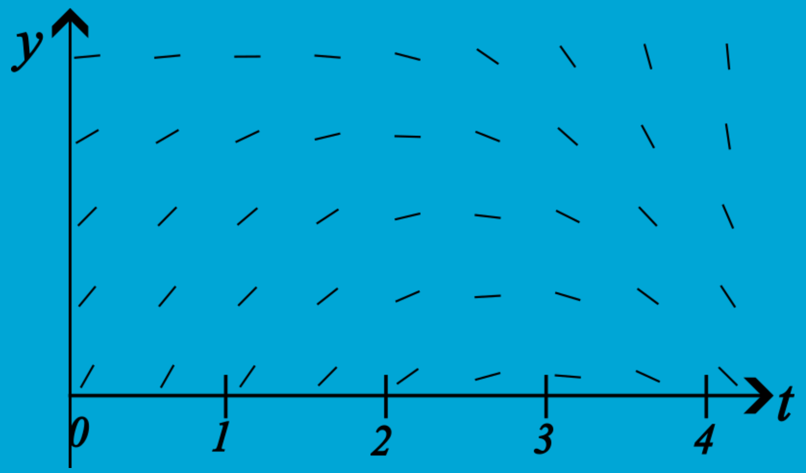
CEGM1000
Modelling, Uncertainty and Data for Engineers

Week 1.3
Numerical modelling (Beyond Fundamentals)

Ronald Brinkgreve, Anna Störiko

Based on a previous version from Jaime Arriaga and the rest of the MUDE team

TU Delft



Learning objectives

At the end of this lecture, you should be able to

- Discuss the characteristics of Explicit and Implicit Numerical schemes
- Schematize numerical solutions of ODEs
- Solve initial and boundary value problems numerically

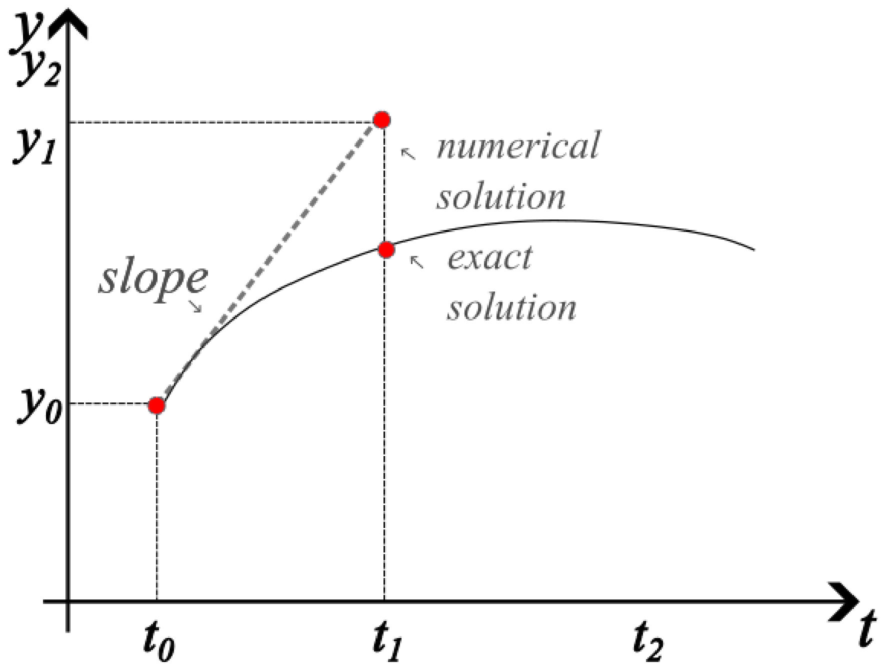
Contents

- Explicit and Implicit numerical schemes: single step
  - Initial Value Problems

- Multiple-step and multi-stage schemes
  - Initial Value Problems

- Second order ODEs
  - Boundary Value Problems

Explicit Euler (Euler Forward)



$$t_{i+1} = t_i + \Delta t$$

$$y_{i+1} = y_i + \Delta t * slope_i$$

$$y_1 = y_0 + \Delta t * slope_0$$

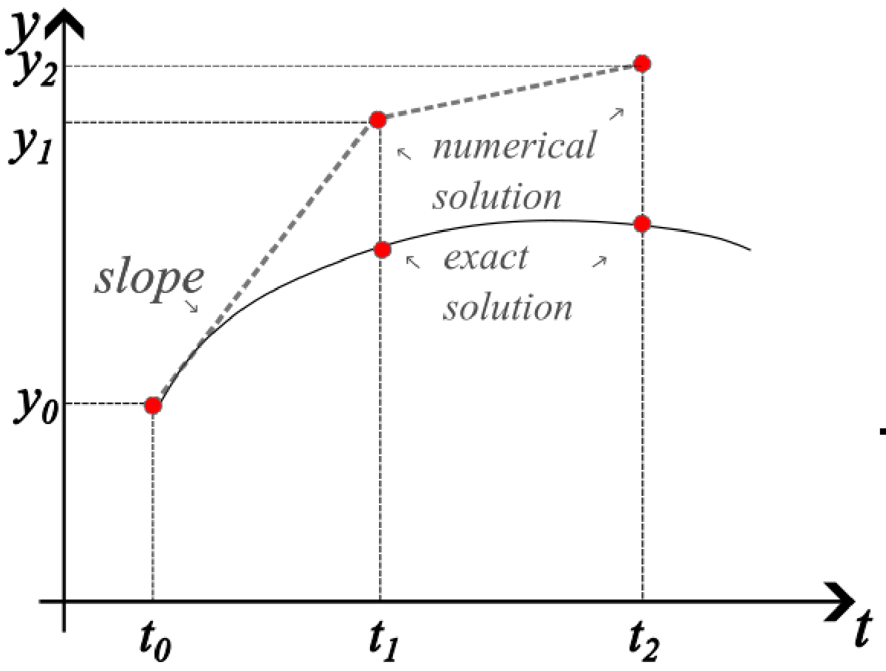
The slope is computed using the Forward Difference formula, first-order accurate.

Truncation error = $y_1^{TSE} - y_1^{FE}$

$$= y_0 + \Delta t y_0' + \frac{\Delta t^2}{2!} y''_0 + \cdots - (y_0 + \Delta t y_0')$$

$$= \frac{\Delta t^2}{2!} y''_0 + \cdots \approx O(\Delta t^2)$$

Explicit Euler (Euler Forward)



$$t_{i+1} = t_i + \Delta t$$

$$y_{i+1} = y_i + \Delta t * slope_i$$

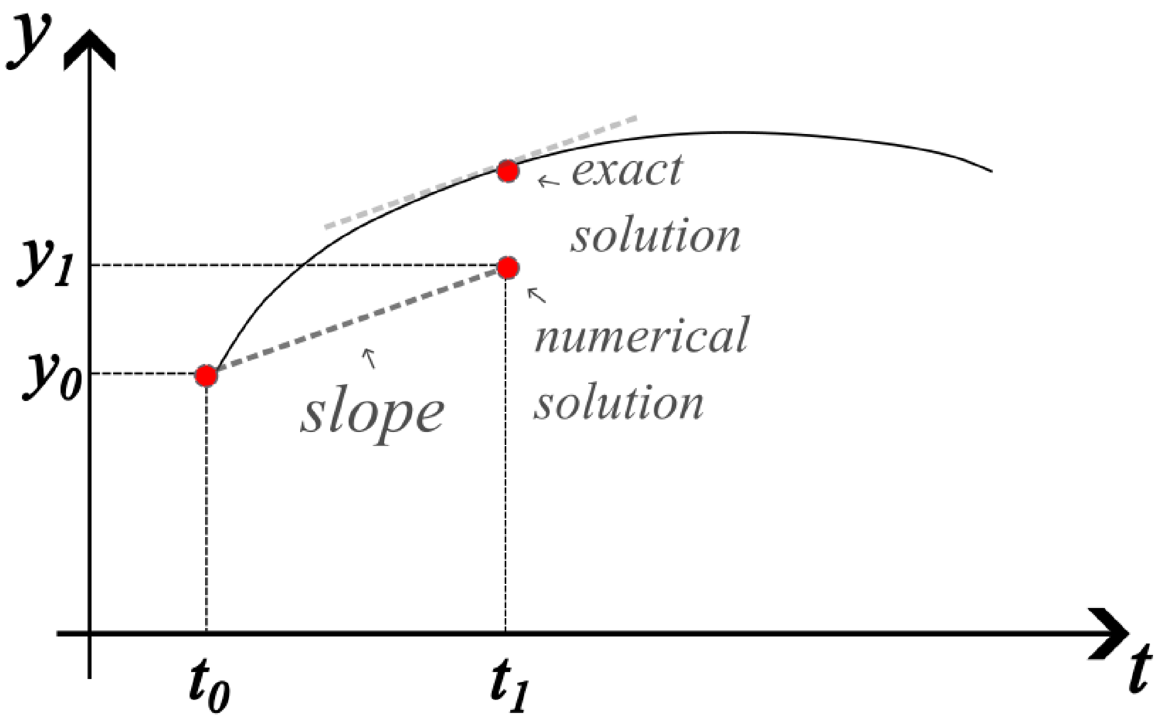
$$y_2 = y_1 + \Delta t * slope_1$$

Total truncation error $= \sum_{i=0}^{n-1} y_{i+1}^{TSE} - y_{i+1}^{FE}$

$$= \sum_{i=0}^{n-1} \frac{\Delta t^2}{2!} y''_i + \cdots \approx \frac{\Delta t^2}{2!} \frac{b-a}{\Delta t} y''_i$$

$$\approx \frac{\Delta t}{2!}(b-a)\overline{y''} \approx O(\Delta t)$$

Implicit Euler (Euler Backward)



$$t_{i+1} = t_i + \Delta t$$

$$y_{i+1} = y_i + \Delta t * slope_{i+1}$$

$$y_1 = y_0 + \Delta t * slope_1$$

The slope is computed using the Backward Difference formula, first-order accurate.

