Numerical Simulation and Scientific Computing II

Lecture 3:
ODE Methods, Finite Volume Method

Josef Weinbub, Paul Manstetten, Heinz Pettermann, Asur Vijaya Kumar, Pavan Kumar, Jesús Carrete Montana, Francesco Zonta, Kevin Sturm

Institute for Microelectronics
TU Wien

nssc@iue.tuwien.ac.at

SS 2023

Questions

ODE Methods

- Q1: What is the order of the forward/backward Euler method?
- Q2: Are the Runge-Kutta methods single-step or multistep methods?
- Q3: How can methods for first order ODEs be applied to higher order ODEs?

Finite volume method

- Q4: Why are finite volume schemes ‘conservative’?
- Q5: What are advantages of the finite volume method over the finite difference method?

Outline

- ODE Methods
  - Example: first order ODEs
  - Single-step methods
  - Multistep methods
  - Embedded methods
  - Higher order ODEs/systems of ODEs

- Finite volume method
  - Concept
  - Domain decomposition
  - Boundary/interface conditions

First order ODE

- First order ODE + initial value

$$\frac{dy}{dt} = f\big(t, y(t)\big)$$
$$y(t_a) = t_a$$

- Simple example (with solution)

$$y' = -y, \qquad y = e^{-t+C}$$
$$y(t_a = 0) = e^1, \qquad y = e^{-t+1}$$

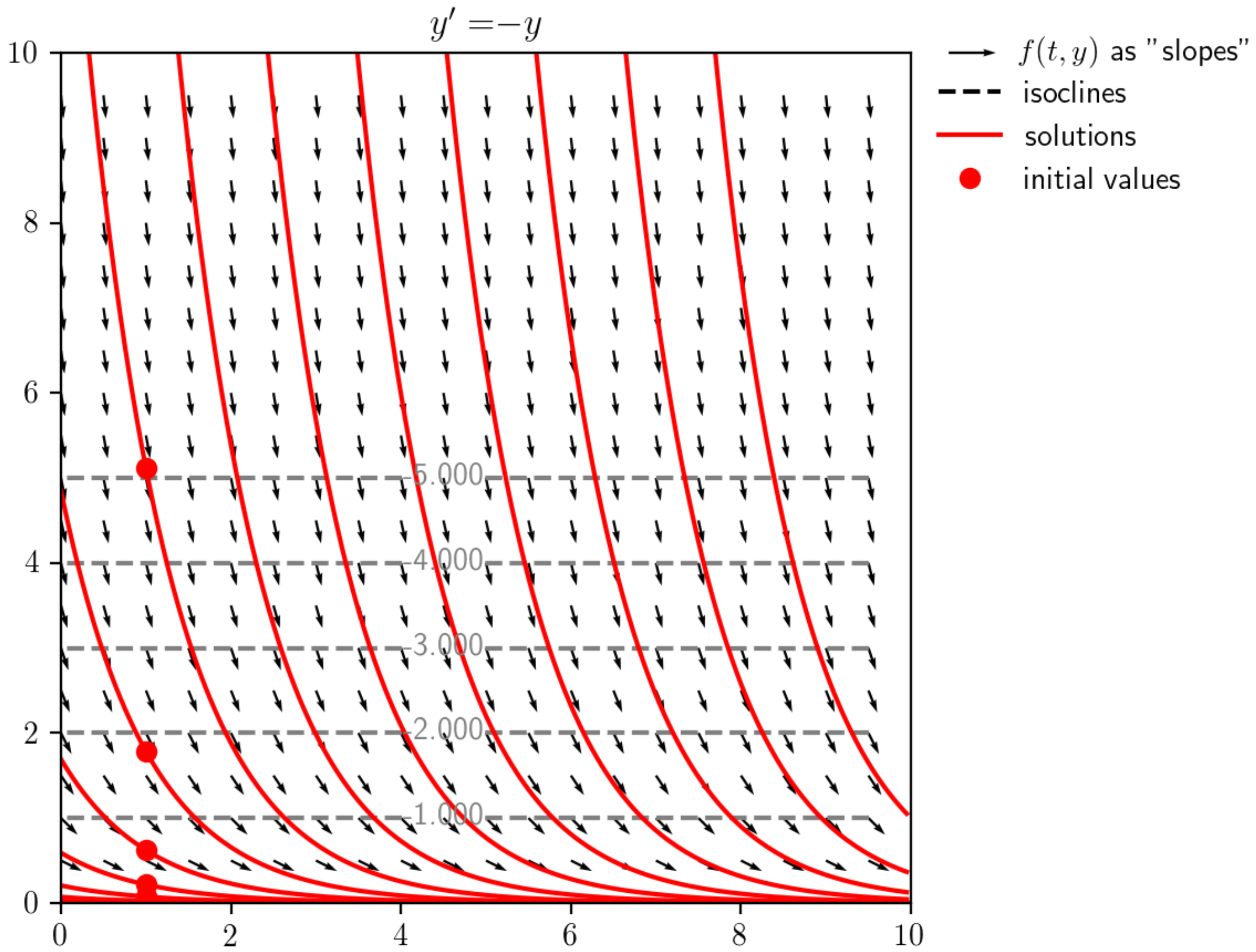
- Methods for numerical solutions from t_a to t_b (or only at t_b)
  - Large collection of methods for problems of this type
  - For “not so nice” f, numerical methods are the only choice
  - “Not so nice” f stem from arbitrary dependencies on t and y

Examples

- Simple example

$$y' = -y, \qquad y = e^{-t+C}$$

- Visualization of solutions in the t, y plane

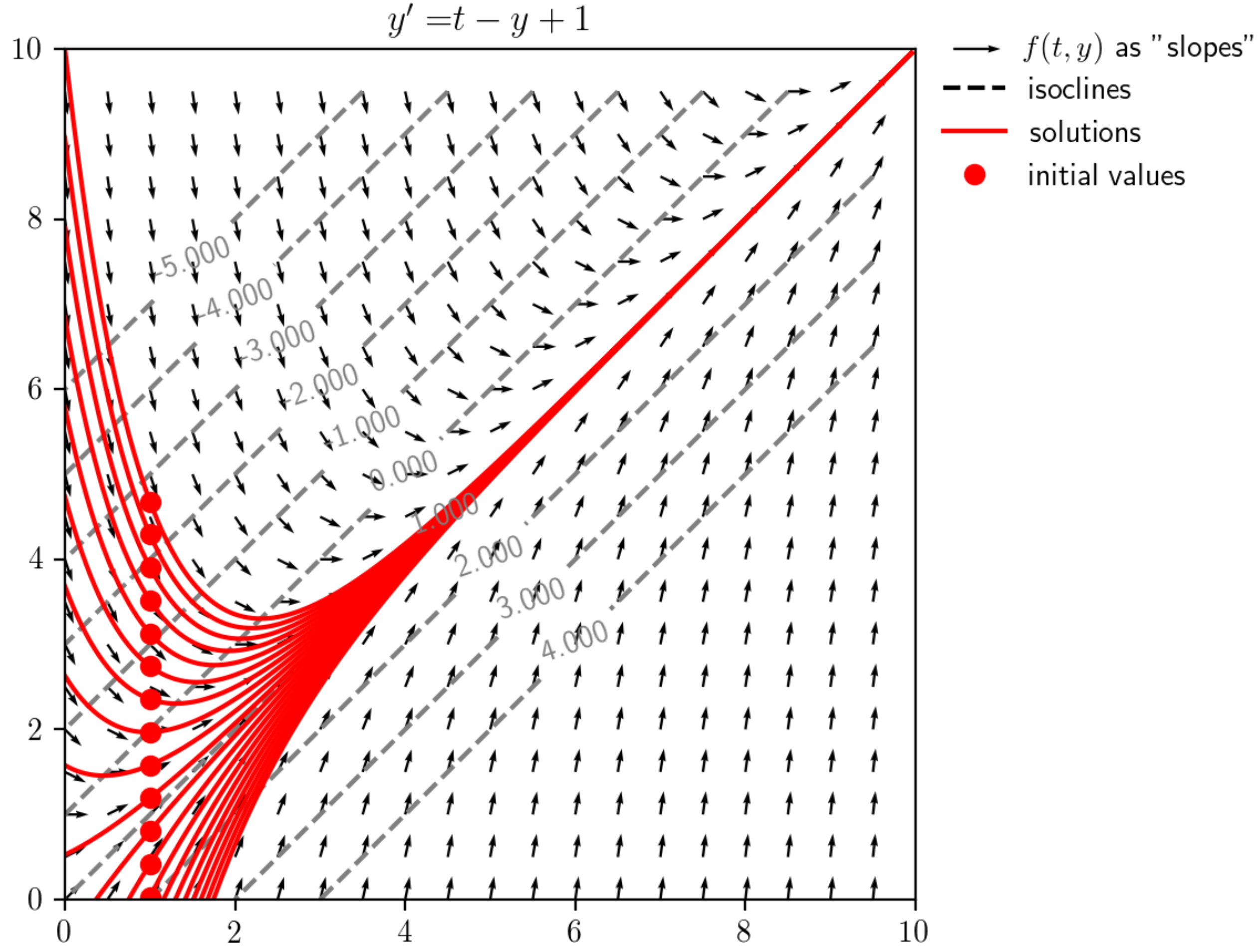


Examples

- Simple example

$$y' = t - y + 1, \qquad y = \frac{te^t + C}{e^t}$$

- Visualization of solutions in the t, y plane (isoclines)



Explicit Euler Method

- Given

$$y' = f\big(t, y(t)\big)$$

- Equidistant discretization in t domain from t_a to t_b

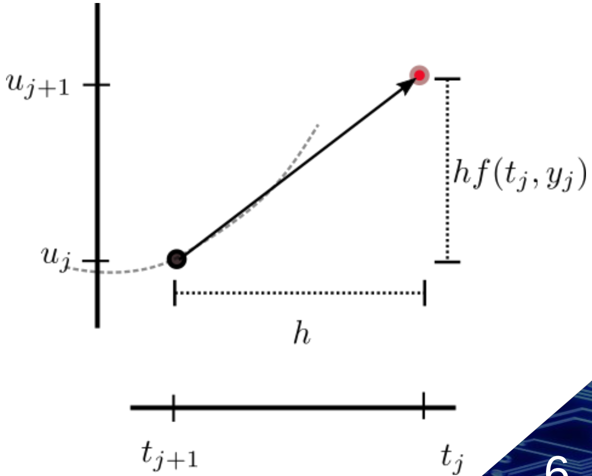
$$t_j = t_a + jh, \quad j = 0,1 \ldots, N, \quad h = \frac{t_b - t_a}{N}$$

