# Numerical Simulation Methods in Geophysics, Part 5: Finite Elements

1. MGPY+MGIN

*thomas.guenther@geophysik.tu-freiberg.de*

TU Bergakademie Freiberg
TUBAF
Die Ressourcenuniversität. Seit 1765.

# Recap Finite Differences

- elliptic (Poisson) or parabolic PDE problems
- replace partial differential operators $\partial$ by finite differences $\Delta$
- transfer PDE into a matrix-vector equation $\mathbf{Au} = \mathbf{b}$
- finite-difference stencil spatial or temporal
- spatial derivative $\Rightarrow$ system matrix $\mathbf{A}$, temporal $\Rightarrow$ identity matrix $\mathbf{I}$
- time-stepping explicit, implicit or mixed (stable & accurate)
- accuracy depends on discretization & parameter contrast

# The Finite Element Method

# History and background

- [1943] Courant: Variational Method
- [1956] Turner, Clough, Martin, Topp: Stiffness
- [1960] Clough: Finite Elements for static elasticity
- [1970-80] extension to structural, thermic and fluid dynamics
- [1990] computational improvements
- now main method for almost all PDE types

Geophysics: Poisson equation in 1970s, revival in 1990s and predominant from 2000s up to now

# Variational formulation of Poisson equation

$$-\boldsymbol{\nabla} \cdot a \boldsymbol{\nabla} u = f$$

Multiplication with test function $w$ and integration $\Rightarrow$ weak form

$$-\int_\Omega w \boldsymbol{\nabla} \cdot a \boldsymbol{\nabla} u \mathrm{d}\Omega = \int_\Omega w f \mathrm{d}\Omega$$

$$\boldsymbol{\nabla} \cdot (b\mathbf{c}) = b \boldsymbol{\nabla} \cdot \mathbf{c} + \boldsymbol{\nabla} b \cdot \mathbf{c}$$

$$\int_\Omega a \boldsymbol{\nabla} w \cdot \boldsymbol{\nabla} u \mathrm{d}\Omega - \int_\Omega \boldsymbol{\nabla} \cdot (w a \boldsymbol{\nabla} u) \mathrm{d}\Omega = \int_\Omega w f \mathrm{d}\Omega$$

# Variational formulation of Poisson equation

$$\int_\Omega a\boldsymbol{\nabla} w \cdot \boldsymbol{\nabla} u \mathrm{d}\Omega - \int_\Omega \boldsymbol{\nabla} \cdot (wa\boldsymbol{\nabla} u)\mathrm{d}\Omega = \int_\Omega wf\mathrm{d}\Omega$$

use Gauss' law $\int_\Omega \boldsymbol{\nabla} \cdot \mathbf{A} = \int_\Gamma \mathbf{A} \cdot \mathbf{n}$

$$\int_\Omega a\boldsymbol{\nabla} w \cdot \boldsymbol{\nabla} u \mathrm{d}\Omega - \int_\Gamma aw\boldsymbol{\nabla} u \cdot \mathbf{n}\mathrm{d}\Gamma = \int_\Omega fw\mathrm{d}\Omega$$

Let $u$ be constructed by shape functions $v$: $u = \sum_i u_i v_i$

$$\int_\Omega a\boldsymbol{\nabla} w \cdot \boldsymbol{\nabla} v_i \mathrm{d}\Omega - \int_\Gamma aw\boldsymbol{\nabla} v_i \cdot \mathbf{n}\mathrm{d}\Gamma = \int_\Omega fw\mathrm{d}\Omega$$

# Galerkin's method

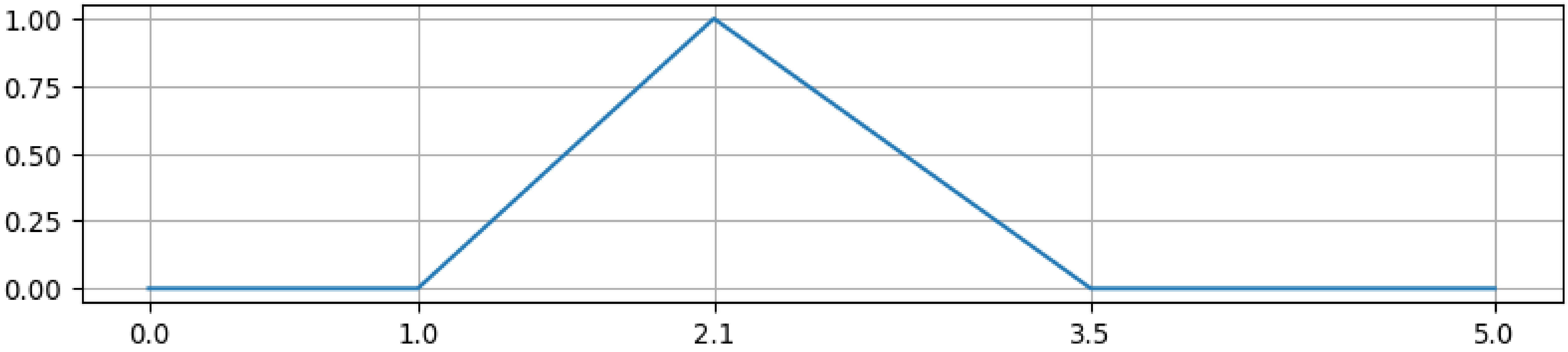
$$\int_\Omega a\boldsymbol{\nabla} w \cdot \boldsymbol{\nabla} v_i \mathrm{d}\Omega - \int_\Gamma a w \boldsymbol{\nabla} v_i \cdot \mathbf{n}\mathrm{d}\Gamma = \int_\Omega f w \mathrm{d}\Omega$$

Test functions the same as shape (trial) functions $w \in v_i$

$$\int_\Omega a\boldsymbol{\nabla} v_j \cdot \boldsymbol{\nabla} v_i \mathrm{d}\Omega - \int_\Gamma a v_j \boldsymbol{\nabla} v_i \cdot \mathbf{n}\mathrm{d}\Gamma = \int_\Omega f v_j \mathrm{d}\Omega$$

