Lecture 01

Introduction to Finite Element Modeling

Anders Logg

May 18, 2018

What will you learn?

- *Introduction to the finite element method*
- *How to derive the variational problem*
- *How to create the finite element function spaces*
- *How to assemble the linear systems*
- *How to solve the linear system*
- *Introduction to FEniCS*
- *Exercise*

Introduction to the finite element method

What is the finite element method?

The finite element method (FEM) is a mathematical framework for discretizing and solving partial differential equations (PDE)

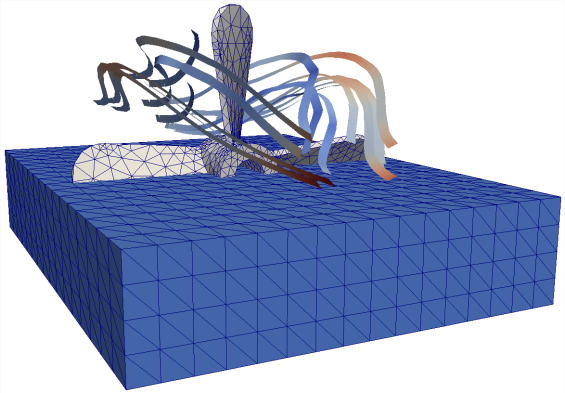
What is the finite element method?

The finite element method (FEM) is a mathematical framework for discretizing and solving partial differential equations (PDE)

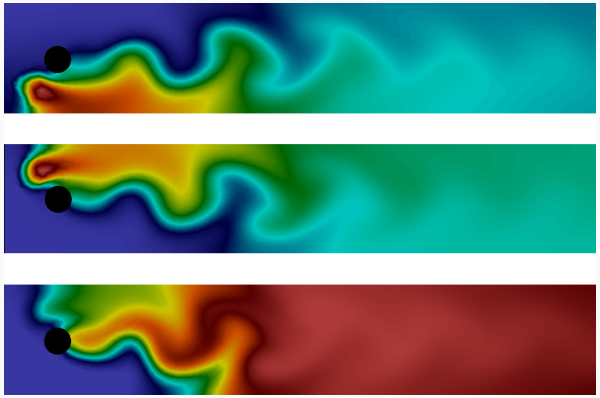
A recipe for discretization of PDE

Partial differential equation
$\mapsto$ Continuous variational problem
$\mapsto$ Discrete variational problem
$\mapsto$ Discrete system of equations

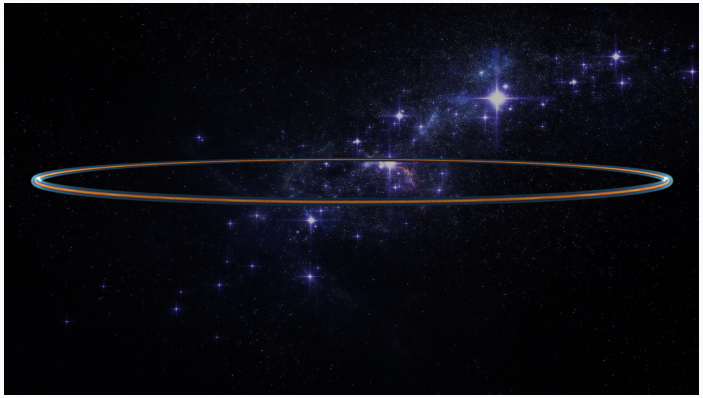
Application examples



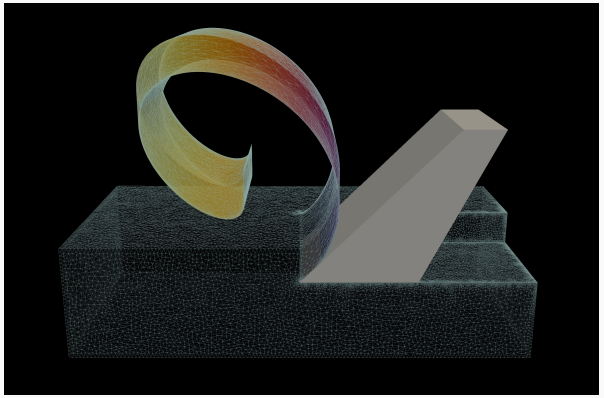
Fluid dynamics



Chemistry



General relativity



Machining

How to derive the variational problem

The FEM cookbook (for a linear PDE)

Partial differential equation (strong form)

$$\mathcal{A}u = f \tag{1}$$

The FEM cookbook (for a linear PDE)

Partial differential equation (strong form)

$$\mathcal{A}u = f \tag{1}$$

Continuous variational problem (weak form)

Find $u \in V$ such that

$$a(u, v) = L(v) \quad \forall v \in V \tag{2}$$

The FEM cookbook (for a linear PDE)

Partial differential equation (strong form)

$$\mathcal{A}u = f \tag{1}$$

Continuous variational problem (weak form)

Find $u \in V$ such that

$$a(u, v) = L(v) \quad \forall v \in V \tag{2}$$

Discrete variational problem (finite element method)

Find $u_h \in V_h$ such that

$$a(u_h, v) = L(v) \quad \forall v \in V_h \tag{3}$$

The FEM cookbook (for a linear PDE)

Partial differential equation (strong form)

$$\mathcal{A}u = f \tag{1}$$

Continuous variational problem (weak form)

Find $u \in V$ such that

$$a(u, v) = L(v) \quad \forall v \in V \tag{2}$$

Discrete variational problem (finite element method)

Find $u_h \in V_h$ such that

$$a(u_h, v) = L(v) \quad \forall v \in V_h \tag{3}$$

Discrete system of equations (linear system)

$$AU = b \tag{4}$$

From strong to weak form: (1) $\rightarrow$ (2)

Partial differential equation (strong form)

$$\mathcal{A}u = f$$

From strong to weak form: (1) $\rightarrow$ (2)

Partial differential equation (strong form)

$$\mathcal{A}u = f$$

Multiply by a test function v and integrate over the domain Ω:

$$\underbrace{\int_\Omega \mathcal{A}uv \,\mathrm{d}x}_{=a(u,v)} = \underbrace{\int_\Omega fv \,\mathrm{d}x}_{=L(v)}$$

From strong to weak form: (1) $\rightarrow$ (2)

Partial differential equation (strong form)

$$\mathcal{A}u = f$$

Multiply by a test function v and integrate over the domain Ω:

$$\underbrace{\int_\Omega \mathcal{A}uv \,\mathrm{d}x}_{=a(u,v)} = \underbrace{\int_\Omega fv \,\mathrm{d}x}_{=L(v)}$$