- Forward Difference (first order approximation)

$$y'(t_{j-1}) = \frac{y(t_j) - y(t_{j-1})}{h} + O(h^2)$$
$$y(t_j) = y(t_{j-1}) + hy'(t_{j-1}) + O(h^2)$$
$$y(t_j) = y(t_{j-1}) + hf\left(t_{j-1}, y(t_{j-1})\right) + O(h^2)$$

- Explicit scheme

$$u_o = y_o$$
$$u_{j+1} = u_j + hf(t_j, u_j)$$

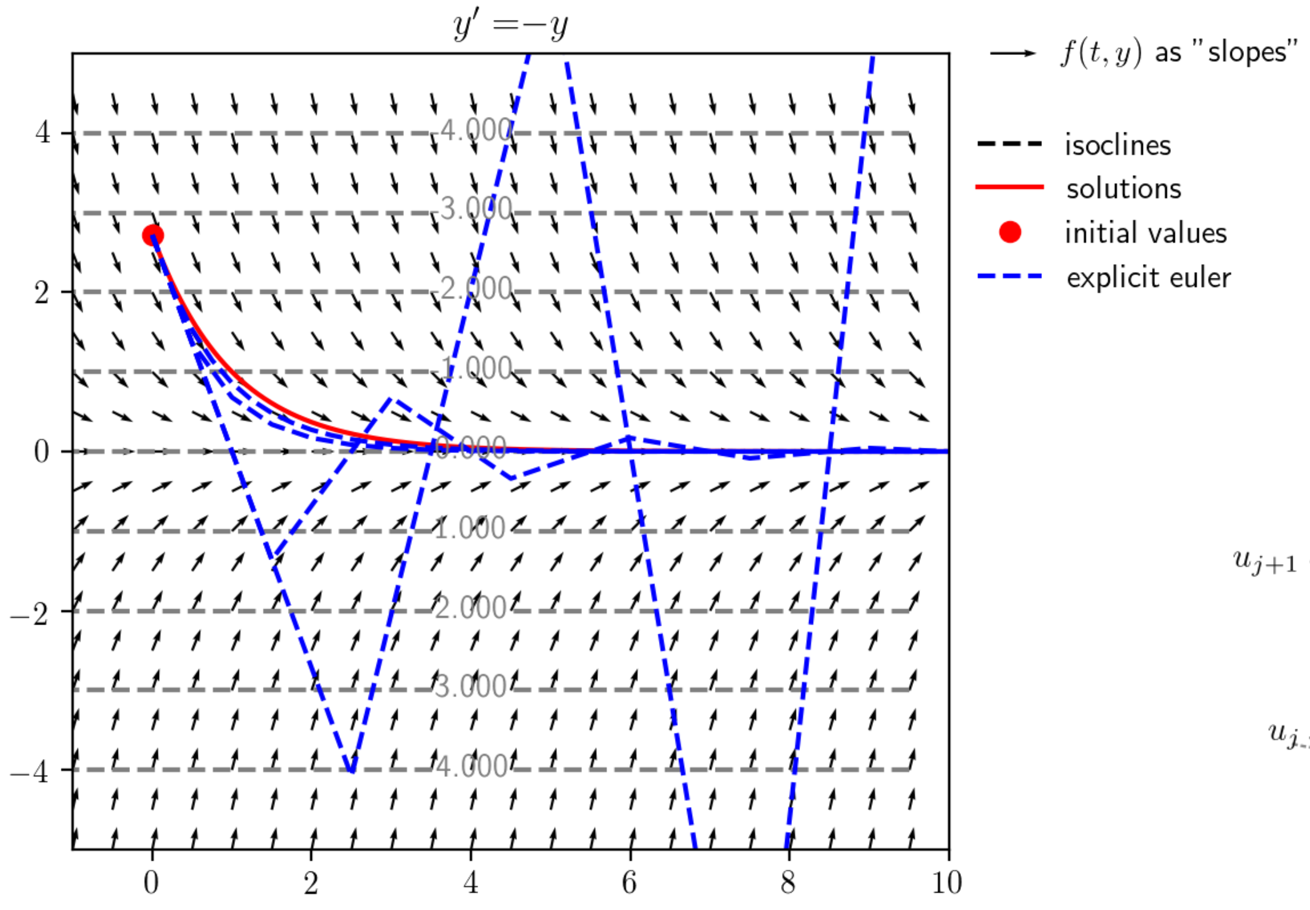


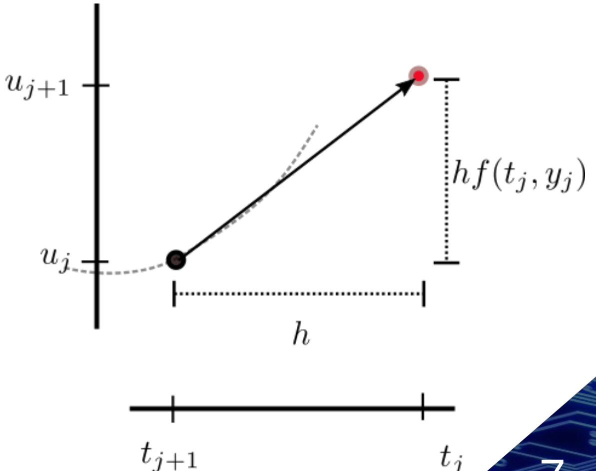
Explicit Euler Method

- Simple example

$$y' = -y, \qquad y = e^{-t+1}$$

- Visualization of approximations in the t, y plane



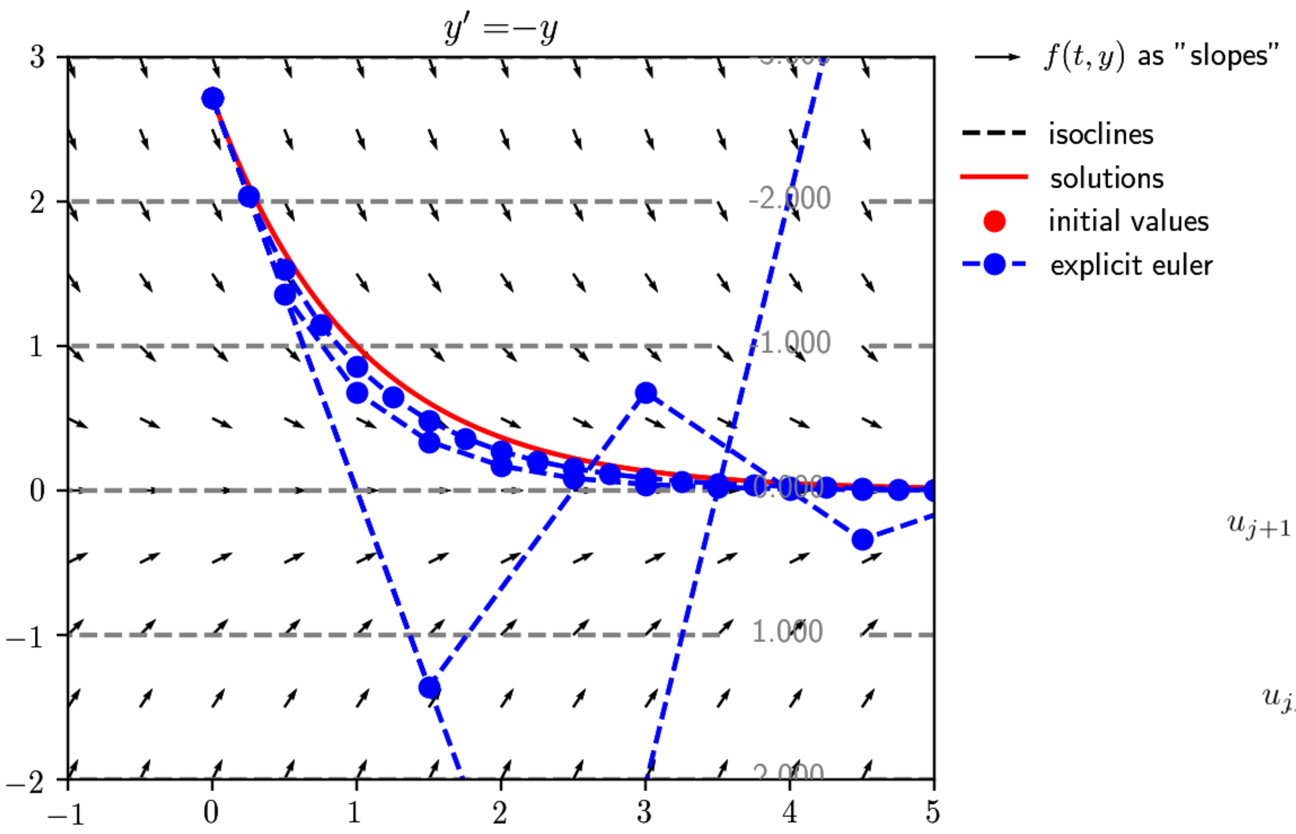


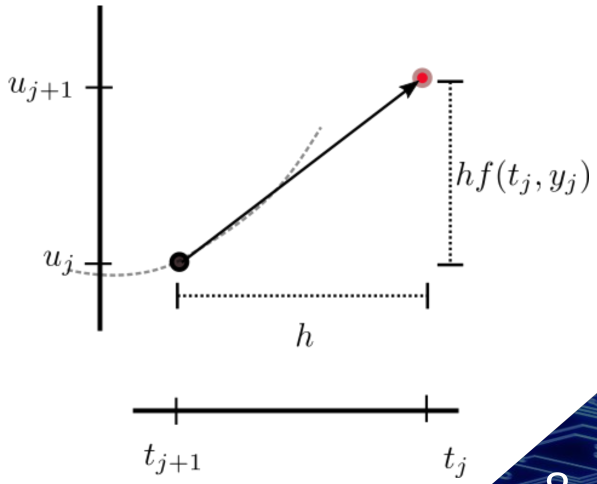
Explicit Euler Method

- Simple example

$$y' = -y, \qquad y = e^{-t+1}$$

- Visualization of approximations in the t, y plane



