Solving Poisson’s equation

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

19 March 2018

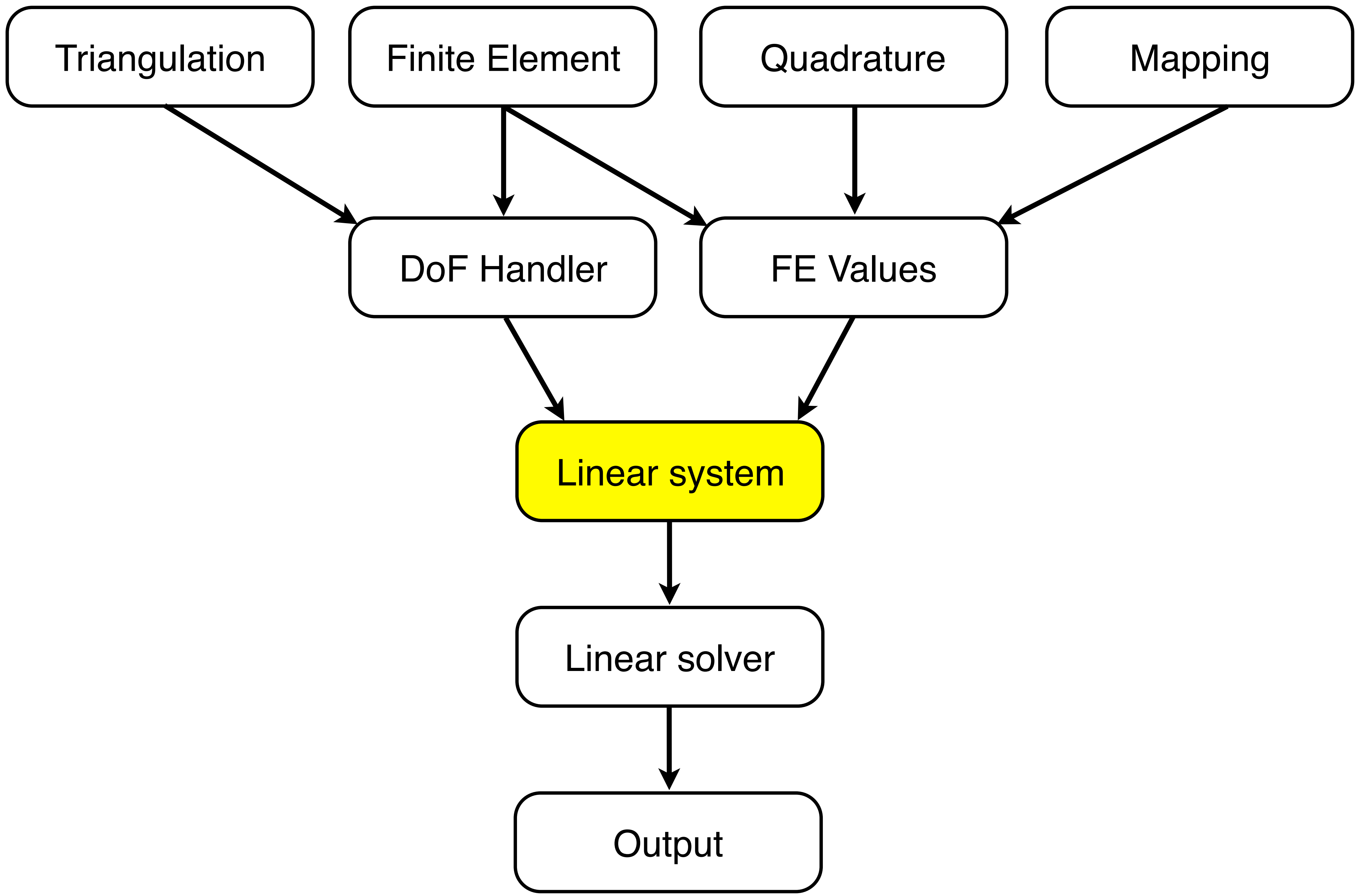
Aims for this module

- First introduction into assembly of sparse linear systems
  - Translation of weak form to assembly loops
  - Applying boundary conditions
- Using linear solvers
- Post-processing and visualisation