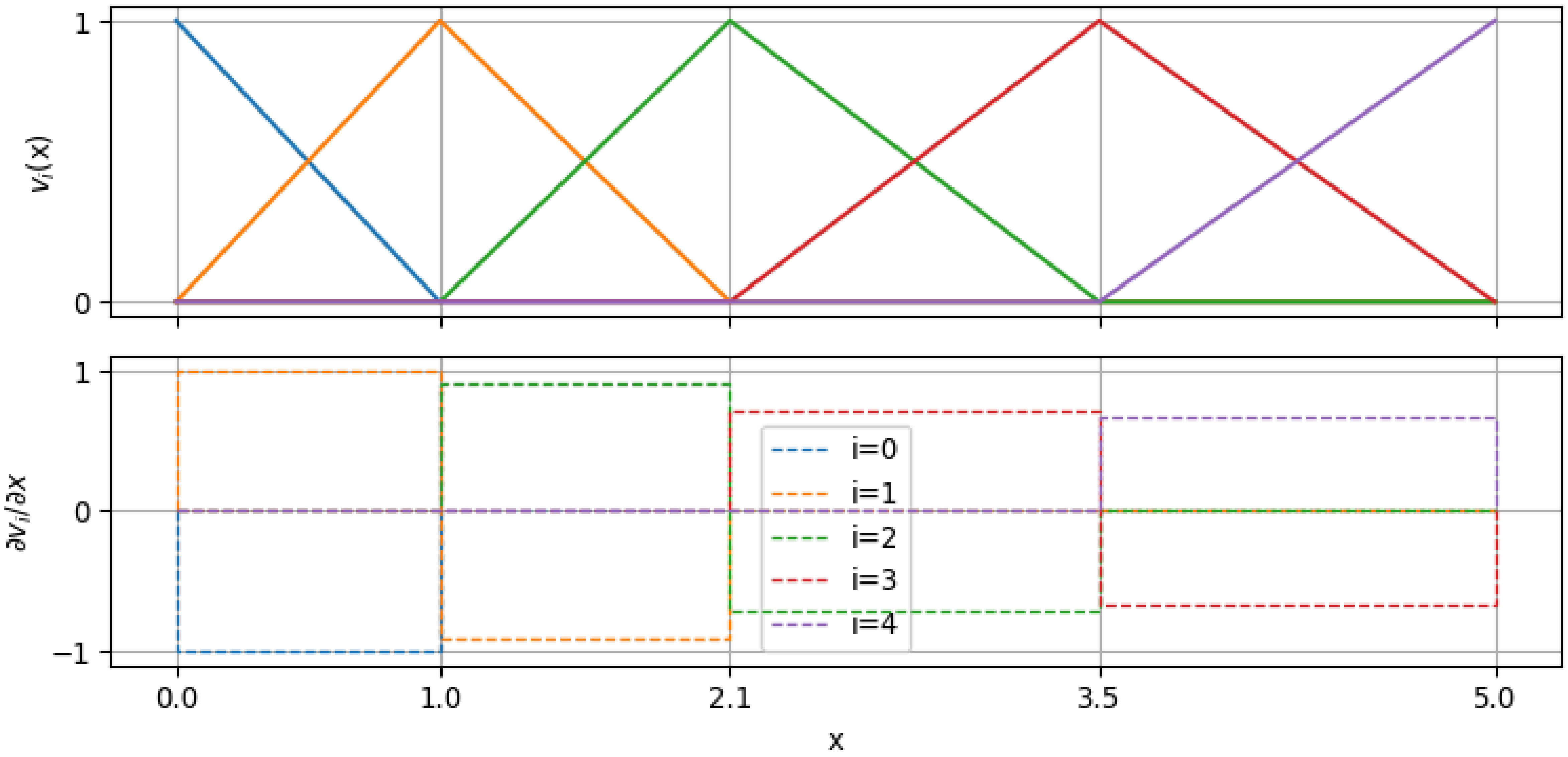
- choose $v_i$ so that $\boldsymbol{\nabla} v_i$ is simple and $\boldsymbol{\nabla} v_i \cdot \boldsymbol{\nabla} v_j$ mostly 0
- divide subsurface in sub-volumes $\Omega_i$ with constant $a_i$ (& $\boldsymbol{\nabla} v_j$)

# FE for 1D Poisson PDE




Every node carries a hat

# 1st-order nodal shape functions (hats)



# Gradients for hat functions

- every element is surrounded by two nodes “carrying” a hat.
- the gradients $v_i'$ are piece-wise constant $\pm 1/\Delta x_i$
- neighboring functions $v_i$ & $v_{i+1}$ only meet between $x_i$ & $x_{i+1}$

$$\int_\Omega a \boldsymbol{\nabla} v_i \cdot \boldsymbol{\nabla} v_{i+1} \mathrm{d}\Omega = \int_{x_i}^{x_{i+1}} a_i v_i' v_{i+1}' \mathrm{d}\Omega = -\frac{a_i}{\Delta x_i^2} \Delta x_i = -\frac{a_i}{\Delta x_i}$$

$$\int_{x_{i-1}}^{x_{i+1}} a v_i' v_i' \mathrm{d}\Omega = \frac{a_{i-1}}{\Delta x_{i-1}^2} \Delta x_{i-1} + \frac{a_i}{\Delta x_i^2} \Delta x_i = \frac{a_{i-1}}{\Delta x_{i-1}} + \frac{a_i}{\Delta x_i}$$

# Integration

Let's write the equation for the first and second nodes in 1D

$$\int_{x_0}^{x_1} u_0 a_0 v_0' v_0' + \int_{x_0}^{x_1} u_1 a_1 v_0' v_1' = \int_{x_0}^{x_1} v_0 f$$

$$\int_{x_0}^{x_1} u_0 a v_0' v_1' + \int_{x_0}^{x_2} u_1 a v_1' v_1' + \int_{x_1}^{x_1} a u_2 v_2' v_1' = \int_{x_0}^{x_2} v_1 f$$

$$u_{i-1} a_{i-1} \int\limits_{x_{i-1}}^{x_i} v_i' v_{i-1}' + u_i a_{i-1} \int\limits_{x_{i-1}}^{x_i} v_i' v_i' + u_i a_i \int\limits_{x_i}^{x_{i+1}} v_i' v_i' + u_{i+1} a_i \int\limits_{x_i}^{x_{i+1}} v_i' v$$