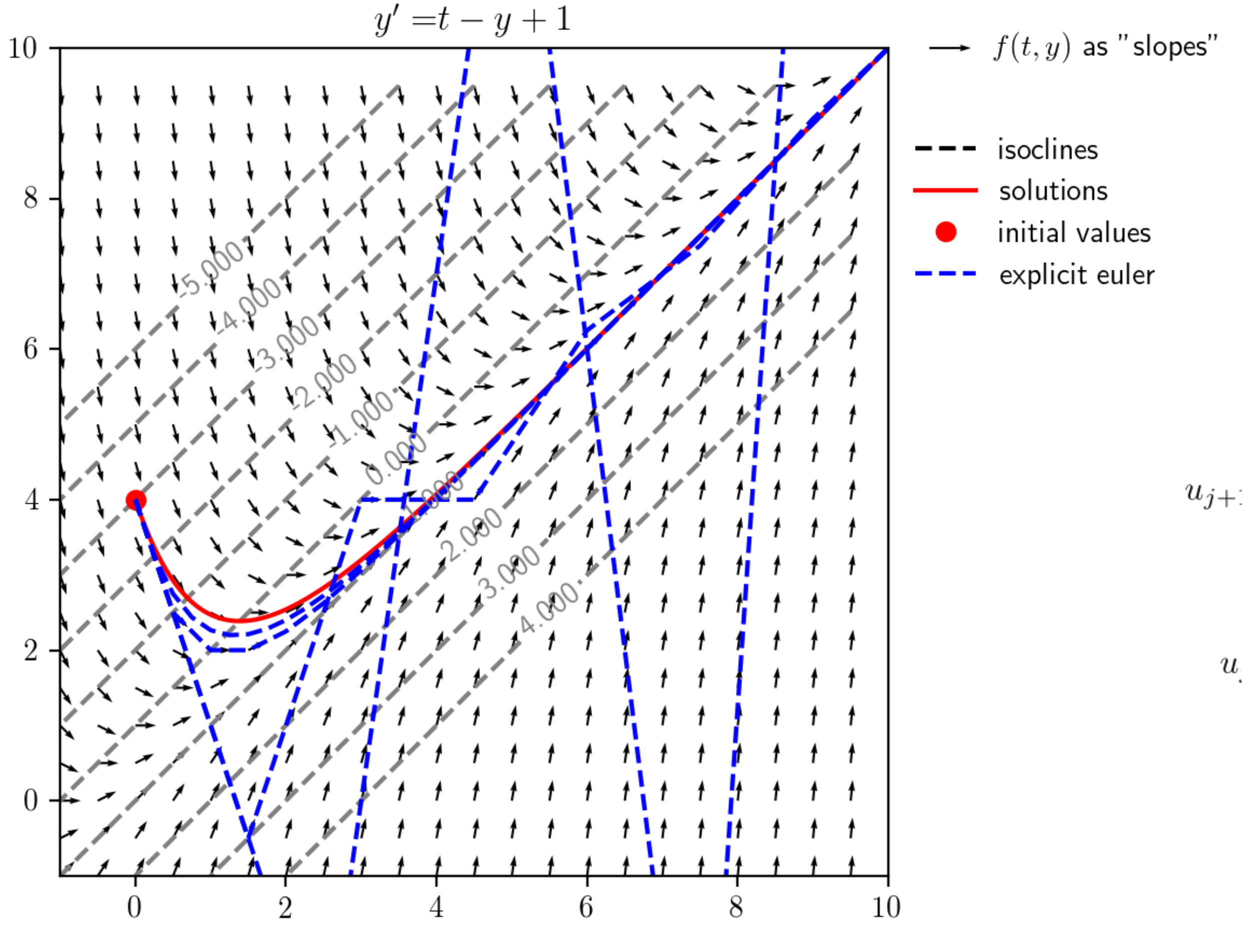


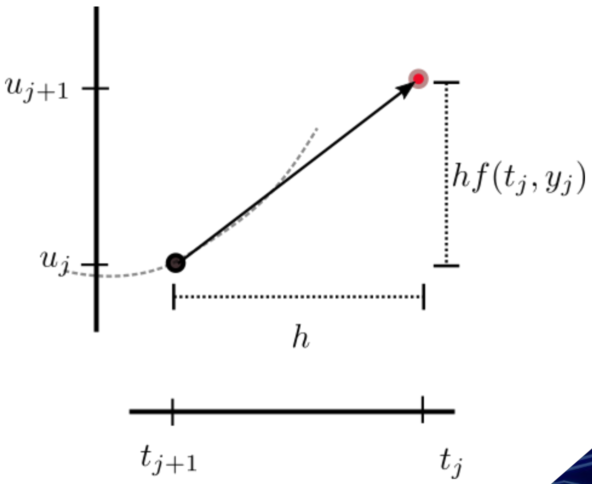
Explicit Euler Method

- Simple example

$$y' = t - y + 1, \qquad y = \frac{te^t + 4}{e^t}$$

- Visualization of approximations in the t, y plane





Implicit Euler Method

- Given

$$y' = f\big(t, y(t)\big)$$

- Equidistant discretization in t domain from t_a to t_b

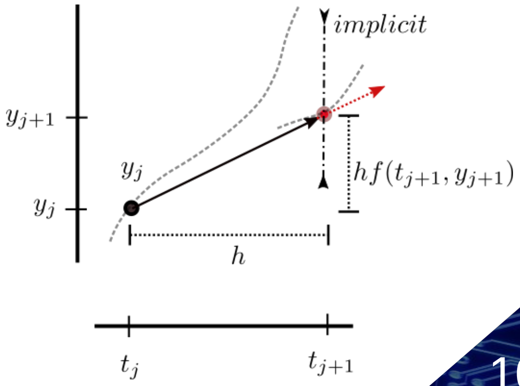
$$t_j = t_a + jh, \quad j = 0,1 \dots, N, \quad h = \frac{t_b - t_a}{N}$$

- Backward Difference (first order approximation)

$$y'(t_j) = \frac{y(t_j) - y(t_{j-1})}{h} + O(h^2)$$
$$y(t_j) = y(t_{j-1}) + hy'(t_j) + O(h^2)$$
$$y(t_j) = y(t_{j-1}) + hf\left(t_j, y(t_j)\right) + O(h^2)$$

- Implicit scheme (in general: solution of nonlinear equation)