Total truncation error $\approx O(\Delta t)$

Stability

ODE: $$\frac{dy}{dt} = -\alpha y$$

Exact solution: $$y(t) = e^{-\alpha t}$$

Stability criterion for **Explicit Euler**: $$\Delta t < \frac{2}{\alpha}$$

Note that the solution may already become highly inaccurate before the limit!

Implicit Euler is unconditionally stable; this does NOT mean it is more accurate!

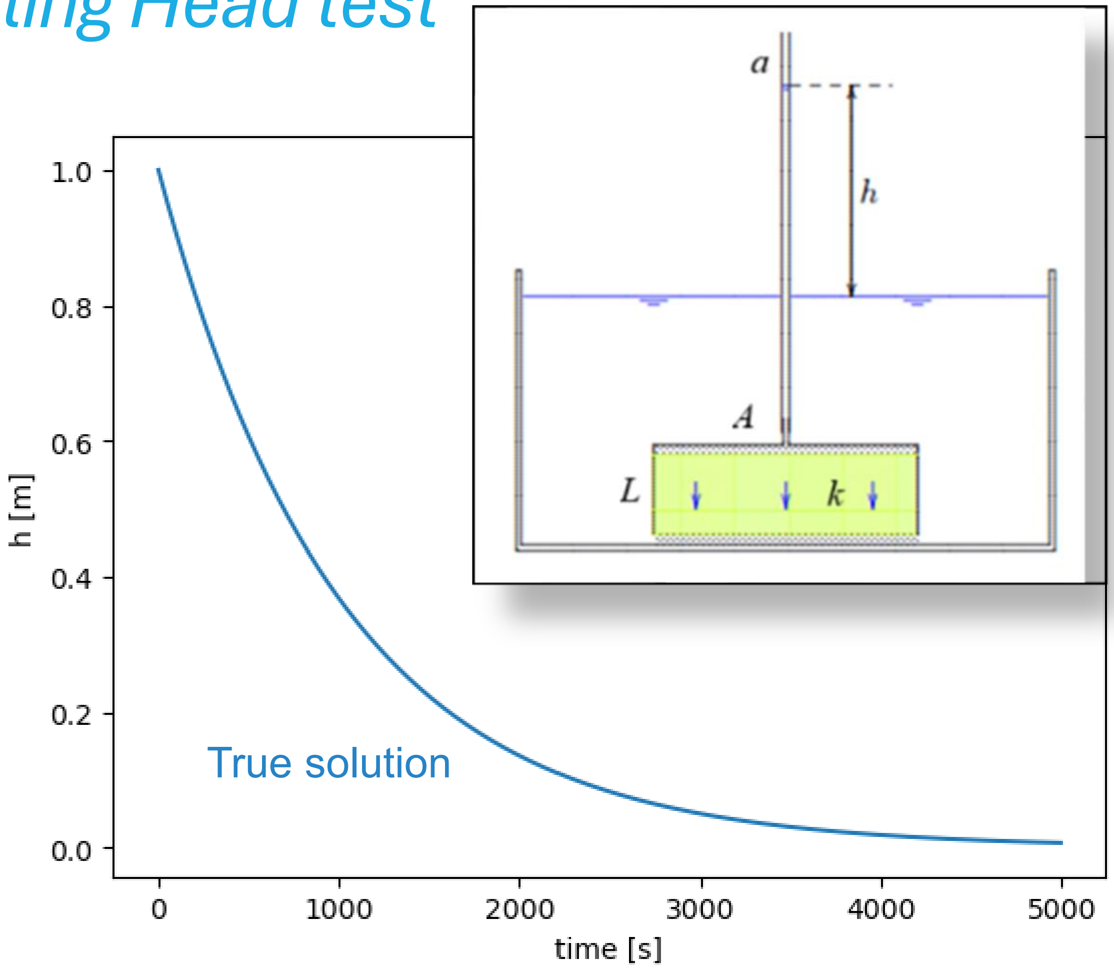
Let's start with a simple example: The Falling Head test

ODE: $$\frac{dh}{dt} = -\frac{kA}{aL}h$$

Initial value: at $t = t_0$: $h = h_0$

Exact solution: $$h(t) = h_0\, e^{-\frac{kAt}{aL}}$$

k	10^{-6}	m/s
A	0.1	m^2
a	0.001	m^2
L	0.1	m
h_0	1.0	m



Purpose of the test on a soil sample:
To measure hydraulic conductivity k

Let’s start with a simple example: The Falling Head test

ODE: $$\frac{dh}{dt} = -\frac{kA}{aL}h$$

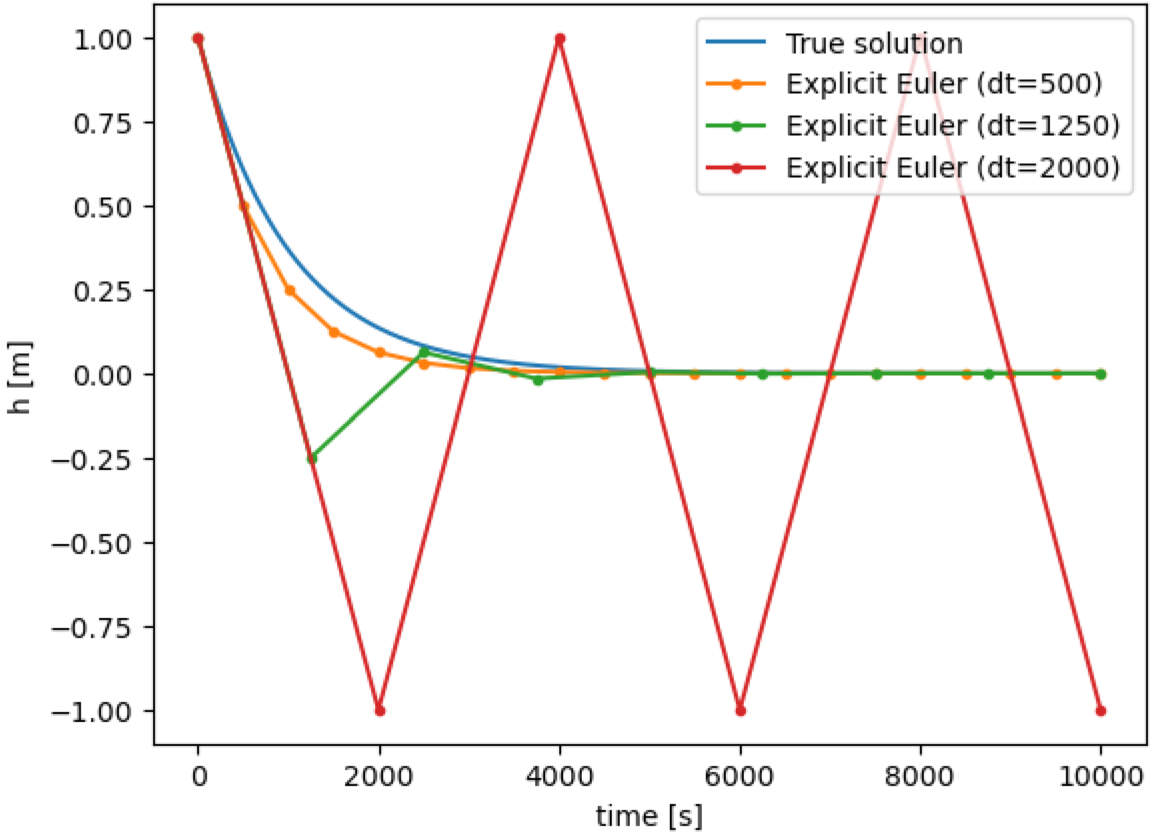
Initial value: at $t = t_0$: $h = h_0$

Explicit Euler: $$\frac{h(t_{i+1}) - h(t_i)}{\Delta t} = -\frac{kA}{aL}h(t_i)$$

$$\Rightarrow \quad h(t_{i+1}) = h(t_i) - \Delta t\frac{kA}{aL}h(t_i)$$

$$= \left(1 - \Delta t\frac{kA}{aL}\right)h(t_i)$$

Stability: $\Delta t < 2\frac{aL}{kA} = 2000$ s




```python
t_n = np.linspace(0, maxtime, n_steps + 1)
h_EF = []
h_EF.append(h0)
for i in range(n_steps):
    h_new = (1.0 - dt * k * A / (a * L)) * h_EF[-1]
    h_EF.append(h_new)
```