# The stiffness matrix

Matrix integrating gradients of base functions for neighbors with $a$

$$\mathbf{A}_{i,i+1} = -\frac{a_i}{\Delta x_i^2} \cdot \Delta x_i = -\frac{a_i}{\Delta x_i}$$

$$A_{i,i} = \int_\Omega a \boldsymbol{\nabla} v_i \cdot \boldsymbol{\nabla} v_i \mathrm{d}\Omega = -A_{i,i+1} - A_{i+1,i}$$

$\Rightarrow$ matrix-vector equation $\mathbf{Au} = \mathbf{b}$ with bending&shear stiffness in $\mathbf{A}$

# Boundary conditions

second term

$$-\int_\Gamma a v_j \boldsymbol{\nabla} v_i \cdot \mathbf{n} \mathrm{d}\Gamma$$

reads in 1D as

$$[a v_i v_j']_{x_0}^{x_N} = a_{N-1} u_N v_N' - a_0 u_0 v_0'$$

$\Rightarrow$ Homogeneous Neumann BC ($v_0' = 0$) are automatically implemented

# Right-hand side vector

The right-hand-side vector $b = \int v_i f \mathrm{d}\Omega$ also scales with $\Delta x$

e.g. $f = \boldsymbol{\nabla} \cdot \mathbf{j}_s \Rightarrow b = \int v_i \boldsymbol{\nabla} \cdot \mathbf{j}_s \mathrm{d}\Omega = \int_\Gamma v_i \mathbf{j}_S \cdot \mathbf{n}$

(system identical to FD for $\Delta x$=1)

**Difference of FE to FD**

Any source function $f(x)$ can be integrated on the whole space!

# Solution

$\mathbf{u}$ holds the coefficient $u_i$ creating $u(x) = \sum u_i v_i(x)$

**Difference of FE to FD**

$u$ is described on the whole space and approximates the solution, not the PDE!

Hat functions: $u_i$ potentials on nodes, $u$ piece-wise linear

**Generality of FE**

Arbitrary base functions $v_i$ can be used to describe $u$

# Method of weighted residuals

PDE $\mathfrak{L}(u) = f \Rightarrow$ approximated by $u_h$

residual $R = L_h(u) - f$ to be minimized, integrating over modelling domain

$$\int_\Omega wR\mathrm{d}\Omega = \int_\Omega w\mathfrak{L}(u_h)\mathrm{d}\Omega - \int_\Omega wf\mathrm{d}\Omega = 0$$

with approximation $u_h(\mathbf{r}) = \sum_j^M u_j \mathbf{v}_j(\mathbf{r})$

($\mathbf{v}$ basis / shape functions, $\mathbf{w}$ test / trial functions)

# Bilinear form for Poisson equation

Solve $\mathbf{Ax} = \mathbf{b}$ with $A_{ij} = (\boldsymbol{\nabla} v_i, a\boldsymbol{\nabla} v_j)$ and $b_i = (\mathbf{v}_i, f)$, where

$$(\mathbf{a}, \mathbf{b}) = \int_\Omega \mathbf{a} \cdot \mathbf{b} \, \mathrm{d}\Omega = \sum_{c=i}^{M} \int_{\Omega_c} \mathbf{a} \cdot \mathbf{b} \, \mathrm{d}\Omega_c$$

Solve the integrals either analytically or numerically

# Coordinate transformation

1D: local coordinate $\xi = \frac{x - x_i}{x_{i+1} - x_i}$ (0..1)

$$u(\xi) = c_1 + c_2\xi$$

$$u_0 = u(0) = c_1, u_1 = u(1) = c_1 + c_2 \Rightarrow c_2 = u_1 - u_0$$

$$\Rightarrow u(\xi) = u_0 + \xi(u_1 - u_0) = u_0(1 - \xi) + u_1\xi = u_0 v_0 + u_1 v_1$$

# Quadratic elements

$$u(\xi) = c_1 + c_2\xi + c_3\xi^2$$

nodes at $x_0$, $x_{1/2}$, $x_1$

$$u_i = u(0) = c_1,\, u_1 = c_1 + c_2 + c_3$$

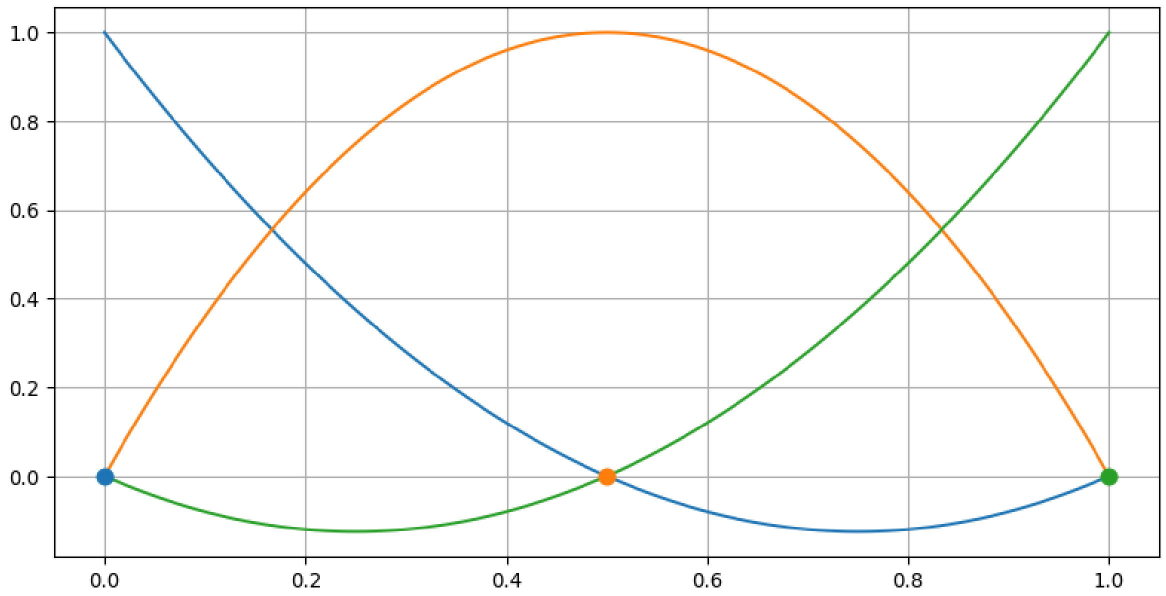
$$u_{1/2} = c_1 + c_2/2 + c_3/4$$

$$u(\xi) = u_0(1 - 3\xi + 2\xi^2) + u_{1/2}(4\xi - 4\xi^2) + u_1(-\xi + 2\xi^2)$$