$$u_o = y_o$$
$$u_{j+1} = u_j + hf(t_{j+1}, u_{j+1})$$

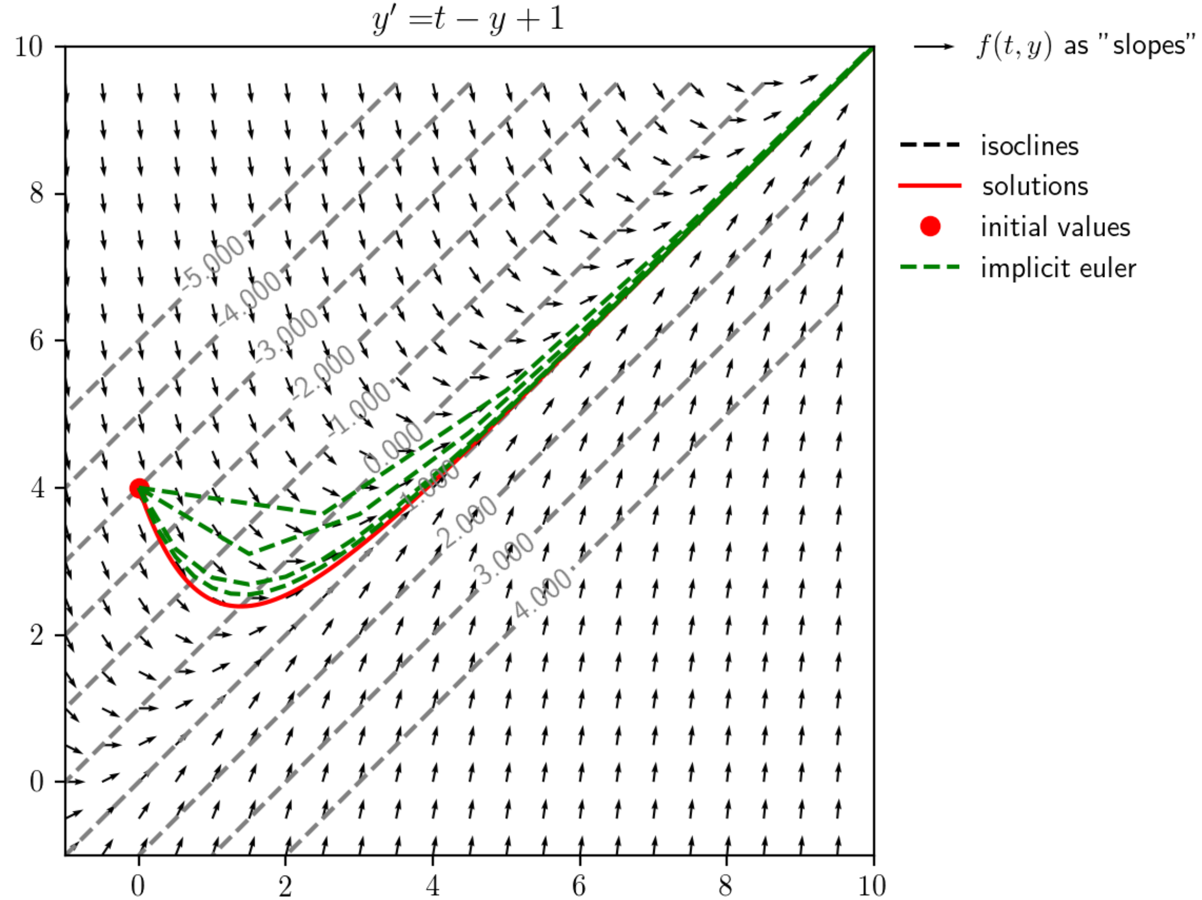


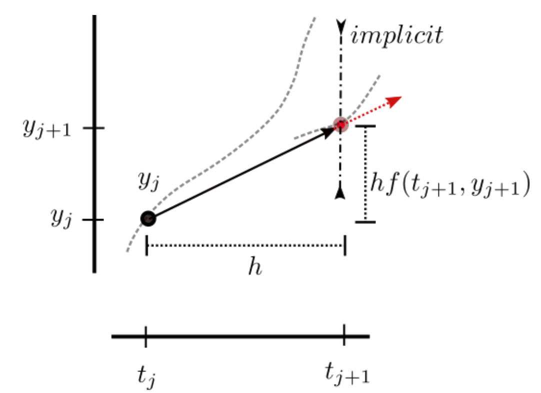
Implicit Euler Method

- Simple example

$$y' = t - y + 1, \qquad y = \frac{te^t + 4}{e^t}$$

- Visualization of approximations in the t, y plane



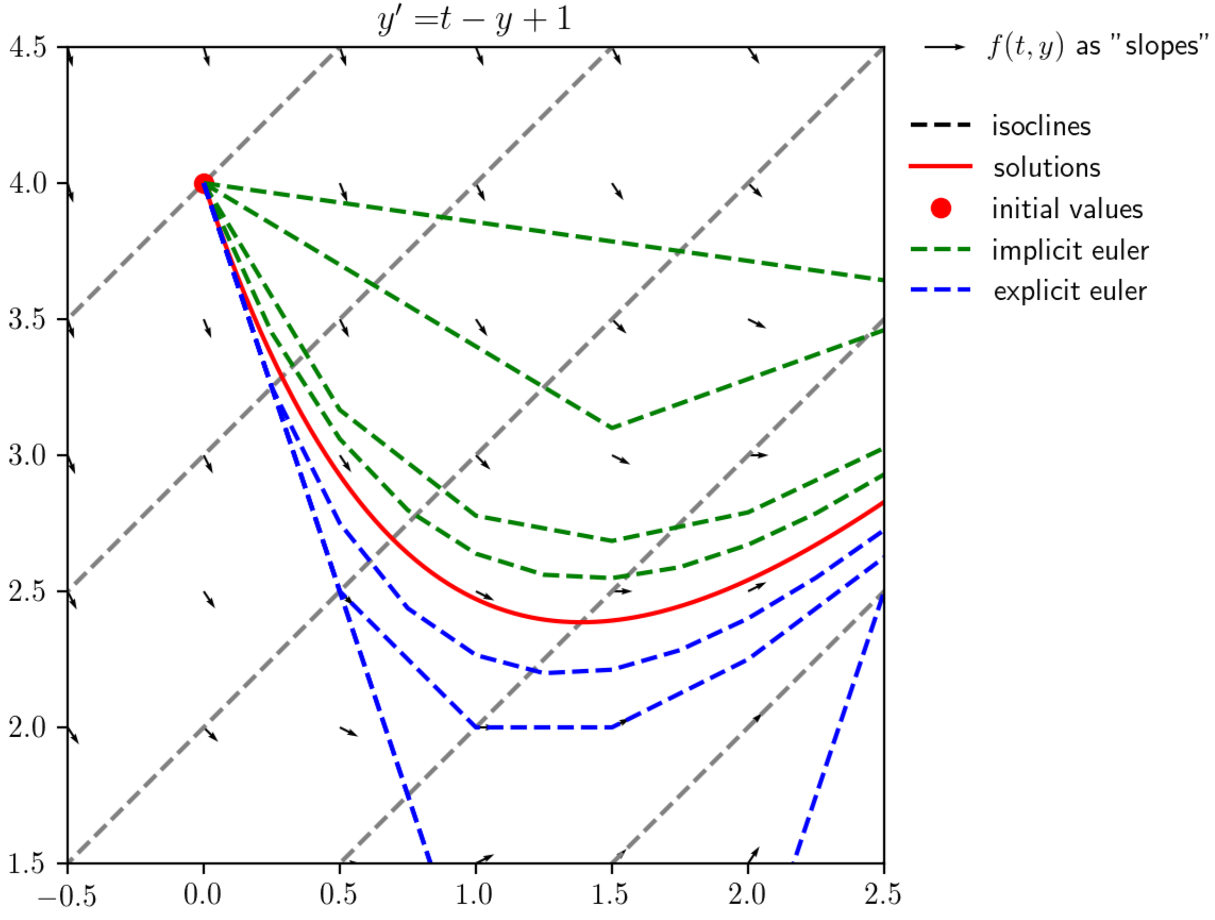


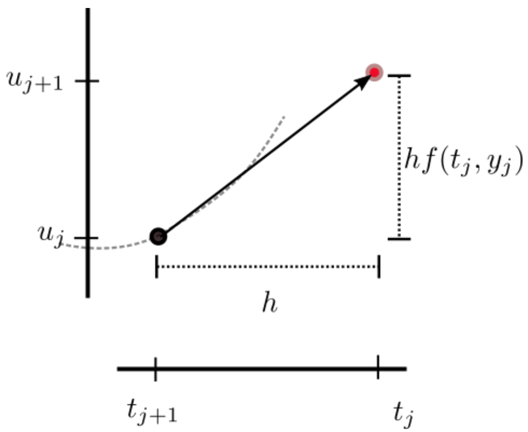
Euler Method

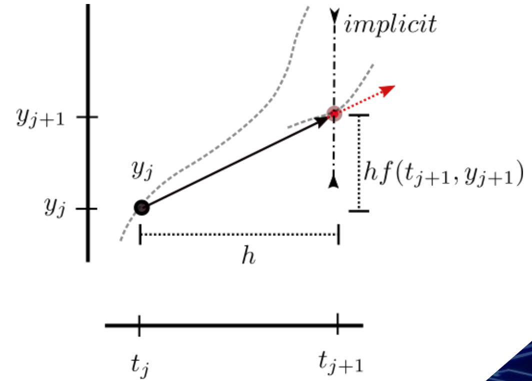
- Simple example

$$y' = t - y + 1, \qquad y = \frac{te^t + 4}{e^t}$$

- Comparison of implicit and explicit approximations







Euler Methods

- Forward Euler method
  - Explicit method
  - Demands small step sizes
- Backward Euler method
  - Implicit method
  - Stable also for large step sizes
- Both methods
  - Are single step methods
  - Produce local truncation error of $O(h^2)$
  - Produce a global error of $O(h)$

Implicit Trapezoidal Rule

- Given

$$y' = f\big(t, y(t)\big)$$

- Equidistant discretization in t domain from t_a to t_b

$$t_j = t_a + jh, \quad j = 0,1 \dots, N, \quad h = \frac{t_b - t_a}{N}$$

- Reformulation and integration (using Trapezoidal rule)

$$y(t_{j+1}) - y(t_j) = \int_{t_j}^{t_{j+1}} f\big(t, y(t)\big)\, dt$$

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

- Implicit scheme

$$u_o = y_o$$

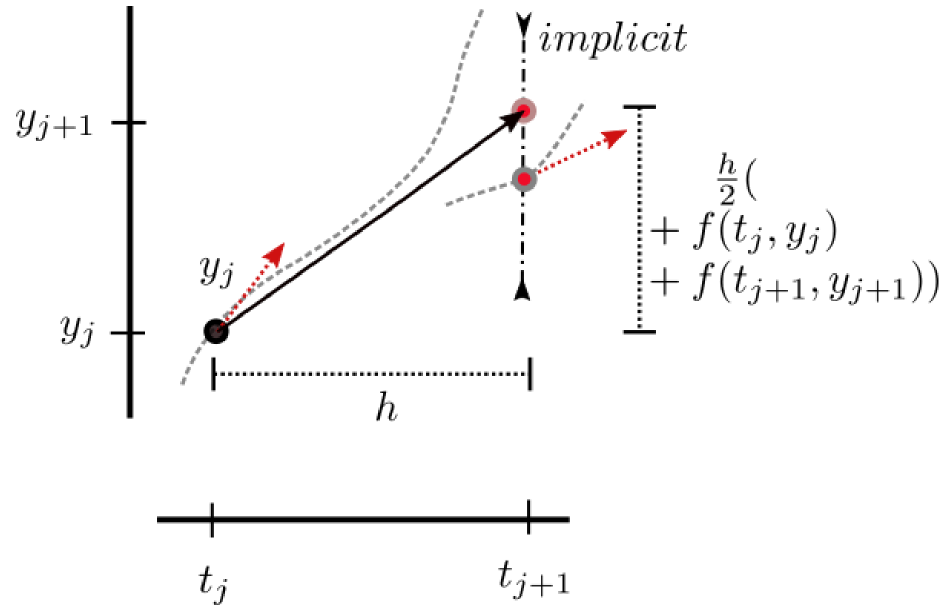
$$u_{j+1} = u_j + \frac{h}{2}\{f\big(t_j, u_j\big) + f\,(t_{j+1}, u_{j+1})\} + O(h^3)$$

Implicit Trapezoidal Rule

- Implicit scheme

$$u_o = y_o$$

$$u_{j+1} = u_j + \frac{h}{2}\{f(t_j, u_j) + f(t_{j+1}, u_{j+1})\} + O(h^3)$$



Explicit Trapezoidal Rule

- Given

$$y' = f\big(t, y(t)\big)$$

- Equidistant discretization in t domain from t_a to t_b

$$t_j = t_a + jh, \quad j = 0,1 \dots, N, \quad h = \frac{t_b - t_a}{N}$$

- Reformulation and integration (using Trapezoidal rule)

$$y(t_{j+1}) - y(t_j) = \int_{t_j}^{t_{j+1}} f\big(t, y(t)\big)\, dt$$

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

- Explicit scheme

$$u_o = y_o$$

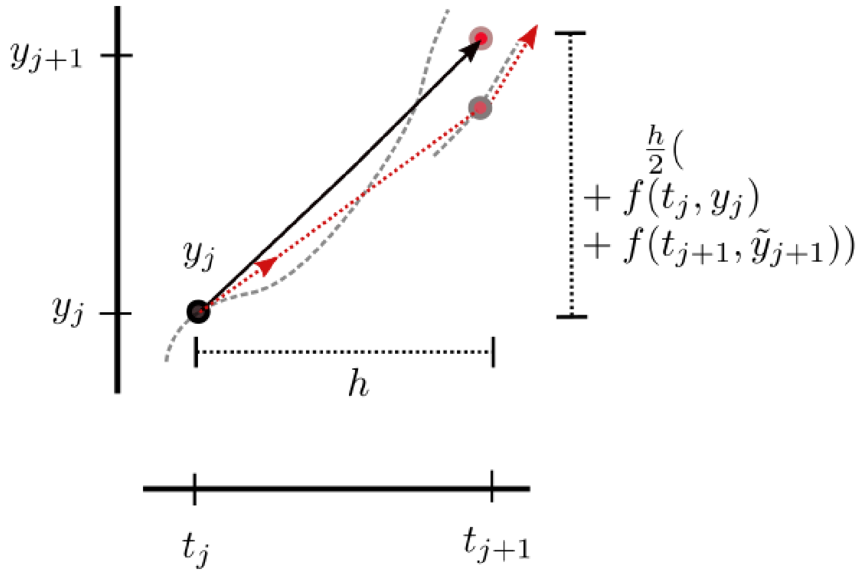
$$\tilde{u}_{j+1} = u_j + hf(t_j, u_j) + O(h^2)$$

$$u_{j+1} = u_j + \frac{h}{2}\{f(t_j, u_j) + f(t_{j+1}, \tilde{u}_{j+1})\} + O(h^3)$$

Explicit Trapezoidal Rule

- Explicit scheme

$$u_o = y_o$$
$$\tilde{u}_{j+1} = u_j + hf(t_j, u_j) + O(h^2)$$
$$u_{j+1} = u_j + \frac{h}{2}\{f(t_j, u_j) + f(t_{j+1}, \tilde{u}_{j+1})\} + O(h^3)$$