$\Rightarrow$ Continuous variational problem (weak form)

Find $u \in V$ such that

$$a(u, v) = L(v) \quad \forall v \in V$$

From weak form to finite element method: (2) $\rightarrow$ (3)

Continuous variational problem (weak form)

Find $u \in V$ such that

$$a(u, v) = L(v) \quad \forall v \in V$$

From weak form to finite element method: (2) $\rightarrow$ (3)

Continuous variational problem (weak form)

Find $u \in V$ such that

$$a(u, v) = L(v) \quad \forall v \in V$$

Let V_h be a discrete finite element subspace of V

From weak form to finite element method: (2) $\rightarrow$ (3)

Continuous variational problem (weak form)

Find $u \in V$ such that

$$a(u, v) = L(v) \quad \forall v \in V$$

Let V_h be a discrete finite element subspace of V

$\Rightarrow$ Discrete variational problem (finite element method)

Find $u_h \in V_h$ such that

$$a(u_h, v) = L(v) \quad \forall v \in V_h$$

From finite element method to linear system: (3) $\rightarrow$ (4)

Discrete variational problem (finite element method)

Find $u_h \in V_h$ such that

$$a(u_h, v) = L(v) \quad \forall v \in V_h$$

From finite element method to linear system: (3) $\rightarrow$ (4)

Discrete variational problem (finite element method)

Find $u_h \in V_h$ such that

$$a(u_h, v) = L(v) \quad \forall v \in V_h$$

Let $\{\phi_j\}_{i=1}^{N}$ be a basis for V_h and make the ansatz

$$u_h(x) = \sum_{j=1}^{N} U_j \phi_j(x)$$

From finite element method to linear system: (3) $\rightarrow$ (4)

Discrete variational problem (finite element method)

Find $u_h \in V_h$ such that

$$a(u_h, v) = L(v) \quad \forall v \in V_h$$

Let $\{\phi_j\}_{i=1}^N$ be a basis for V_h and make the ansatz

$$u_h(x) = \sum_{j=1}^{N} U_j \phi_j(x)$$

$\Rightarrow$ Discrete system of equations (linear system)

$$AU = b$$

$U \in \mathbb{R}^N$ is the vector of *degrees of freedom*

From finite element method to linear system: details

Insert the ansatz $u_h(x) = \sum_{j=1}^{N} U_j \phi_j(x)$ into the variational problem $a(u_h, v) = L(v)$ and take $v = \phi_i$ for $i = 1, 2, \ldots, N$:

$$a\Big(\sum_{j=1}^{N} U_j \phi_j, \phi_i\Big) = L(\phi_i), \quad i = 1, 2, \ldots, N$$

From finite element method to linear system: details

Insert the ansatz $u_h(x) = \sum_{j=1}^{N} U_j \phi_j(x)$ into the variational problem $a(u_h, v) = L(v)$ and take $v = \phi_i$ for $i = 1, 2, \ldots, N$:

$$a\Big(\sum_{j=1}^{N} U_j \phi_j, \phi_i\Big) = L(\phi_i), \quad i = 1, 2, \ldots, N$$

$$\sum_{j=1}^{N} U_j \underbrace{a(\phi_j, \phi_i)}_{=A_{ij}} = \underbrace{L(\phi_i)}_{=b_i}, \quad i = 1, 2, \ldots, N$$

From finite element method to linear system: details

Insert the ansatz $u_h(x) = \sum_{j=1}^{N} U_j \phi_j(x)$ into the variational problem $a(u_h, v) = L(v)$ and take $v = \phi_i$ for $i = 1, 2, \ldots, N$:

$$a\Big(\sum_{j=1}^{N} U_j \phi_j, \phi_i\Big) = L(\phi_i), \quad i = 1, 2, \ldots, N$$

$$\sum_{j=1}^{N} U_j \underbrace{a(\phi_j, \phi_i)}_{=A_{ij}} = \underbrace{L(\phi_i)}_{=b_i}, \quad i = 1, 2, \ldots, N$$

$$\sum_{j=1}^{N} A_{ij} U_j = b_i, \quad i = 1, 2, \ldots, N \quad \Leftrightarrow \quad AU = b$$