# Quadratic elements

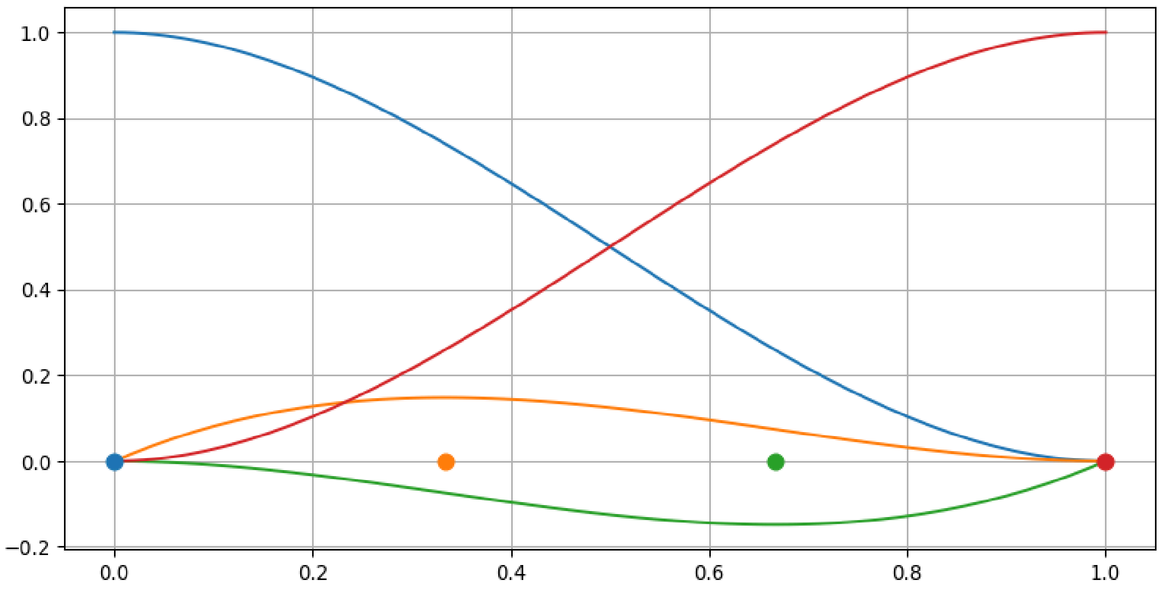
$$u(\xi) = u_0(3\xi + 2\xi^2) + u_{1/2}(4\xi - 4\xi^2) + u_1(-\xi + 2\xi^2)$$

```python
import numpy as np
import matplotlib.pyplot as plt
x=np.linspace(0, 1, 101)

# Plot velocity distribution.
plt.plot(x, 1-3*x+2*x**2)
plt.plot(x, 4*x-4*x**2)
plt.plot(x, -x+2*x**2)
plt.plot(0, 0, "o", color="C0", ms=8)
plt.plot(0.5, 0, "o", color="C1", ms=8)
plt.plot(1, 0, "o", color="C2", ms=8)
plt.grid()
```



# Cubic elements

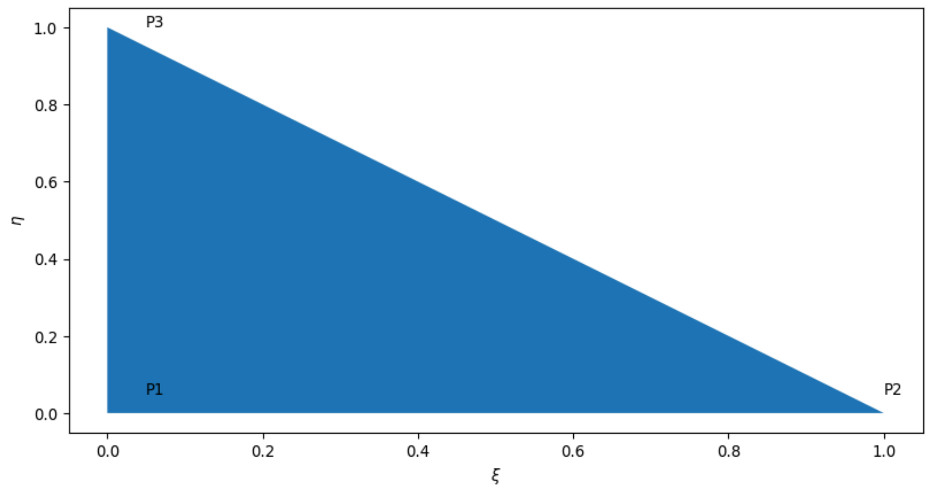
```python
import numpy as np
import matplotlib.pyplot as plt
x=np.linspace(0, 1, 101)

# Plot velocity distribution.
plt.plot(x, 1-3*x**2+2*x**3)
plt.plot(x, x-2*x**2+x**3)
plt.plot(x, -x**2+x**3)
plt.plot(x, 3*x**2-2*x**3)
for i in range(4):
    plt.plot(i/(3), 0, "o",
             color=f"C{i}", ms=8)
plt.grid()
```



