Master in High Performance Computing

Numerical Solution of PDEs using the Finite Element Method

A Poisson solver - deal.II step-3

Luca Heltai <luca.heltai@sissa.it>

International School for Advanced Studies (www.sissa.it)
Mathematical Analysis, Modeling, and Applications (math.sissa.it)
Master in High Performance Computing (www.mhpc.it)
SISSA mathLab (mathlab.sissa.it)

sissa mathLab innovating with mathematics

The Abdus Salam International Centre for Theoretical Physics

SISSA

Aims for this Lecture

- 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/current/doxygen/deal.II/step_3.html
- Documentation
  - https://www.dealii.org/current/doxygen/deal.II/group__FE__vs__Mapping__vs__FEValues.html
  - https://www.dealii.org/current/doxygen/deal.II/group__UpdateFlags.html

Recap of Poisson Problem

Variational, continuous problem, infinte dimensional space:

$$\int_\Omega \nabla u \cdot \nabla v = \int_\Omega f v \qquad \forall v \in H^1_0(\Omega)$$

Variational, discrete problem, finite dimensional space:

$$\int_\Omega \nabla u_h \cdot \nabla v_h = \int_\Omega f v_h \qquad \forall v_h \in V_h \subset H^1_0(\Omega)$$

Recap of Poisson Problem

$$\int_\Omega \nabla u_h \cdot \nabla v_h = \int_\Omega f v_h \qquad \forall v_h \in V_h = \text{span}\{v_i\}_{i=1}^N$$

$$\Longleftrightarrow$$

$$A_{ij} u^j = F_i \qquad u_h := u^i v_i$$

$$A_{ij} := \int_\Omega \nabla v_j \nabla v_i \qquad F_i := \int_\Omega f v_i$$

Split Assembly on cells

$$A_{ij} := \int_\Omega \nabla v_j \cdot \nabla v_i d\Omega \qquad F_i := \int_\Omega f v_i d\Omega$$

$$A_{ij} = \sum_m \int_{T_m} \nabla v_j \cdot \nabla v_i dT_m$$

To make this efficiently, we need a smart way to map local dofs to global dofs

Split Assembly on cells

$$A_{ij} = \sum_m \int_{T_m} \nabla v_j \cdot \nabla v_i dT_m = \sum_m \int_{\hat{T}} [(\nabla v_j) \circ F_m] \cdot [(\nabla v_i) \circ F_m] J_m d\hat{T}$$

$$v_i \circ F_m |_{T_m} = \sum_\alpha P_{mi\alpha} \hat{v}_\alpha$$

$$P_{mi\alpha} = \begin{cases} 1 & \text{if local dof } \alpha \text{ on element } T_m \text{ maps to global dof } i \\ 0 & \text{otherwise} \end{cases}$$

Split Assembly on cells

$$A_{ij} = \sum_m \int_{T_m} \nabla v_j \cdot \nabla v_i dT_m = \sum_m \int_{\hat{T}} [(\nabla v_j) \circ F_m] \cdot [(\nabla v_i) \circ F_m] J_m d\hat{T}$$

$$A_{ij} = \sum_m \sum_\alpha \sum_\beta P_{mi\alpha} \int_{\hat{T}} [(DF_m^{-T} \hat{\nabla} \hat{v}_\alpha)] \cdot [DF_m^{-T} (\hat{\nabla} \hat{v}_\beta)] J_m d\hat{T} \; P_{mj\beta}$$

$$A_{ij} = \sum_m \sum_\alpha \sum_\beta \sum_q P_{mi\alpha} [(DF_m^{-T} \hat{\nabla} \hat{v}_\alpha)](\hat{x}_q) \cdot [DF_m^{-T} (\hat{\nabla} \hat{v}_\beta)](\hat{x}_q) J_m(\hat{x}_q) w_q \; P_{mj\beta}$$

$m \in [0, N_{\text{cell}})$ $\alpha, \beta \in [0, N_{\text{localdofs}})$ $i, j \in [0, N_{\text{dofs}})$ $q \in [0, N_{\text{qpoints}})$