Summary

- FEM transforms a linear PDE to a linear system

$$\mathcal{A}u = f \quad \mapsto \quad AU = b$$

Summary

- FEM transforms a linear PDE to a linear system

$$\mathcal{A}u = f \quad \mapsto \quad AU = b$$

- FEM transforms a nonlinear PDE to a nonlinear system

$$\mathcal{A}(u) = f \quad \mapsto \quad F(U) = 0$$

Summary

- FEM transforms a linear PDE to a linear system

$$\mathcal{A}u = f \quad \mapsto \quad AU = b$$

- FEM transforms a nonlinear PDE to a nonlinear system

$$\mathcal{A}(u) = f \quad \mapsto \quad F(U) = 0$$

- Use a linear solver to solve the linear system

Summary

- FEM transforms a linear PDE to a linear system

$$\mathcal{A}u = f \quad \mapsto \quad AU = b$$

- FEM transforms a nonlinear PDE to a nonlinear system

$$\mathcal{A}(u) = f \quad \mapsto \quad F(U) = 0$$

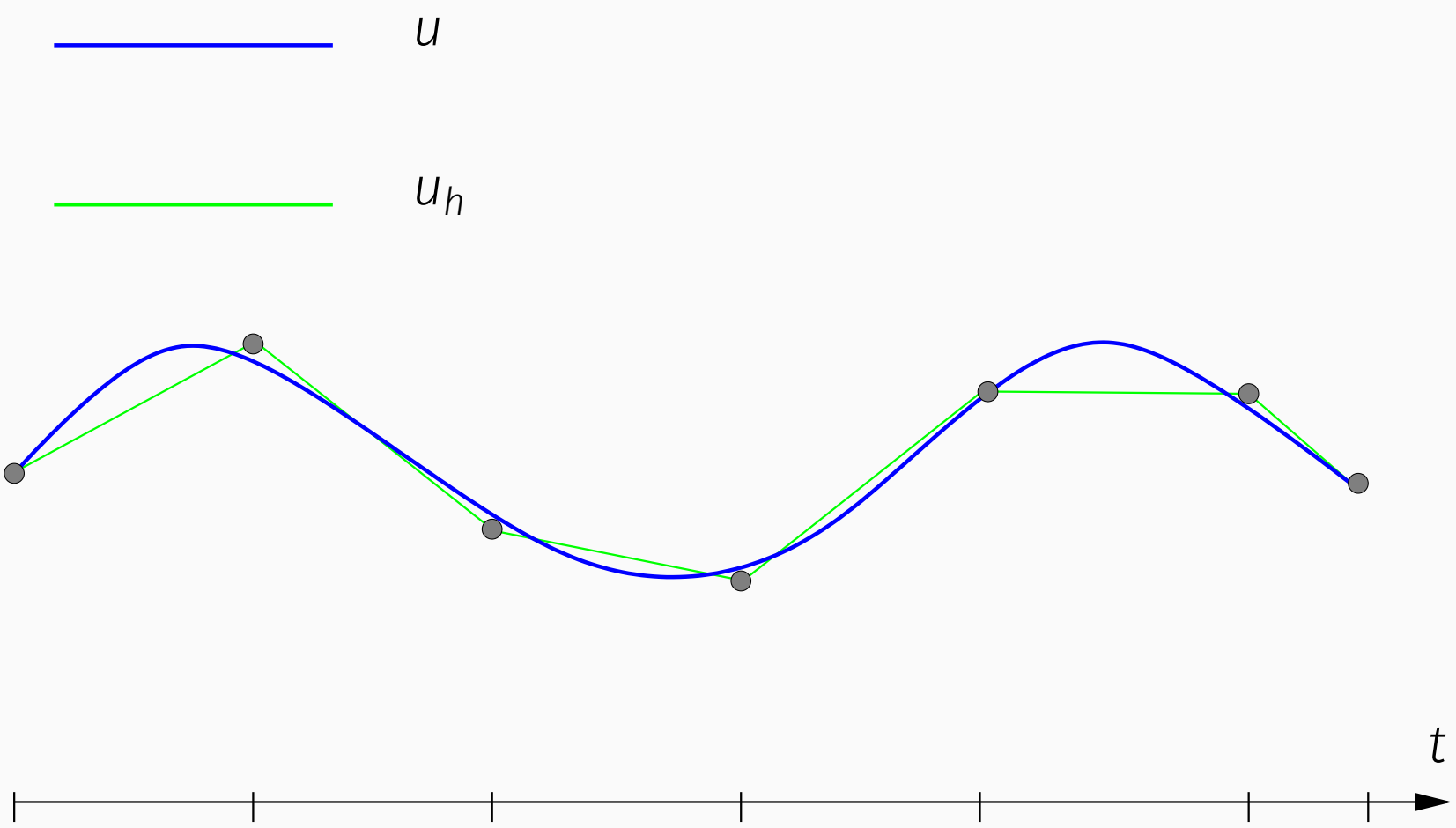
- Use a linear solver to solve the linear system
- Use a nonlinear solver to solve the nonlinear system (Newton's method)

Important topics

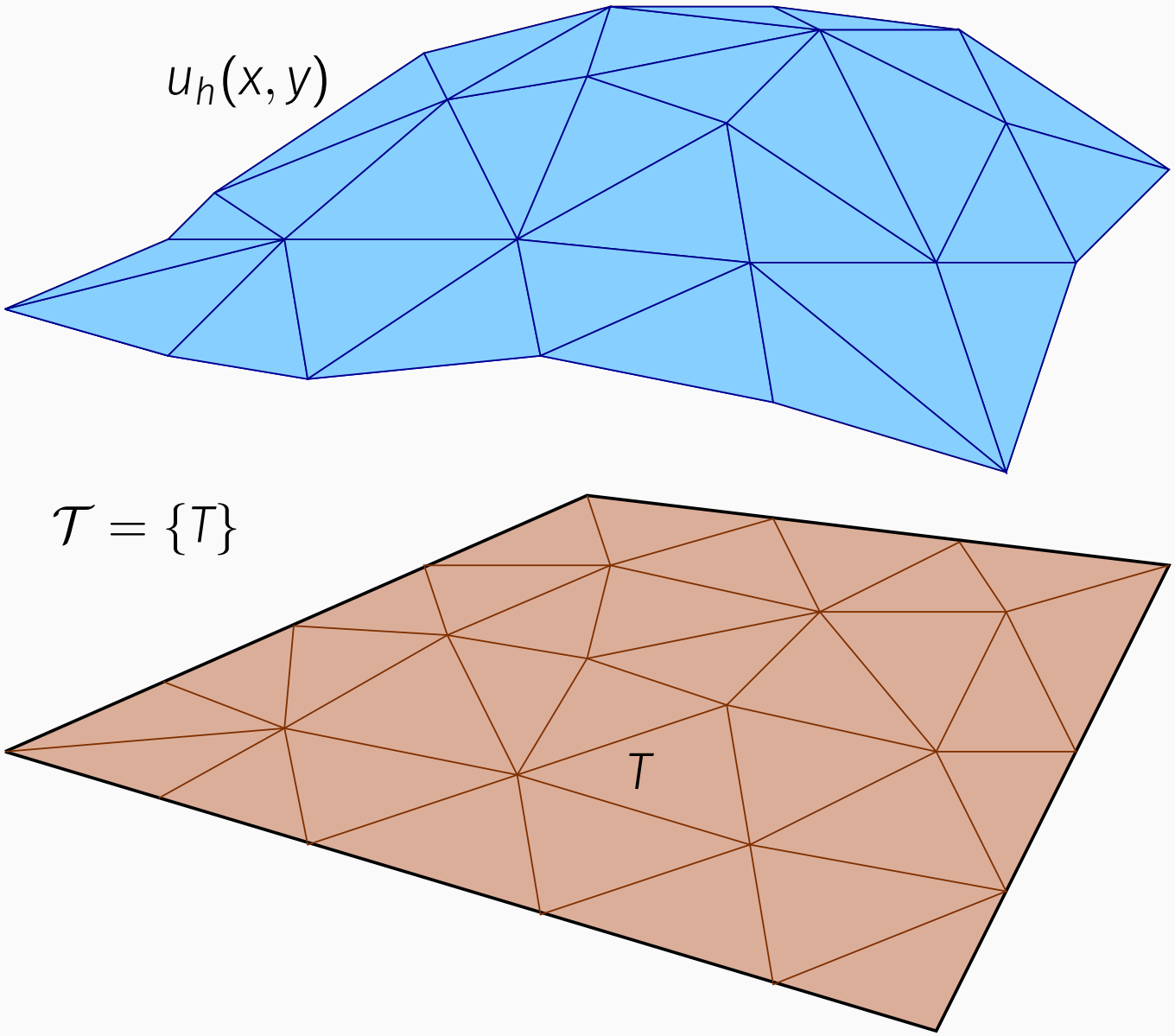
- *How to derive the variational problem*
- *How to create the finite element function space V_h*
- *How to assemble the linear system $AU = b$*
- *How to solve the linear system $AU = b$*