Reference material

- Tutorials
  - Step-3
    https://dealii.org/8.5.1/doxygen/deal.II/step_3.html
- Documentation
  - https://www.dealii.org/developer/doxygen/deal.II/group__FE__vs__Mapping__vs__FEValues.html
  - https://www.dealii.org/developer/doxygen/deal.II/group__UpdateFlags.html

Structure of a prototypical FE problem



Sparse linear systems

- Minimise data storage
  - Evaluate grid connectivity
- Functions to help set up
  - Connectivity
  - Constraints
- Minimal access times
  - Direct manipulation of (non-zero) entries
  - Matrix-vector operations
    - Skip over zero-entries
- Types
  - Unity (monolithic, contiguous)
  - Block sparse structures
- Sub-organisation (e.g. component-wise)

$$[K]\,\{d\} = \{F\}$$

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix}$$

- $$\left(K_{11} - K_{12}K_{22}^{-1}K_{21}\right) d_1 = F_1 - K_{12}K_{22}^{-1}F_2$$
- $$d_2 = K_{22}^{-1}\left(F_2 - K_{21}d_1\right)$$

Constraints on sparse linear systems

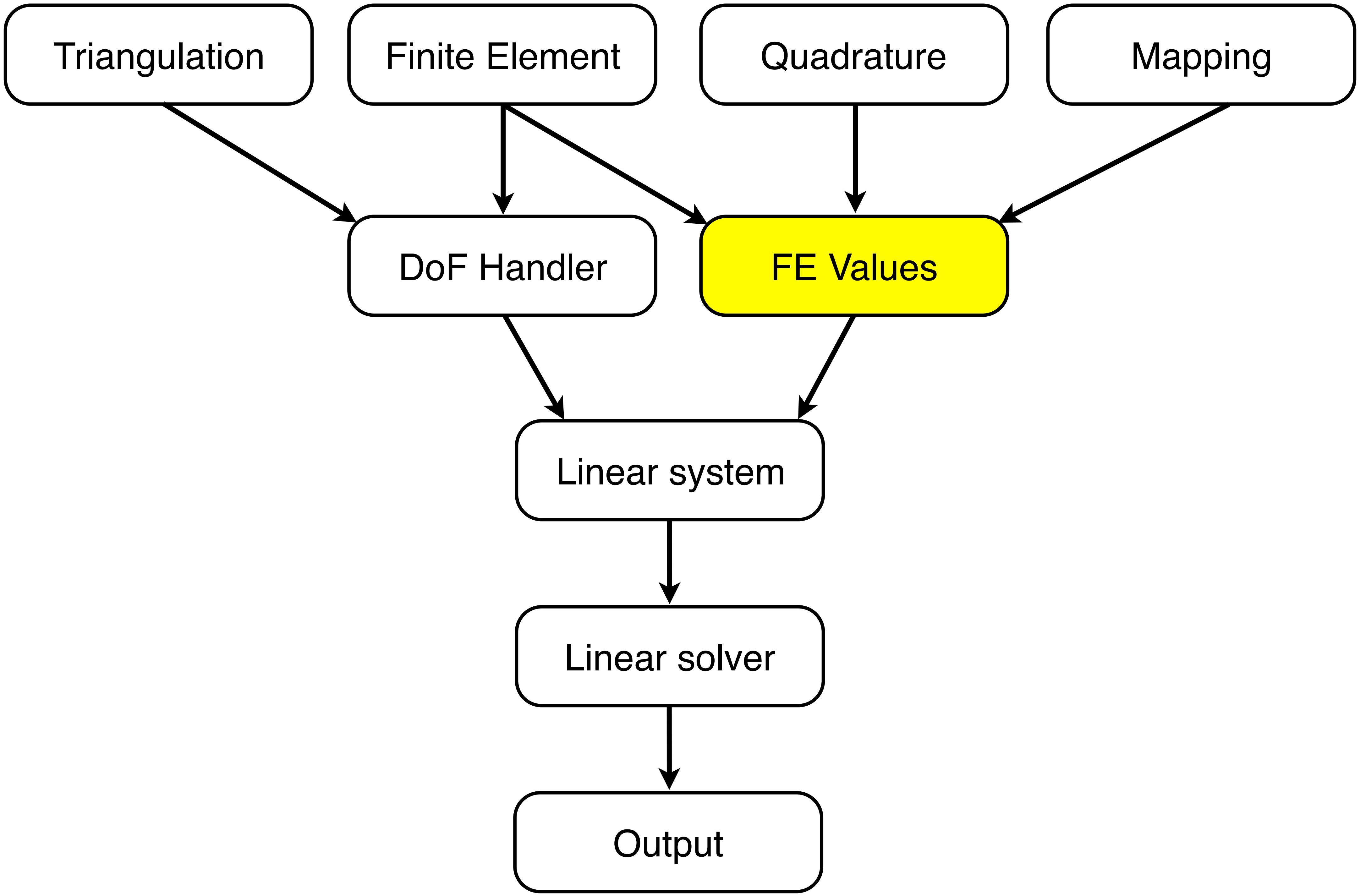
- Strong Dirichlet boundary conditions
  - Apply user-defined spatially-dependent functions to specific boundaries
  - Can restrict to components of a multi-dimensional field
  - Limited to fields with support points on faces
  - Possible to scale system matrix/RHS vector accordingly
    - Better matrix conditioning
- Neumann boundary conditions
  - Implementation dependent
- Other constraints need special consideration
  - Periodic boundary conditions
  - Refinement with hanging nodes
  - Some time-dependent formulations