Local VS global matrix

$$A_{ij} = \sum_m \int_{T_m} \nabla v_j \cdot \nabla v_i dT_m = \sum_m \int_{\hat{T}} [(\nabla v_j) \circ F_m] \cdot [(\nabla v_i) \circ F_m] J_m d\hat{T}$$

$$A_{ij} = \sum_m \sum_\alpha \sum_\beta \sum_q P_{mi\alpha} [(DF_m^{-T} \hat{\nabla} \hat{v}_\alpha)](\hat{x}_q) \cdot [DF_m^{-T} (\hat{\nabla} \hat{v}_\beta)](\hat{x}_q) J_m(\hat{x}_q) w_q \; P_{mj\beta}$$

$$a_{m\;\alpha\beta} := \sum_q [(DF_m^{-T} \hat{\nabla} \hat{v}_\alpha)](\hat{x}_q) \cdot [DF_m^{-T} (\hat{\nabla} \hat{v}_\beta)](\hat{x}_q) J_m(\hat{x}_q) w_q$$

$$A = \sum_m P_m^T a_m P_m$$

Local VS global right-hand-side

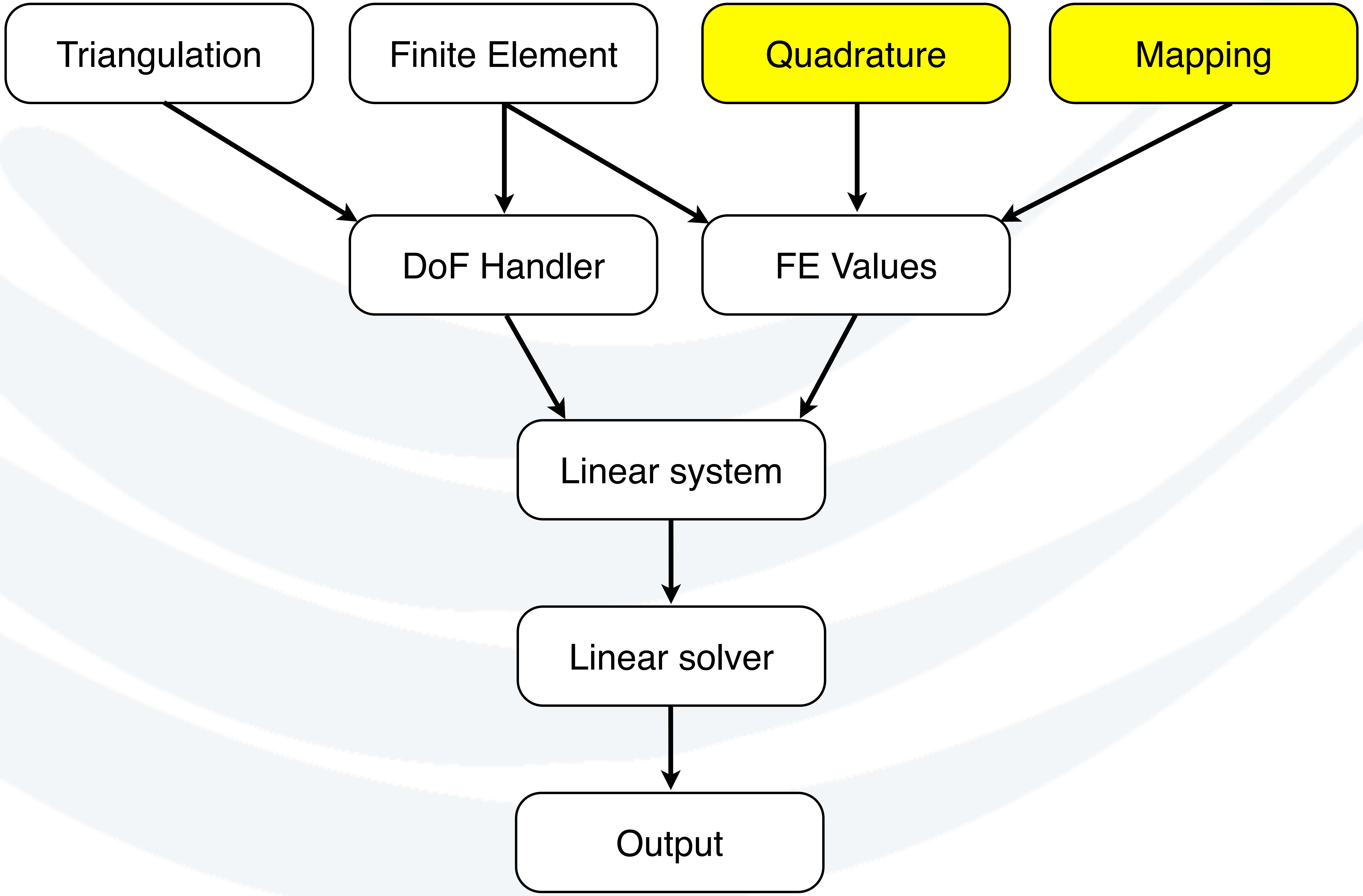
$$F_i = \sum_m \int_{T_m} f v_i dT_m = \sum_m \int_{\hat{T}} [f \circ F_m][v_i \circ F_m] J_m d\hat{T}$$