How to create the finite element function space V_h

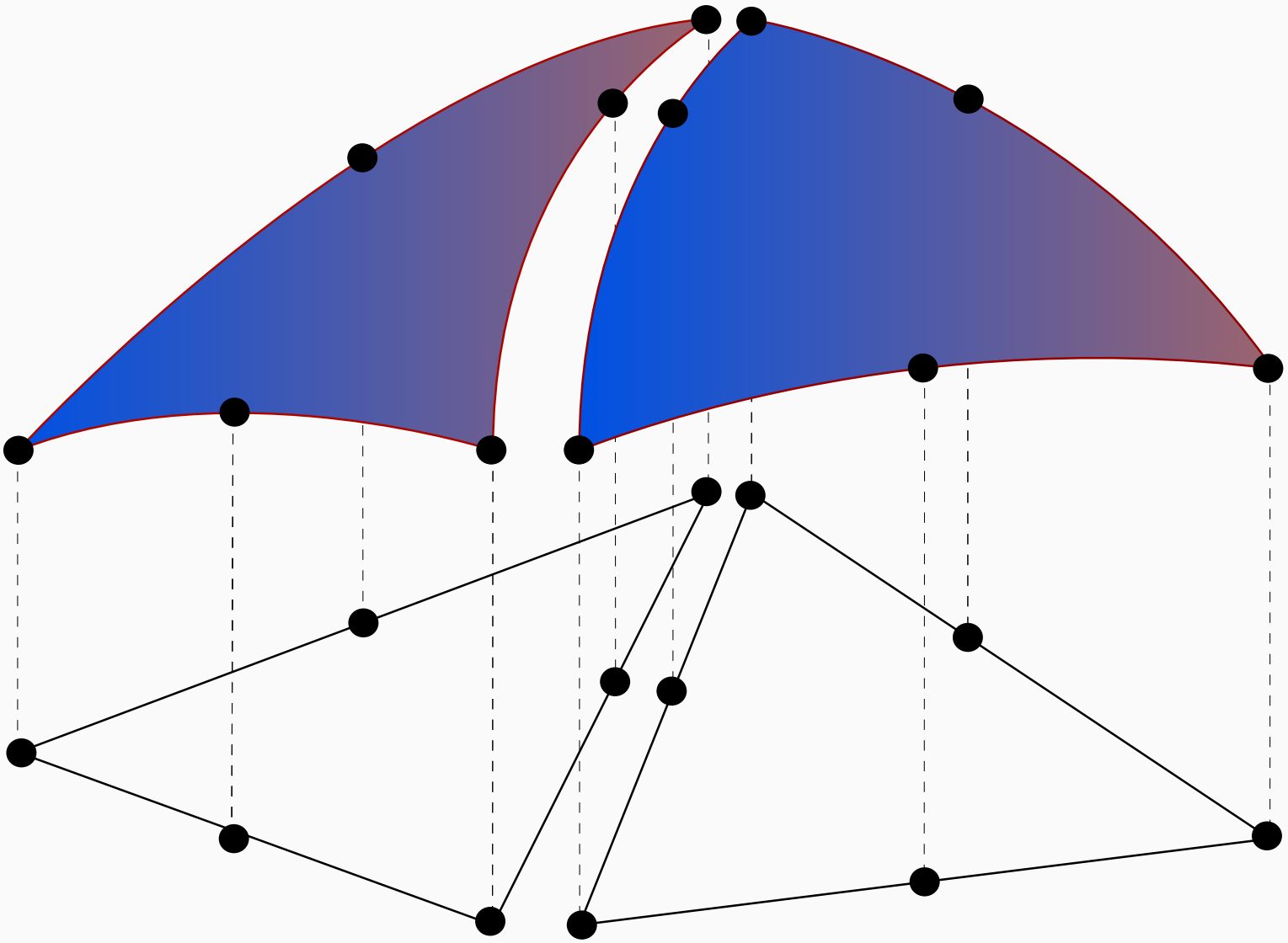
Piecewise linear functions in 1D: $\mathrm{P}_1(\Delta_1)$



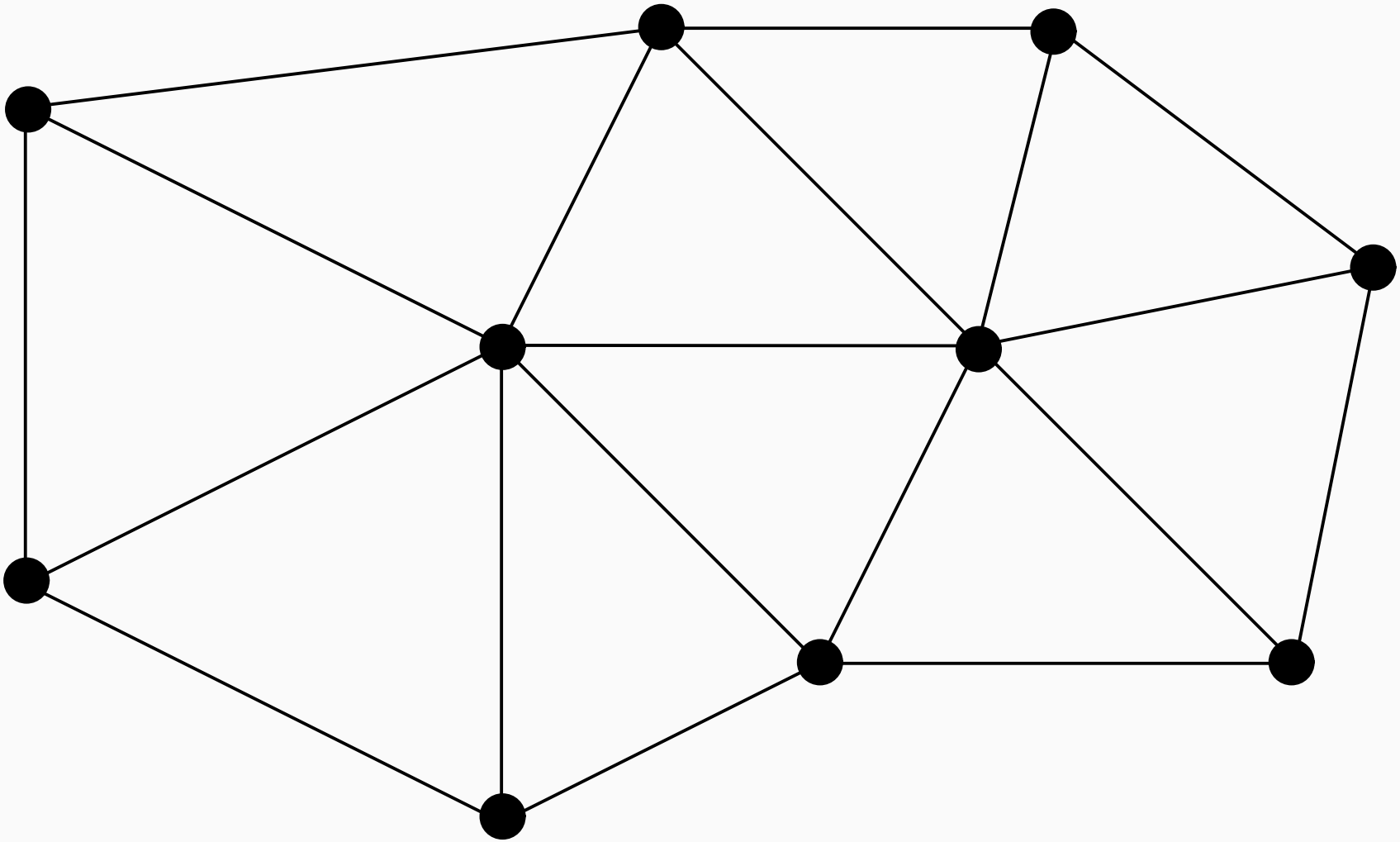
Piecewise linear functions in 2D: $\mathrm{P}_1(\Delta_2)$