$$[K]\{d\} = \{F\}$$

$$\begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \begin{Bmatrix} d_1 \\ d_2 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \end{Bmatrix}$$

- $$\left(K_{11} - K_{12}K_{22}^{-1}K_{21}\right) d_1 = F_1 - K_{12}K_{22}^{-1}F_2$$
- $$d_2 = K_{22}^{-1}\left(F_2 - K_{21}d_1\right)$$

Structure of a prototypical FE problem



Integration on a cell: the FEValues class

$$K = \int_{\Omega} \nabla \delta \phi \cdot k \nabla \phi dV$$

$$\approx \delta \phi^I \sum_K \left(\int_{\Omega^h} \nabla N^I \cdot k \nabla N^J dV^h \right) \phi^J$$

$$\approx \delta \phi^I \underbrace{\sum_K \left(\sum_q \nabla N^I \left(\mathbf{x}_q\right) \cdot k_q \nabla N^J \left(\mathbf{x}_q\right) \, w_q \right)}_{K_{IJ} = (\nabla \phi^I, k \nabla \phi^J)} \phi^J$$

$$\approx \delta \phi^I \underbrace{\sum_K \boxed{\left(\sum_q \left[J_{\square}^K\right]_q^{-1} \nabla_{\square} N^I \left(\mathbf{x}_q\right) \cdot k_q \left[J_{\square}^K\right]_q^{-1} \nabla_{\square} N^J \left(\mathbf{x}_q\right) | \det J_q | w_q \right)}}_{K_{IJ}} \phi^J$$

$$J_{\square}^h = \frac{\partial \mathbf{X}^{\boldsymbol{\xi}}}{\partial \mathbf{X}}$$