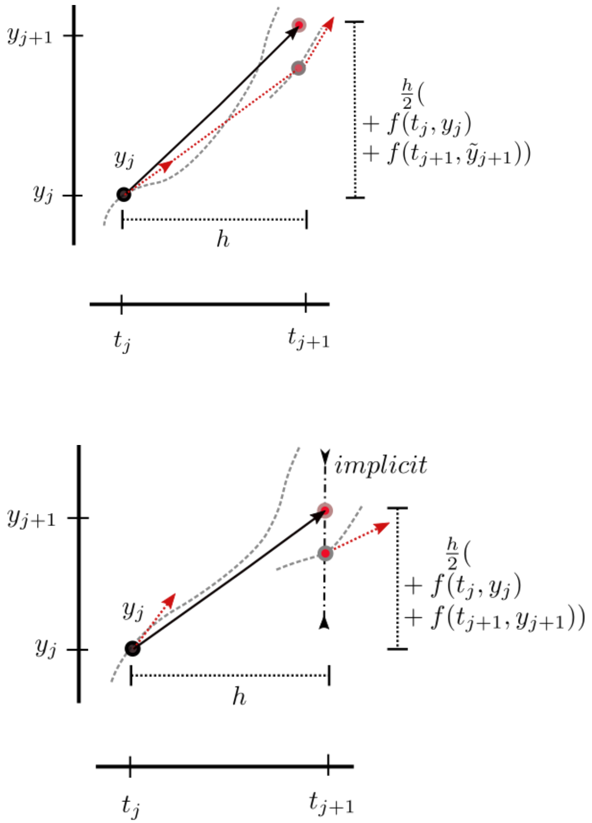


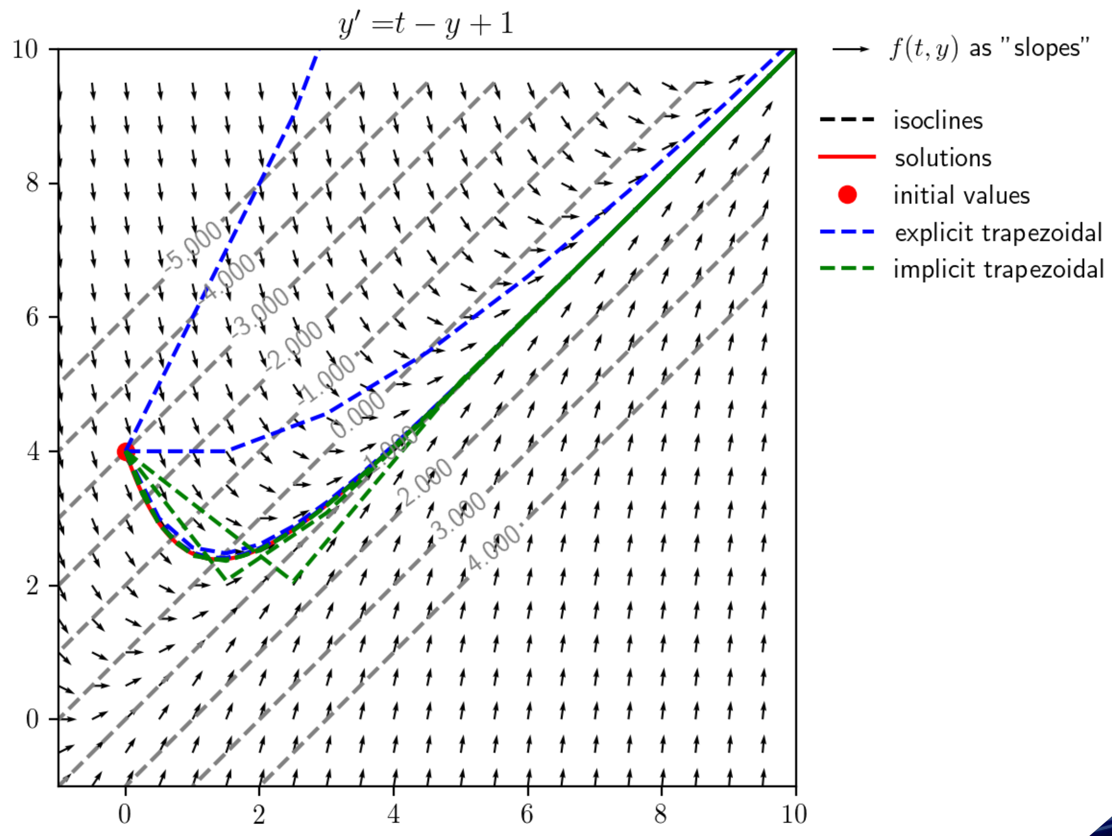
Trapezoidal Rule

- Simple example

$$y' = t - y + 1, \qquad y = \frac{te^t + 4}{e^t}$$

- Comparison of implicit and explicit approximations



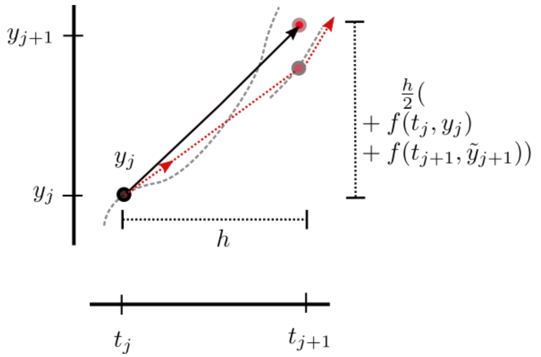


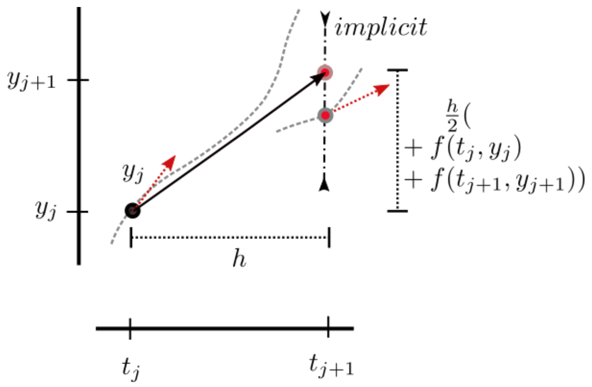
Trapezoidal Rule

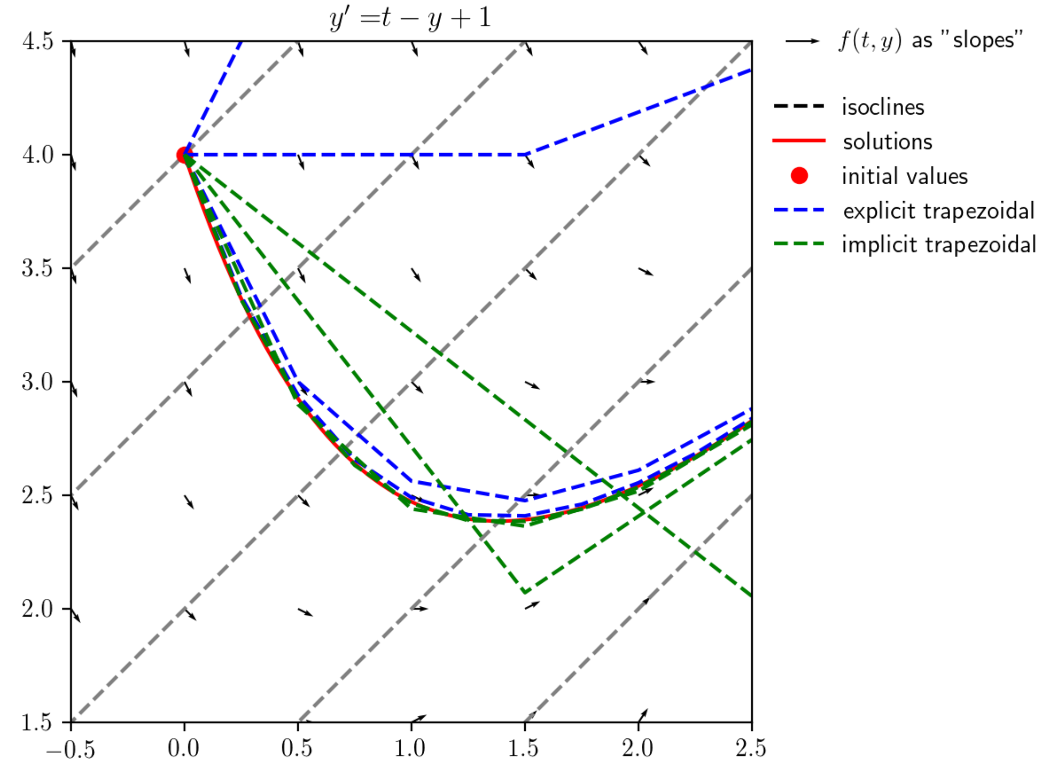
- Simple example

$$y' = t - y + 1, \qquad y = \frac{te^t + 4}{e^t}$$

- Comparison of implicit and explicit approximations







Trapezoidal Rule

- Implicit Trapezoidal Rule
  - Implicit method
  - Stable also for large steps
  - Suitable for stiff problems
- Explicit Trapezoidal Rule
  - Explicit method
  - Improved accuracy compared the forward Euler method
- Both methods
  - Are single step methods
  - Produce local truncation error of $O(h^3)$
  - Produce a global error of $O(h^2)$ and are therefore second order methods

Classic Runge-Kutta Method

- Given

$$y' = f\big(t, y(t)\big)$$

- Equidistant discretization in t domain from t_a to t_b

$$t_j = t_a + jh, \quad j = 0,1\,\ldots, N, \quad h = \frac{t_b - t_a}{N}$$

- Reformulation and integration (using Simpson's rule)

$$y(t_{j+1}) - y(t_j) = \int_{t_j}^{t_{j+1}} f\big(t, y(t)\big)\,dt$$

$$y(t_{j+1}) = y(t_j) + \frac{h}{6}\{f^{\,j} + 4f^{\,j+1/2} + f^{\,j+1}\}$$

$$y(t_{j+1}) = y(t_j) + \frac{h}{6}\{f^{\,j} + 2\hat{f}^{\,j+1/2} + 2\tilde{f}^{\,j+1/2} + f^{\,j+1}\}$$

- Idea: successively approximate f for increasing t

Classic Runge-Kutta Method

$$y(t_{j+1}) = y(t_j) + \frac{h}{6}\{f^j + 2\hat{f}^{j+1/2} + 2\tilde{f}^{j+1/2} + \bar{f}^{j+1}\}$$

- Forward Euler

$$\hat{f}^{j+1/2} = f\left(t_{j+1/2}, u_j + \frac{h}{2} f(t_{j,} u_j)\right)$$

- Backward Euler

$$\tilde{f}^{j+1/2} = f\left(t_{j+1/2}, u_j + \frac{h}{2}\hat{f}^{j+1/2}\right)$$

