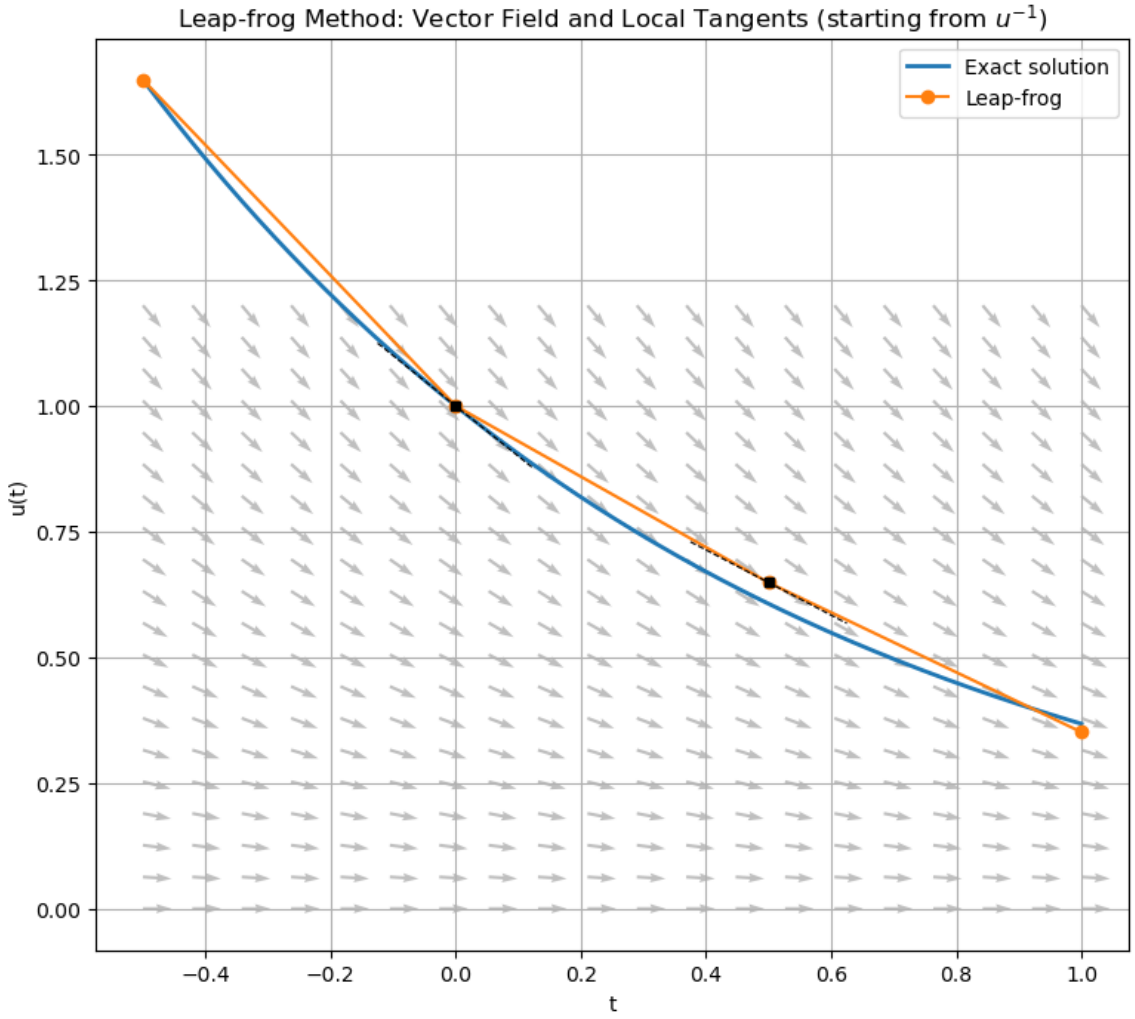
$$\partial_t u + if u = -\epsilon u + \tau$$

The forward method applied to a simple oscillation equation (f = 1 and with no damping, epsilon = 0). The solutions all grow with time but as a function of the time-step.



Time-stepping scheme

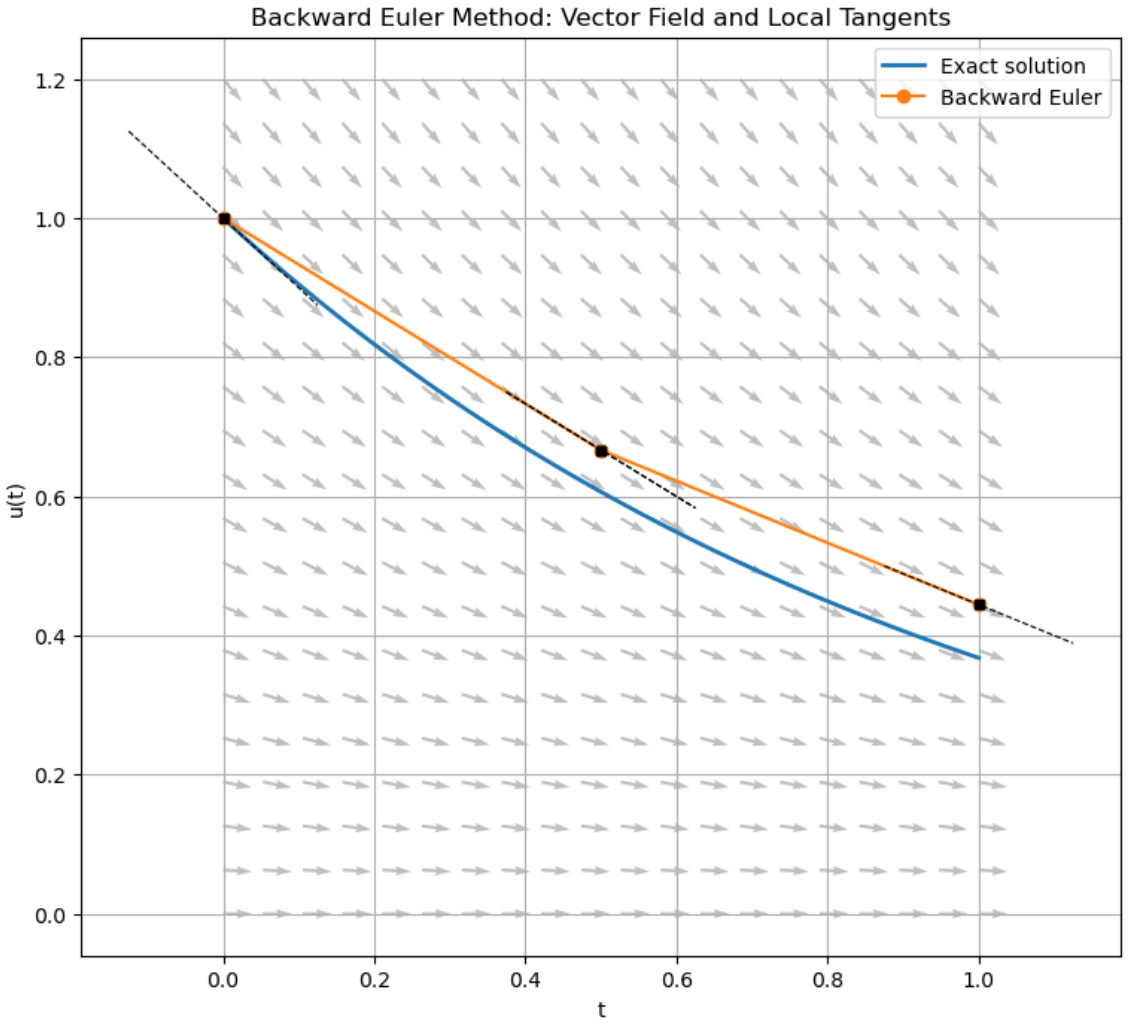
- Leap-frog $$\frac{u^{n+1} - u^{n-1}}{2\Delta t} = f\left(u^n, t^n\right)$$



$$u^{n+1} = u^{n-1} + 2\Delta t\, f(u^n, t^n)$$

Time-stepping scheme

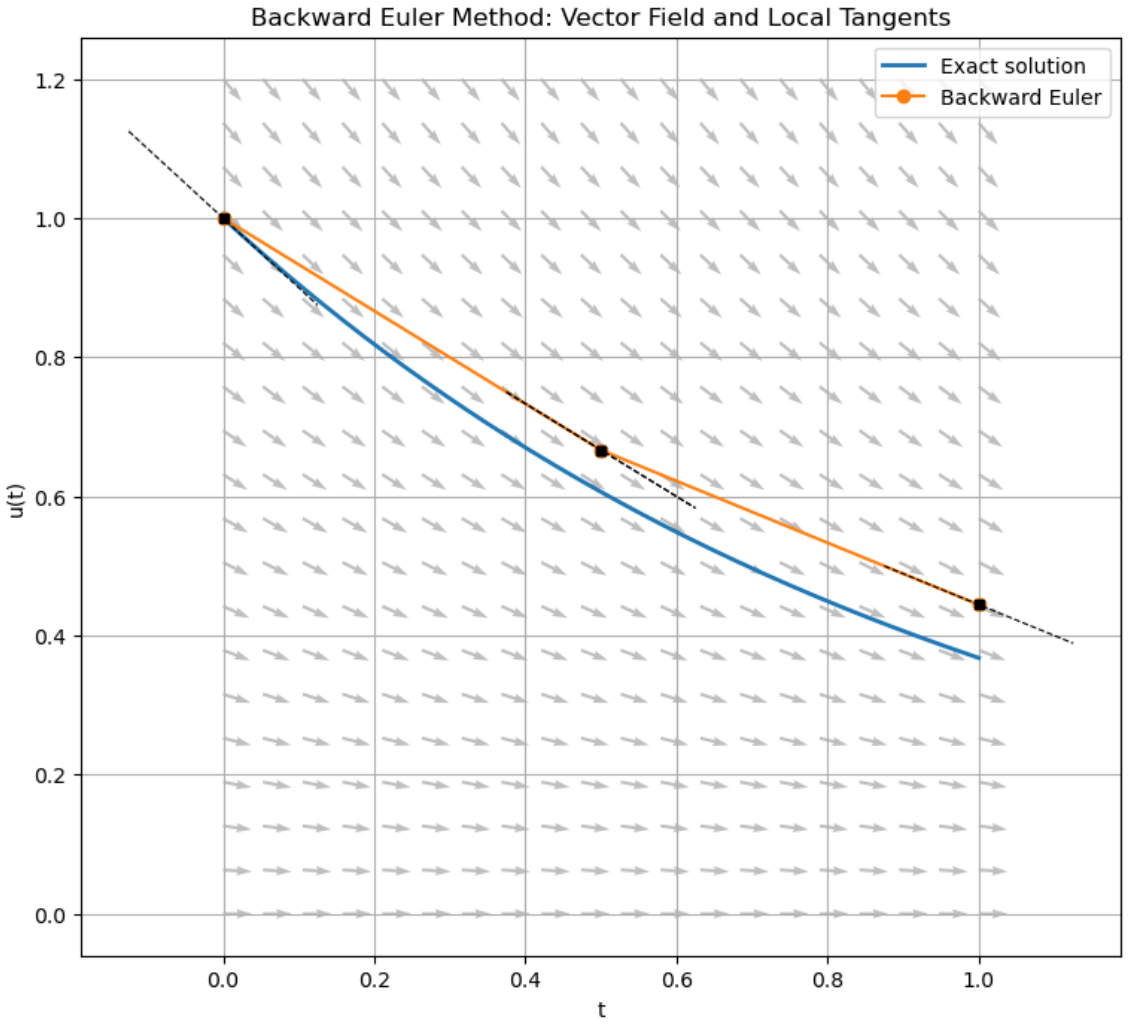
- Backward (Implicit) Euler: $$\frac{u^{n+1}-u^n}{\Delta t}=f\left(u^{n+1},t^{n+1}\right)$$



$$u^{n+1}=u^n+\Delta t\,f\left(u^{n+1},t^{n+1}\right)$$

Time-stepping scheme

- Backward (Implicit) Euler: $$\frac{u^{n+1} - u^n}{\Delta t} = f\left(u^{n+1}, t^{n+1}\right)$$