$$F_i = \sum_m \sum_\alpha \sum_q P_{mi\alpha} [f \circ F_m](\hat{x}_q) \hat{v}_\alpha(\hat{x}_q) J_m(\hat{x}_q) w_q$$

$$f_{m\ \alpha} := \sum_\alpha \sum_q [f \circ F_m](\hat{x}_q) \hat{v}_\alpha(\hat{x}_q) J_m(\hat{x}_q) w_q$$

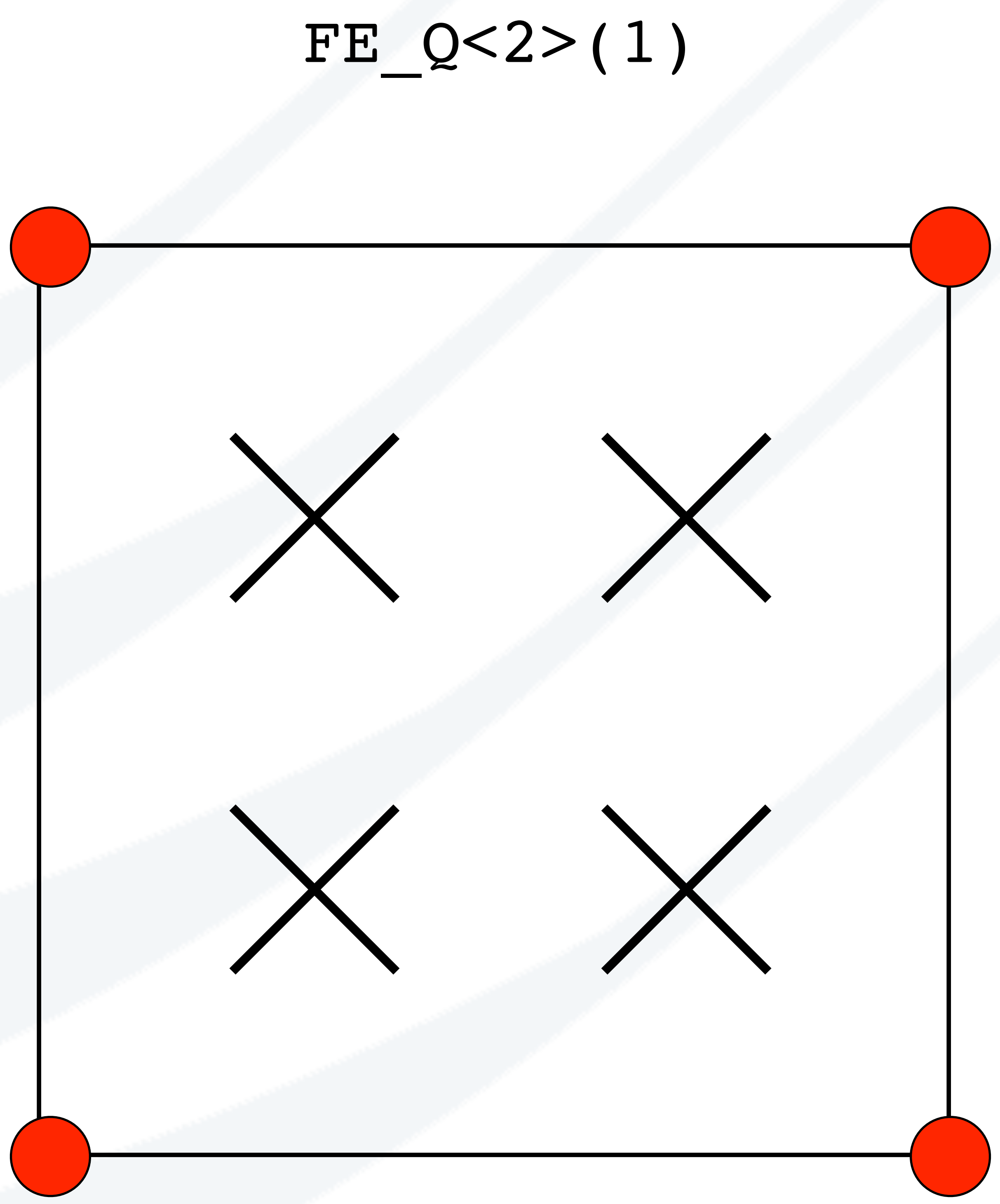
$$F = \sum_m P_m^T f_m$$

Structure of a prototypical FE problem



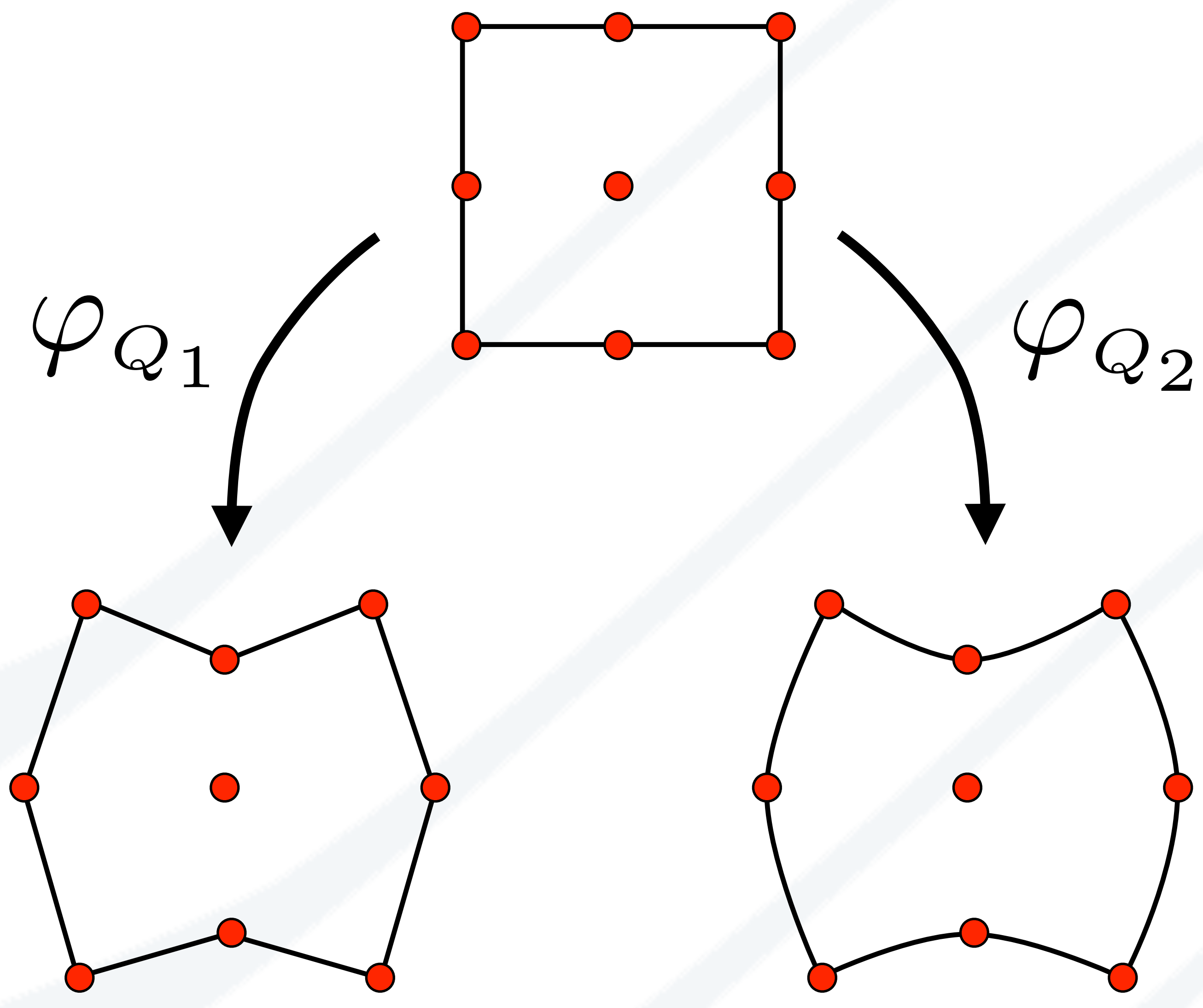
Integration on a cell: the Quadrature classes