- Midpoint method

$$\bar{f}^{j+1} = f(t_{j+1}, u_j + h\tilde{f}^{j+1/2})$$

- Explicit scheme

$$u_o = y_o$$

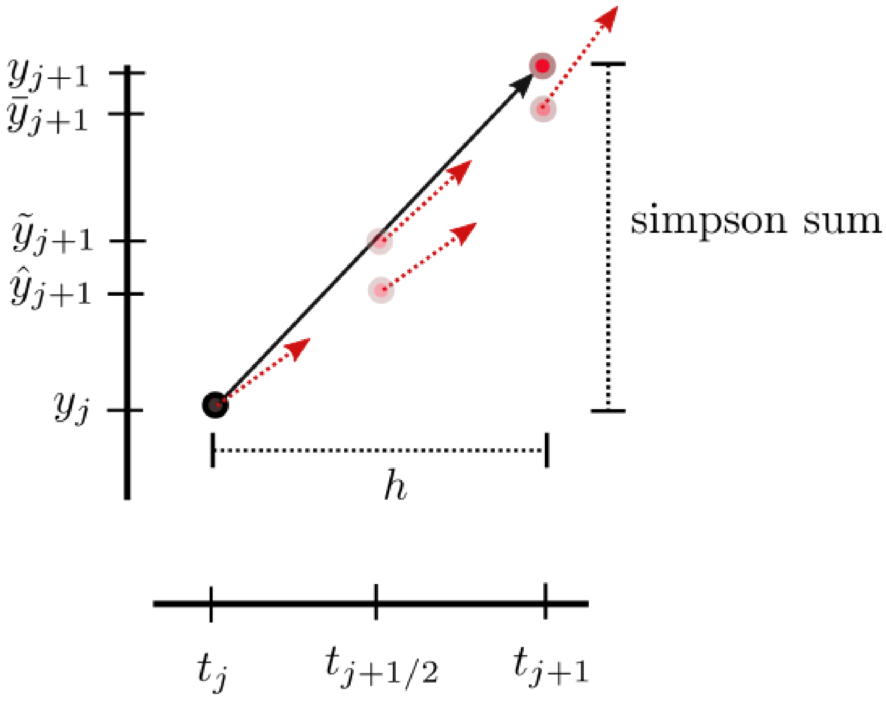
$$u_{j+1} = u_j + \frac{h}{6}\{f^j + 2\hat{f}^{j+1/2} + 2\tilde{f}^{j+1/2} + \bar{f}^{j+1}\}$$

Classic Runge-Kutta Method

- Explicit scheme

$$u_o = y_o$$

$$u_{j+1} = u_j + \frac{h}{6}\{f^j + 2\hat{f}^{j+1/2} + 2\tilde{f}^{j+1/2} + \bar{f}^{j+1}\}$$

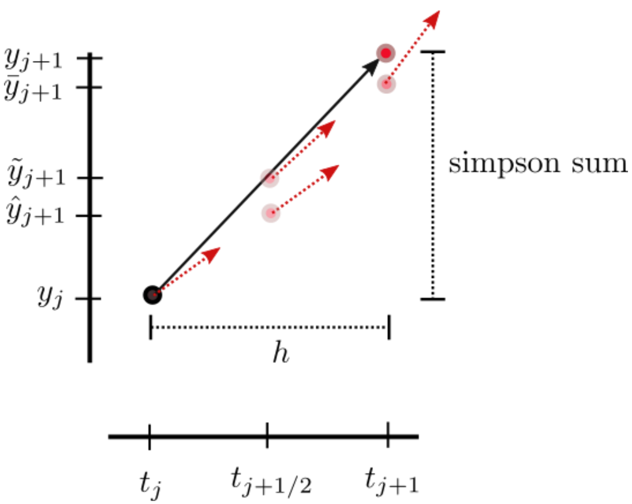


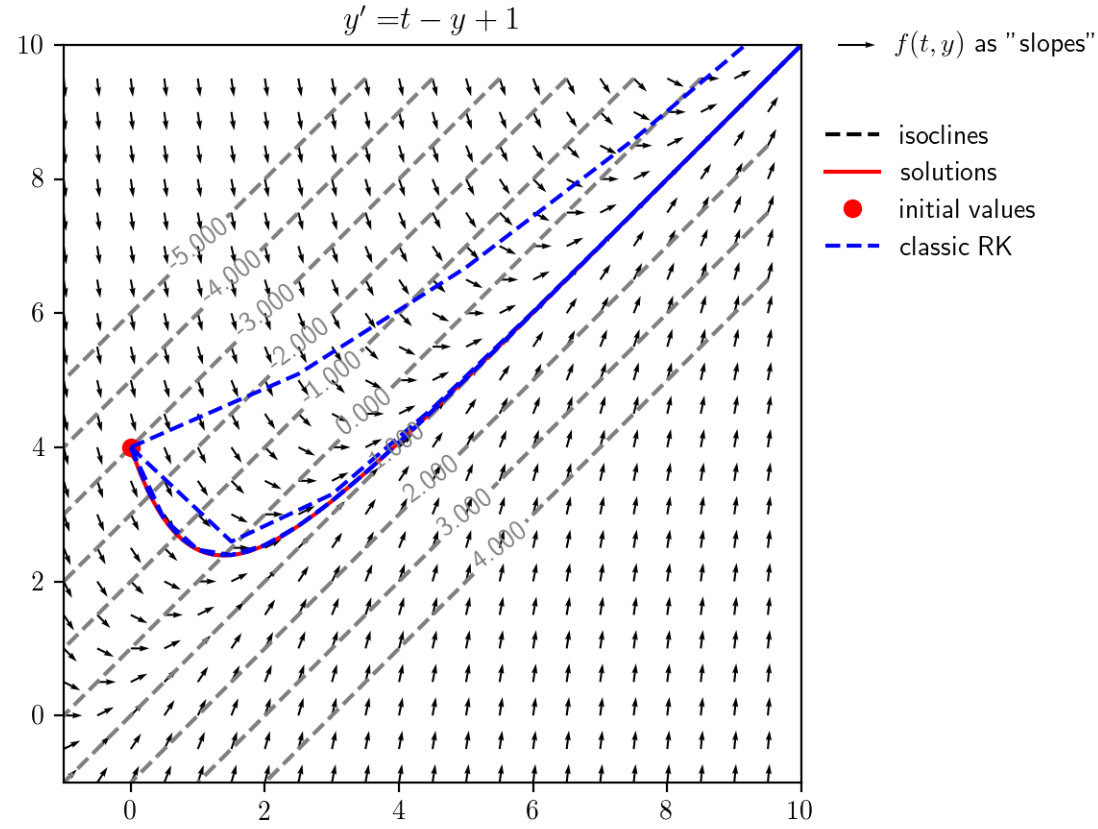
Classic Runge-Kutta Method

- Simple example

$$y' = t - y + 1, \qquad y = \frac{te^t + 4}{e^t}$$

- Approximation using classic Runge-Kutta method





Classic Runge-Kutta Method

- Simple example

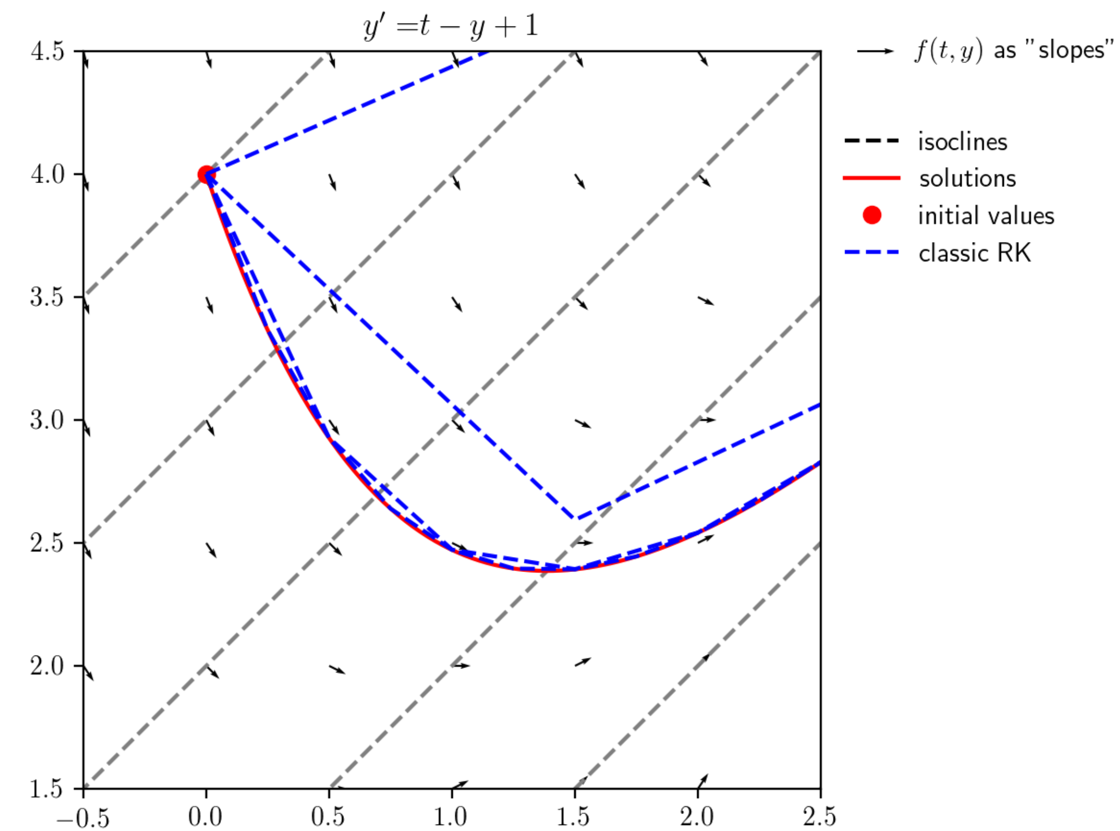
$$y' = t - y + 1, \qquad y = \frac{te^t + 4}{e^t}$$

- Approximation using the classic Runge-Kutta method

- Simple example

$$y' = t - y + 1, \qquad y = \frac{te^t + 4}{e^t}$$

- Approximation using the classic Runge-Kutta method



Runge-Kutta Methods

- Classic Runge-Kutta method
  - Single step method
  - Explicit method
  - Produces a local truncation error of $O(h^5)$
  - Produces a global error of $O(h^4)$
  - Widely used
  - Not useful for stiff problems
- Generalization of Runge-Kutta methods
  - Butcher-Tableau: array notation of the characteristic coefficients
  - (Semi)Implicit Runge-Kutta methods
- Embedded Runge-Kutta methods
  - Runge-Kutta-Fehlberg (RK45) method is a prominent example
    - 4^{th} order method using an embedded 5^{th} order method to approximate local truncation error (adaptive step width control)