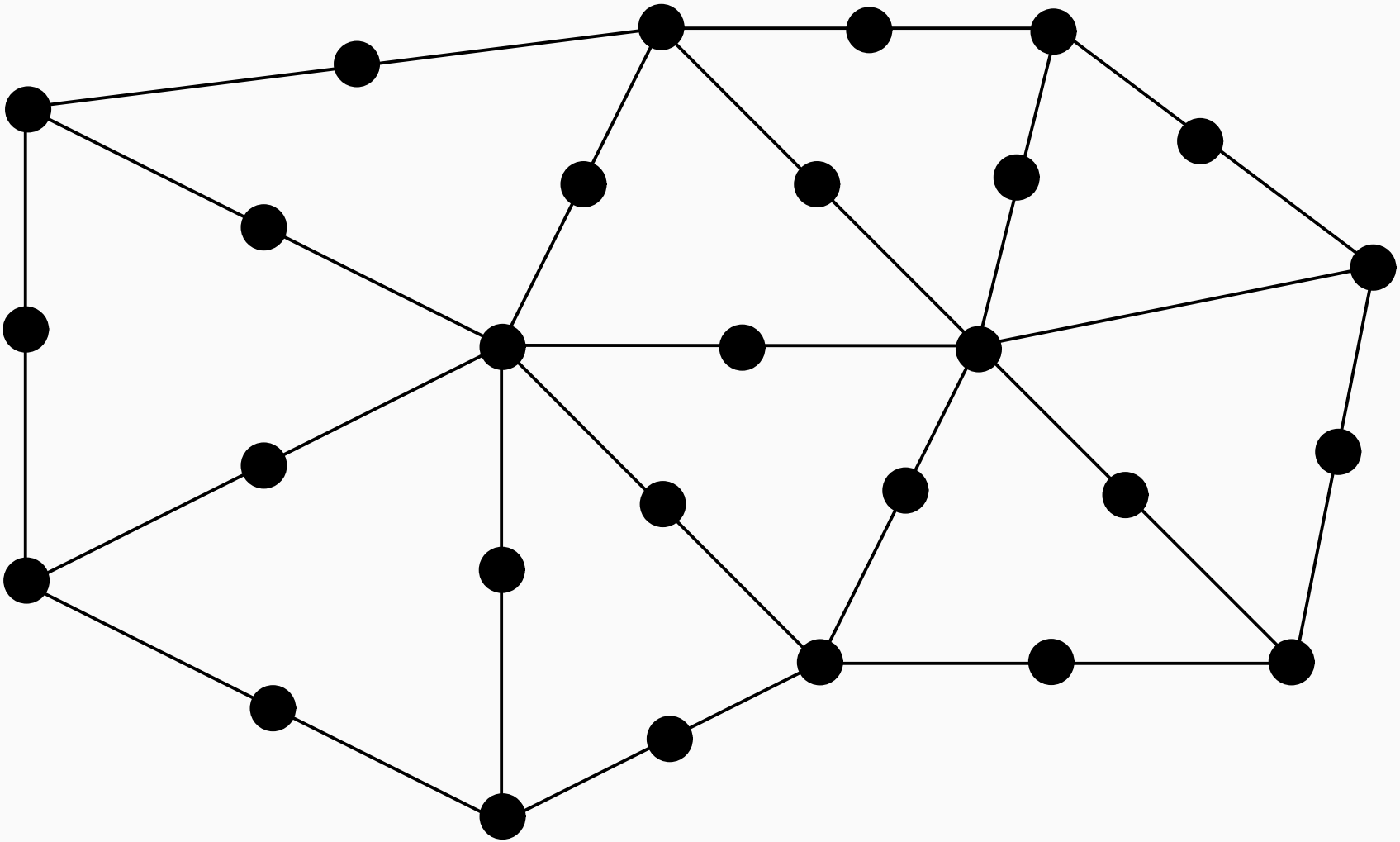
Piecewise quadratic functions in 2D: $P_2(\Delta_2)$



Degrees of freedom for $P_1(\Delta_2)$



Degrees of freedom for $P_2(\Delta_2)$



Periodic Table of the Finite Elements

How to assemble the linear system

$AU = b$

Recall the definition of the linear system $AU = b$

The stiffness matrix

$$A_{ij} = a(\phi_j, \phi_i), \quad i, j = 1, 2, \ldots, N$$

Recall the definition of the linear system $AU = b$

The stiffness matrix

$$A_{ij} = a(\phi_j, \phi_i), \quad i, j = 1, 2, \ldots, N$$

The load vector

$$b_i = L(\phi_i), \quad i = 1, 2, \ldots, N$$

Recall the definition of the linear system $AU = b$

The stiffness matrix

$$A_{ij} = a(\phi_j, \phi_i), \quad i, j = 1, 2, \ldots, N$$

The load vector

$$b_i = L(\phi_i), \quad i = 1, 2, \ldots, N$$

Note that N is large, think $N \sim 10^6$

Also note that A is *sparse*, meaning that A_{ij} is mostly zero and the number of nonzero entries is $\sim N$