$$u^{n+1} = u^n + \Delta t\, f\left(u^{n+1}, t^{n+1}\right)$$

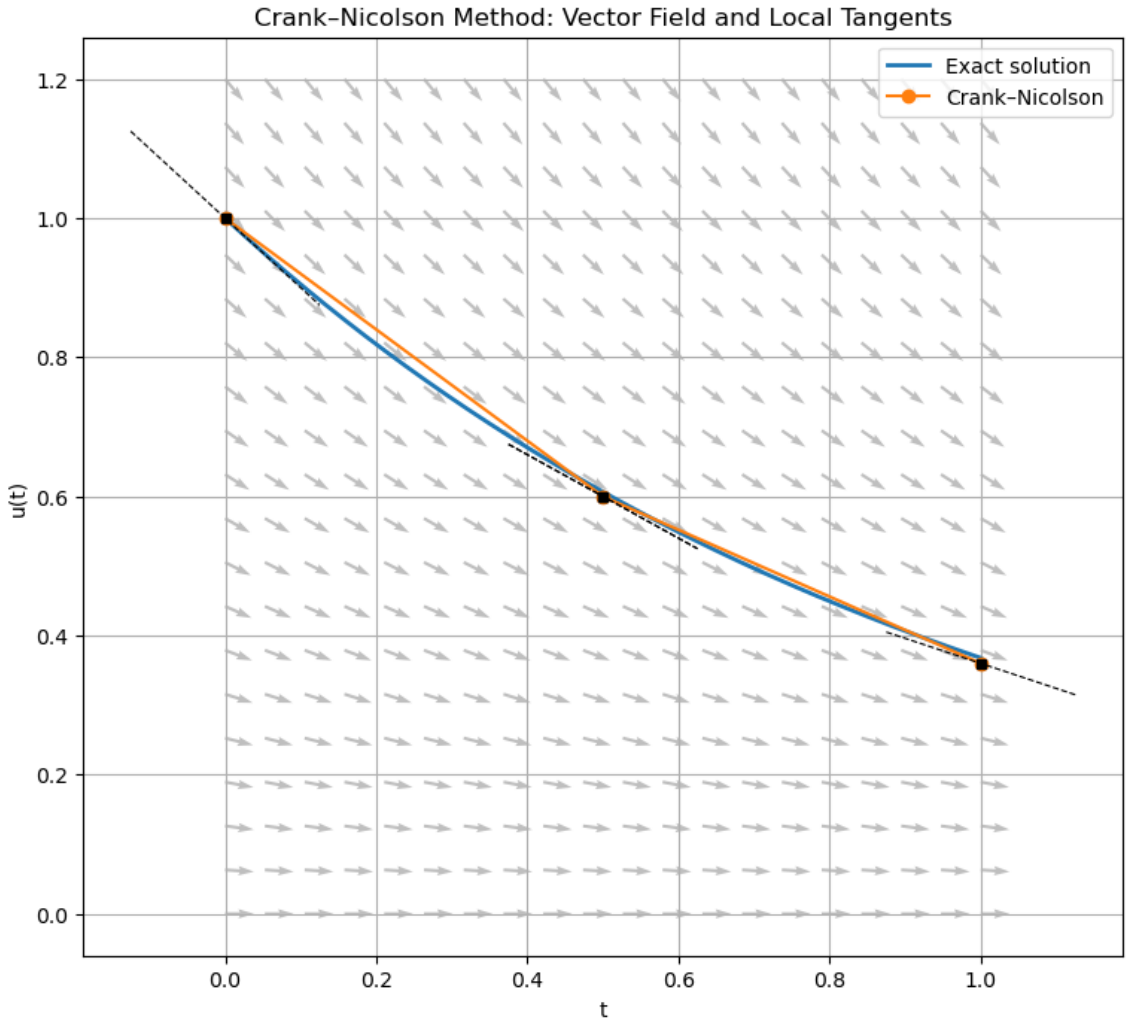
Implicit method

= *System to solve*

Time-stepping scheme

- Crank–Nicolson (Implicit):

$$u^{n+1} = u^n + \frac{\Delta t}{2}\left(f\left(u^n, t^n\right) + f\left(u^{n+1}, t^{n+1}\right)\right).$$



Example: standard time-stepping schemes

Explicit (uses only known variables)

Forward Euler	$u^{n+1} = u^n + f(t^n, u^n)\Delta t + \mathcal{O}(\Delta t)^2$
Backward Euler	$u^{n+1} = u^n + f(t^{n+1}, u^{n+1})\Delta t + \mathcal{O}(\Delta t)^2$
Crank-Nicolson	$u^{n+1} = u^n + \frac{1}{2}[f(t^n, u^n) + f(t^{n+1}, u^{n+1})]\Delta t + \mathcal{O}(\Delta t)^3$

Implicit

Explicit vs. implicit time discretization

Pros and cons of explicit schemes

- ⊕ easy to implement and parallelize, low cost per time step
- ⊕ a good starting point for the development of CFD software
- ⊖ small time steps are required for stability reasons, especially if the velocity and/or mesh size are varying strongly

Pros and cons of implicit schemes

- ⊕ stable over a wide range of time steps, sometimes unconditionally
- ⊕ constitute excellent iterative solvers for steady-state problems
- ⊖ difficult to implement and parallelize, high cost per time step
- ⊖ insufficiently accurate for truly transient problems at large Δt

Predictor-corrector and multipoint methods

Objective: to combine the simplicity of explicit schemes and robustness of implicit ones in the framework of a fractional-step algorithm, e.g.,

1. Predictor	$\tilde{u}^{n+1} = u^n + f(t^n, u^n)\Delta t$		forward Euler
2. Corrector	$u^{n+1} = u^n + \frac{1}{2}[f(t^n, u^n) + f(t^{n+1}, \tilde{u}^{n+1})]\Delta t$		Crank-Nicolson
or	$u^{n+1} = u^n + f(t^{n+1}, \tilde{u}^{n+1})\Delta t$		backward Euler

Remark. Stability still leaves a lot to be desired, additional correction steps usually do not pay off since iterations may diverge if Δt is too large

Order barrier: two-level methods are at most second-order accurate, so extra points are needed to construct higher-order integration schemes

Adams methods	$t^{n+1}, \ldots, t^{n-m}, \quad m = 0, 1, \ldots$
Runge-Kutta methods	$t^{n+\alpha} \in [t^n, t^{n+1}], \quad \alpha \in [0, 1]$

Runge-Kutta methods

Multipredictor-multicorrector algorithms of order p

$p = 2$

$$\tilde{u}^{n+1/2} = u^n + \tfrac{\Delta t}{2} f(t^n, u^n)$$ forward Euler / predictor

$$u^{n+1} = u^n + \Delta t f(t^{n+1/2}, \tilde{u}^{n+1/2})$$ midpoint rule / corrector

$p = 4$

$$\tilde{u}^{n+1/2} = u^n + \tfrac{\Delta t}{2} f(t^n, u^n)$$ forward Euler / predictor

$$\hat{u}^{n+1/2} = u^n + \tfrac{\Delta t}{2} f(t^{n+1/2}, \tilde{u}^{n+1/2})$$ backward Euler / corrector

$$\tilde{u}^{n+1} = u^n + \Delta t f(t^{n+1/2}, \hat{u}^{n+1/2})$$ midpoint rule / predictor

$$u^{n+1} = u^n + \tfrac{\Delta t}{6}[f(t^n, u^n) + 2f(t^{n+1/2}, \tilde{u}^{n+1/2})$$ Simpson rule

$$+ 2f(t^{n+1/2}, \hat{u}^{n+1/2}) + f(t^{n+1}, \tilde{u}^{n+1})]$$ corrector

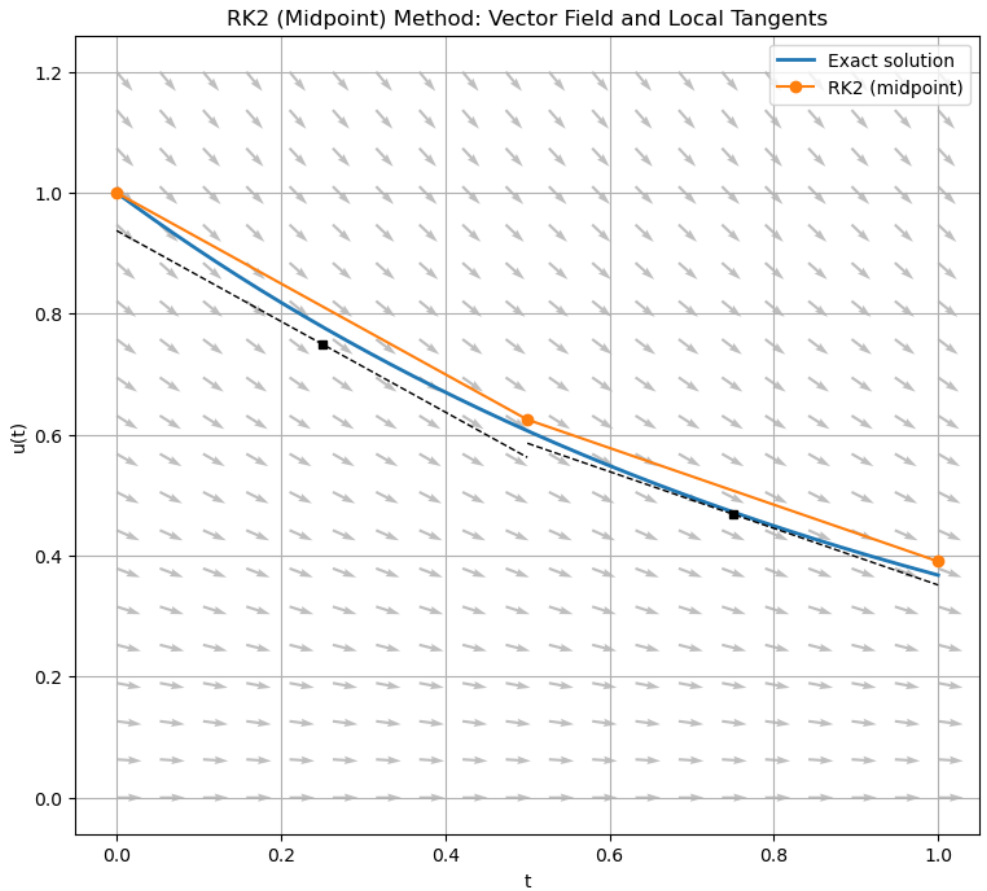
Runge-Kutta methods

$p = 2$

$$\tilde{u}^{n+1/2} = u^n + \frac{\Delta t}{2} f(t^n, u^n)$$ forward Euler / predictor

$$u^{n+1} = u^n + \Delta t f(t^{n+1/2}, \tilde{u}^{n+1/2})$$ midpoint rule / corrector

RK2



Runge-Kutta methods

$p = 4$

$$\tilde{u}^{n+1/2} = u^n + \frac{\Delta t}{2} f(t^n, u^n) \qquad \text{forward Euler / predictor}$$