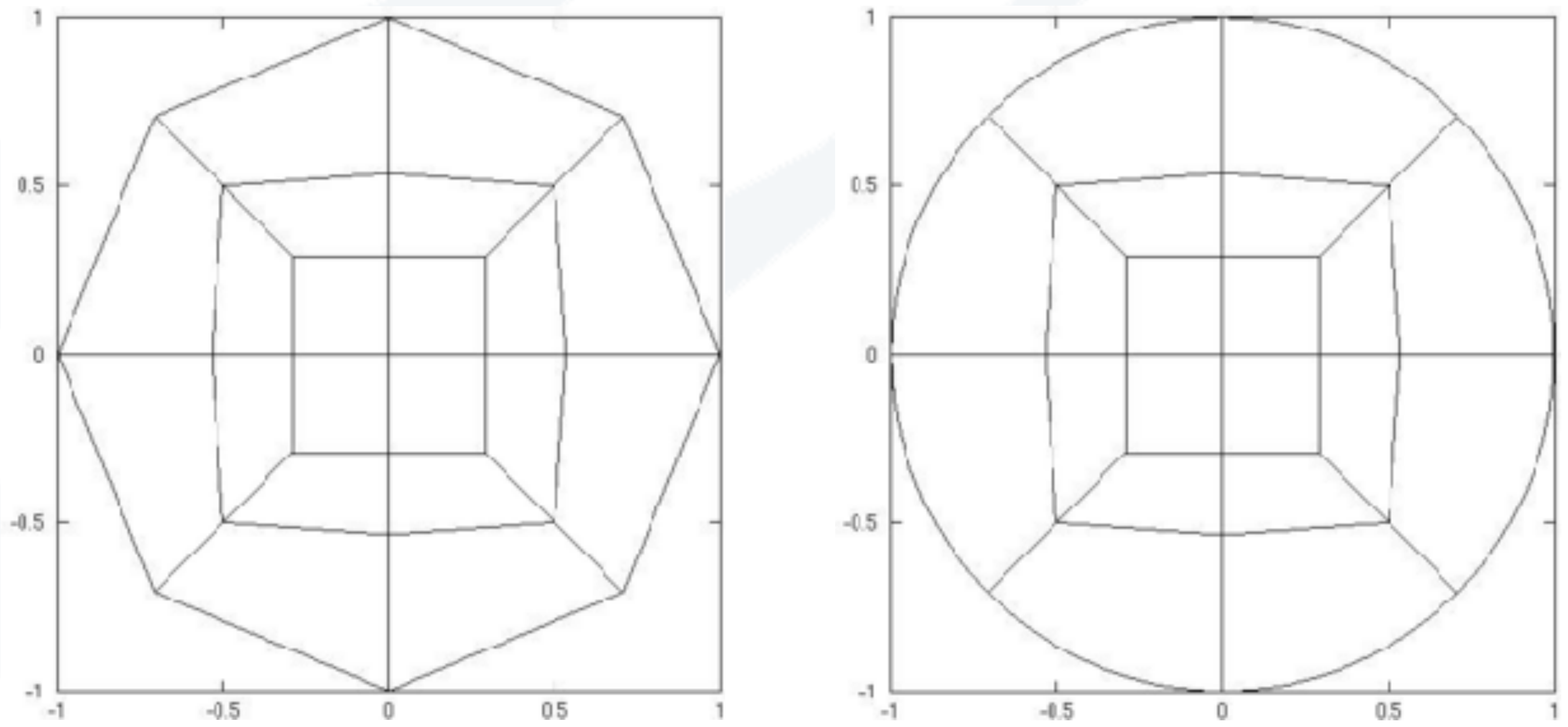
- n-Order Gauss quadrature
- Other rules
  - Gauss Lobatto
  - Simpson
  - Trapezoidal
  - Midpoint
  - A few others
- Anisotropic
  - Tensor product