Multistep Methods

- Single-step methods
  - Use only the previous evaluation point
  - Intermediate values (to construct higher order methods) are withdrawn after a step is performed
- Multistep methods
  - Previous evaluation points are reused to construct higher order methods (to gain efficiency)
  - Linear multistep methods use a linear combination of previous evaluation points
  - Adams-Bashforth Methods (explicit)
  - Adams-Moulton Methods (implicit)
  - Backward differentiation formulas (implicit)

Adams-Bashforth Methods

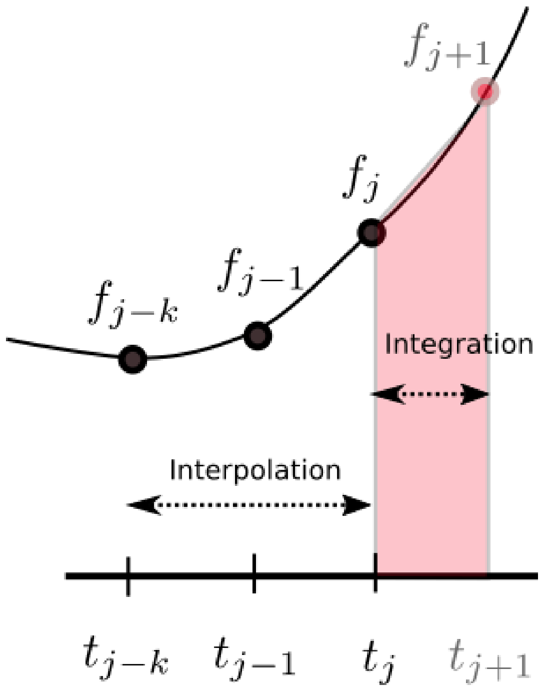
- Given

$$y' = f\big(t, y(t)\big)$$

- Reformulation and polynomial approximation

$$y(t_{j+1}) = y(t_j) + \int_{t_j}^{t_{j+1}} \underbrace{f\big(t, y(t)\big)}_{P_k(t)}\, dt$$

- $P_k(t)$ for f constructed from t_j to t_{j-k} (explicit)



Adams-Moulton Methods

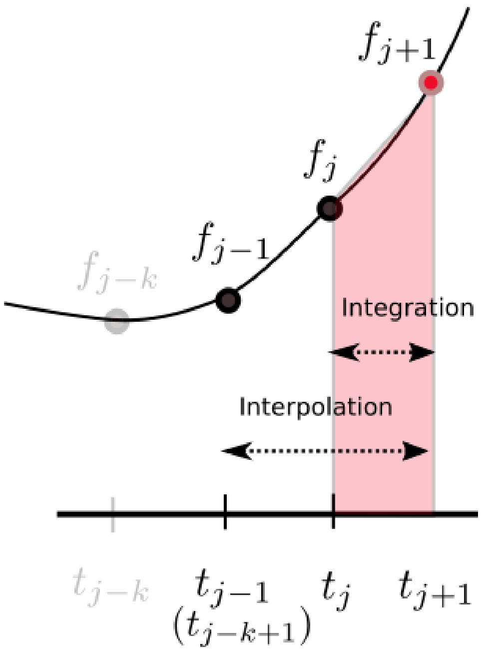
- Given

$$y' = f\big(t, y(t)\big)$$

- Reformulation and Polynomial approximation

$$y(t_{j+1}) = y(t_j) + \int_{t_j}^{t_{j+1}} \underbrace{f\big(t, y(t)\big)}_{P_k(t)}\, dt$$

- $P_k(t)$ for f constructed from t_{j+1} to t_{j-k+1} (implicit)



Backward Differentiation Formulas

- Given

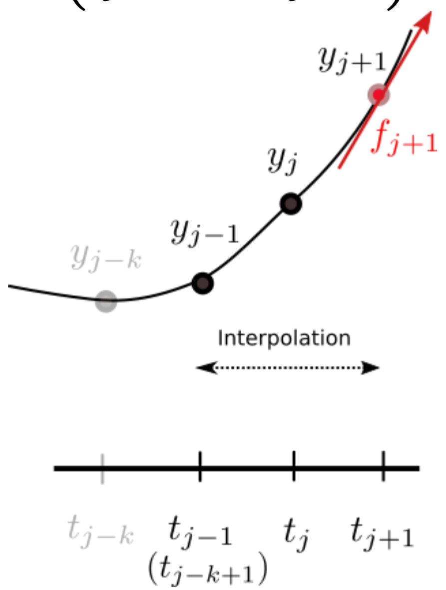
$$y' = f\big(t, y(t)\big)$$

- Reformulation and polynomial approximation

$$\underbrace{y\big(t_{j+1}\big)}_{P_k} = f\left(t_{j+1}, y_{j+1}\right)$$

- $P_k(t)$ directly for y using t_{j+1} to t_{j-k+1} (implicit)
- Condition for $P_k(t)$ at t_{j+1}

$$P_k'\left(t_{j+1}\right) = f\left(t_{j+1}, P_k\big(t_{j+1}\big)\right) = f\left(t_{j+1}, y\big(t_{j+1}\big)\right)$$



Higher Order ODEs

- Any explicit (system of) higher order ODEs can be transformed into an equivalent system of first order ODEs

$$\frac{d^2y}{dt^2} = -ky, \qquad \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} y \\ y' \end{bmatrix}, \quad \begin{bmatrix} z_1' \\ z_2' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

$$my'' + by' + ky = f(t) \qquad \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} y \\ y' \end{bmatrix}, \quad \begin{bmatrix} z_1' \\ z_2' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -b/m & -k/m \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} 0 \\ f(t)/m \end{bmatrix}$$

  - Semi-implicit schemes
    - Sequential evaluation of coupled ODEs
    - Incorporation of already updated values
- Partial Differential Equations
  - Spatial discretization leads to system of coupled ODEs
  - Time integration schemes for ODEs can be used
  - Tailored/adopted schemes for specific problem domains

Conclusions: ODE Methods

- Single-Step
  - Explicit
    - Small step size required, but ‘just’ evaluations, no equations to solve
  - Implicit
    - Large step size, requires solution to (in general) non-linear equation
  - Intermediate evaluations: Runge-Kutta Methods
    - Classic Runge-Kutta: Higher order accuracy using multiple evaluations
- Multistep
  - Idea: efficiency through reuse of previous steps
  - Explicit/Implicit Methods
  - Reuse of past values
- Embedded methods
  - Idea: perform multiple (two) approximations of different order
  - Model local truncation error as difference between the approximations
  - Use error model for ‘automatic’ step size control

Next: finite volume method

Finite Volume Method

- Idea: Reformulate a differential equation using divergence theorem
- Example: Poisson equation

$$\nabla^2 u = \nabla \cdot \nabla u = f$$

- Apply divergence theorem

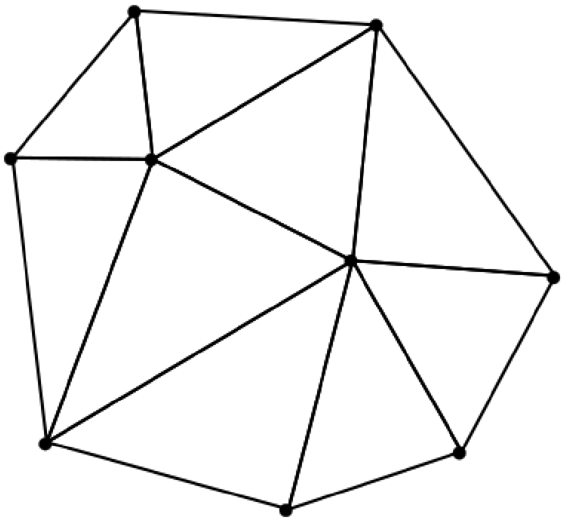
$$\int \nabla \cdot \nabla u dV = \oint \nabla u dA = \int f dV$$

- Finite Volume Method
  - Decomposing the domain into discrete cells (e.g., triangles, boxes)
  - Construct a corresponding dual representation (control volumes)
  - Setup of neighbour information between control volumes sharing surface regions
  - Approximate surface integrals on shared surfaces
  - Define surface integrals on domain boundary according to boundary conditions
  - Assemble linear system of equations