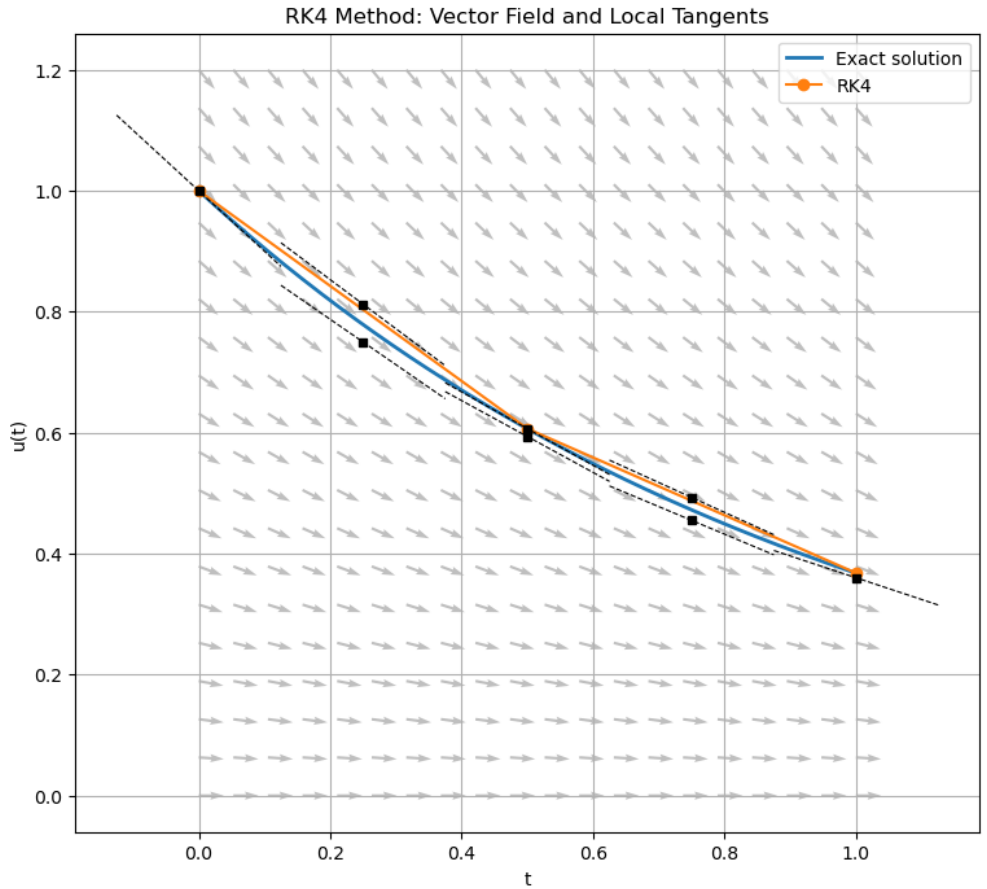
$$\hat{u}^{n+1/2} = u^n + \frac{\Delta t}{2} f(t^{n+1/2}, \tilde{u}^{n+1/2}) \qquad \text{backward Euler / corrector}$$

$$\tilde{u}^{n+1} = u^n + \Delta t f(t^{n+1/2}, \hat{u}^{n+1/2}) \qquad \text{midpoint rule / predictor}$$

$$u^{n+1} = u^n + \frac{\Delta t}{6}[f(t^n, u^n) + 2f(t^{n+1/2}, \tilde{u}^{n+1/2}) \qquad \text{Simpson rule}$$

$$+ 2f(t^{n+1/2}, \hat{u}^{n+1/2}) + f(t^{n+1}, \tilde{u}^{n+1})] \qquad \text{corrector}$$

RK4



Runge-Kutta methods

Pros and cons of Runge-Kutta methods

- ⊕ self-starting, easy to operate with variable time steps
- ⊕ more stable and accurate than Adams methods of the same order
- ⊖ high order approximations are rather difficult to derive; p function evaluations per time step are required for a $p-$th order method
- ⊖ more expensive than Adams methods of comparable order

Adaptive time-stepping strategy $\Delta t \quad \Delta t \quad \Delta t \quad \Delta t \quad \Delta t \quad \Delta t \quad \Delta t \quad \Delta t$

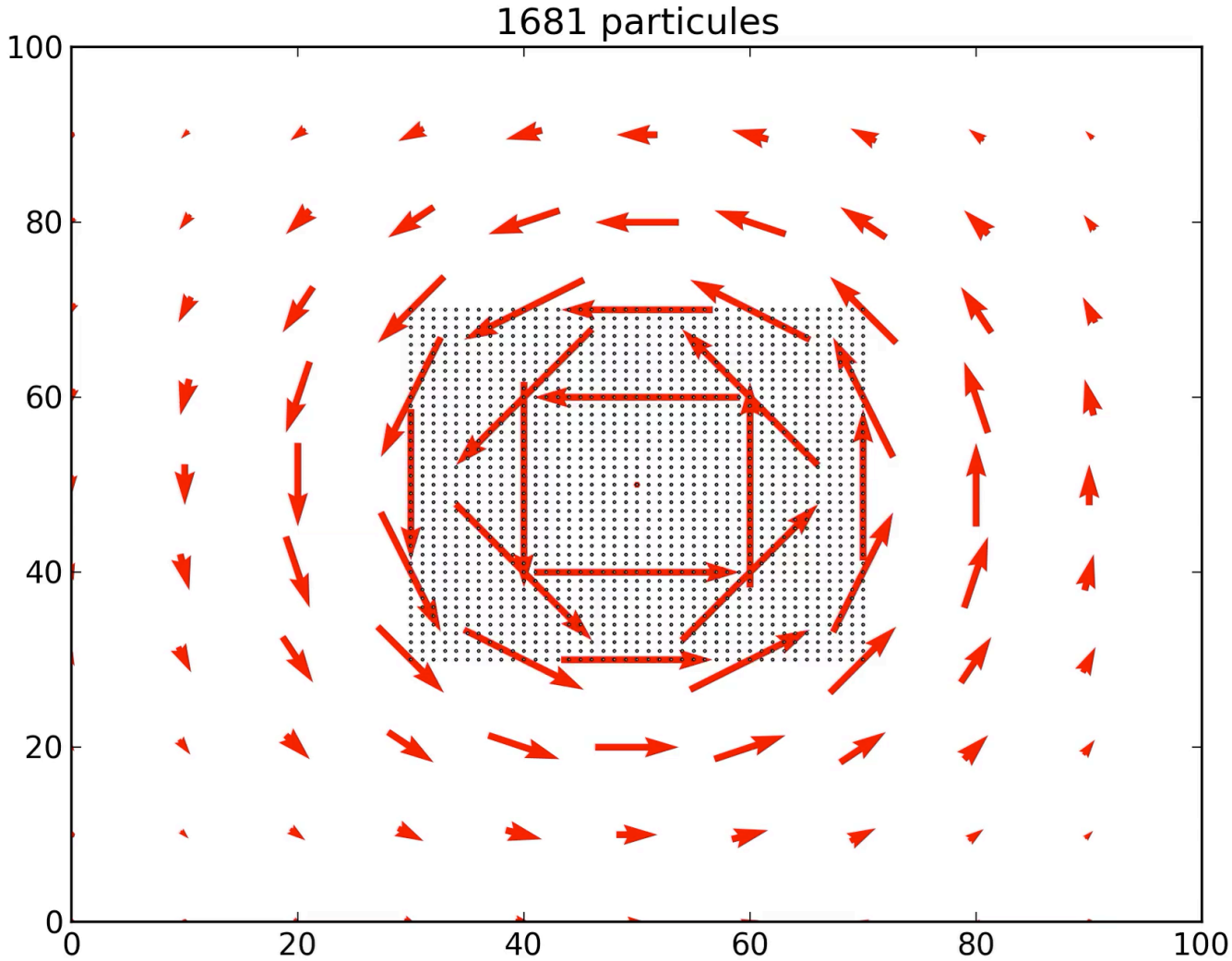
makes it possible to achieve the desired accuracy at a relatively low cost

Explicit methods: *use the largest time step satisfying the stability condition*

Implicit methods: *estimate the error and adjust the time step if necessary*

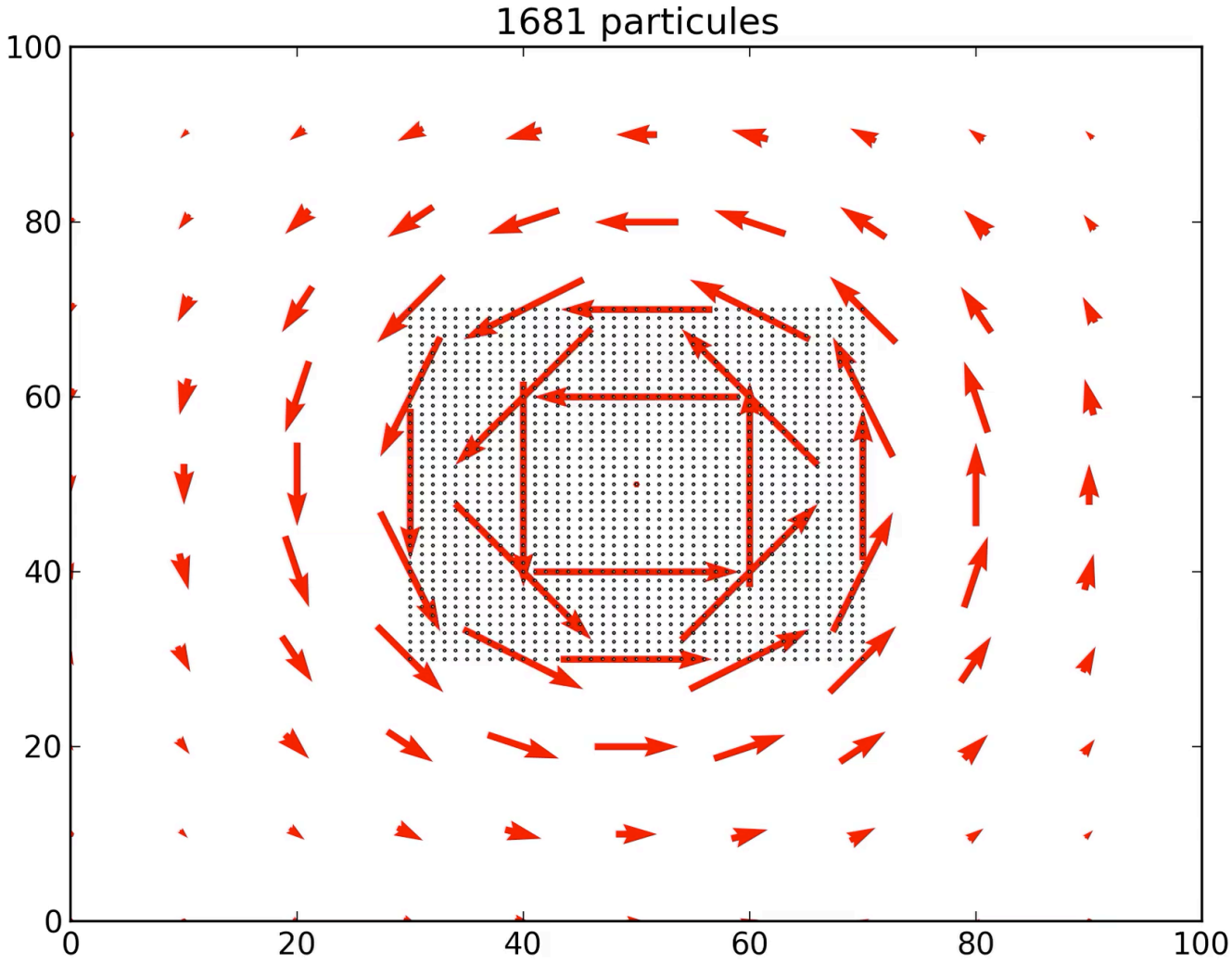
Time-stepping scheme

- Example using a FE time-stepping scheme for advection of Lagrangian particles in a time-invariant velocity field:

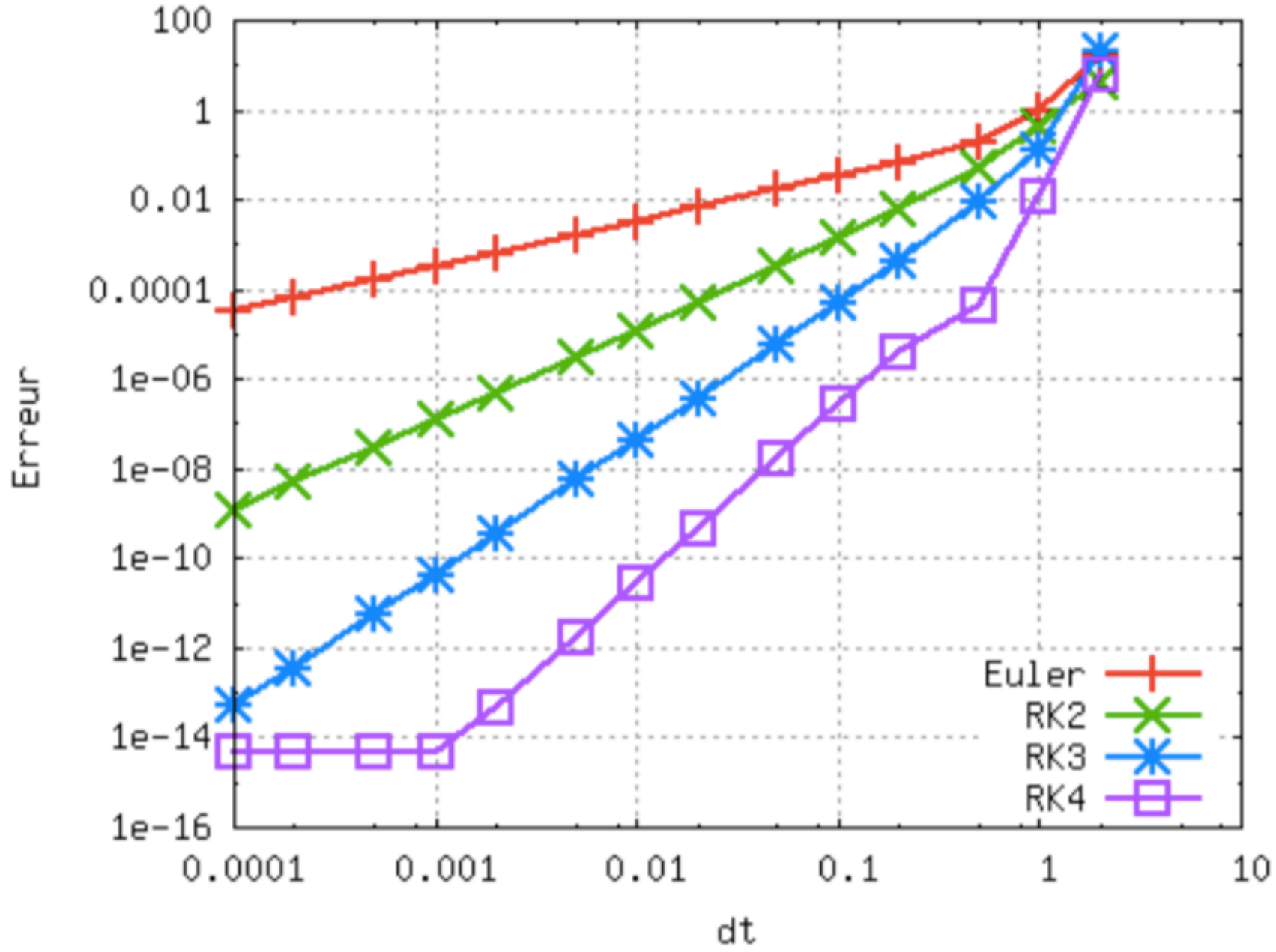


Time-stepping scheme

- Example using a RK4 time-stepping scheme for advection of Lagrangian particles in a time-invariant velocity field:



Time-stepping scheme

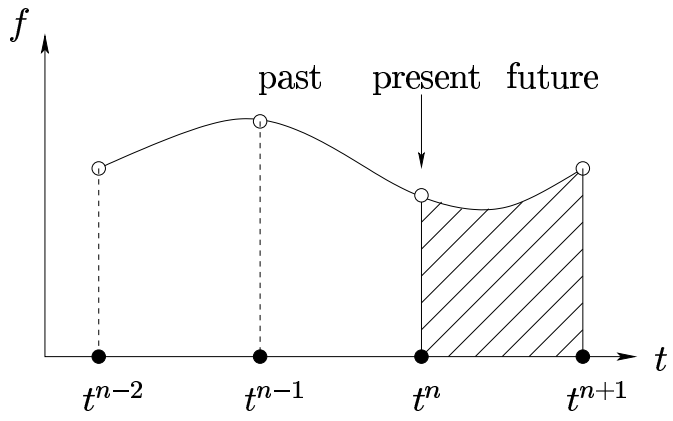


Adams methods

Derivation: *polynomial fitting*

Truncation error: $\epsilon_\tau^{\text{glob}} = \mathcal{O}(\Delta t)^p$

for polynomials of degree $p-1$ which interpolate function values at p points



Adams-Bashforth methods (explicit)

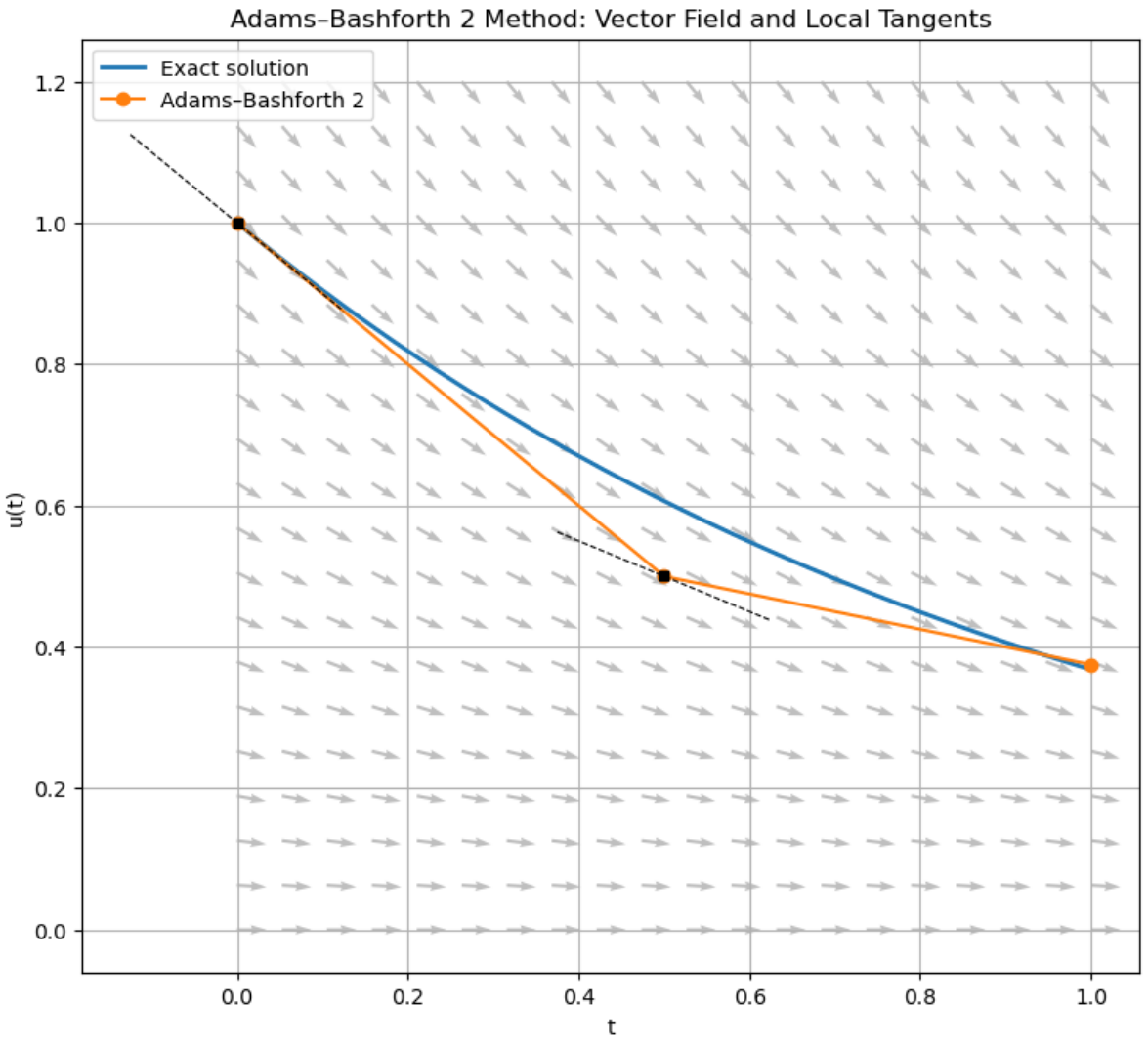
$p = 1$ $\quad u^{n+1} = u^n + \Delta t f(t^n, u^n)$ forward Euler

$p = 2$ $\quad u^{n+1} = u^n + \frac{\Delta t}{2}[3f(t^n, u^n) - f(t^{n-1}, u^{n-1})]$

$p = 3$ $\quad u^{n+1} = u^n + \frac{\Delta t}{12}[23f(t^n, u^n) - 16f(t^{n-1}, u^{n-1}) + 5f(t^{n-2}, u^{n-2})]$

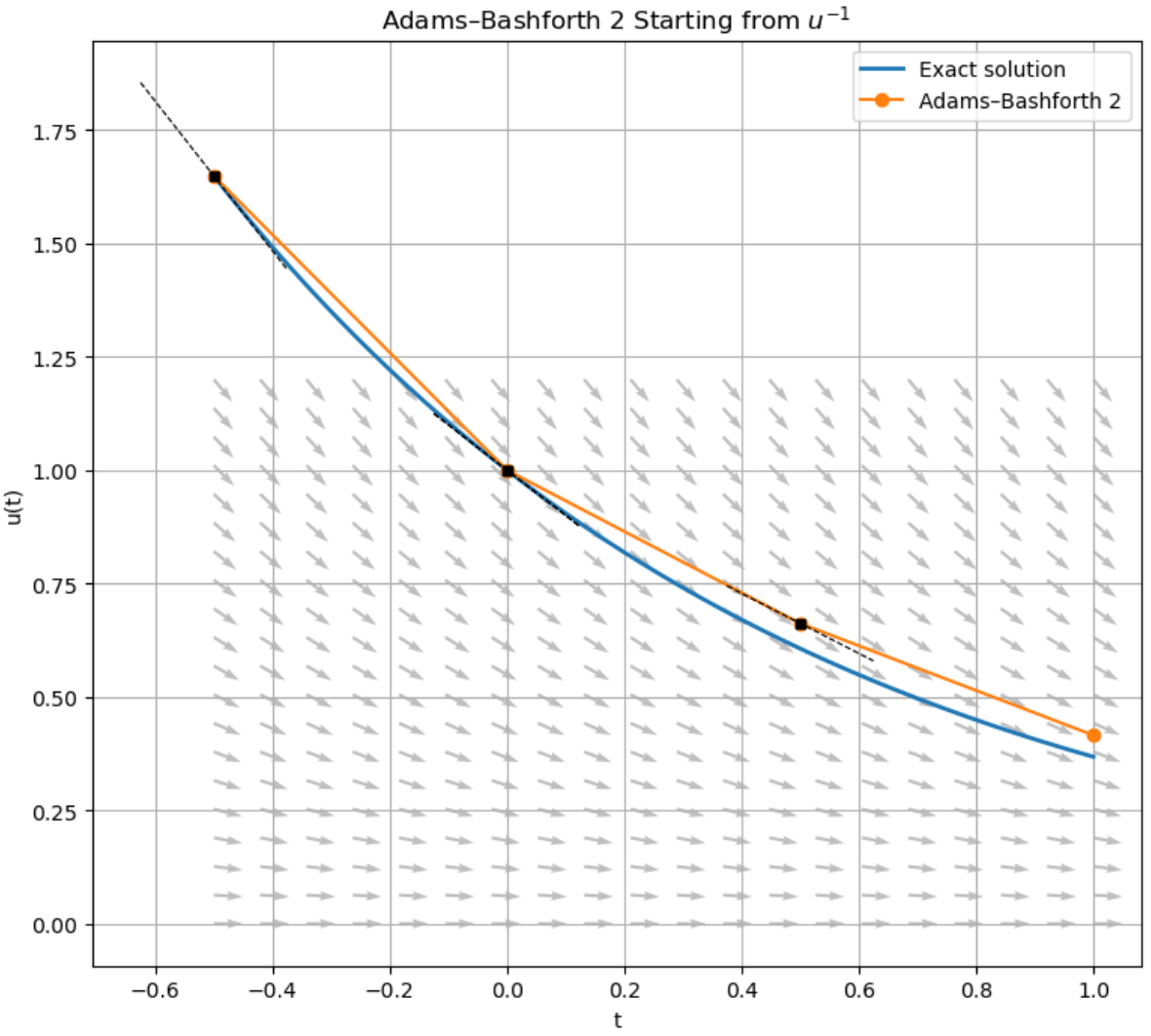
Adams methods

$p = 2$ $\qquad u^{n+1} = u^n + \frac{\Delta t}{2}[3f(t^n, u^n) - f(t^{n-1}, u^{n-1})]$



Adams methods

$$p = 2 \qquad u^{n+1} = u^n + \frac{\Delta t}{2}[3f(t^n, u^n) - f(t^{n-1}, u^{n-1})]$$