Let's start with a simple example: The Falling Head test

ODE: $$\frac{dh}{dt} = -\frac{kA}{aL}h$$

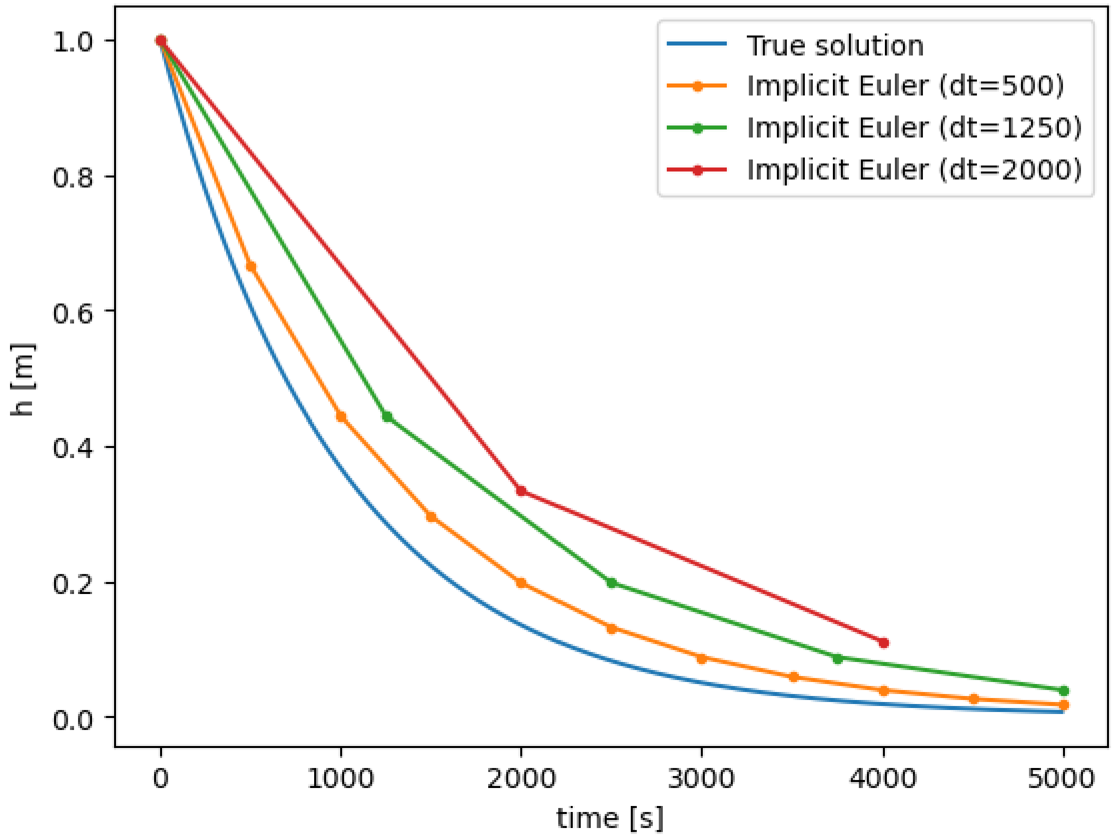
Initial value: at $t = t_0$: $h = h_0$

Implicit Euler: $$\frac{h(t_{i+1}) - h(t_i)}{\Delta t} = -\frac{kA}{aL}h(t_{i+1})$$

$$\Rightarrow \quad h(t_{i+1})\left(1 + \Delta t\frac{kA}{aL}\right) = h(t_i)$$

$$h(t_{i+1}) = h(t_i)/\left(1 + \Delta t\frac{kA}{aL}\right)$$

Stability: Not an issue (unconditionally stable)



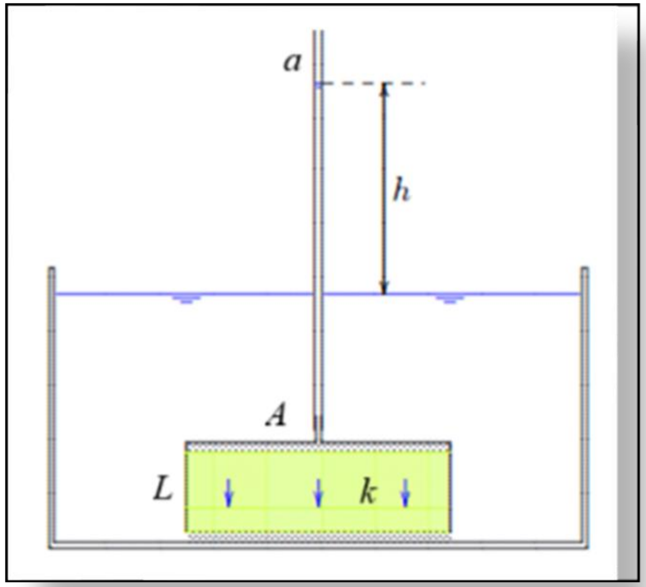

```
h_EB = []
h_EB.append(h0)
for i in range(n_steps):
    h_new = h_EB[-1] / (1.0 + dt * k * A / (a * L))
    h_EB.append(h_new)
```

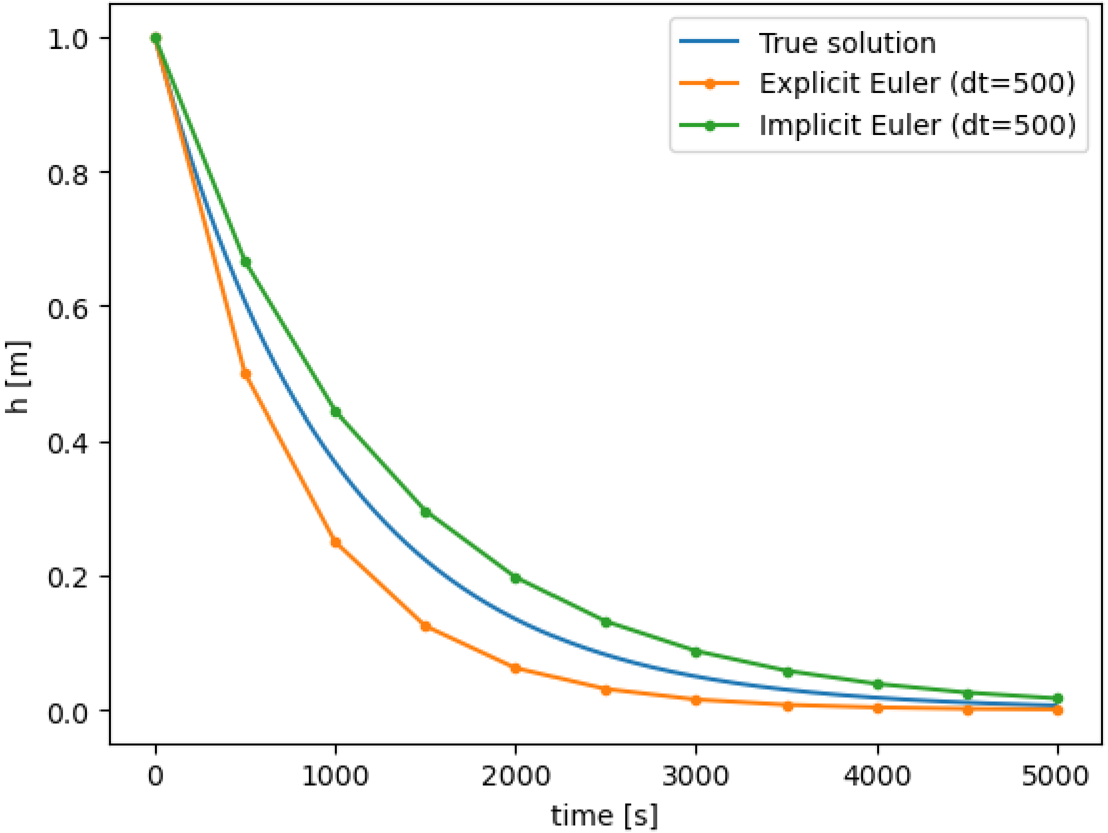
Let's start with a simple example: The Falling Head test

ODE: $$\frac{dh}{dt} = -\frac{kA}{aL}h$$

Initial value: at $t = t_0$: $h = h_0$

Comparison of solutions:

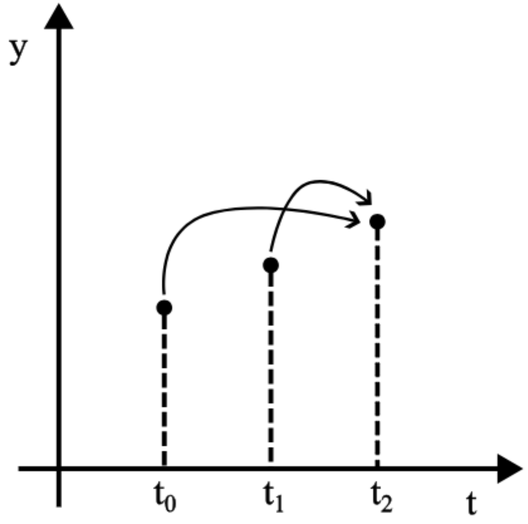




Multi-step methods

Adams-Bashfort
second order accurate

$$y_{i+1} = y_i + \frac{\Delta t}{2}(3y_i' - y'_{i-1})$$



Not enough info at t_1

- Adams-Bashfort are explicit methods of higher accuracy

- AB is not self starting!