Naïve assembly algorithm

$A = 0$

for $i = 1, \ldots, N$

    for $j = 1, \ldots, N$

        $A_{ij} = a(\phi_j, \phi_i)$

    end for

end for

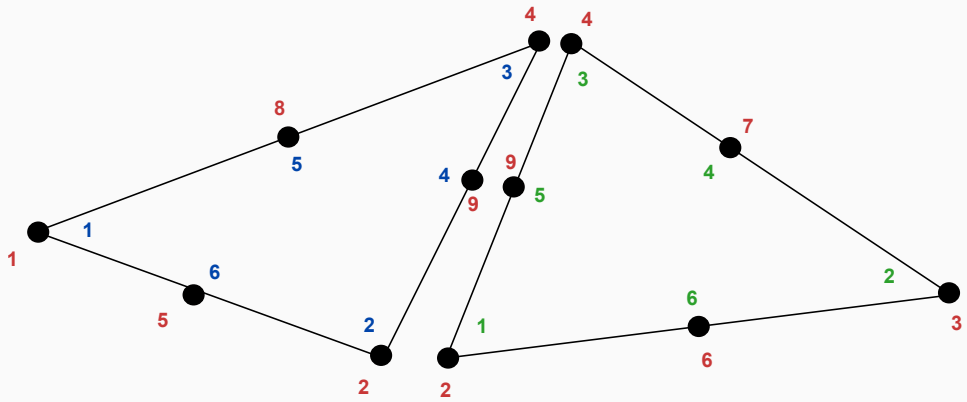
Efficient assembly algorithm

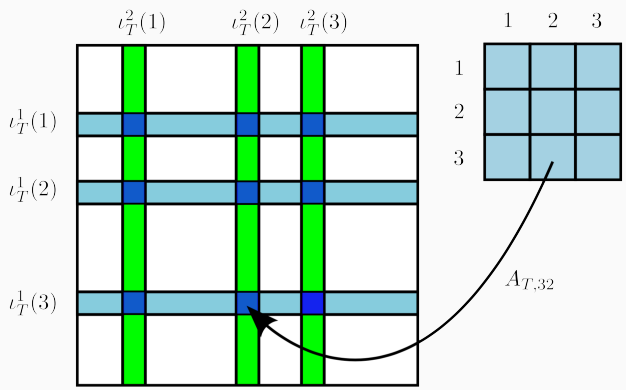
- Naïve assembly algorithm is extremely inefficient: $\mathcal{O}(N^2)$
- Computes all N^2 entries (including zeros)
- Does not reuse element computations

Efficient assembly algorithm

- Naïve assembly algorithm is extremely inefficient: $\mathcal{O}(N^2)$
- Computes all N^2 entries (including zeros)
- Does not reuse element computations

- Can be improved by *element-wise assembly* of A
- Fully automated in FEniCS
- Fully parallelized in FEniCS





How to solve the linear system

$AU = b$

Overview of direct methods

- Gaussian elimination
  - Requires $\sim \frac{2}{3}N^3$ operations
- LU factorization: $A = LU$
  - Solve requires $\sim \frac{2}{3}N^3$ operations
  - Reuse L and U for repeated solves
- Cholesky factorization: $A = LL^\top$
  - Works if A is symmetric and positive definite
  - Solve requires $\sim \frac{1}{3}N^3$ operations
  - Reuse L for repeated solves

Overview of iterative methods

Krylov subspace methods

- GMRES (Generalized Minimal RESidual method)
- CG (Conjugate Gradient method)
  - Works if A is symmetric and positive definite
- BiCGSTAB, MINRES, TFQMR, ...

Multigrid methods

- GMG (Geometric MultiGrid)
- AMG (Algebraic MultiGrid)

Preconditioners

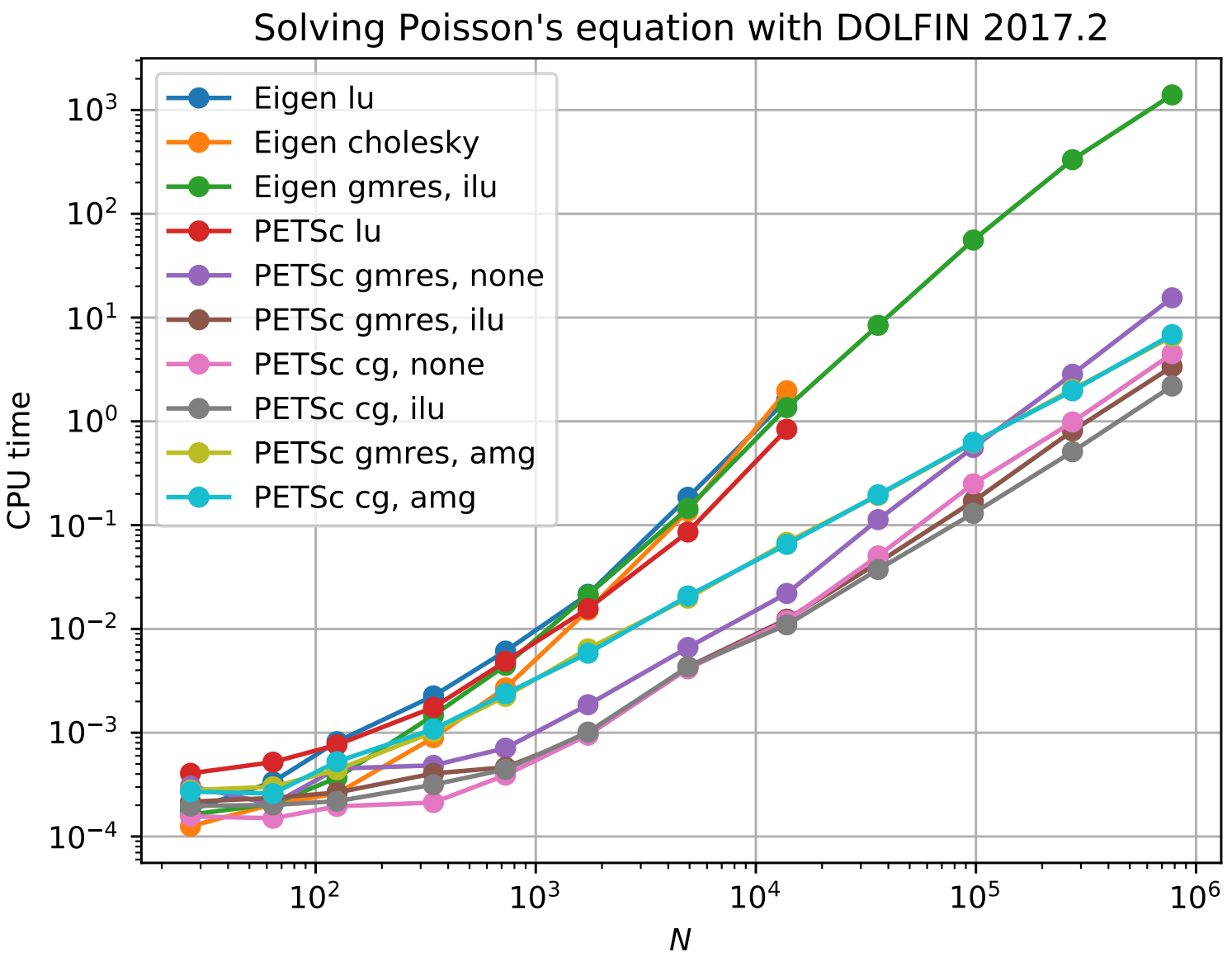
- ILU, ICC, SOR, AMG, Jacobi, block-Jacobi, additive Schwarz, ...