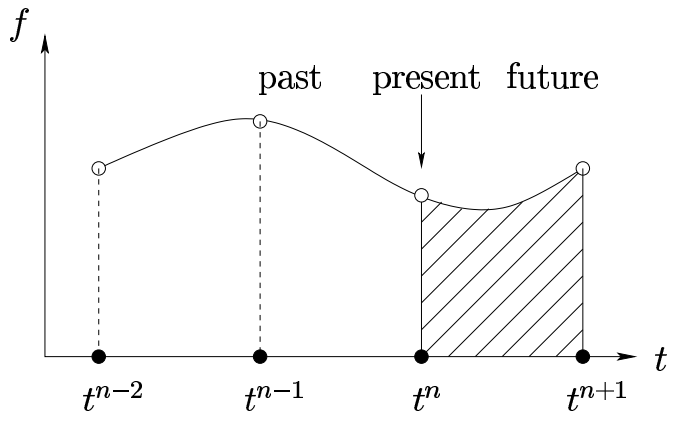


Adams methods

Derivation: *polynomial fitting*

Truncation error: $\epsilon_\tau^{\text{glob}} = \mathcal{O}(\Delta t)^p$

for polynomials of degree $p-1$ which interpolate function values at p points



Adams-Bashforth methods (explicit)

$p=1$ $\quad u^{n+1} = u^n + \Delta t f(t^n, u^n)$ forward Euler

$p=2$ $\quad u^{n+1} = u^n + \frac{\Delta t}{2}[3f(t^n, u^n) - f(t^{n-1}, u^{n-1})]$

$p=3$ $\quad u^{n+1} = u^n + \frac{\Delta t}{12}[23f(t^n, u^n) - 16f(t^{n-1}, u^{n-1}) + 5f(t^{n-2}, u^{n-2})]$

Adams-Moulton methods (implicit)

$p=1$ $\quad u^{n+1} = u^n + \Delta t f(t^{n+1}, u^{n+1})$ backward Euler

$p=2$ $\quad u^{n+1} = u^n + \frac{\Delta t}{2}[f(t^{n+1}, u^{n+1}) + f(t^n, u^n)]$ Crank-Nicolson

$p=3$ $\quad u^{n+1} = u^n + \frac{\Delta t}{12}[5f(t^{n+1}, u^{n+1}) + 8f(t^n, u^n) - f(t^{n-1}, u^{n-1})]$

Adams methods

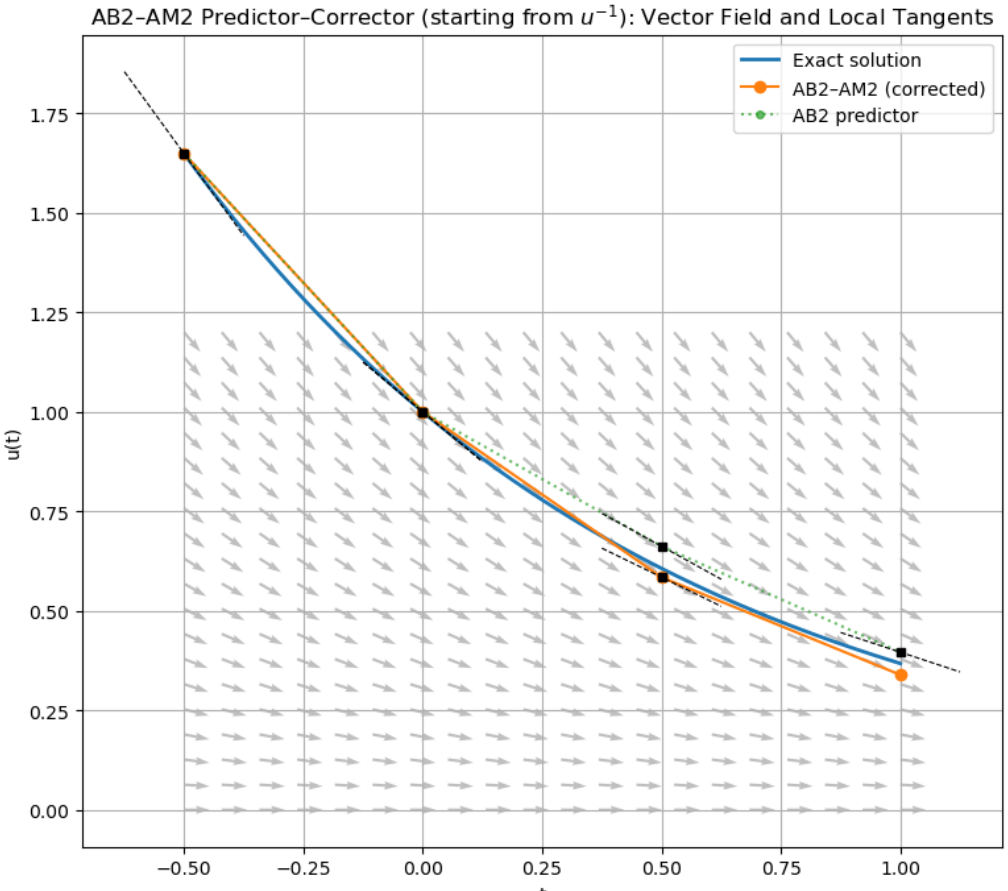
Predictor-corrector algorithm

1. Compute $\tilde{u}^{n+1}$ using an Adams-Bashforth method of order $p-1$
2. Compute u^{n+1} using an Adams-Moulton method of order p with predicted value $f(t^{n+1}, \tilde{u}^{n+1})$ instead of $f(t^{n+1}, u^{n+1})$

ex: AB2 - AM2

Predict: $\tilde{u}^{n+1} = u^n + \Delta t \left(\frac{3}{2} f^n - \frac{1}{2} f^{n-1} \right),$

Correct: $u^{n+1} = u^n + \frac{\Delta t}{2} \left(f(\tilde{u}^{n+1}, t^{n+1}) + f^n \right)$



Adams methods

Predictor-corrector algorithm

1. Compute $\tilde{u}^{n+1}$ using an Adams-Bashforth method of order $p-1$
2. Compute u^{n+1} using an Adams-Moulton method of order p with predicted value $f(t^{n+1}, \tilde{u}^{n+1})$ instead of $f(t^{n+1}, u^{n+1})$

Pros and cons of Adams methods

- $\oplus$ methods of any order are easy to derive and implement
- $\oplus$ only one function evaluation per time step is performed
- $\oplus$ error estimators for ODEs can be used to adapt the order
- $\ominus$ other methods are needed to start/restart the calculation
- $\ominus$ time step is difficult to change (coefficients are different)
- $\ominus$ tend to be unstable and produce nonphysical oscillations

Finite difference schemes

- A finite difference scheme is produced when the partial derivatives in the partial differential equations governing a physical phenomenon are replaced by a finite difference approximation.
- The result is a system of algebraic equations which, when solved, provide an approximation to the solution of the original partial differential equations at selected points of a solution grid.

- The Most important characteristic of a scheme are: Consistence, Convergence and Stability

Finite difference schemes

- Consistence = A finite difference scheme is consistent if the operator reduces to the original differential equation as the increments in the independent variables vanish ($\delta x_i, \delta t \to 0$)

- Convergence = The finite-difference solution approaches the true solution to the partial differential equation as the increments go to zero ($\delta x_i, \delta t \to 0$)

- Stability = The error caused by a small perturbation in the numerical solution remains bound.

- *for a well-posed initial-value problem, the Lax-Richtmyer theorem states that, for a consistent numerical method, stability and convergence are equivalent.*

Stability:

- The basic idea for the convergence and stability analysis for a linear partial differential equation consists in writing the solution to the equation as a complex Fourier series and analyzing a generic component of the solution.

- Example: ***The advection equation***

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} = 0$$

Stability:

Example: ***The advection equation (u>0)*** $\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} = 0$

We use a forward Euler in time and a downstream scheme in space:

$$\frac{\partial T}{\partial t} = \frac{T(t+\delta t) - T(t)}{\delta t} = \frac{T_{i,n+1} - T_{i,n}}{\delta t}$$

$$\frac{\partial T}{\partial x} = \frac{T(x+\delta x) - T(x)}{\delta x} = \frac{T_{i+1,n} - T_{i,n}}{\delta x}$$

Such that:

$$T_{i,n+1} = T_{i,n} - u\frac{\delta t}{\delta x}(T_{i+1,n} - T_{i,n})$$

Stability:

Example: ***The advection equation*** $\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} = 0$

$$T_{i,n+1} = T_{i,n} - u\frac{\delta t}{\delta x}(T_{i+1,n} - T_{i,n})$$

We introduce: $T_{i,n} = \hat{T}_n e^{ikx}$

We get $\hat{T}_{n+1}e^{ikx} = \hat{T}_n e^{ikx} - u\frac{\delta t}{\delta x}(\hat{T}_n e^{ik(x+\delta x)} - \hat{T}_n e^{ikx})$

Stability:

Example: ***The advection equation*** $\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} = 0$

$$T_{i,n+1} = T_{i,n} - u\frac{\delta t}{\delta x}(T_{i+1,n} - T_{i,n})$$

We introduce: $T_{i,n} = \hat{T}_n e^{ikx}$

We get $\hat{T}_{n+1}e^{ikx} = \hat{T}_n e^{ikx} - u\frac{\delta t}{\delta x}(\hat{T}_n e^{ik(x+\delta x)} - \hat{T}_n e^{ikx})$

The amplification is $A = \frac{\hat{T}_{n+1}}{\hat{T}_n} = 1 - u\frac{\delta t}{\delta x}(e^{ik\delta x} - 1)$

Stability:

The amplification is $$A = \frac{\hat{T}_{n+1}}{\hat{T}_n} = 1 - u\frac{\delta t}{\delta x}(e^{ik\delta x} - 1)$$

In general a numerical scheme is

unstable if $|A| > 1$

stable if $|A| < 1$

For the Euler/downstream: $$|A|^2 = |1 - C(e^{ik\delta x} - 1)|^2$$

With $$C = u\frac{\delta t}{\delta x}$$

So $$|A|^2 = (1 - C\cos(k\delta x) + C)^2 + C^2\sin(k\delta x)^2$$

Stability:

$$|A|^2 = (1 - C\cos(k\delta x) + C)^2 + C^2\sin(k\delta x)^2$$

$$|A|^2 = (1 + C)^2 - 2(1 + C)C\cos(k\delta x) + C^2$$

$$|A|^2 = 1 + 2(1 + C)C(1 - \cos(k\delta x))$$

$$> 0$$

For positive u: $|A| > 1$

The Euler/downstream is unconditionally unstable

Stability:

- Example: ***The advection equation*** $$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} = 0$$

- Activity:
  - Write a discrete equation using a forward Euler in time and an upstream scheme in space:
  - Check its stability

Stability

Courant-Friedrichs-Levy (CFL) stability criterion (1d) :

$$C = u\frac{\delta t}{\delta x} \leq C_{max}$$

… the time step must be kept small enough so that information has enough time to propagate through the space discretization

- For an explicit solver, typically: $C_{max} = 1$
- The CFL condition is a necessary condition, but may not be sufficient for stability.

Stability

Courant-Friedrichs-Levy (CFL) stability criterion :

$$C = \delta t \sum_{i=1}^{n} \frac{u_i}{\delta x_i} \leq C_{max}$$

… the time step must be kept small enough so that information has enough time to propagate through the space discretization

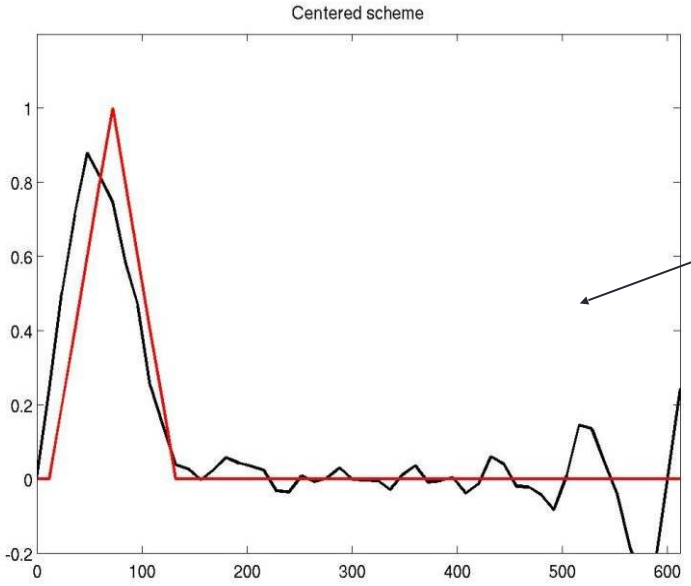
- For an explicit solver, typically: $C_{max} = 1$
- The CFL condition is a necessary condition, but may not be sufficient for stability.