- Adams-Moulton is an implicit method of higher accuracy

- AM is self starting but requires iterations

Multi-step means:
To calculate a new point, it uses more than one previous point

Multi-stage methods

The main error of simple single step methods is assuming that the slope does not change between i and $i+1$

- Modified Euler $\quad y_{i+1} = y_i + \Delta t\,(y'_i + y'_{i+1^*})/2$

- Midpoint $\quad y_{i+1} = y_i + \Delta t\, y'_{i+\frac{1}{2}^*}$

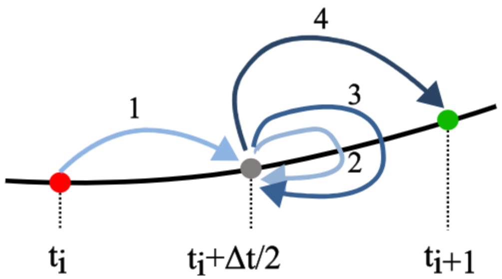
- Heun's method (= 2-stage RK)

$$y_{i+1} = y_i + \Delta t\,(k_1 + k_2)\,/\,2$$
$$k_1 = y'_i$$
$$k_2 = y'_{i+1*} \;\; with \;\; y_{i+1*} = y_i + \Delta t\, k_1$$

- 4-stage Runge-Kutta

$$y_{i+1} = y_i + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) + \mathcal{O}(\Delta t^5)$$

$$\begin{cases} k_1 &= y'_i \\ k_2 &= y'_{i+1/2} \text{ with } y_{i+1/2} = y_i + \frac{\Delta t}{2} k_1 \\ k_3 &= y'_{i+1/2} \text{ with } y_{i+1/2} = y_i + \frac{\Delta t}{2} k_2 \\ k_4 &= y'_{i+1} \text{ with } y_{i+1} = y_i + \Delta t\, k_3 \end{cases}$$



Multi-stage means:
To calculate a new point, it uses intermediate estimates of the new point

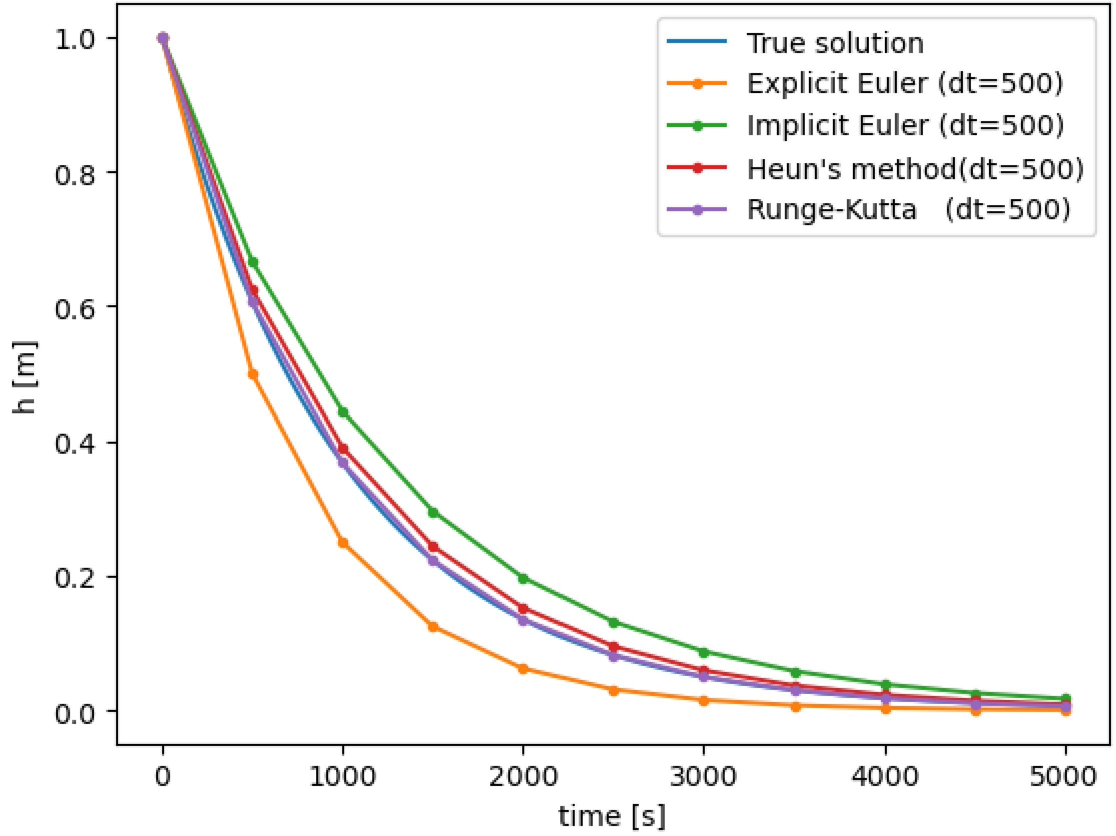
Back to our simple example: The Falling Head test

ODE: $$\frac{dh}{dt} = -\frac{kA}{aL}h$$

Initial value: at $t = t_0$: $h = h_0$

Comparison of solutions:

```
h_RK4 = []
h_RK4.append(h0)
for i in range(n_steps):
    fac = -k * A / (a * L)
    k1 = fac * h_RK4[-1]
    k2 = fac * (h_RK4[-1] + dt / 2 * k1)
    k3 = fac * (h_RK4[-1] + dt / 2 * k2)
    k4 = fac * (h_RK4[-1] + dt * k3)
    h_new = h_RK4[-1] + dt / 6 * (k1 + 2 * k2 + 2 * k3 + k4)
    h_RK4.append(h_new)
```



Non-linear ODE

$$\frac{dp(t)}{dt} = -p^{3/2} + 5 \cdot p_{cst}(1 - e^{-t})$$

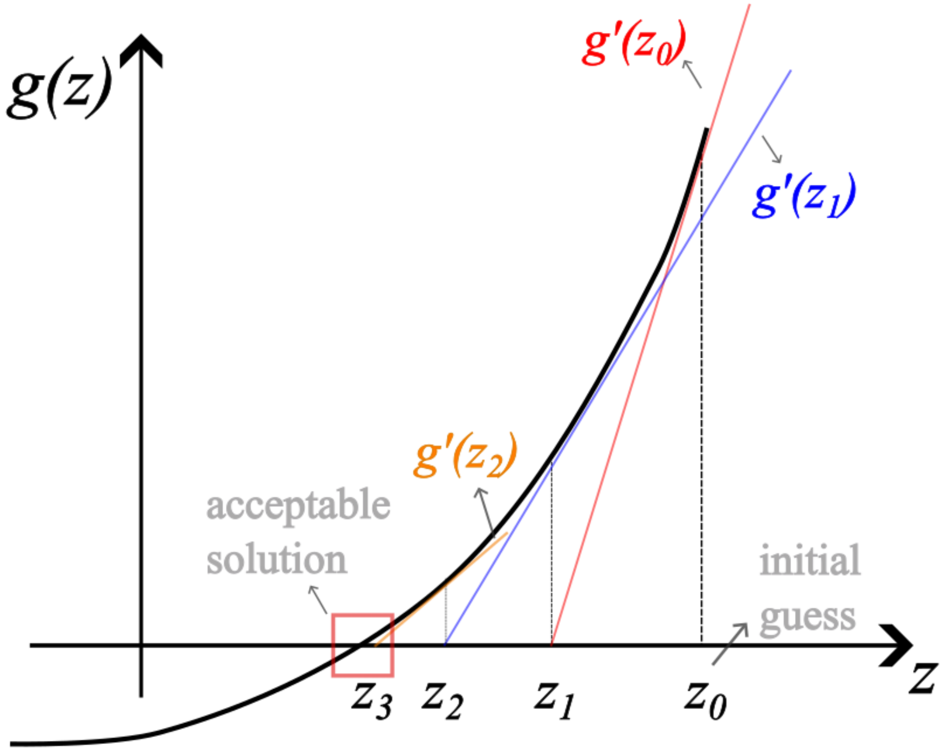
$$p_{i+1} = p_i + \Delta t \left(-p_i^{3/2} + 5p_{cst} \cdot (1 - e^{-t_i}) \right)$$

$$p_{i+1} = p_i + \Delta t \left(-p_{i+1}^{3/2} + 5p_{cst} * (1 - e^{-t_{i+1}}) \right)$$

Euler forward (explicit):
Can be solved directly

Euler backward (implicit):
Can only be solved iteratively

Non-linear ODE: Newton-Raphson (iterative) method



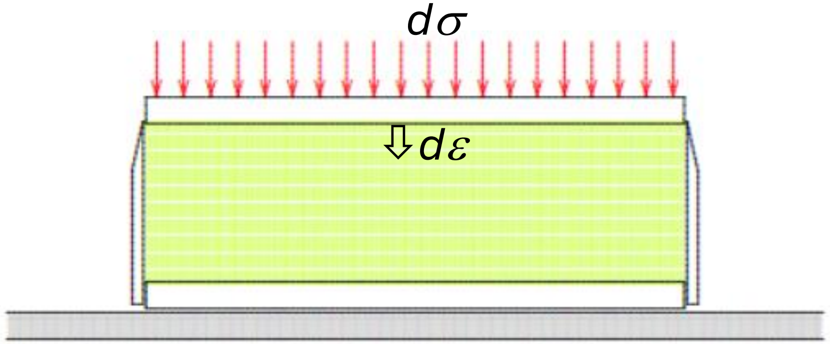
Newton-Raphson updating: $$z_{i+1} = z_i - \frac{g(z_i)}{g'(z_i)}$$

Let’s take a break..