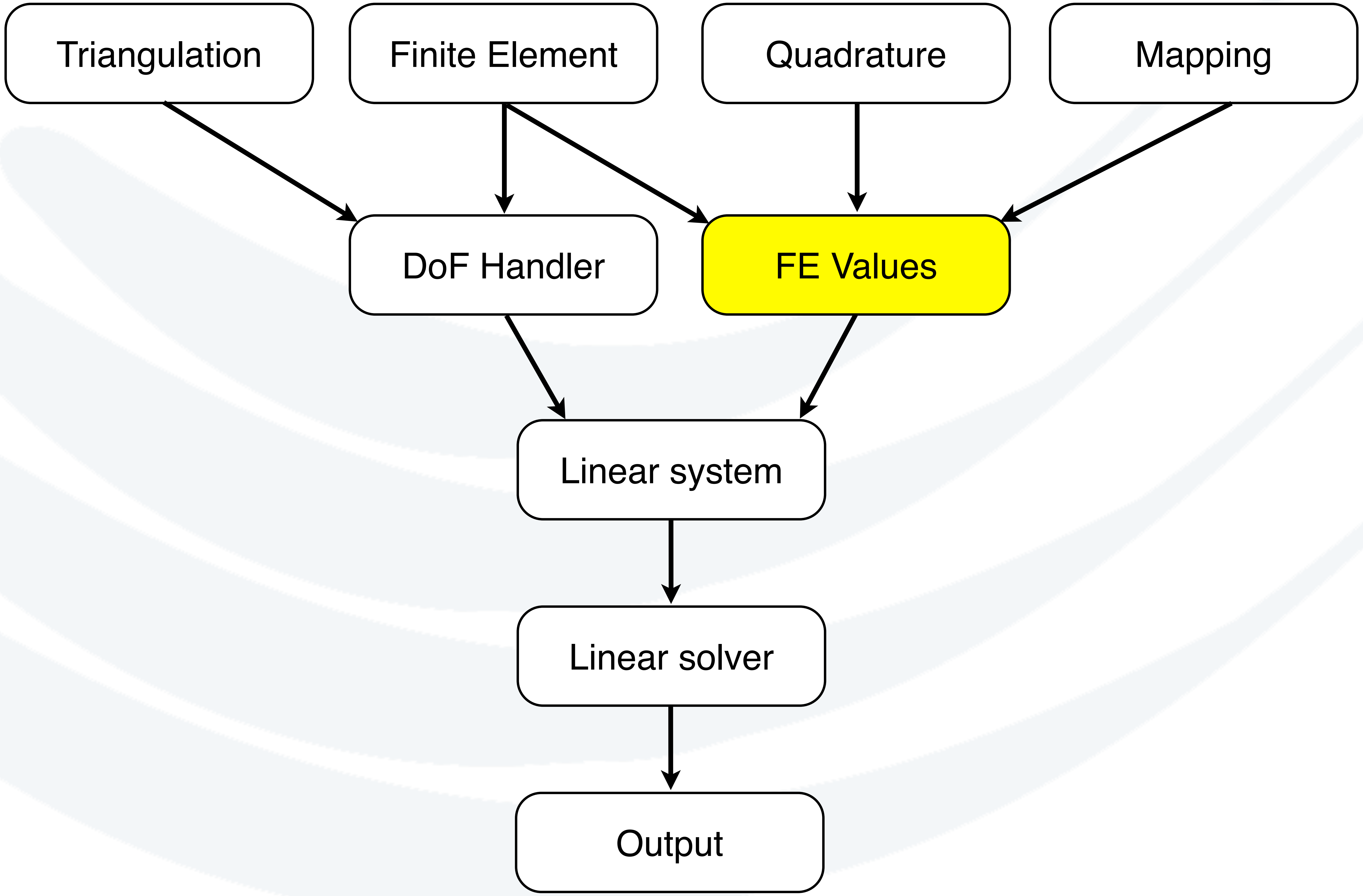
Integration on a cell: the Mapping classes

