A refresher on the Finite Element Method

Jean-Paul Pelteret (jean-paul.pelteret@fau.de)
Luca Heltai (luca.heltai@sissa.it)

19 March 2018

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Implementing the finite element method

Brief re-hash of the FEM, using the Poisson equation:

We start with the strong form:

$$\begin{aligned} -\Delta u &= f \qquad \text{in } \Omega \\ u &= 0 \qquad \text{on } \partial\Omega \end{aligned}$$

Implementing the finite element method

Brief re-hash of the FEM, using the Poisson equation:

We start with the strong form:

$$-\Delta u = f$$

...and transform this into the weak form by multiplying *from the left* with a test function:

$$(\nabla\varphi,\nabla u)=(\varphi,f) \quad \forall\varphi$$

The solution of this is a function *u(x)* from an infinite-dimensional function space.

Implementing the finite element method

Since computers can't handle objects with infinitely many coefficients, we seek a finite dimensional function of the form

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

To determine the *N* coefficients, test with the *N* basis functions:

$$(\nabla \varphi_i, \nabla u_h) = (\varphi_i, f) \quad \forall i = 1 \ldots N$$

If basis functions are linearly independent, this yields *N* equations for *N* coefficients.
This is called the *Galerkin* method.

Implementing the finite element method

Practical question 1: How to define the basis functions?

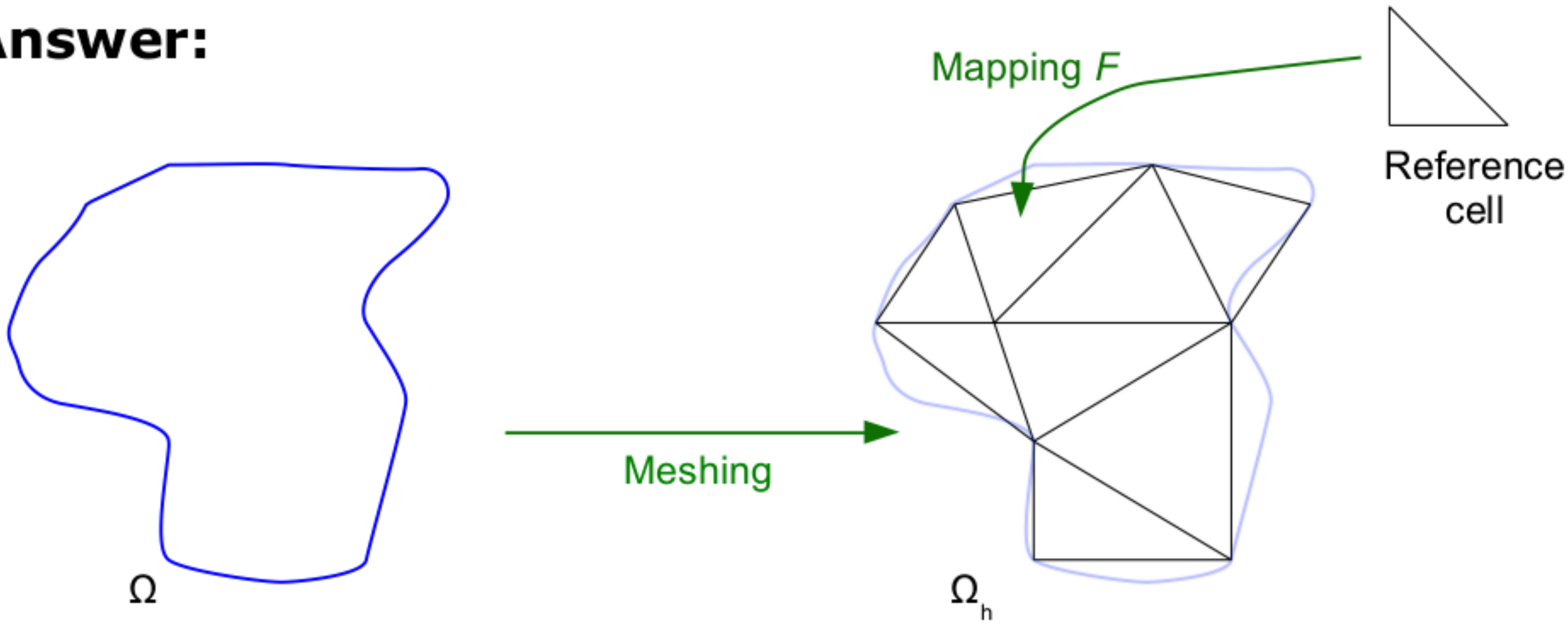
Answer: In the finite element method, this is done using the following concepts:

- Subdivision of the domain into a mesh
- Each cell of the mesh is a mapping of the reference cell
- Definition of basis functions on the reference cell
- Each shape function corresponds to a degree of freedom on the global mesh

Implementing the finite element method

Practical question 1: How to define the basis functions?