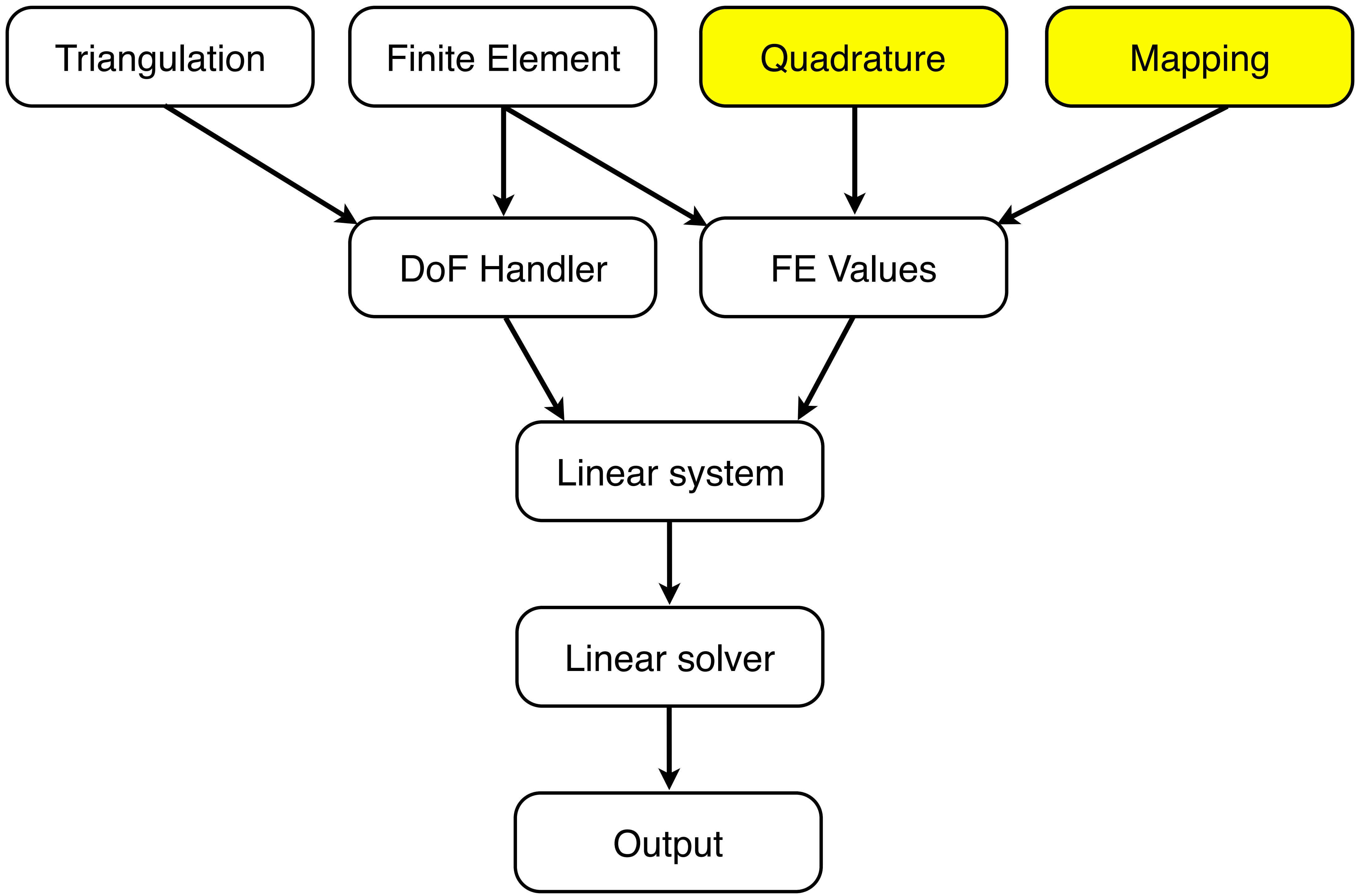
Integration on a cell: the FEValues class