Which method should I use?

Rules of thumb

- Use direct methods for “small” systems
- Use iterative methods for “large” systems
- Break-even at ca 100–1000 degrees of freedom
- Use a symmetric method for a symmetric system
  - Cholesky factorization (direct)
  - CG (iterative)
- Use a multigrid preconditioner for Poisson-like systems
- GMRES with ILU preconditioning is a good default choice
- Use a problem-specific preconditioner for optimal performance

Current timings (2018–05–14) using FEniCS 2017.2



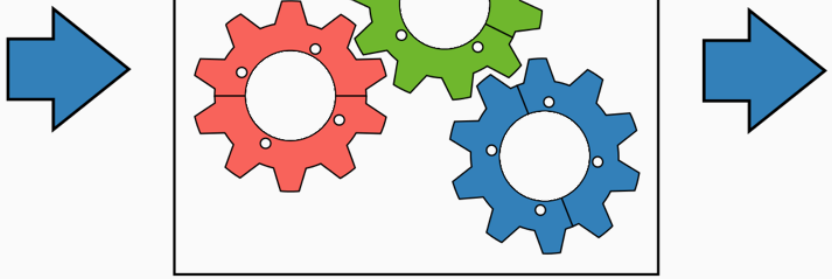
Introduction to FEniCS

FEniCS is an open-source computing platform for PDE/FEM

- C++/Python library
- Initiated 2003 in Chicago
- Automates the finite element method
  - Expressing variational forms
  - Creating function spaces
  - Assembling linear/nonlinear systems
  - Solving linear/nonlinear systems
- Intuitive programming interface
- High-performance (parallel) computing
- Licensed under the GNU LGPL

http://fenicsproject.org

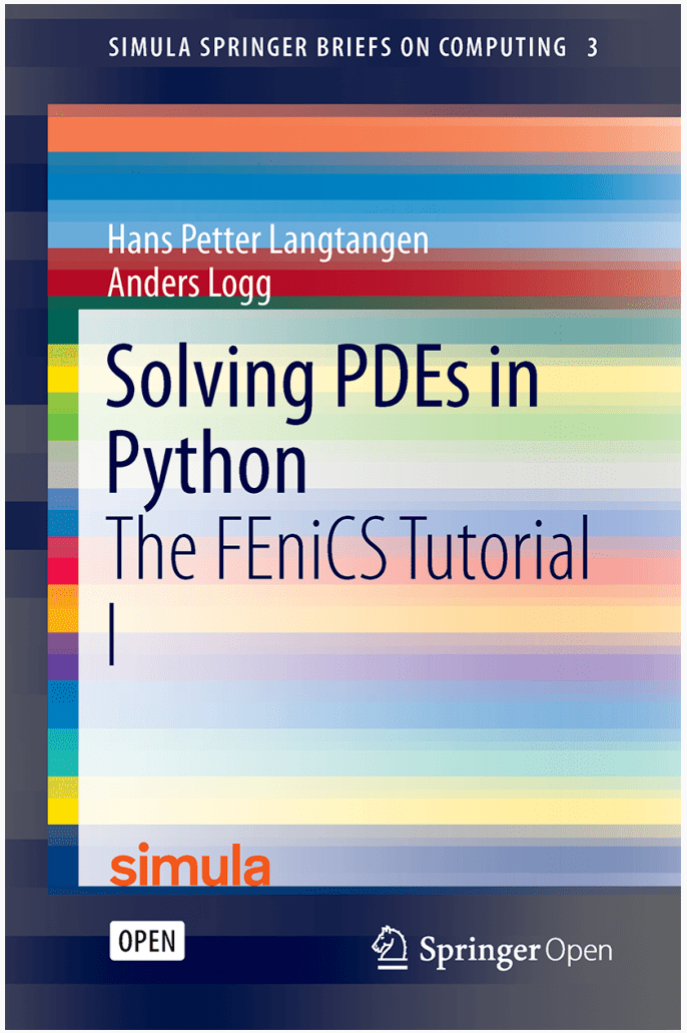
Finite element code generation



```
/// Tabulate the tensor for the contribution from a local cell
virtual void tabulate_tensor(double* A,
                             const double * const * w,
                             const double* vertex_coordinates,
                             int cell_orientation) const
{
  // Number of operations (multiply-add pairs) for Jacobian data:      3
  // Number of operations (multiply-add pairs) for geometry tensor:    8
  // Number of operations (multiply-add pairs) for tensor contraction: 11
  // Total number of operations (multiply-add pairs):                  22

  // Compute Jacobian
  double J[4];
  compute_jacobian_triangle_2d(J, vertex_coordinates);

  // Compute Jacobian inverse and determinant
  double K[4];
  double detJ;
  compute_jacobian_inverse_triangle_2d(K, detJ, J);

  // Set scale factor
  const double det = std::abs(detJ);

  // Compute geometry tensor
  const double G0_0_0 = det*K[0]*K[0] + K[1]*K[1];
  const double G0_0_1 = det*K[0]*K[2] + K[1]*K[3];
  const double G0_1_0 = det*K[2]*K[0] + K[3]*K[1];
  const double G0_1_1 = det*K[2]*K[2] + K[3]*K[3];

  // Compute element tensor
  A[0] = 0.49999999999999*G0_0_0 + 0.5*G0_0_1 + 0.5*G0_1_0 + 0.5*G0_1_1;
  A[1] = -0.49999999999999*G0_0_0 - 0.5*G0_1_0;
  A[2] = -0.5*G0_0_1 - 0.5*G0_1_1;
  A[3] = -0.49999999999999*G0_0_0 - 0.5*G0_0_1;
  A[4] = 0.49999999999999*G0_0_0;
  A[5] = 0.5*G0_0_1;
  A[6] = -0.5*G0_1_0 - 0.5*G0_1_1;
  A[7] = 0.5*G0_1_0;
  A[8] = 0.5*G0_1_1;
}
```