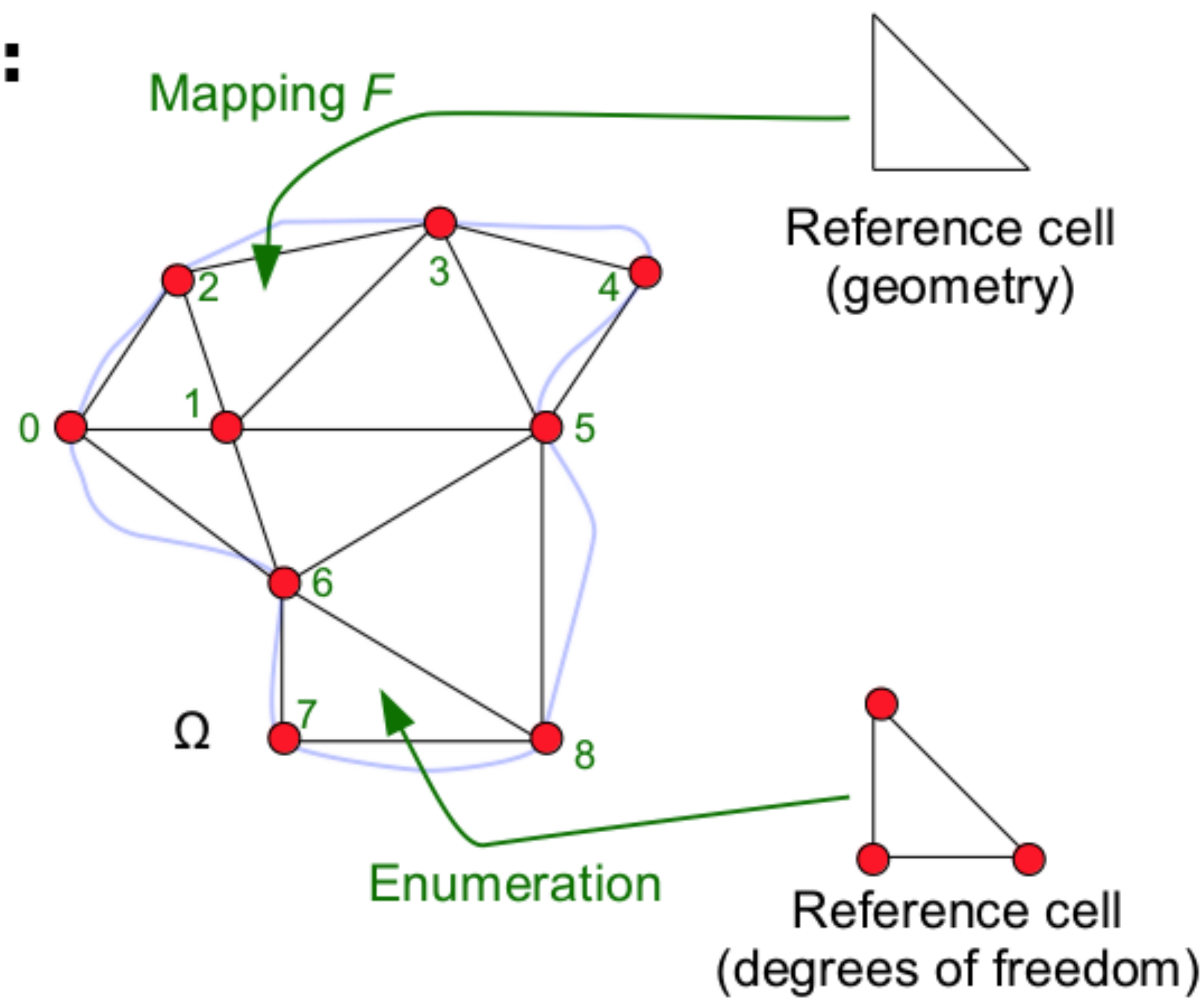
Answer:



Implementing the finite element method

Practical question 1: How to define the basis functions?