Stability:

- Example: ***The advection equation*** $$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} = 0$$

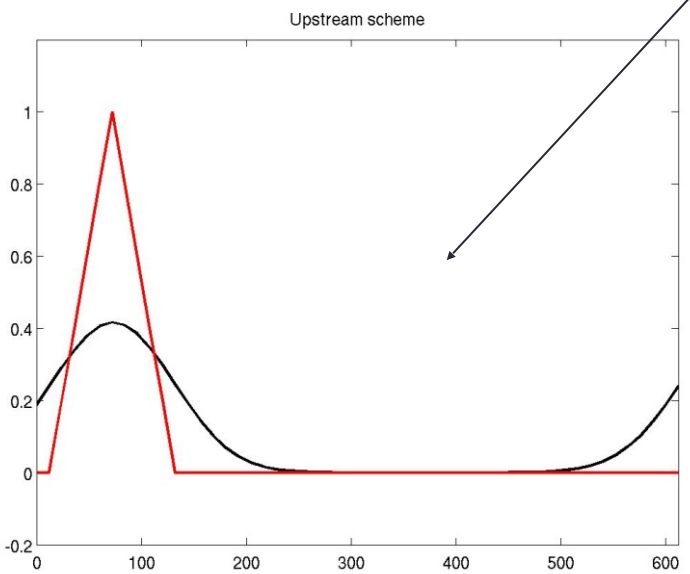
- Activity:
  - Write a discrete equation using a forward Euler in time and **a centered scheme** in space:
  - Check its stability

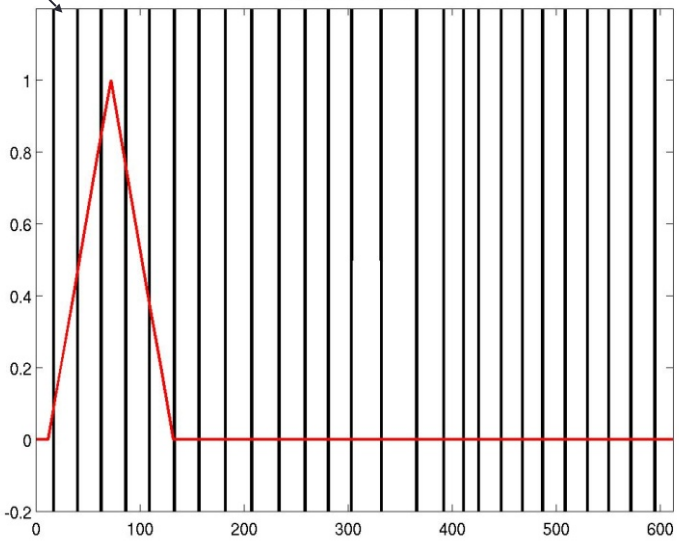
Numerical properties



A numerical scheme can be:

- **Dispersive**: ripples, overshoot and extrema (Leapfrog/Centered)
- **Diffusive** (Euler /Upstream)
- **Unstable** (Euler/Centered)





Stability:

- Example: ***The advection equation*** $$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} = 0$$

- One way to stabilise the Euler/Centered is to use a **Lax Friedrichs**:

$$T_{i,n}$$

$$\downarrow$$

$$T_{i,n+1} = \frac{T_{i+1,n} + T_{i-1,n}}{2} - u\frac{\delta t}{\delta x}\frac{T_{i+1,n} - T_{i-1,n}}{2}$$

Stable (if C<1) but diffusive

Stability:

- Example: ***The advection equation*** $\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} = 0$

- We can have a 2nd order in time by using:

$$T_{n+1} = T_n + \delta t T_t + \frac{1}{2}\delta t^2 T_{tt} + \ldots$$

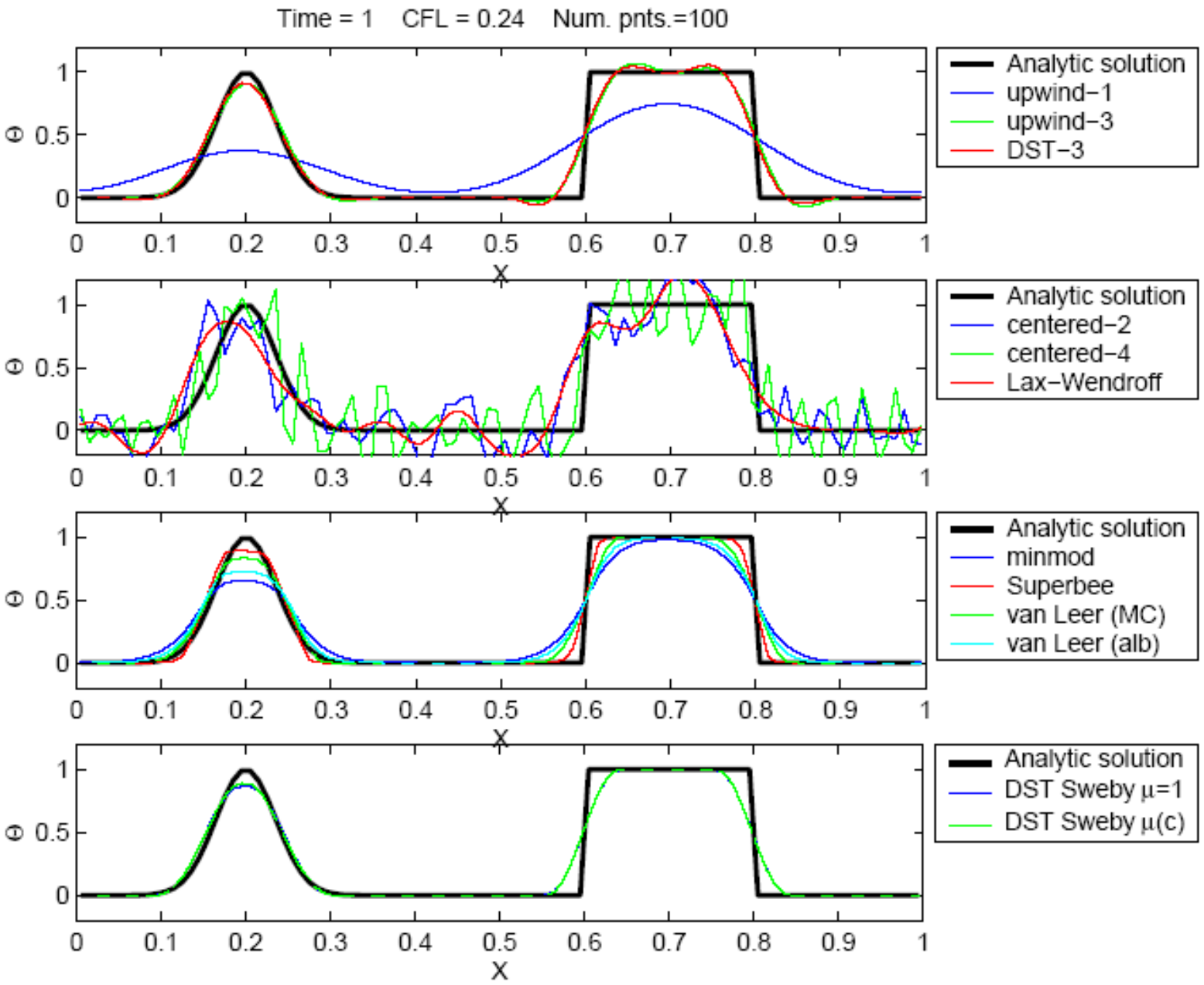
$$T_{tt} = -uT_{tx} = u^2 T_{xx}$$

- Such that

$$T_{i,n+1} = T_{i,n} - u\frac{\delta t}{\delta x}\frac{T_{i+1,n} - T_{i-1,n}}{2} + \left(u\frac{\delta t}{\delta x}\right)^2 \frac{T_{i+1,n} + T_{i-1,n} - 2T_{i,n}}{2}$$

It is the **Lax-Wendroff** Scheme - Stable but dispersive!

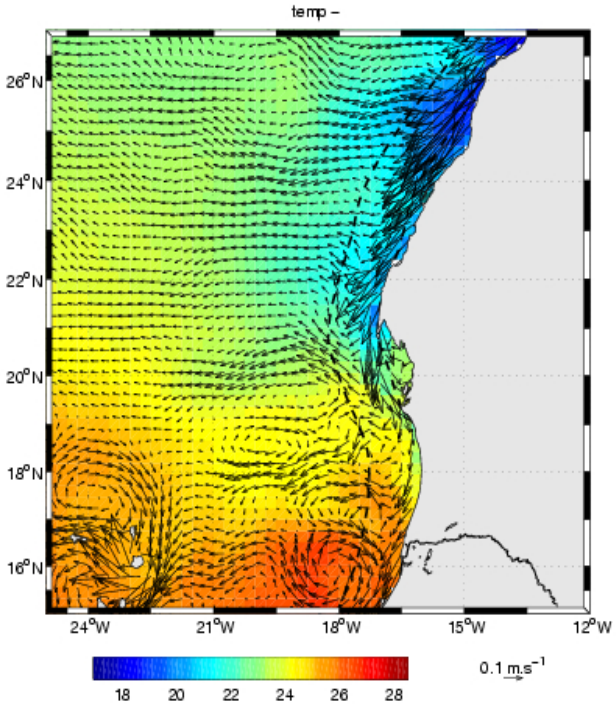
Illustration of different numerical schemes



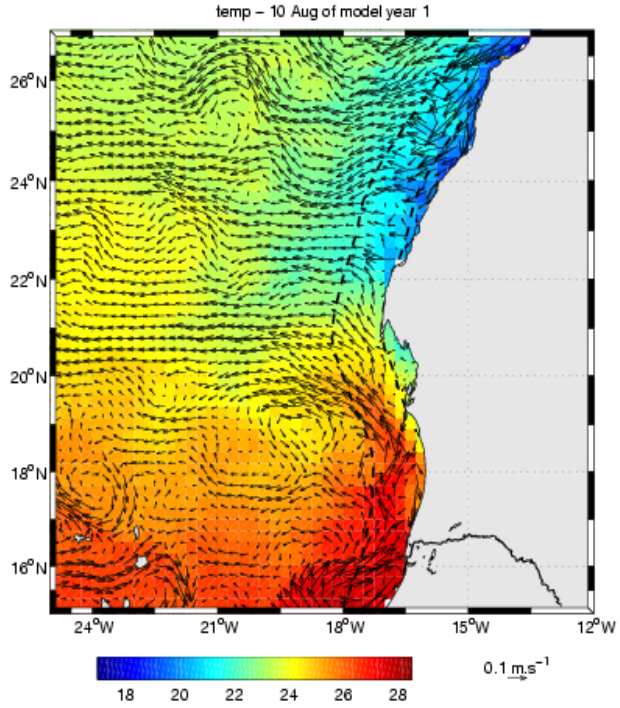
Spatial advection scheme in ROMS/CROCO

- 3rd or 5th order, upstream-biased advection scheme : allows the generation of steep gradient, with a weak dispersion and weak diffusion.
- No need to impose explicit diffusion/ viscosity to avoid numerical noise (in case of 3D modeling)
- Effective resolution is improved :

NEMO - 0.25 deg



ROMS - 0.25 deg



Upwind biased schemes :

- Upwind biased schemes:

Ex: $$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0$$

First order

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + a\underbrace{\frac{u_i^n - u_{i-1}^n}{\Delta x}}_{u_x^-} = 0 \quad \text{for} \quad a > 0$$

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + a\underbrace{\frac{u_{i+1}^n - u_i^n}{\Delta x}}_{u_x^+} = 0 \quad \text{for} \quad a < 0$$

2nd order

$$u_x^- = \frac{3u_i^n - 4u_{i-1}^n + u_{i-2}^n}{2\Delta x} \qquad u_x^+ = \frac{-u_{i+2}^n + 4u_{i+1}^n - 3u_i^n}{2\Delta x}$$

3nd order

$$u_x^- = \frac{2u_{i+1} + 3u_i - 6u_{i-1} + u_{i-2}}{6\Delta x} \quad u_x^+ = \frac{-u_{i+2} + 6u_{i+1} - 3u_i - 2u_{i-1}}{6\Delta x}$$

Available schemes in CROCO

- See https://croco-ocean.gitlabpages.inria.fr/croco_doc/model/model.numerics.advec.html
-

Stability

- Courant-Friedrichs-Levy (CFL) stability criterion :

$$C = \delta t \sum_{i=1}^{n} \frac{u_i}{\delta x_i} \leq C_{max}$$

- Approximate limitations on time-step due to propagation pro cesses in the ocean :

Process	Speed	Formula for maximum Δt	Maximum Δt using Δx= 20 km	Maximum Δt using Δx= 200 km
Sound waves, c_s	1500 m s^{-1}	$\Delta x/c_s$	15 sec	2 min
External waves, $\sqrt{gH}$	200 m s^{-1}	$\Delta x/\sqrt{gH}$	2 min	15 min
Internal waves, NH	3 m s^{-1}	$\Delta x/NH$	2 hour	18 hour
Jets, U	2 m s^{-1}	$\Delta x/U$	3 hour	1 day
Interior flow, U	0.1 m s^{-1}	$\Delta x/U$	2 days	3 weeks

Split-explicit method:

- Since the limitation on time-step due to external gravity waves is due only to the depth integrated equations we can try split out that part of the system and integrate it with a shorter time step than the rest of the model.