- 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



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

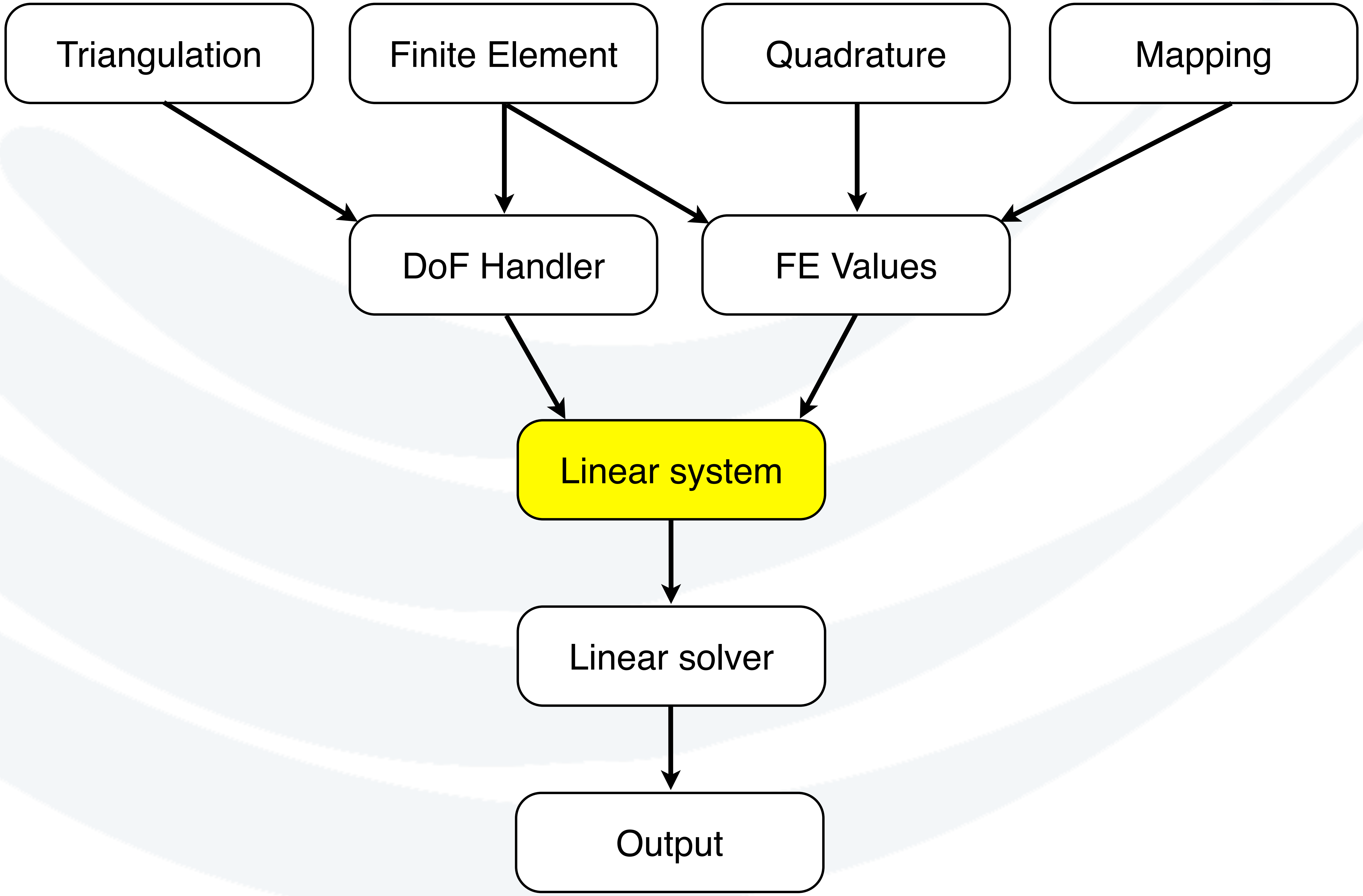
$$a_{IJ} := \sum_q [(DF_m^{-T}\hat{\nabla}\hat{v}_I)](\hat{x}_q) \cdot [DF_m^{-T}(\hat{\nabla}\hat{v}_J)](\hat{x}_q) J_m(\hat{x}_q) w_q$$

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

The Abdus Salam International Centre for Theoretical Physics

SISSA

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)

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

$$\left[\begin{array}{c|c} K_{11} & K_{12} \\ \hline K_{21} & K_{22} \end{array}\right] \left\{\begin{array}{c} d_1 \\ \hline d_2 \end{array}\right\} = \left\{\begin{array}{c} F_1 \\ \hline F_2 \end{array}\right\}$$

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

Solving Poisson’s equation

- Demonstration: Step-3
  https://www.dealii.org/current/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