# Triangles with linear shape functions

$$x = x_1 + (x_2 - x_1)\xi + (x_3 - x_1)\eta$$

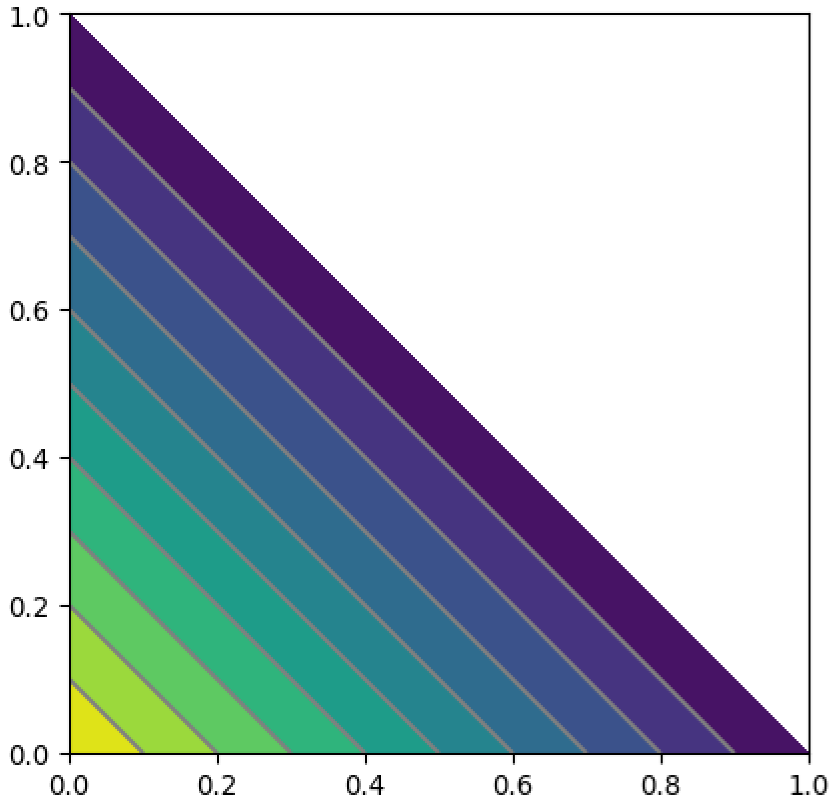
$$y = y_1 + (y_2 - y_1)\xi + (y_3 - y_1)\eta$$



# Triangle

$$u(\xi) = u_1(1-\xi-\eta) + u_2\xi + u_3\eta$$

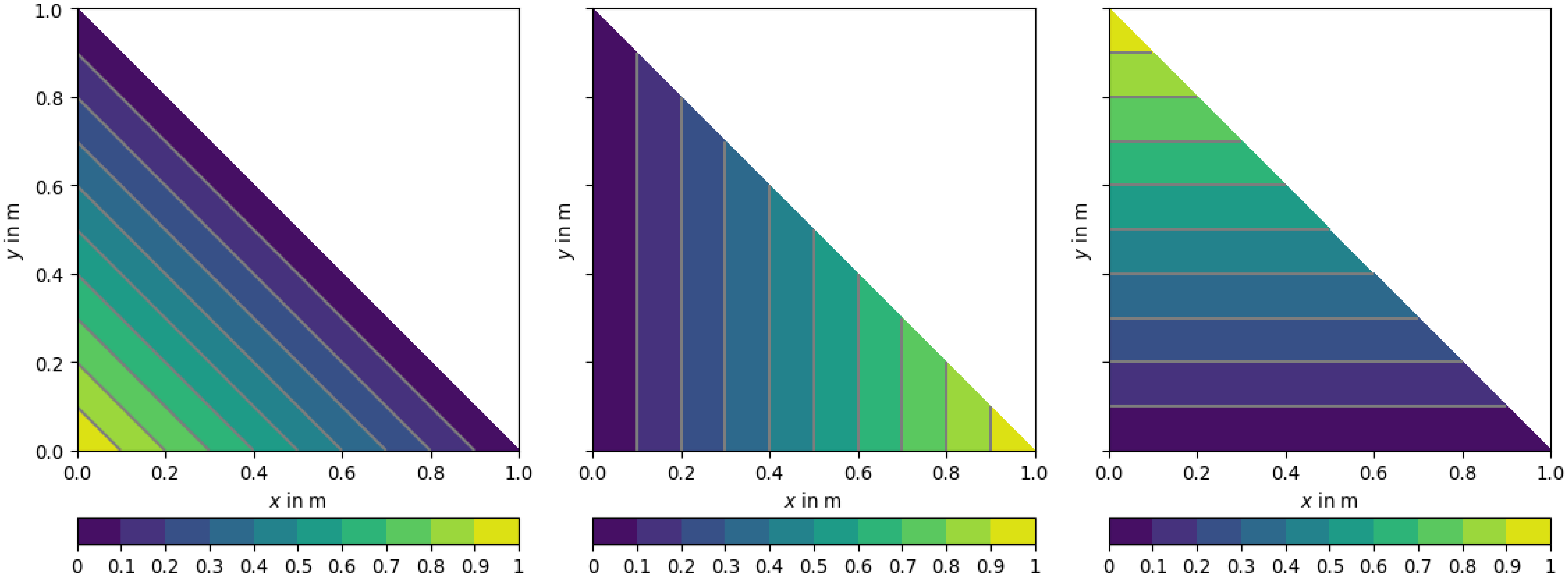
```
import pygimli as pg
import pygimli.meshtools as mt

shape = mt.createPolygon(
    [[0, 0], [1, 0],[0, 1]],
    isClosed=True)
mesh = mt.createMesh(shape, area=0.01)
mx = pg.x(mesh)
my = pg.y(mesh)
# Plot velocity distribution.
fig, ax = plt.subplots()
pg.show(mesh, 1-mx-my, ax=ax,
        nLevs=11, label="u");
```