- We first obtain an approximation for the barotropic momentum equations:

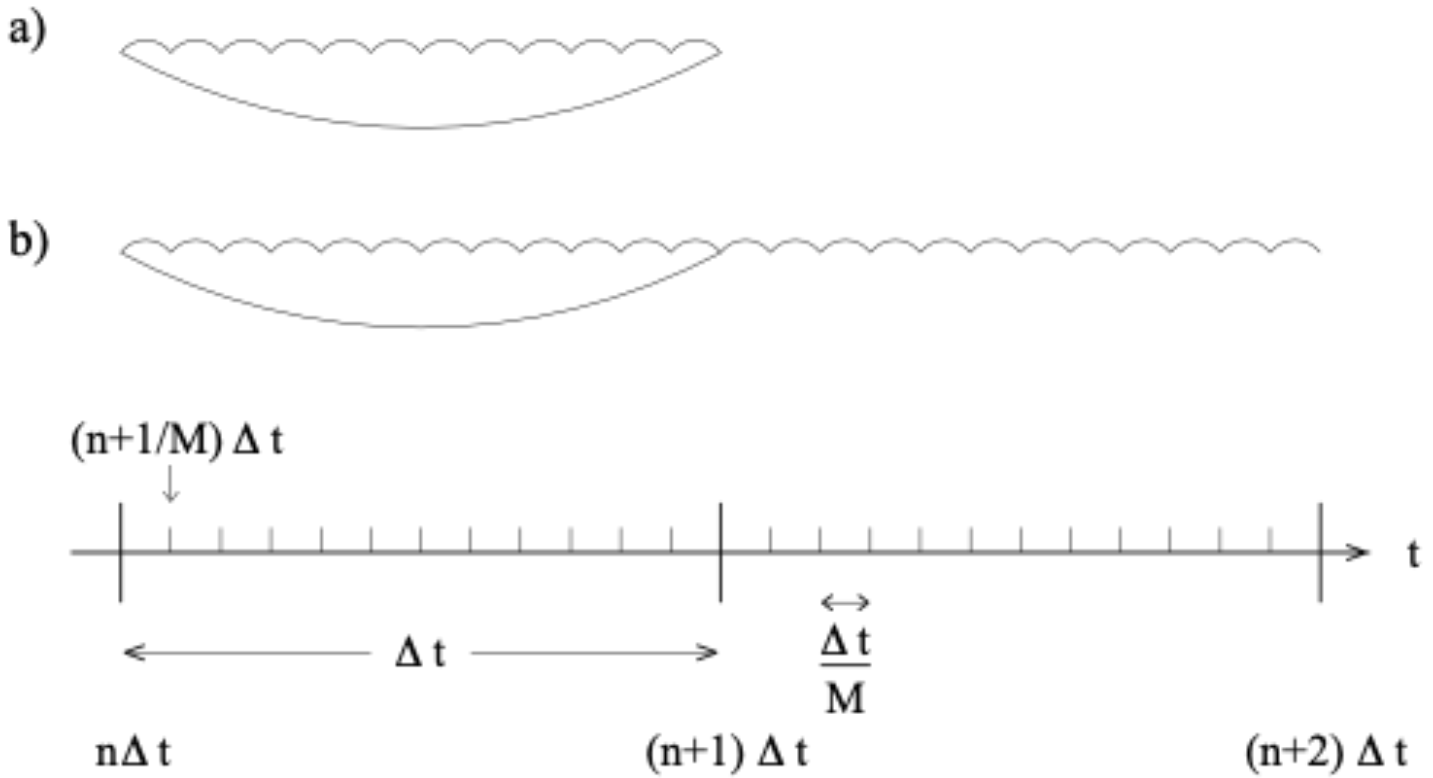
$$\partial_t < \vec{v}_h > + g\nabla\eta = < \vec{G}_h >$$

where < Gh > is the depth average of all the terms in the full momentum equations.

- We then integrate this equation with the free-surface equation forward using a short time step, $\Delta t/M$, where Δ is the regular time-step of the full model.

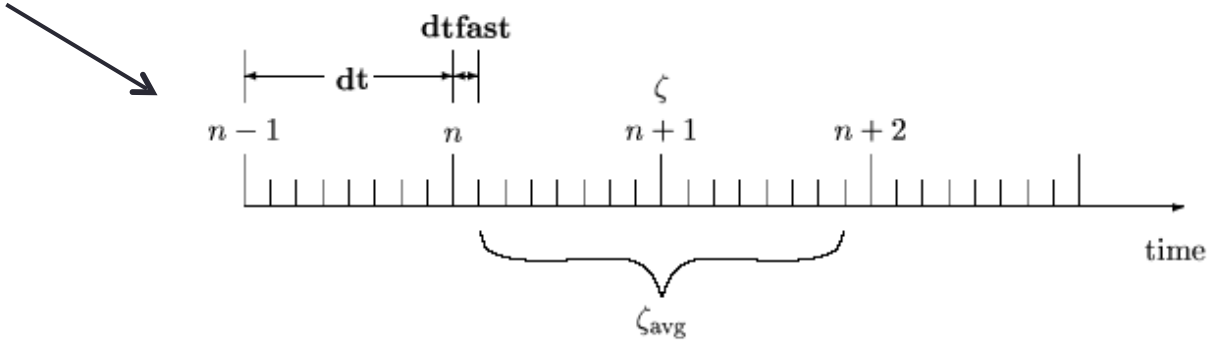
Split-explicit method:

- We then integrate this equation with the free-surface equation forward using a short time step, $\Delta t/M$, where Δt is the regular time-step of the full model.



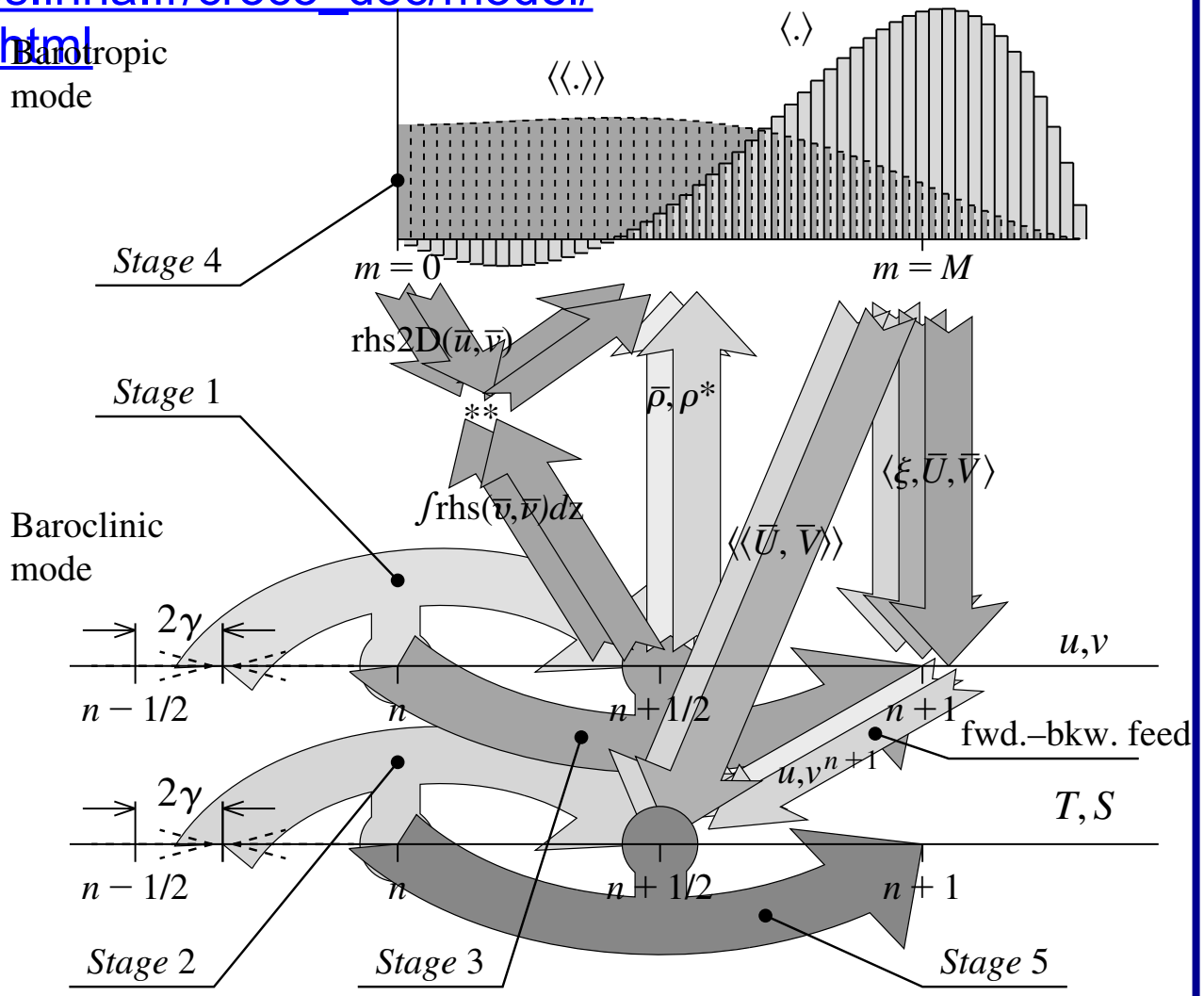
Time Stepping in ROMS

- **ROMS resolve free surface using the time-splitting method :**
  - Direct integration of the barotropic equations
  - Getting the free surface is straight forward
  - Good parallelization performances
  - BUT difficulties to separate fast and slow modes : possible instabilities. To avoid that:
    - time averaging over the barotropic subcycle



Time Stepping in ROMS

https://croco-ocean.gitlabpages.inria.fr/croco_doc/model/model.numerics.timestepping.html



LeapFrog – third-order Adams-Moulton (LF–AM3) predictor-corrector step for the baroclinic (3D) mode with mode coupling during the corrector stage.

Time Stepping in ROMS

- Shchepetkin, A., and J. McWilliams, 2008: Computational kernel algorithms for fine- scale, multiprocess, longtime oceanic simulations. Handb. Nu- mer. Anal., 14, 121–183, doi:10.1016/S1570-8659(08)01202-0.

http://jgula.fr/ModNum/ShchepetkinMcWilliams08.pdf

Time Stepping in ROMS

- https://croco-ocean.gitlabpages.inria.fr/croco_doc/model/model.numerics.timestepping.html#stability-constraints

  - **Barotropic mode** (note that considering an Arakawa C-grid divides the theoretical stability limit by a factor of 2)

$$\Delta t \sqrt{gH \left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} \right)} \leq 0.89$$

- Barotropic time-step is dt/NDTFAST (See croco.in)

```
time_stepping: NTIMES   dt[sec]  NDTFAST  NINFO
                16320     3600       60      1
```

Time Stepping in ROMS

- Baroclinic time-step is dt (See croco.in), it is limited by:

  - **3D advection**

$$\alpha^{x}_{\mathrm{adv}} + \alpha^{y}_{\mathrm{adv}} + \beta\alpha^{z}_{\mathrm{adv}} \leq \alpha^{\star}_{\mathrm{horiz}}$$

where $\alpha^{x}_{\mathrm{adv}}$, $\alpha^{y}_{\mathrm{adv}}$, and $\alpha^{z}_{\mathrm{adv}}$ are the Courant numbers in each direction and $\beta = \alpha^{\star}_{\mathrm{horiz}}/\alpha^{\star}_{\mathrm{vert}}$ a coefficient arising from the fact that different advection schemes with different stability criteria may be used in the horizontal and vertical directions. Typical CFL values for $\alpha^{\star}_{\mathrm{horiz}}$ and $\alpha^{\star}_{\mathrm{vert}}$ with Croco time-stepping algorithm are

Advection scheme	Max Courant number ($\alpha^{\star}$)
C2	1.587
UP3	0.871
SPLINES	0.916
C4	1.15
UP5	0.89
C6	1.00

  - **Internal waves**

$$\Delta t c_1 \sqrt{\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}} \leq 0.843686$$

where c_1 the phase speed associated with the first (fastest) baroclinic mode.