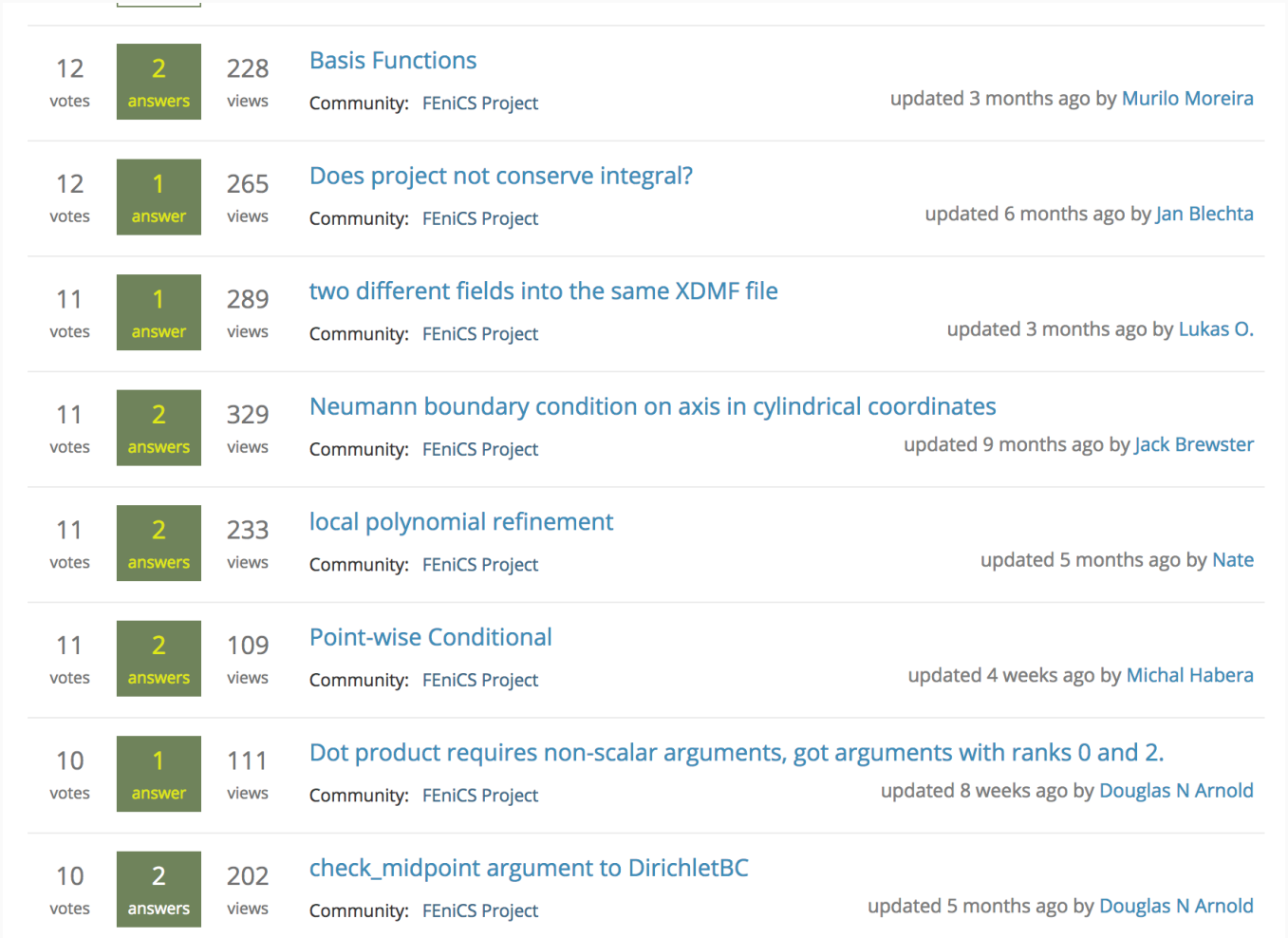
Literature and documentation

- The FEniCS Book (2012)
- The FEniCS Tutorial (2017)
- Free E-book download
- Introduction to finite element analysis
- Introduction to Python programming
- Numerous example problems and codes



Community resources

- Documented FEniCS demos
- FEniCS AllAnswered forum



votes	answers	views		
12 votes	2 answers	228 views	Basis Functions Community: FEniCS Project	updated 3 months ago by Murilo Moreira
12 votes	1 answer	265 views	Does project not conserve integral? Community: FEniCS Project	updated 6 months ago by Jan Blechta
11 votes	1 answer	289 views	two different fields into the same XDMF file Community: FEniCS Project	updated 3 months ago by Lukas O.
11 votes	2 answers	329 views	Neumann boundary condition on axis in cylindrical coordinates Community: FEniCS Project	updated 9 months ago by Jack Brewster
11 votes	2 answers	233 views	local polynomial refinement Community: FEniCS Project	updated 5 months ago by Nate
11 votes	2 answers	109 views	Point-wise Conditional Community: FEniCS Project	updated 4 weeks ago by Michal Habera
10 votes	1 answer	111 views	Dot product requires non-scalar arguments, got arguments with ranks 0 and 2. Community: FEniCS Project	updated 8 weeks ago by Douglas N Arnold
10 votes	2 answers	202 views	check_midpoint argument to DirichletBC Community: FEniCS Project	updated 5 months ago by Douglas N Arnold

Exercise

Exercise 1: Getting started

- Download the FEniCS Tutorial
- Install FEniCS 2017.2
- Verify the installation

```
from fenics import *
mesh = UnitCubeMesh(8, 8, 8)
plot(mesh)
```

http://fenicsproject.org/download/

http://fenicsproject.org/tutorial/