Domain Decomposition

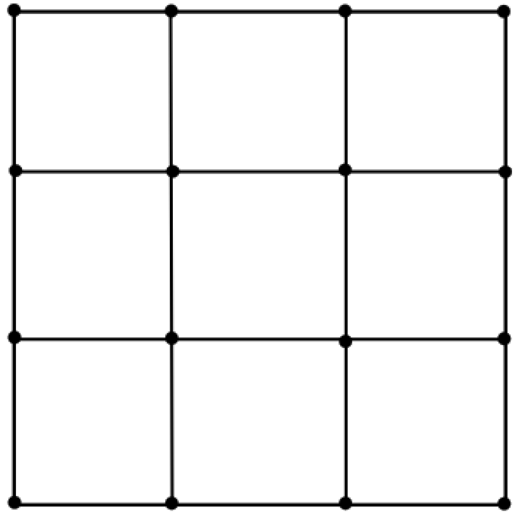
Triangles

Regular Grid

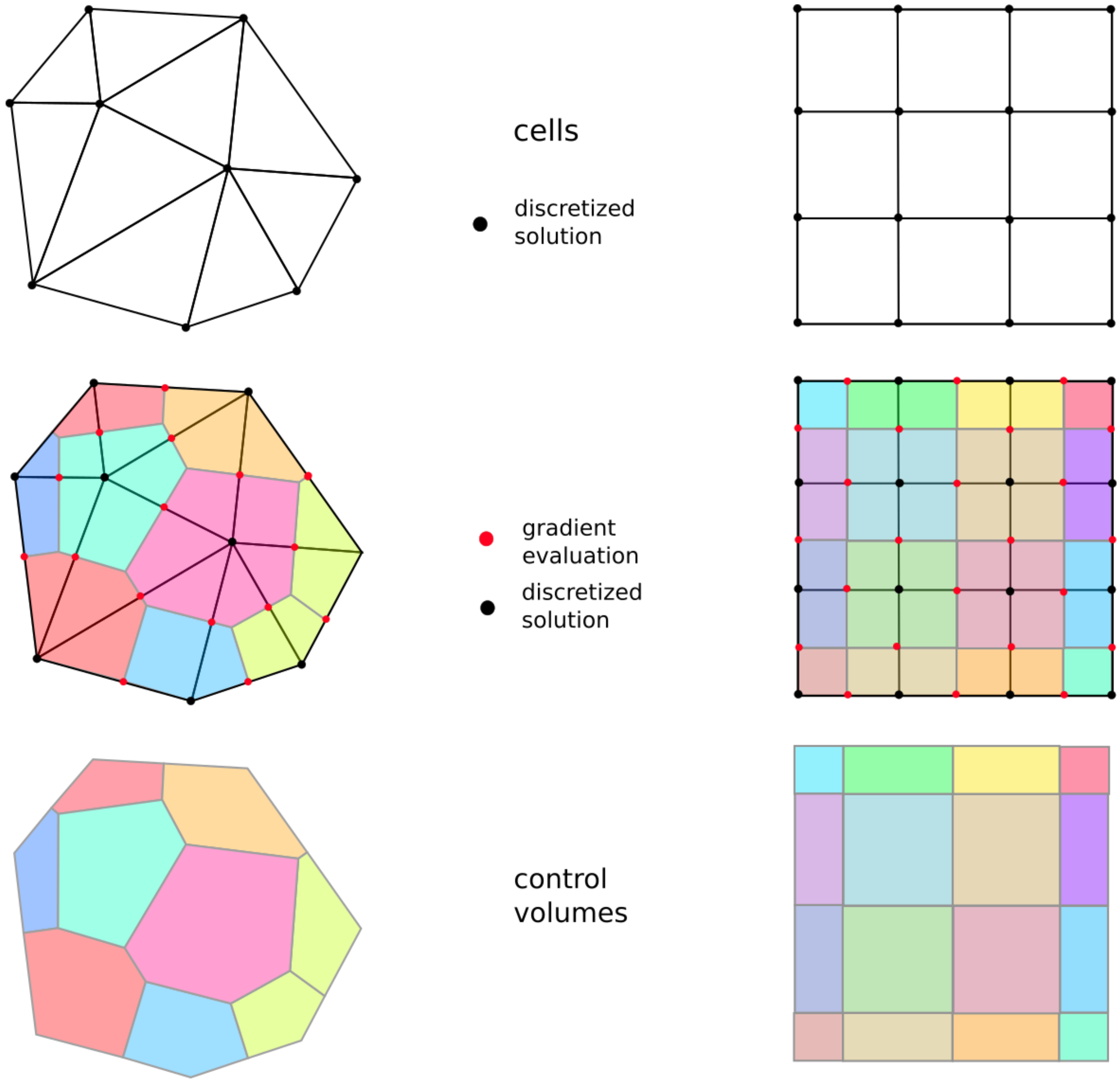


cells

discretized solution



Control Volumes



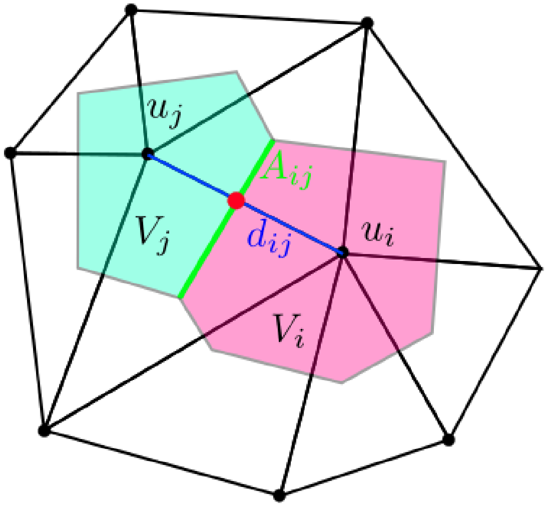
Discrete Gradient

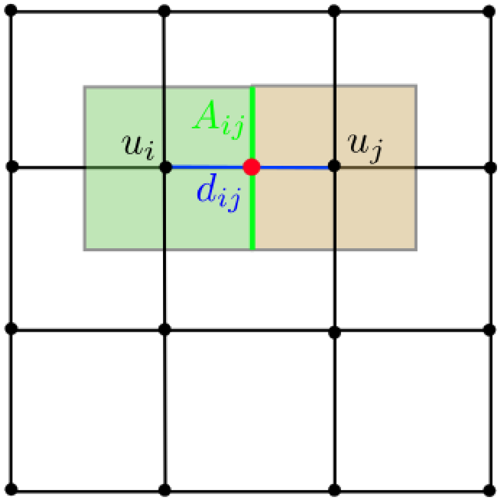
- First order difference approximation of gradient normal (outward) to shared surfaces

$$\oint \nabla u dA = \int f dV$$

$$\sum_j \underbrace{\frac{u_j - u_i}{d_{ij}}}_{\nabla u} A_{ij} = f_i V_i$$

- Conservative discretization





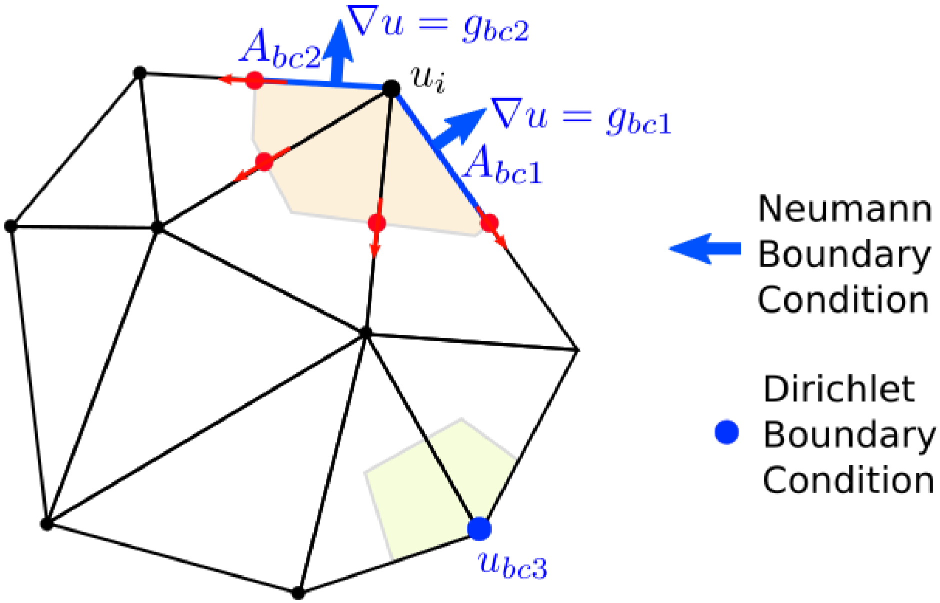
Boundary/Interface Conditions

- Neumann boundary condition

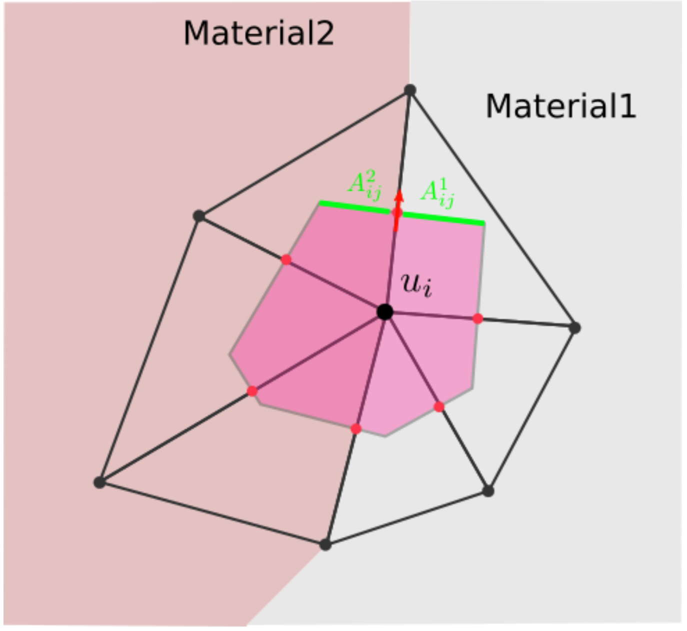
$$\oint \nabla u dA = \int f dV$$

$$\sum_j \frac{u_j - u_i}{d_{ij}} A_{ij} + g_{bc1} A_{bc1} + g_{bc2} A_{bc2} = f_i V_i$$

Boundary condition



Interface condition



Relation to FDM

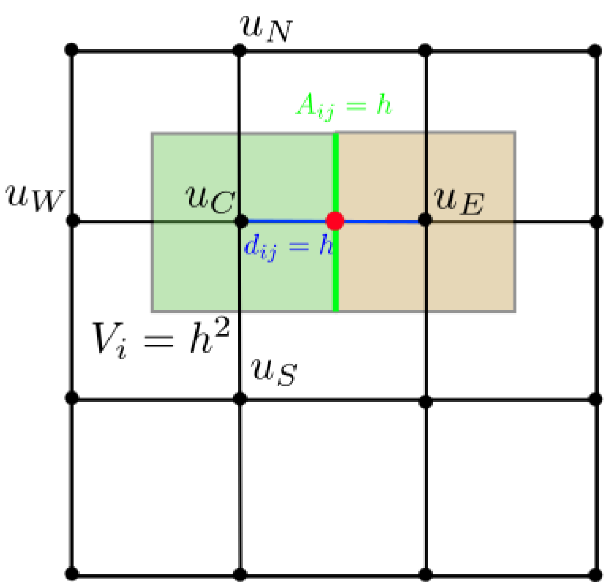
- For regular grid: FVM equal to FDM discretization
  - Finite Volume

$$\sum_{j=N,S,E,W} \underbrace{\frac{u_j - u_C}{h}}_{\nabla u} h = \frac{u_N + u_S + u_E + u_W - 4u_C}{h} h = f_C h^2$$

  - Finite Differences (second order central)

$$(u_{xx} + u_{yy}) \approx \left(\frac{(u_N - 2u_C + u_S)}{h^2} + \frac{(u_W - 2u_C + u_E)}{h^2} \right) =$$

$$= \frac{1}{h^2}(u_N + u_S + u_E + u_W - 4u_C) = f_C$$



Conclusions: Finite Volume Method

Using triangles (2D) or tetrahedral (3D) as cells:

- Advantages
  - Complex domains possible
  - Straightforward handling of discontinuous properties/interfaces
  - Adaptive discretization resolution straightforward
  - Mesh Requirements: Delaunay triangulation
  - Conservative method
- Disadvantages
  - No implicit connectivity (explicit neighbourhood information required)
  - No constant stencil coefficients (coefficients stored for each control volume)

Conclusions

- Q1: What is the order of the forward/backward Euler method?
- Q2: Are the Runge-Kutta methods single-step or multistep methods?
- Q3: How can first order ODEs methods be applied to higher order ODEs?
- Q4: Why are finite volume schemes ‘conservative’?
- Q5: What are advantages of the Finite Volume Method over the Finite Difference Method?

References

- Norbert Köckler – Numerische Mathematik (2006)
- Svein Linge – Programming for Computations (2016)