  - **Coriolis**

$$f\Delta t \leq 1.58$$

Activity 3

- See https://www.jgula.fr/ModNum/Activity3.html

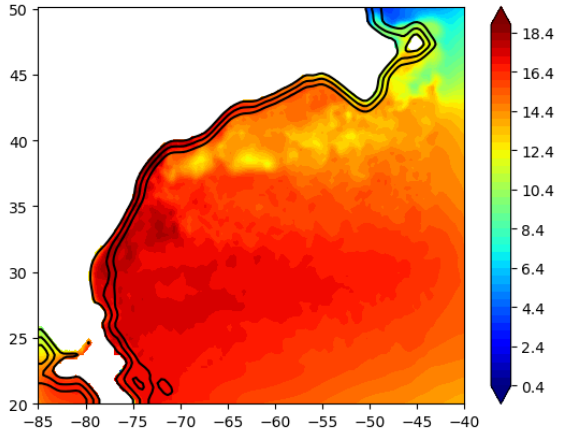
For Activity : Computing z and vertical interpolations

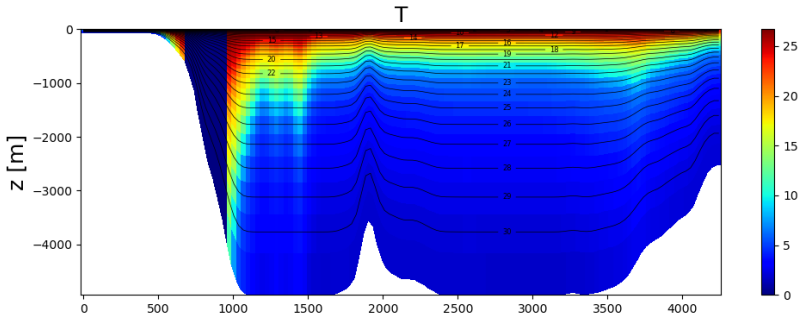
- See example in :

https://github.com/Mesharou/mesharou.github.io/blob/master/ModNum/example_croco.ipynb

```
# some tools to interpolate croco outputs vertically

import tools as to
```

```
###############################################################################
#Load variables and parameters
###############################################################################

nc = Dataset(ncfile, 'r')

temp3d=np.array(nc.variables['temp'][-1,:,:,:])


#Load some parameters

zeta=nc.variables['zeta'][-1,:,:]
topo=nc.variables['h'][:]
pm=nc.variables['pm'][:]
pn=nc.variables['pn'][:]

hc = nc.hc
Cs_r = nc.Cs_r
Cs_w = nc.Cs_w

#close netcdf file
nc.close()

###############################################################################
#Compute vertical coordinates
###############################################################################

(z_r,z_w) =  to.zlevs(topo,zeta, hc, Cs_r, Cs_w)

###############################################################################
#Interpolate a variable on a given depth
###############################################################################

t400 = to.vinterp(temp3d,z_r,-500,topo=topo,cubic=1)
```

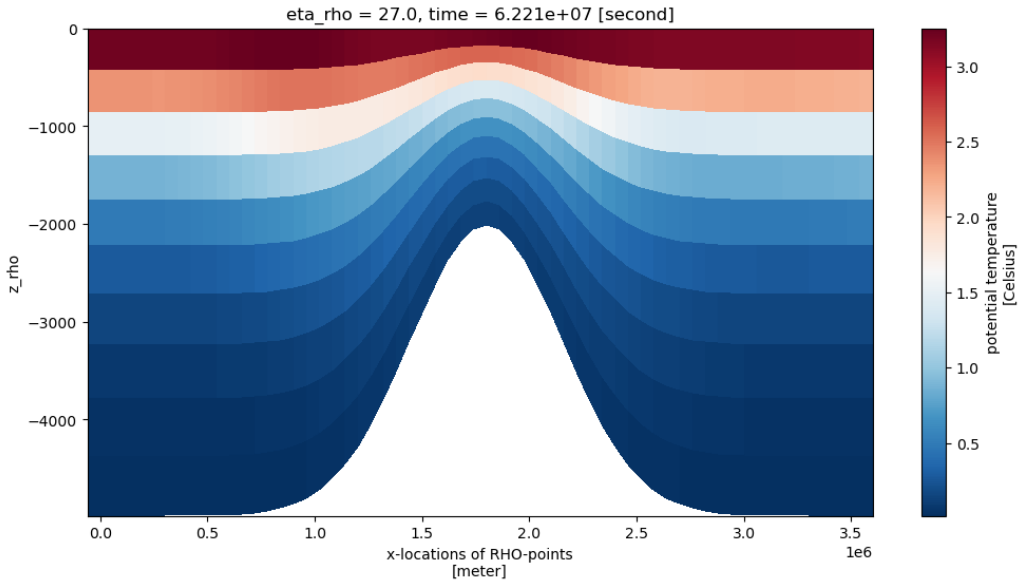




For Activity : Computing z and vertical interpolations

- An another using xarray in

https://github.com/Mesharou/mesharou.github.io/blob/master/ModNum/example_croco_xarray.ipynb



Example: 1D convection-diffusion equation

$$\begin{cases} \frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = d\frac{\partial^2 u}{\partial x^2} & \text{in } (0,X)\times(0,T) \\ u(0) = u(1) = 0, \quad u|_{t=0} = u_0 \end{cases}$$

$$f(t,u(t)) = -v\frac{\partial u}{\partial x} + d\frac{\partial^2 u}{\partial x^2}$$

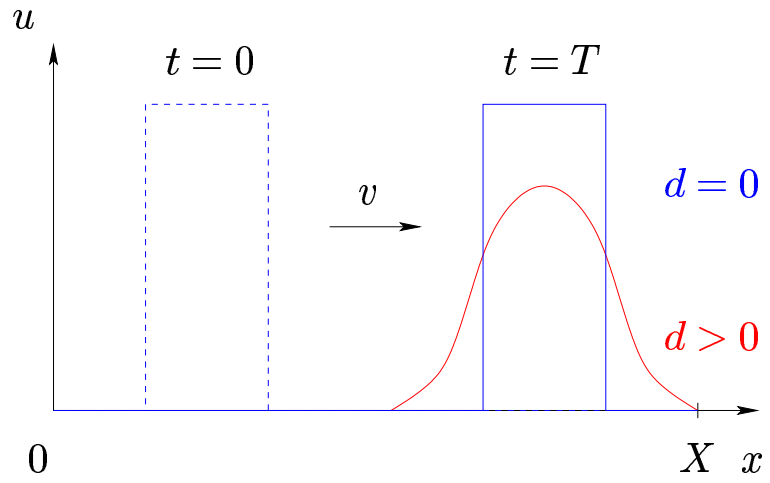
Lagrangian representation $\frac{du(x(t),t)}{dt} = d\frac{\partial^2 u}{\partial x^2}$ pure diffusion equation

where $\frac{d}{dt}$ is the substantial derivative along the characteristic lines $\frac{dx(t)}{dt} = v$

Initial profile is convected at speed v and smeared by diffusion if $d > 0$

$$d = 0 \quad \Rightarrow \quad \frac{du(x(t),t)}{dt} = 0$$

For the pure convection equation u is constant along the characteristics



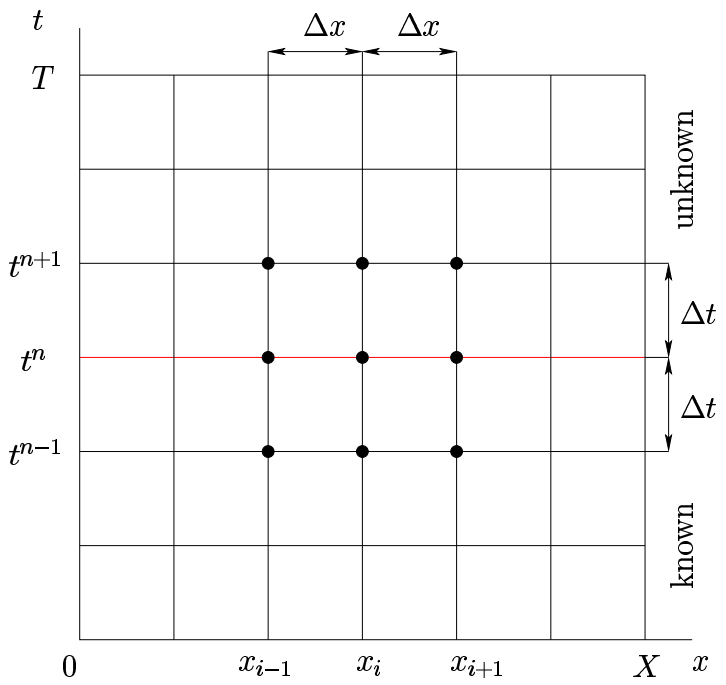
Example: 1D convection-diffusion equation

Uniform space-time mesh

$$x_i = i\Delta x, \quad \Delta x = \frac{X}{N}, \quad i = 0, \ldots, N$$

$$t^n = n\Delta t, \quad \Delta t = \frac{T}{M}, \quad n = 0, \ldots, M$$

$$u^n \longrightarrow u^{n+1}, \quad u_i^0 = u_0(x_i)$$



Fully discretized equation $\qquad 0 \le \theta \le 1$

$$u_i^{n+1} = u_i^n + [\theta f_h^{n+1} + (1-\theta) f_h^n]\Delta t$$

Central difference / lumped-mass FEM

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \theta \left[-v \frac{u_{i+1}^{n+1} - u_{i-1}^{n+1}}{2\Delta x} + d \frac{u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}}{(\Delta x)^2} \right]$$

$$+ \quad (1-\theta) \left[-v \frac{u_{i+1}^{n} - u_{i-1}^{n}}{2\Delta x} + d \frac{u_{i-1}^{n} - 2u_i^{n} + u_{i+1}^{n}}{(\Delta x)^2} \right]$$

a sequence of tridiagonal linear systems $\qquad i = 1, \ldots, N, \quad n = 0, \ldots M-1$

Example: 1D convection-diffusion equation

Standard $\theta-$scheme (two-level)

$$\frac{u_i^{n+1}-u_i^n}{\Delta t}+v\frac{u_{i+1}^{n+\theta}-u_{i-1}^{n+\theta}}{2\Delta x}=d\frac{u_{i-1}^{n+\theta}-2u_i^{n+\theta}+u_{i+1}^{n+\theta}}{(\Delta x)^2}$$

$$u_i^{n+\theta}=\theta u_i^{n+1}+(1-\theta)u_i^n, \qquad 0\le\theta\le 1$$

Forward Euler $(\theta=0)$ $\quad u_i^{n+1}=h(u_{i-1}^n,u_i^n,u_{i+1}^n)$

Backward Euler $(\theta=1)$ $\quad u_i^{n+1}=h(u_{i-1}^{n+1},u_i^n,u_{i+1}^{n+1})$

Crank-Nicolson $(\theta=\frac{1}{2})$ $\quad u_i^{n+1}=h(u_{i-1}^{n+1},u_{i-1}^n,u_i^n,u_{i+1}^n,u_{i+1}^{n+1})$

Leapfrog time-stepping $\quad u_i^{n+1}=u_i^{n-1}+2\Delta t f_h^n$

$$\frac{u_i^{n+1}-u_i^{n-1}}{2\Delta t}+v\frac{u_{i+1}^n-u_{i-1}^n}{2\Delta x}=d\frac{u_{i-1}^n-2u_i^n+u_{i+1}^n}{(\Delta x)^2}$$

(explicit, three-level) $\quad u_i^{n+1}=h(u_{i-1}^n,u_i^{n-1},u_i^n,u_{i+1}^n)$