- Object that helps perform integration
- Combines information of:
    - Cell geometry
    - Finite-element system
    - Quadrature rule
    - Mappings
- Can provide:
    - Shape function data
    - Quadrature weights and mapping jacobian at a point
    - Normal on face surface
    - Covariant/contravariant basis vectors
- More ways it can help:
    - Object to extract shape function data for individual fields
    - Natural expressions when coding
- Low level optimisations

$$K_{IJ} = \sum_{K} \left(\sum_{q} \boxed{\left[J_{\square}^{K}\right]_{q}^{-1} \nabla_{\square} N^{I}\left(\mathbf{x}_{q}\right) \cdot k_{q} \left[J_{\square}^{K}\right]_{q}^{-1} \nabla_{\square} N^{J}\left(\mathbf{x}_{q}\right) \left|\det J_{q}\right| w_{q}} \right)$$

```
cell_matrix(I,J) += k
      * fe_values.shape_grad (I, q_point)
      * fe_values.shape_grad (J, q_point)
      * fe_values.JxW (q_point);
```

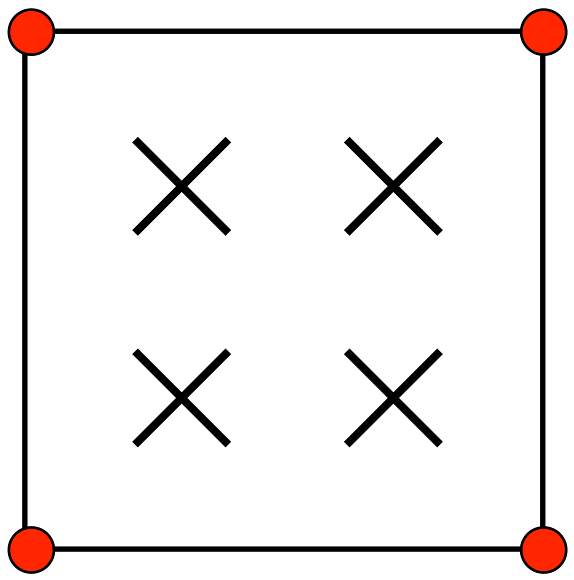
Structure of a prototypical FE problem



Integration on a cell: the Quadrature classes

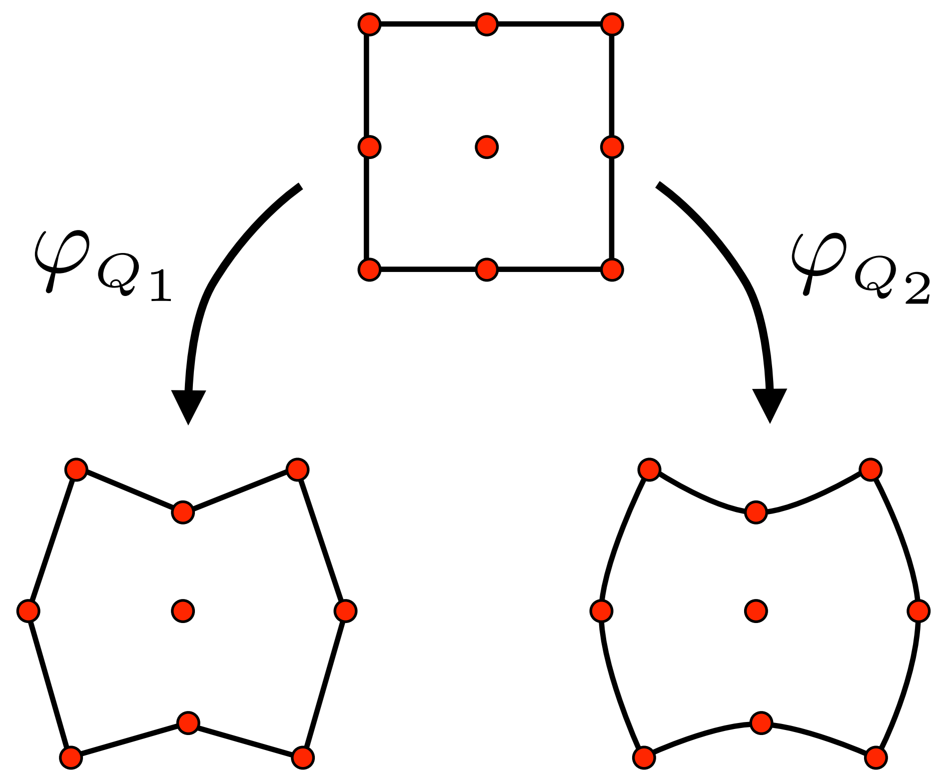
- n-Order Gauss quadrature
- Other rules
  - Gauss Lobatto
  - Simpson
  - Trapezoidal
  - Midpoint
  - A few others
- Anisotropic
  - Tensor product