Answer:



Implementing the finite element method

Practical question 1: How to define the basis functions?

Answer: In the finite element method, this is done using the following concepts:

- Subdivision of the domain into a mesh
- Each cell of the mesh is a mapping of the reference cell
- Definition of basis functions on the reference cell
- Each shape function corresponds to a degree of freedom on the global mesh

Concepts in red will correspond to things we need to implement in software, explicitly or implicitly.

Implementing the finite element method

Given the definition $u_h = \sum_{j=1}^{N} U_j \varphi_j(x)$, we can expand the bilinear form

$$(\nabla \varphi_i, \nabla u_h) = (\varphi_i, f) \quad \forall i = 1 \ldots N$$

to obtain:

$$\sum_{j=1}^{N} (\nabla \varphi_i, \nabla \varphi_j) U_j = (\varphi_i, f) \quad \forall i = 1 \ldots N$$

This is a linear system

$$AU = F$$

with

$$A_{ij} = (\nabla \varphi_i, \nabla \varphi_j) \qquad F_i = (\varphi_i, f)$$

Implementing the finite element method

Practical question 2: How to compute

$$A_{ij}=(\nabla\varphi_i,\nabla\varphi_j) \qquad F_i=(\varphi_i,f)$$

Answer: By mapping back to the reference cell...

$$\begin{aligned} A_{ij} &= (\nabla\varphi_i,\nabla\varphi_j) \\ &= \sum_K \int_K \nabla\varphi_i(x)\cdot\nabla\varphi_j(x) \\ &= \sum_K \int_{\hat K} J_K^{-1}(\hat x)\hat\nabla\hat\varphi_i(\hat x) \cdot J_K^{-1}(\hat x)\hat\nabla\hat\varphi_j(\hat x)\ |\det J_K(\hat x)| \end{aligned}$$

...and quadrature:

$$A_{ij} \approx \sum_K \sum_{q=1}^Q J_K^{-1}(\hat x_q)\hat\nabla\hat\varphi_i(\hat x_q) \cdot J_K^{-1}(\hat x_q)\hat\nabla\hat\varphi_j(\hat x_q)\ \underbrace{|\det J_K(\hat x_q)|\ w_q}_{=:\mathrm{JxW}}$$

Similarly for the right hand side *F*.

Implementing the finite element method

Practical question 3: How to store the matrix and vectors of the linear system

$$AU = F$$

Answers:

- *A* is sparse, so store it in compressed row format
- *U*,*F* are just vectors, store them as arrays
- Implement efficient algorithms on them, e.g. matrix-vector products, preconditioners, etc.
- For large-scale computations, data structures and algorithms must be parallel