Please, take a seat…

Example: Non-linear stress-strain relation in one-dimensional compression

Oedometer test: to measure soil compressibility (or stiffness)



Stress-strain relationship:

$$d\sigma = C\left(\frac{\sigma}{\sigma^{ref}}\right)^m d\varepsilon$$

C → stiffness at ref. stress

As ODE:

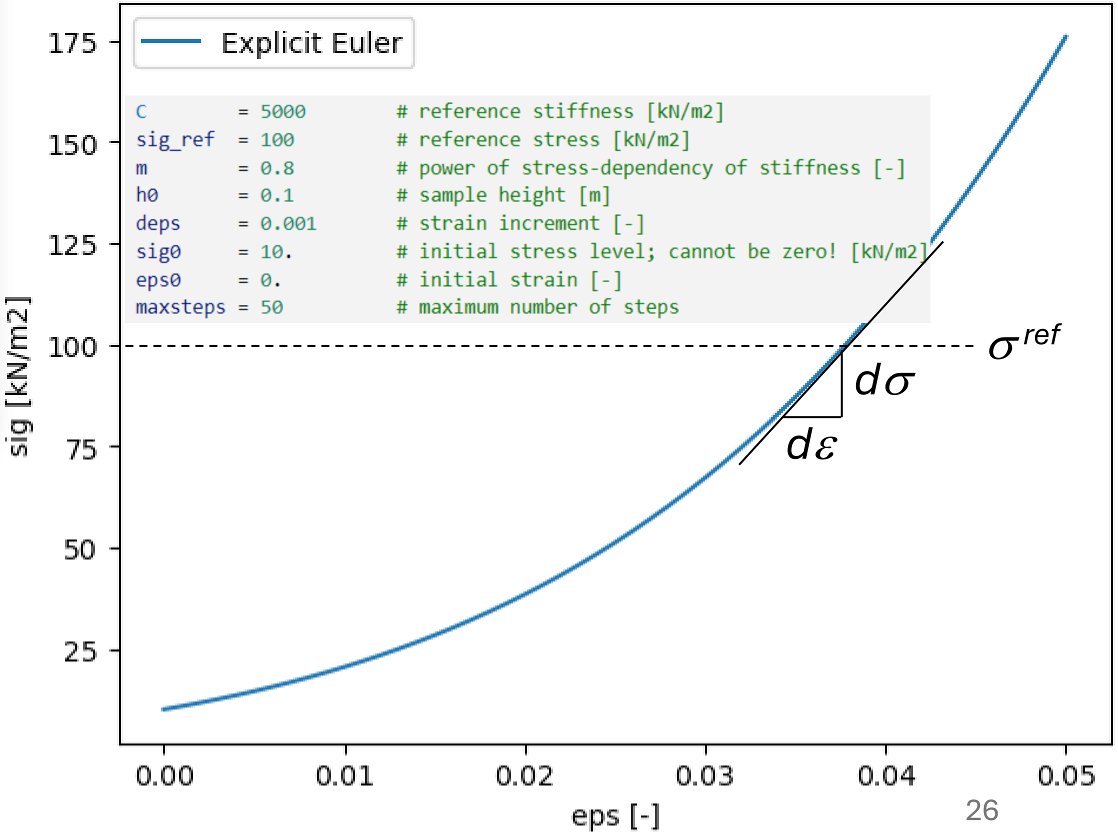
$$\frac{d\sigma}{d\varepsilon} = C\left(\frac{\sigma}{\sigma^{ref}}\right)^m = C^*\sigma^m$$

with $\quad C^* = \dfrac{C}{(\sigma^{ref})^m}$

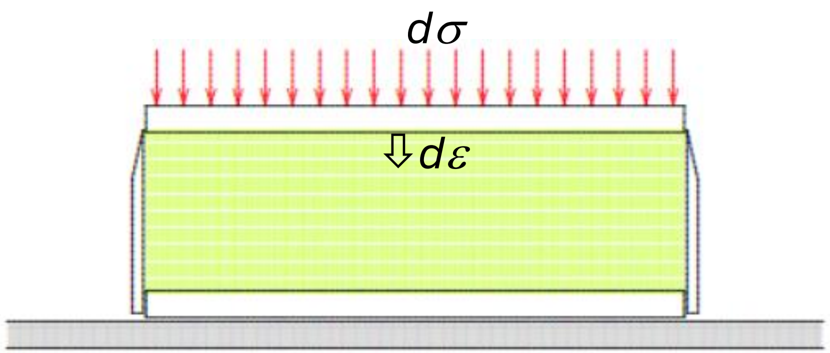
Explicit Euler:

$$\frac{\sigma_{i+1} - \sigma_i}{\Delta\varepsilon} = C^*\sigma_i^m$$

$$\sigma_{i+1} = \sigma_i + \Delta\varepsilon C^*\sigma_i^m$$



Example: Non-linear stress-strain relation in one-dimensional compression



ODE: $$\frac{d\sigma}{d\varepsilon} = C\left(\frac{\sigma}{\sigma^{ref}}\right)^m = C^*\sigma^m$$

Implicit Euler: $$\frac{\sigma_{i+1} - \sigma_i}{\Delta\varepsilon} = C^*\left(\sigma_{i+1}\right)^m$$

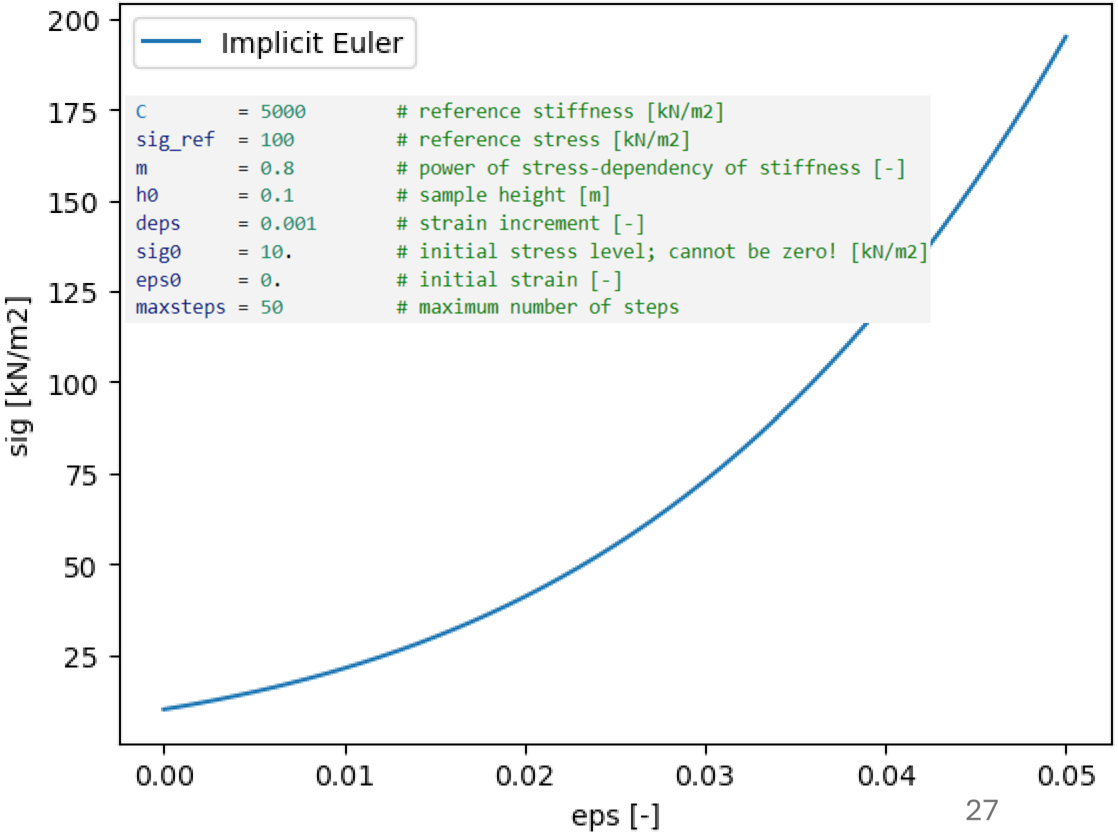
Possible solution:
(solved iteratively)
$$\sigma_{i+1}^{j+1} = \sigma_i + \Delta\varepsilon C^*\left(\sigma_{i+1}^{j}\right)^m$$

Not so good solution:
(requires more
iterations)

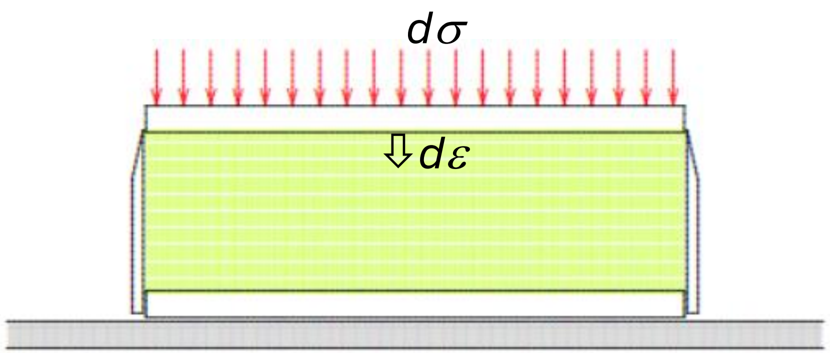
$$\sigma_{i+1} - \Delta\varepsilon C^*\left(\sigma_{i+1}\right)^m = \sigma_i$$

$$\sigma_{i+1}\left[1 - \Delta\varepsilon C^*\left(\sigma_{i+1}\right)^{m-1}\right] = \sigma_i$$

$$\sigma_{i+1}^{j+1} = \frac{\sigma_i}{\left[1 - \Delta\varepsilon C^*\left(\sigma_{i+1}^{j}\right)^{m-1}\right]}$$



Example: Non-linear stress-strain relation in one-dimensional compression



ODE: $$\frac{d\sigma}{d\varepsilon} = C\left(\frac{\sigma}{\sigma^{ref}}\right)^m = C^*\sigma^m$$