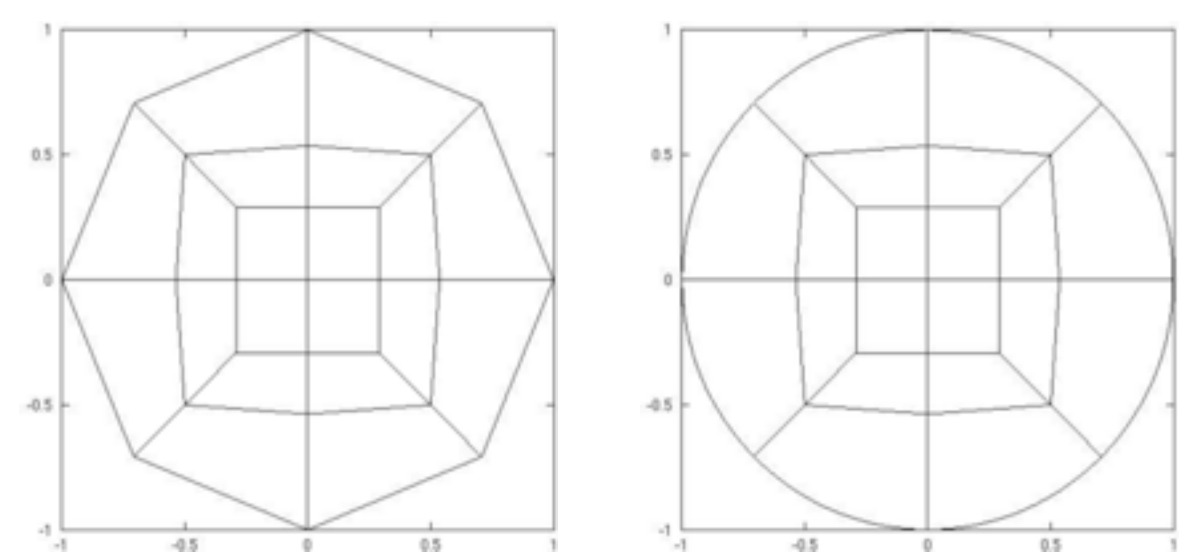
`FE_Q<2>(1)`



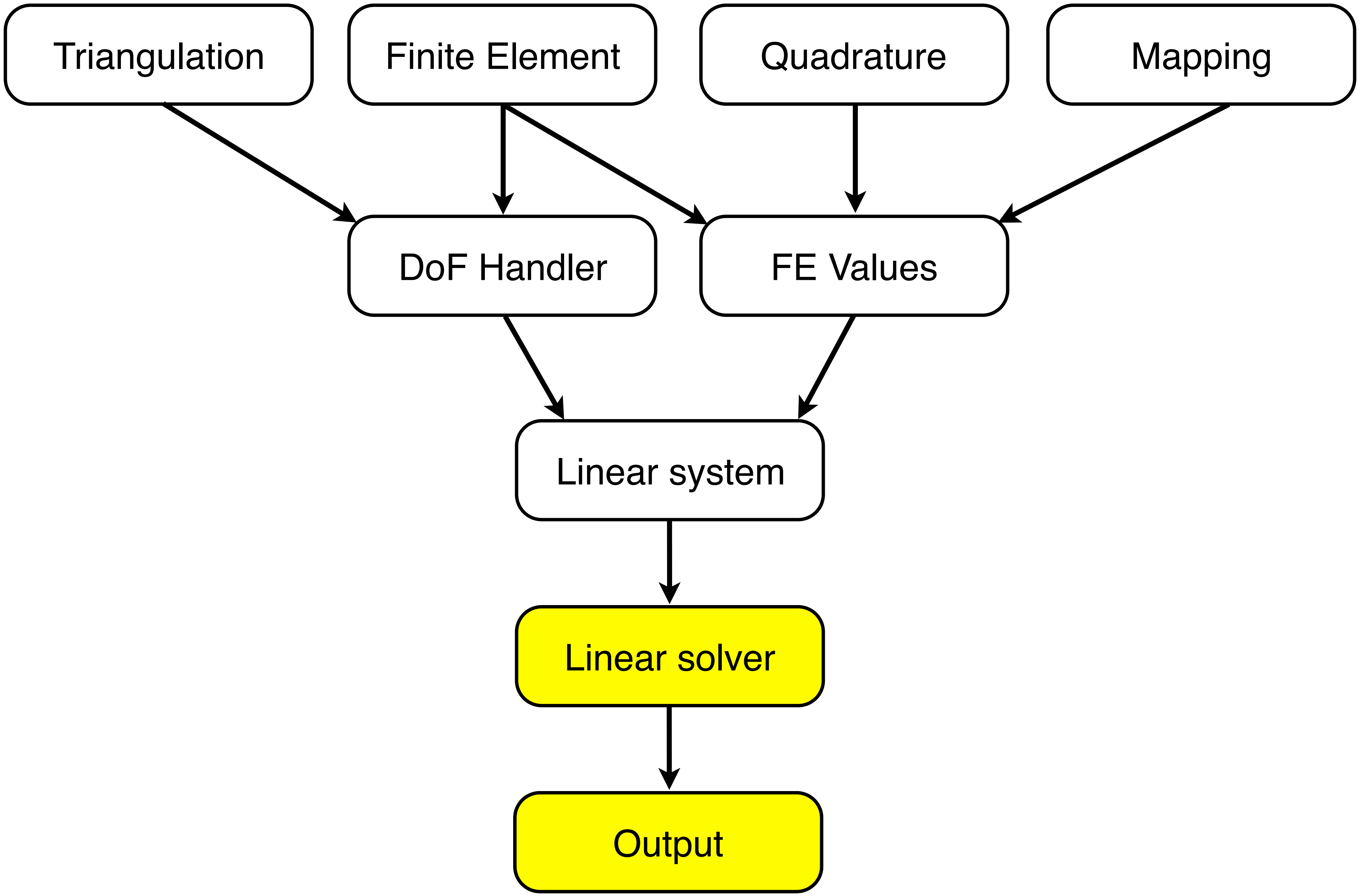
Integration on a cell: the Mapping classes

- n-order mappings
    - Increase accuracy of:
        - Integration schemes
        - Surface basis vectors
- Lagrangian / Eulerian
    - Latter useful for fluid and contact problems, data visualisation
- Boundary and interior manifolds





Structure of a prototypical FE problem



Solving Poisson's equation

- Demonstration: Step-3
  https://www.dealii.org/8.5.1/doxygen/deal.II/step_3.html
  http://www.math.colostate.edu/~bangerth/videos.676.10.html

- Key points
  - Local assembly + quadrature rules
  - Distribution of local contributions to the global linear system
  - Application of boundary conditions
  - Solving a linear system
  - Output for visualisation