<Figure size 960x480 with 0 Axes>

# Triangle linear shape functions

$$u(\xi) = u_1(1 - \xi - \eta) + u_2\xi + u_3\eta$$



# The general solution

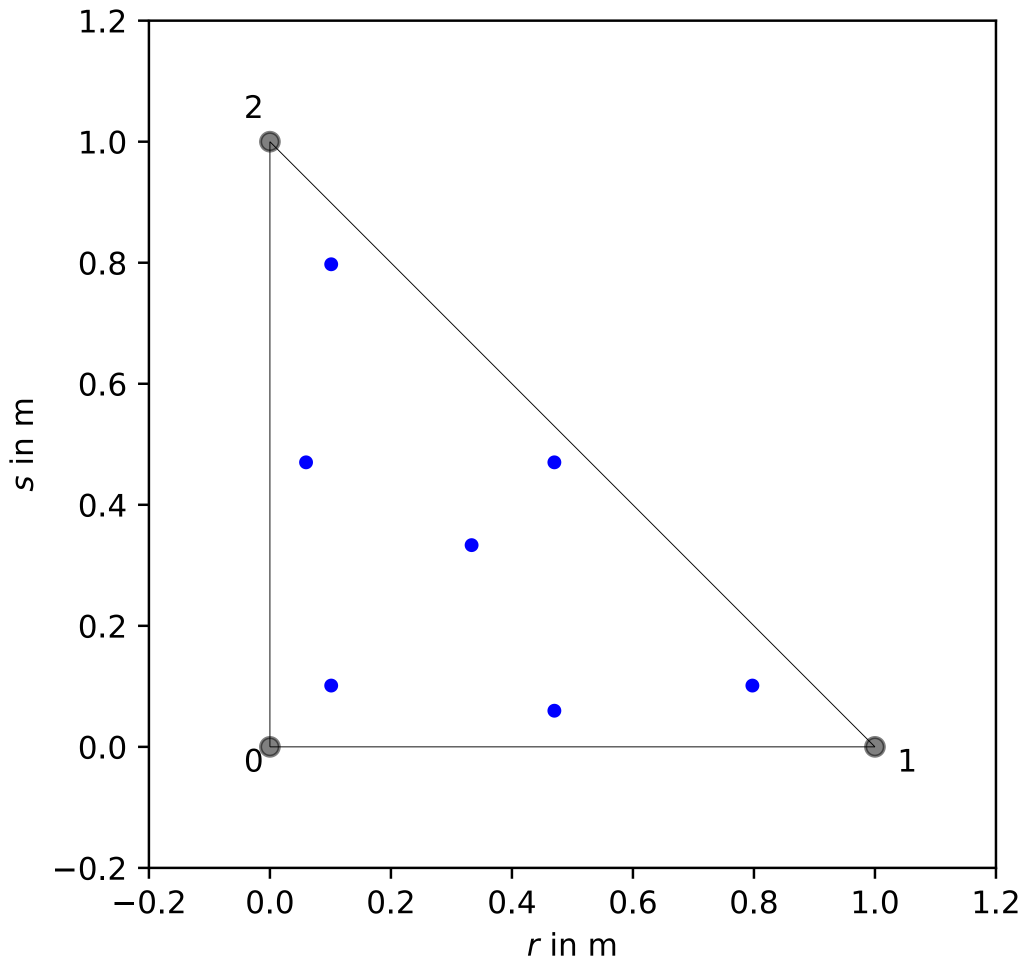
Solving any integral using (Gaussian) quadrature

$$\int g(x)\mathrm{d}x \approx \sum_q g(x_q)w_q$$

$$f_i^c = \int_{\Omega_c} v_i f \mathrm{d}x \approx \sum_q v(x_q^c) f(x_q^c) w_q^c$$

$$a_{ij}^c = \int_c a_c \nabla v_i \cdot \nabla v_j = \sum a_c \nabla v_i(x_q^c) \cdot \nabla v_j(x_q^c) w_q^c$$

# Gaussian quadrature

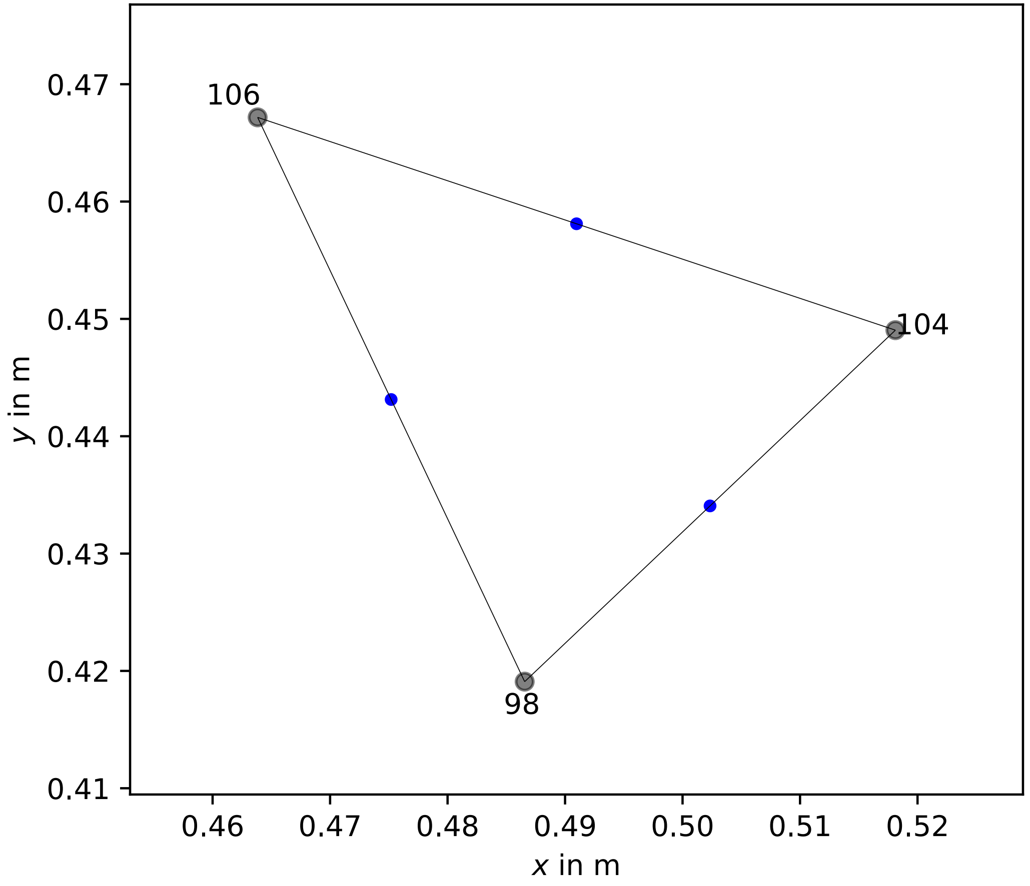



Quadrature points

```
quadratureRules(c.shape(), 5)
```

- optimum quadrature on reference triangle for a given order (5)