Implementing the finite element method

Practical question 4: How to solve the linear system

$$AU = F$$

Answers: In practical computations, we need a variety of

- Direct solvers
- Iterative solvers
- Parallel solvers

Implementing the finite element method

Practical question 5: What to do with the solution of the linear system

$$AU = F$$

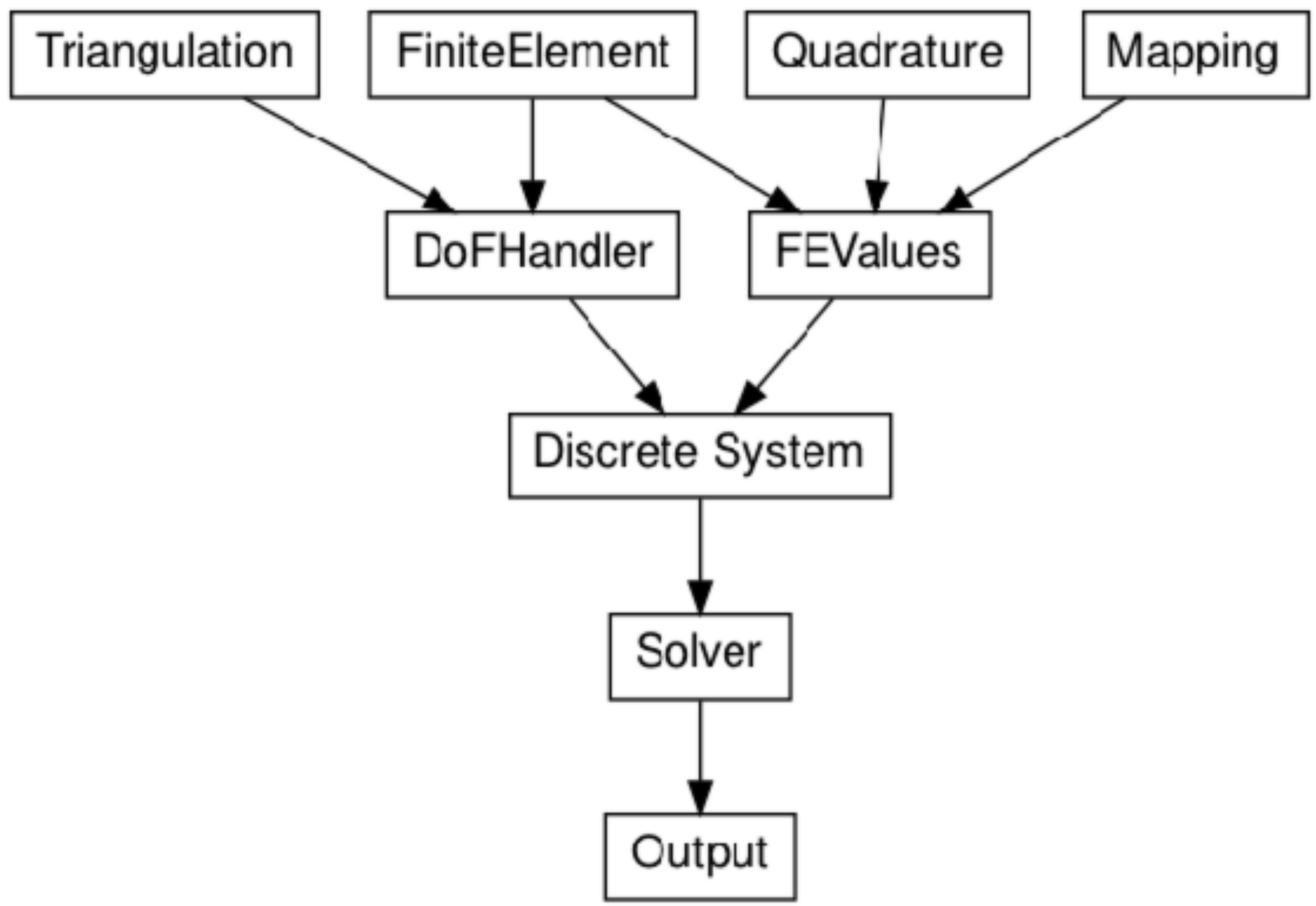
Answers: The goal is not to solve the linear system, but to do something with its solution:

- Visualize
- Evaluate for quantities of interest
- Estimate the error

These steps are often called *postprocessing the solution*.

Implementing the finite element method

Together, the concepts we have identified lead to the following components that all appear (explicitly or implicitly) in finite element codes:



Implementing the finite element method

Summary:

- By going through the mathematical description of the FEM, we have identified *concepts* that need to be represented by *software components*.
- Other components relate to what we *want to do* with numerical solutions of PDEs.
- The next few lectures will show the software realization of these concepts.

On using state-of-the-art tools

All of us have our favorite editor, often the first one we learned well:

- vi/vim/gvim
- emacs
- ...

But: Just because we know them well, this doesn't mean:

- That they are well suited to the task
- That they are state of the art
- That they are the tools that let you be most productive.

The rarest resource is *your* time, not CPU time etc.
You *must* be willing to keep learning new tools!

IDEs

All of us have our favorite editor, often the first one we learned well:

- vi/vim/gvim
- emacs
- ...

The problem with most of these:

- They are (good) editors but not code exploration tools
- They are text-based, not graphical

Excellent, modern tool are for example:

- eclipse
- kdevelop
- Xcode, Microsoft Visual C/C++

IDEs

What an IDE can do for you:

IDEs "know" about your code base, i.e., they *parse all* of the files that belong to your project.

Thus, the IDE...

- Knows where a variable is declared (even if in a different file)
- Knows its type and can help you with *code completion*
- Can *rename* a variable everywhere
- Can keep declaration and definition in sync
- Can help you with function arguments
- Makes you *faster* and make *far fewer mistakes.*