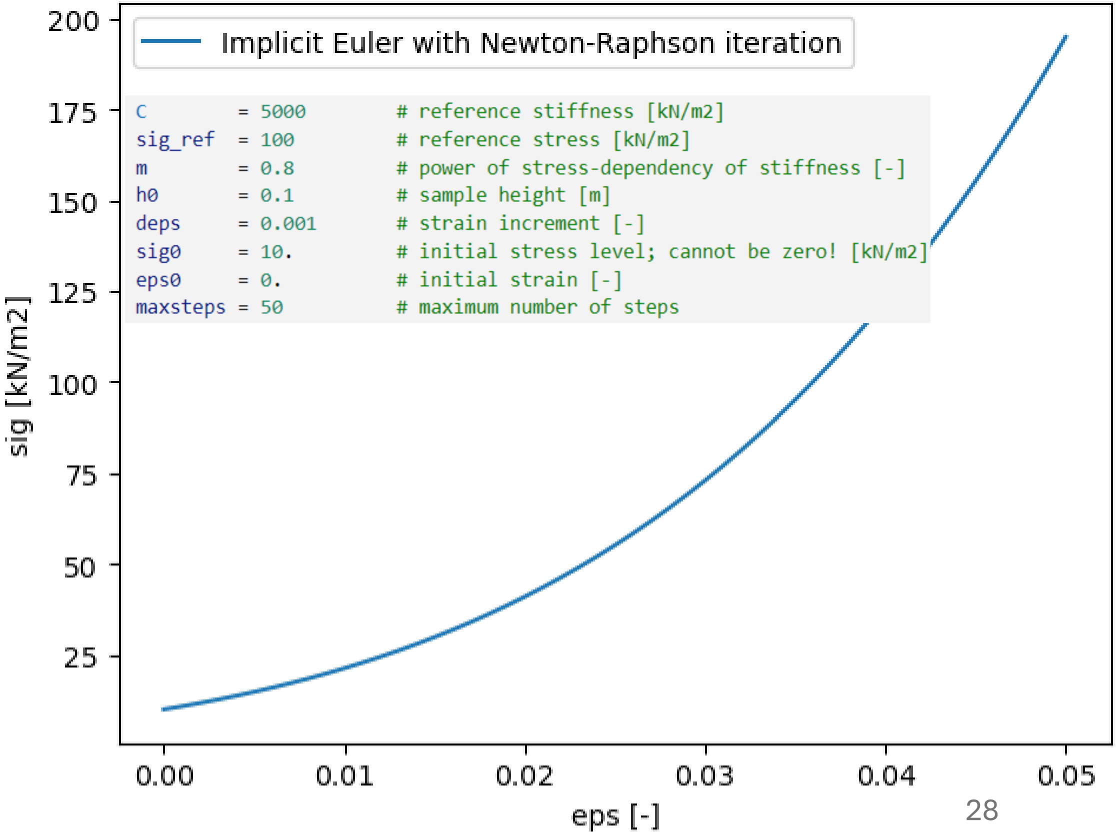
Best solution (least number of iterations):
Implicit Euler with Newton-Raphson iterations:

$g(\sigma_{i+1}) = 0$ with $g(\sigma_{i+1}) = \sigma_{i+1} - \Delta\varepsilon C^*\left(\sigma_{i+1}\right)^m - \sigma_i$

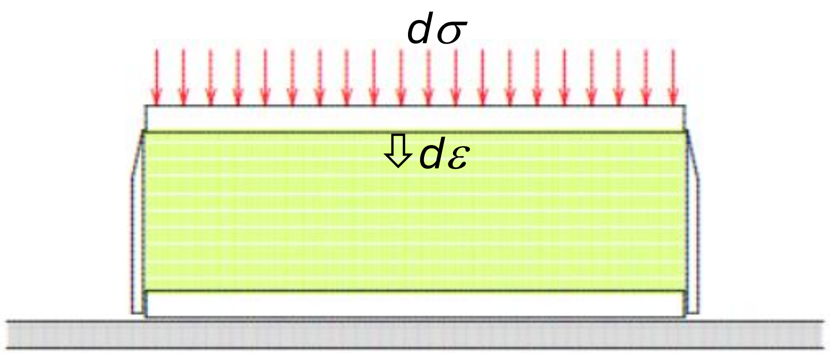
$$\sigma_{i+1}^{j+1} = \sigma_{i+1}^{j} - \frac{g(\sigma_{i+1}^{j})}{g'(\sigma_{i+1}^{j})}$$

$$g(\sigma_{i+1}^{j}) = \sigma_{i+1}^{j} - \Delta\varepsilon C^*\left(\sigma_{i+1}^{j}\right)^m - \sigma_i$$

$$g'(\sigma_{i+1}^{j}) = \frac{dg}{d\sigma_{i+1}} = 1 - m\Delta\varepsilon C^*\left(\sigma_{i+1}^{j}\right)^{m-1}$$



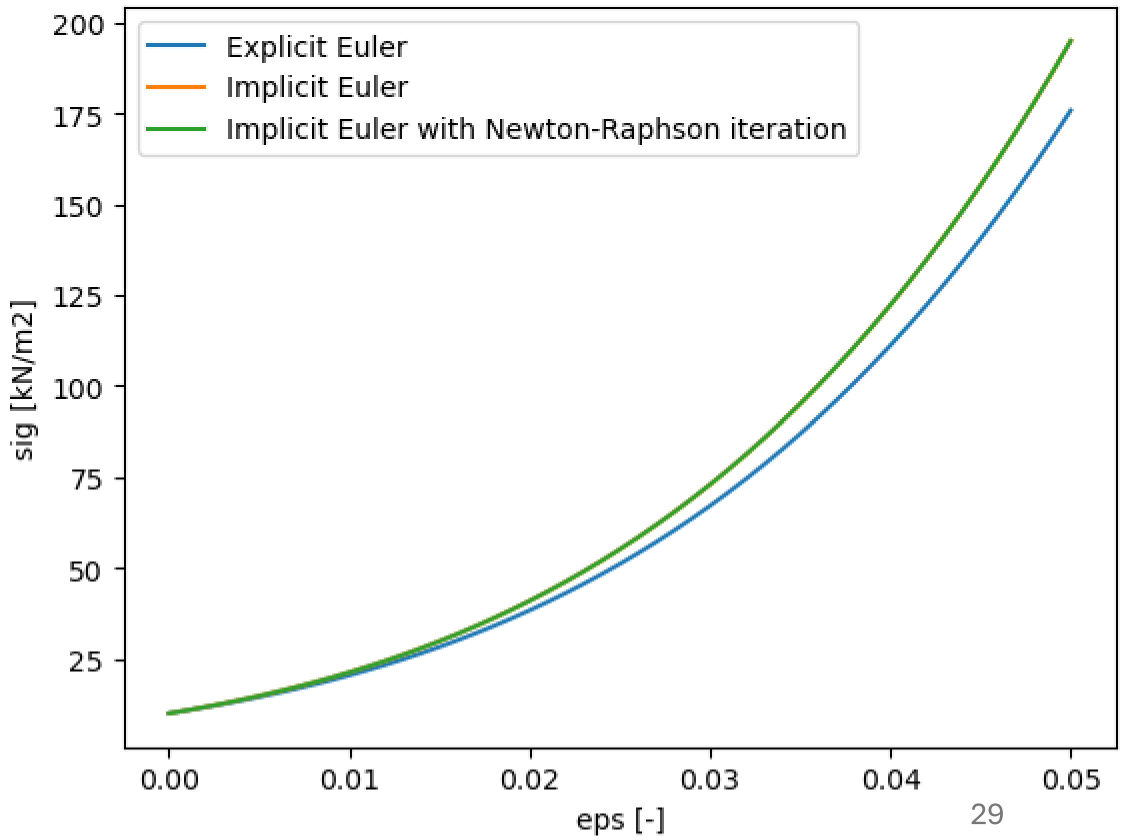
Example: Non-linear stress-strain relation in one-dimensional compression



ODE: $$\frac{d\sigma}{d\varepsilon} = C\left(\frac{\sigma}{\sigma^{ref}}\right)^m = C^*\sigma^m$$

Comparison of solutions:

```
C        = 5000        # reference stiffness [kN/m2]
sig_ref  = 100         # reference stress [kN/m2]
m        = 0.8         # power of stress-dependency of stiffness [-]
h0       = 0.1         # sample height [m]
deps     = 0.001       # strain increment [-]
sig0     = 10.         # initial stress level; cannot be zero! [kN/m2]
eps0     = 0.          # initial strain [-]
maxsteps = 50          # maximum number of steps
```



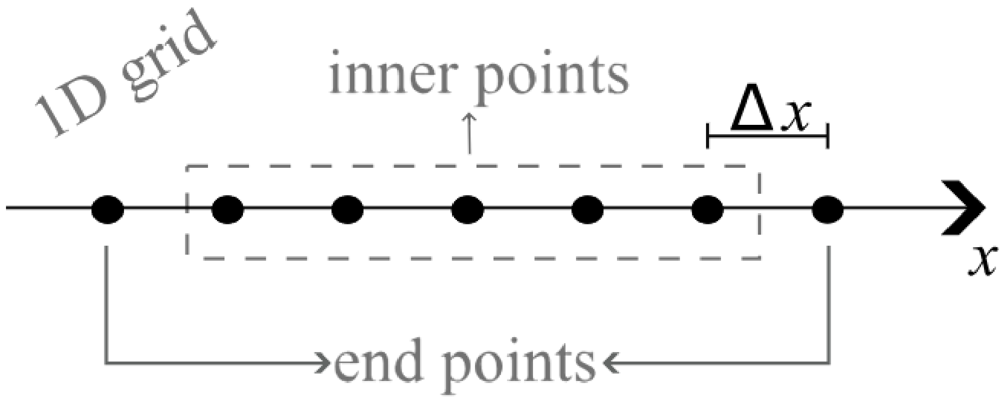
Final sample height:

```
h_end = h0 (1-eps) = 0.095 m
```