# Gaussian quadrature




Quadrature points

```
quadratureRules(c, 2)
```

- optimum quadrature on arbitrary triangle for order 2

# Verification

1. Method of Manufactured Solutions (MMS)

- manufacture a smooth u
- generate $f$ matching approximation of $u$

2. Method of Exact Solutions (MES)

- find parameters for which an analytic solution exists

3. Perform convergence tests for increasingly smaller $h$

- approximation error $E(h) < Ch^n$ test for some $h$

# Green's functions

The Green's function $G$ is the solution for a Dirac source $\delta$

$$\mathfrak{L}G = \delta(\mathbf{r})$$

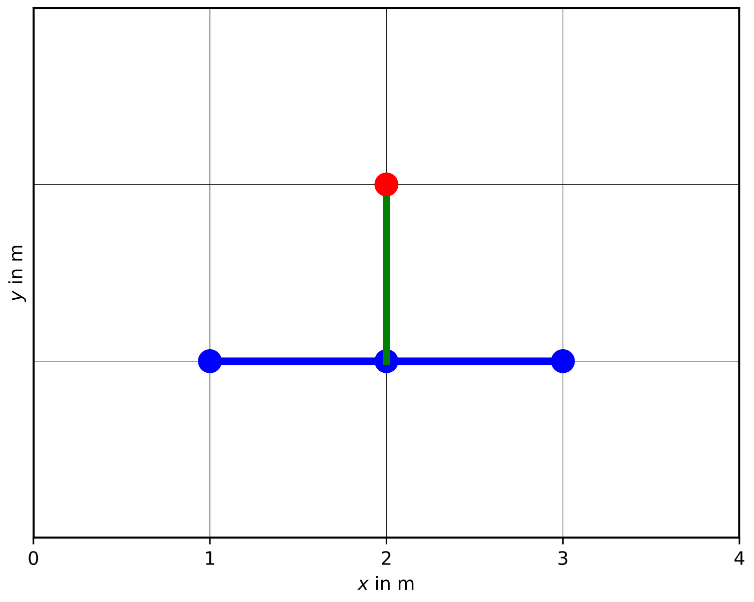
The solution can then be obtained by convolution

$$u(\mathbf{r}) = G(\mathbf{r} - \mathbf{r}') * f(\mathbf{r}') =$$

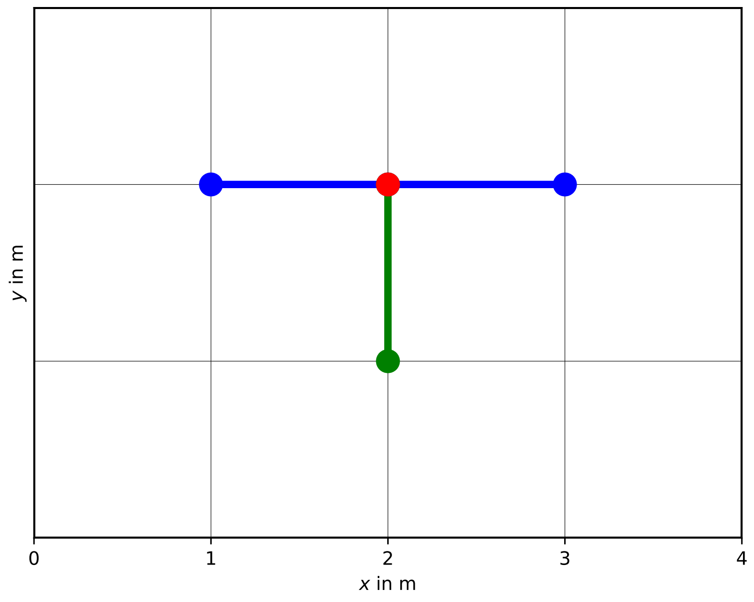
# Time-stepping in FE

# Recap time-stepping in FD

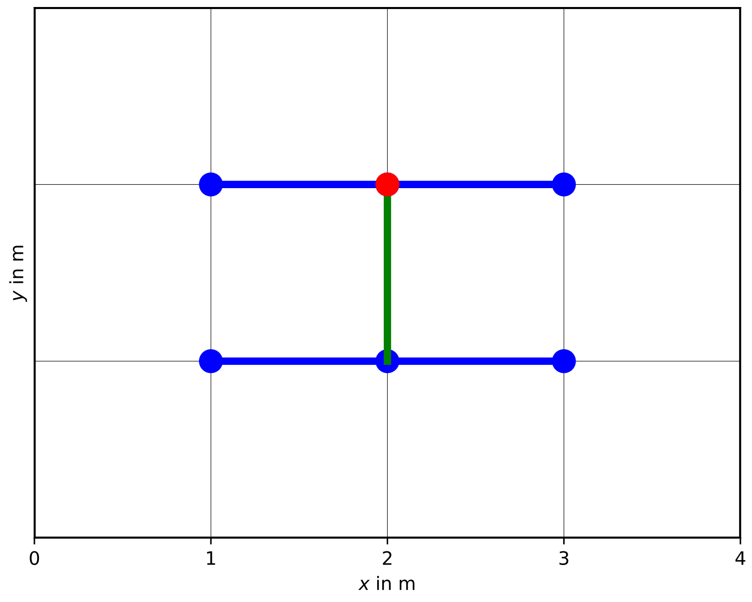
| | | | | |
|---|---|---|---|---|
| **Explicit:** | | $\mathbf{u}^{n+1}$ | $=$ | $(\mathbf{I} - \Delta t\mathbf{A}) \quad \mathbf{u}^{n}$ |
| **Implicit:** | $(\mathbf{I} + \Delta t\mathbf{A})$ | $\mathbf{u}^{n+1}$ | $=$ | $\mathbf{u}^{n}$ |
| **Mixed:** | $(2\mathbf{I} + \Delta t\mathbf{A})$ | $\mathbf{u}^{n+1}$ | $=$ | $(2\mathbf{I} - \Delta t\mathbf{A}) \quad \mathbf{u}^{n}$ |




Explicit




Implicit




Mixed

# Variational formulation of Diffusion equation

$$\frac{\partial u}{\partial t} - \boldsymbol{\nabla} \cdot a \boldsymbol{\nabla} u = f$$