Second order ODE

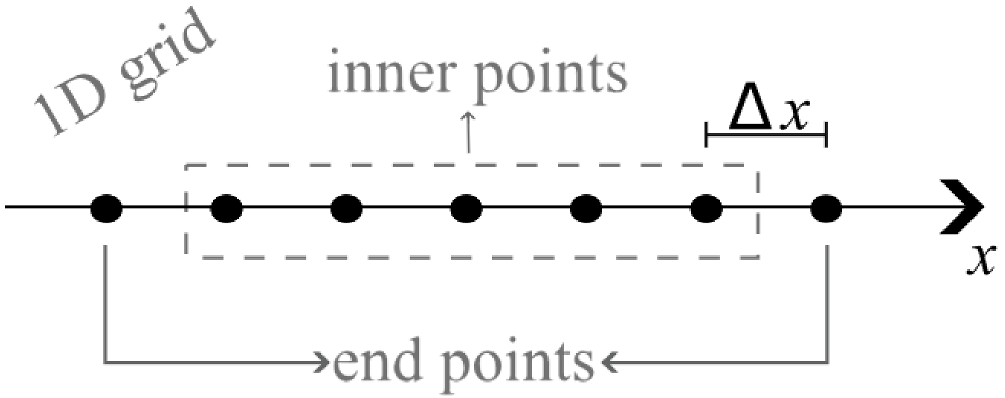
$$\frac{d^2y}{dt^2} = \frac{dy}{dt} - y + \cos t = 0$$ ➔ IVP

$$\frac{d^2T}{dx^2} - \alpha_1(T - T_s) = 0$$ ➔ BVP



Second order ODE: Boundary Conditions

$$\frac{d^2T}{dx^2} - \alpha_1(T - T_s) = 0$$



$$\frac{d^2y}{dx^2} = g(x, y, \frac{dy}{dx})$$

- **Dirichlet**

$$y(x = a) = Y_a \text{ and } y(x = b) = Y_b$$

- **Neumann**

$$\left.\frac{dy}{dx}\right|_{x=a} = D_a \text{ and } \left.\frac{dy}{dx}\right|_{x=b} = D_b$$

- Mixed

Join the Vevox session

Go to **vevox.app**

Enter the session ID: **123-957-664**

Or scan the QR code

Give an example of a Neumann boundary condition for a deformation or a flow problem

0

Join at: **vevox.app**

ID: **123-957-664**

Showing Results

Give an example of a Neumann boundary condition for a deformation or a flow problem

RESULTS SLIDE

Exercise – Temperature distribution across a thin plate

Consider the following differential equation:

$$\frac{d^2T}{dx^2} = \alpha(T - T_s) \quad \text{or} \quad \frac{d^2T}{dx^2} - \alpha(T - T_s) = 0$$

in the domain x ∈ $(0, 0.1)$

with the conditions: $T(0) = 473[K]$, $T(0.1) = 293[K]$

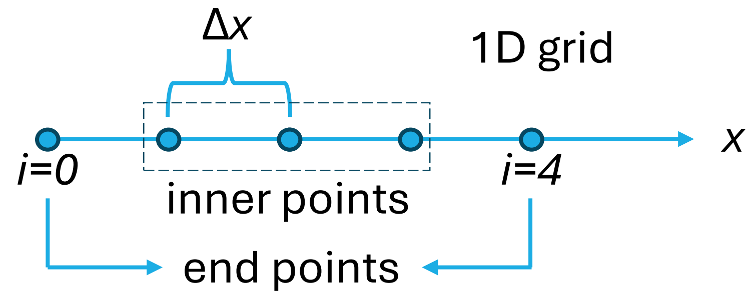
What type of boundary conditions are these?

Central Difference scheme:

$$f''(x_0) \approx \frac{f(x_0 - \Delta x) - 2f(x_0) + f(x_0 + \Delta x)}{(\Delta x)^2}$$

or

$$f''(x_i) \approx \frac{f(x_{i-1}) - 2f(x_i) + f(x_{i+1})}{(\Delta x)^2}$$



Elaborate the general numerical equation in T_{i-1} , T_i , T_{i+1} according to the Central Difference scheme.

Elaborate the scheme further using 3 inner grid points and add the boundary conditions.
(i.e. consider i = 1, 2, 3 in the equation and apply the boundary conditions at the end points i = 0, 4)

Formulate the result as a matrix-vector system.

Exercise – Temperature distribution across a thin plate

$$\frac{T_{i-1} - 2T_i + T_{i+1}}{\Delta x^2} - \alpha(T_i - T_s) = 0 \quad \Leftrightarrow \quad T_{i-1} - 2T_i + T_{i+1} - \alpha\Delta x^2(T_i - T_s) = 0$$

$$T_{i-1} - (2 + \alpha\Delta x^2)T_i + T_{i+1} = \alpha\Delta x^2 T_s$$

$$\begin{aligned}
T_0 &= T(0)\\
T_0 - (2+\alpha\Delta x^2)T_1 + T_2 &= \alpha\Delta x^2 T_s\\
T_1 - (2+\alpha\Delta x^2)T_2 + T_3 &= \alpha\Delta x^2 T_s\\
T_2 - (2+\alpha\Delta x^2)T_3 + T_4 &= \alpha\Delta x^2 T_s\\
T_4 &= T(0.1)
\end{aligned}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & -(2+\alpha\Delta x^2) & 1 & 0 & 0 \\ 0 & 1 & -(2+\alpha\Delta x^2) & 1 & 0 \\ 0 & 0 & 1 & -(2+\alpha\Delta x^2) & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} T_0 \\ T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} T(0) \\ \alpha\Delta x^2 T_s \\ \alpha\Delta x^2 T_s \\ \alpha\Delta x^2 T_s \\ T(0.1) \end{bmatrix}$$

What do you notice? Why?

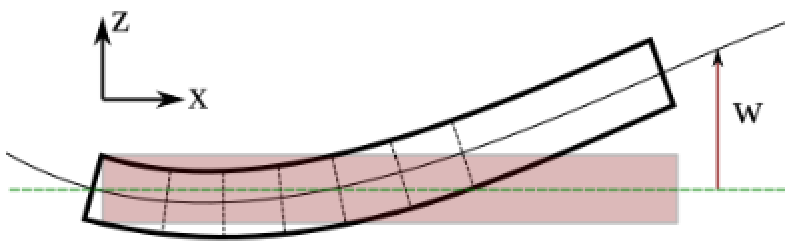
```
M = np.zeros([gridpoints, gridpoints])
RHS = np.zeros(gridpoints)

for i in range(1, gridpoints-1):
    M[i, i-1] = -1.
    M[i, i  ] = (2 + alfa*dx*dx)
    M[i, i+1] = -1.
    RHS[i] = -alfa*dx*dx*Ts
M[0, 0] = 1.
M[gridpoints-1, gridpoints-1] = 1.
RHS[0] = 473.
RHS[gridpoints-1] = 293.

T = np.linalg.solve(M, RHS)
```

Example: Euler-Bernoulli beam deflection

$$\frac{d^2M}{dx^2} = -q$$ and $$M = -EI\frac{d^2w}{dx^2}$$



M = bending moment
w = beam deflection (lateral displacement)
EI = flexural rigidity (bending stiffness)
q = distributed load

Central difference scheme for both equations:

$$\frac{M_{i-1} - 2M_i + Mi + 1}{\Delta x^2} = -q_i$$ or $$\boxed{M_{i-1} - 2M_i + M_{i+1} = -\Delta x^2 q_i}$$

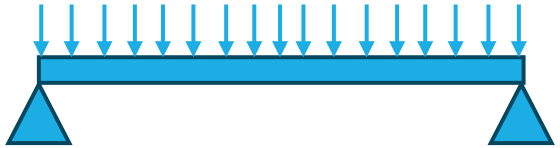
$$-\frac{M_i}{EI} = \frac{w_{i-1} - 2w_i + w_{i+1}}{\Delta x^2}$$ or $$\boxed{w_{i-1} - 2w_i + w_{i+1} = -\frac{\Delta x^2}{EI}M_i}$$

Example: Euler-Bernoulli beam deflection

Consider a simply supported beam at both ends with the following boundary conditions:

at x = 0: w = 0 and M = 0

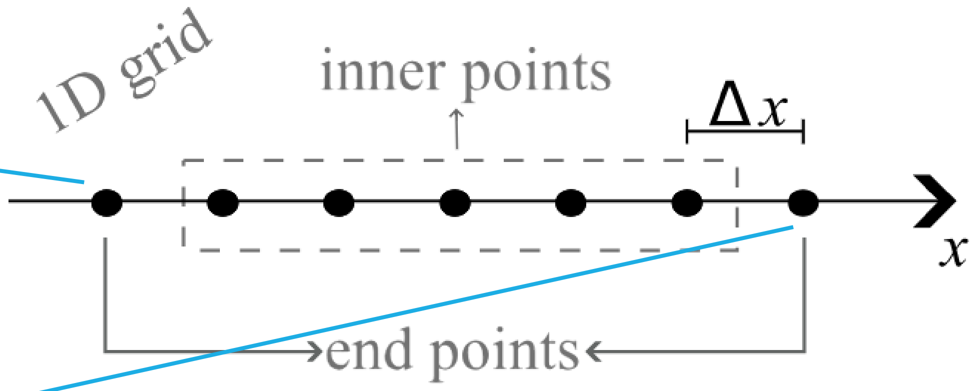
at x = L: w = 0 and M = 0



First, solve the bending moments: $\boxed{M_{i-1} - 2M_i + M_{i+1} = -\Delta x^2 q_i}$

This gives a system of equations: $\boldsymbol{A}\,M = f$ => $M = \boldsymbol{A}^{-1} f$ (in Python: *M* = numpy.linalg.solve(*A*,*f*))

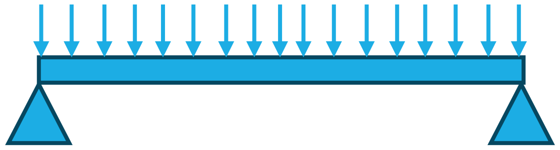
$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & & & & \\ 1 & -2 & 1 & 0 & & & 0 & \\ 0 & 1 & -2 & 1 & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & 1 & -2 & 1 & 0 & \\ & 0 & & 0 & 1 & -2 & 1 & \\ & & & 0 & 0 & 0 & 1 & \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} \\ \\ M_i \\ \\ \\ \end{bmatrix}}_{M} = \underbrace{\begin{bmatrix} 0 \\ \\ -\Delta x^2 q_i \\ \\ 0 \end{bmatrix}}_{f}$$



Boundary conditions

Example: Euler-Bernoulli beam deflection

Consider a simply supported beam at both ends with the following boundary conditions:



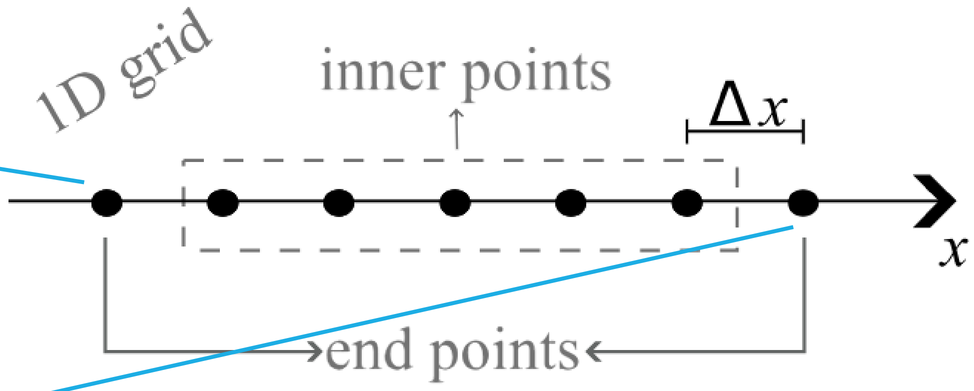
at x = 0: w = 0 and M = 0

at x = L: w = 0 and M = 0

Then, solve the beam deflections: $$\boxed{w_{i-1} - 2w_i + w_{i+1} = -\frac{\Delta x^2}{EI} M_i}$$

This gives a system of equations: ***A*** *w* = *f* => *w* = ***A***$^{-1}$ *f* (in Python: w = numpy.linalg.solve(A,f))

$$\begin{array}{cccc} A & w & & f \\ \begin{bmatrix} 1 & 0 & 0 & 0 & & & \\ 1 & -2 & 1 & 0 & & 0 & \\ 0 & 1 & -2 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & 1 & -2 & 1 & 0 \\ & 0 & & 0 & 1 & -2 & 1 \\ & & & 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} \\ \\ \\ w_i \\ \\ \\ \\ \end{bmatrix} & = & \begin{bmatrix} 0 \\ \\ \\ -\Delta x^2 M_i / EI \\ \\ \\ 0 \end{bmatrix} \end{array}$$



Boundary conditions

Example: Euler-Bernoulli beam deflection

Alternatively, both differential equations can be solved simultaneously:

$$\boxed{M_{i-1} - 2M_i + M_{i+1} = -\Delta x^2 q_i}$$

$$\boxed{w_{i-1} - 2w_i + w_{i+1} = -\frac{\Delta x^2}{EI} M_i}$$

$$\Downarrow$$

$$\frac{\Delta x^2}{EI} M_i + w_{i-1} - 2w_i + w_{i+1} = 0$$

(RHS term is now integrated in the matrix)

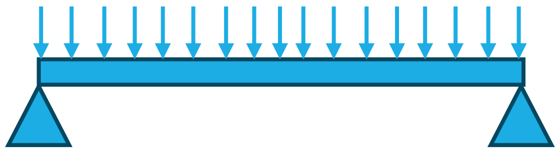
Example: Euler-Bernoulli beam deflection

Alternatively, both differential equations can be solved simultaneously:

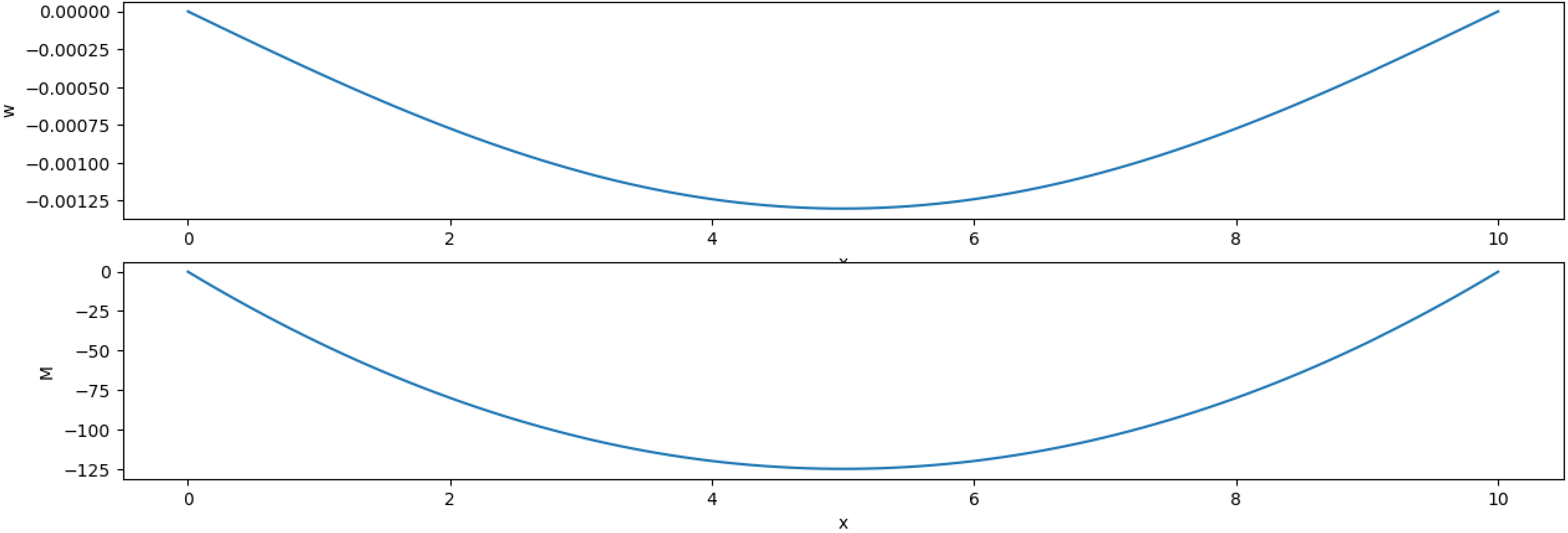
Boundary conditions

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & & & & & & & & \\
1 & -2 & 1 & 0 & & 0 & & & & & & \\
0 & 1 & -2 & 1 & & & & & & & & \\
 & & \ddots & \ddots & \ddots & & & & & 0 & & \\
 & & & 1 & -2 & 1 & 0 & & & & & \\
 & 0 & & 0 & 1 & -2 & 1 & & & & & \\
 & & & 0 & 0 & 0 & 1 & & & & & \\
0 & & & & & & & 1 & & & & \\
 & \Delta x^2/EI & & & & & & 1 & -2 & 1 & & \\
 & & \Delta x^2/EI & & & & & & 1 & -2 & 1 & \\
 & & & \ddots & & & & & \ddots & \ddots & \ddots & \\
 & & & & \Delta x^2/EI & & & & & 1 & -2 & 1 \\
 & & & & & \Delta x^2/EI & & & & & 1 & -2 & 1 \\
 & & & & & & 0 & & & & & & 1
\end{bmatrix}
\begin{bmatrix}
\\ \\ M_i \\ \\ \\ \\ \\ w_i \\ \\ \\
\end{bmatrix}
=
\begin{bmatrix}
0 \\ \\ -\Delta x^2 q_i \\ \\ 0 \\ 0 \\ \\ 0 \\ \\ 0
\end{bmatrix}
$$

Example: Euler-Bernoulli beam deflection



Results for L=10m, EI=10^6 kNm^2, q=-10 kN/m^2, Δx=0.1



Summary

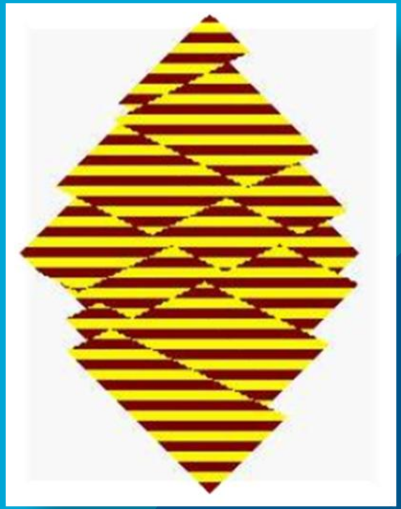
- Differential equations:
  - Initial Value Problems
  - Boundary Value Problems
- Numerical solutions:
  - Explicit Euler Stability criterion!
  - Implicit Euler
  - Example: Falling head test
- Non-linear ODE: Iterative solution (Newton-Raphson):
- Multiple-step and multi-stage schemes
  - Heun
  - Runge-Kutta
- Second order ODEs
  - Boundary Value Problems
  - Example: Bending beam

What you can expect further this week…

- Programming Assignment 1.3:
  - Coding collaboration using VS Live Share
  - Code completion using IntelliSense
  - Programming: Matrix and vector manipulations

- Wednesday 10:45-12:30: Workshop 1.3:
  - Solving initial value problem: Falling head test (hydraulic conductivity)
  - Solving boundary value problem: Bending beam
  - Solving two coupled ODEs in one system

- Friday 9:45-12:30: Group Assignment 1.3:
  - Solving non-linear ODE (initial value problem) using Newton-Raphson iterations: Hydrological discharge model



Join us for the information session on
the **Geotechnical Engineering** track
Tuesday 16 September 12:45-13:30 room D

25
sept

Geodrinks XL
(FREE pizza & drinks)

17:00 in exhibition room (glass part 1st floor)

TU Delft

Ronald Brinkgreve, Anna Störiko, Jaime Arriaga

r.b.j.brinkgreve@tudelft.nl