Finite Difference in Time (NOT in space)

$$\frac{u^{n+1} - u^n}{\Delta t} - \boldsymbol{\nabla} \cdot a \boldsymbol{\nabla} u = f$$

# Variational formulation

$$\frac{u^{n+1} - u^n}{\Delta t} - \boldsymbol{\nabla} \cdot a \boldsymbol{\nabla} u = f$$

Multiplication with test function $w$ and integration $\Rightarrow$ weak form

$$1/\Delta t\left(\int_\Omega w u^{n+1} \mathrm{d}\Omega - \int_\Omega w u^n \mathrm{d}\Omega\right) - \int_\Omega w \boldsymbol{\nabla} \cdot a \boldsymbol{\nabla} u \mathrm{d}\Omega = \int_\Omega w f \mathrm{d}\Omega$$

$$1/\Delta t\left(\int_\Omega w u^{n+1} \mathrm{d}\Omega - \int_\Omega w u^n \mathrm{d}\Omega\right) - \int_\Omega a \boldsymbol{\nabla} w \cdot \boldsymbol{\nabla} u \mathrm{d}\Omega = \int_\Omega w f \mathrm{d}\Omega$$

# Variational formulation of diffusion equation

$u$ is constructed of shape functions $\mathbf{v}_i$ that are identical to $w$

The integral over the Poisson term

$$-\int_\Omega a\boldsymbol{\nabla} w \cdot \boldsymbol{\nabla} u \mathrm{d}\Omega$$

is represented by $\mathbf{A}\mathbf{v}$ using the stiffness matrix

$$\mathbf{A}_{i,j} = \int_\Omega \sigma\boldsymbol{\nabla} v_i \cdot \boldsymbol{\nabla} v_j \mathrm{d}\Omega$$

# Variational formulation of diffusion equation

Weighted integrals over both $u$ are represented by the mass matrix $\mathbf{Mv}$

$$\mathbf{M}_{i,j} = \int_{\Omega} v_i \cdot v_j \mathrm{d}\Omega$$

explicit method (use $u^n$): $\mathbf{M}\mathbf{u}^{n+1} = (\mathbf{M} - \Delta t \mathbf{A})\mathbf{u}^n$

implicit method (use $u^{n+1}$): $(\mathbf{M} + \Delta t \mathbf{A})\mathbf{u}^{n+1} = \mathbf{M}\mathbf{u}^n$

mixed method (mix $u^n$/$u^{n+1}$):
$(2\mathbf{M} + \Delta t \mathbf{A})\mathbf{u}^{n+1} = (2\mathbf{M} - \Delta t \mathbf{A})\mathbf{u}^n$

# Time-stepping in FE

| | | | | | |
|---|---|---|---|---|---|
| **Explicit:** | $\mathbf{M}$ | $\mathbf{u}^{n+1}$ | $=$ | $(\mathbf{M} - \Delta t\mathbf{A})$ | $\mathbf{u}^{n}$ |
| **Implicit:** | $(\mathbf{M} + \Delta t\mathbf{A})$ | $\mathbf{u}^{n+1}$ | $=$ | $\mathbf{M}$ | $\mathbf{u}^{n}$ |
| **Mixed:** | $(2\mathbf{M} + \Delta t\mathbf{A})$ | $\mathbf{u}^{n+1}$ | $=$ | $(2\mathbf{M} - \Delta t\mathbf{A})$ | $\mathbf{u}^{n}$ |

**Tip**

same as in FD but with FE mass matrix $\mathbf{M}$ instead of $\mathbf{I}$

# The mass matrix in 1D

$$\mathbf{M}_{i,j} = \int_\Omega v_i \cdot v_j \mathrm{d}\Omega$$

$$\mathbf{M}_{i,i+1} = \int_{x_i}^{x_{i+1}} \frac{x_{i+1} - x}{\Delta x_i} \frac{x - x_i}{\Delta x_i} \mathrm{d}x = \int_0^1 (1-\xi)\xi \Delta x_i \mathrm{d}\xi$$

$$\Rightarrow \mathbf{M}_{i,i+1} = \Delta x_i \int_0^1 (\xi - \xi^2) = \Delta x_i \left| \frac{1}{2}\xi^2 - \frac{1}{3}\xi^3 \right|_0^1 = \frac{\Delta x_i}{6}$$

# The mass matrix in 1D

$$\mathbf{M}_{i,i} = \Delta x_{i-1} \int_0^1 \xi^2 \mathrm{d}\xi + \Delta x_i \int_0^1 (1-\xi)^2 \mathrm{d}\xi$$

$$\mathbf{M}_{i,i} = \Delta x_{i-1} \left| \frac{1}{3}\xi^3 \right|_0^1 - \Delta x_i \left| \frac{1}{3}\xi^3 \right|_1^0$$

$$\Rightarrow \mathbf{M}_{i,i} = \frac{\Delta x_{i-1}}{3} + \frac{\Delta x_i}{3} = 2(\mathbf{M}_{i,i-1} + \mathbf{M}_{i,i+1})$$

$\Delta x = 1 \quad \Rightarrow \quad [1, 4, 1]$ (stiffness was [-1, 2, -1])

# Inner vs. outer nodes

distinguish dofs into inner and outer $[\mathbf{u}_i, \mathbf{u}_o]^T$

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{ii} & \mathbf{A}_{io} \\ \mathbf{A}_{oi} & \mathbf{A}_{oo} \end{pmatrix}$$

$$\mathbf{A} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{u}_o \end{bmatrix} = \begin{bmatrix} \mathbf{f}_i \\ \mathbf{f}_o \end{bmatrix}$$

$$\Rightarrow \mathbf{A}_{ii}\mathbf{u}_i = \mathbf{f}_i - \mathbf{A}_{oi}\mathbf{u}_o$$

# Tasks

1. Write a function computing the FE stiffness matrix for 1D discretization
2. Test it by solving the Poisson equation with $f = 1$ (analytical solution)
3. Compare with analytical and FD solutions
4. Write a function computing the FE mass matrix for 1D discretization
5. Repeat the time-stepping tasks from FD with FE