Master in High Performance Computing

Continuum mechanics and fluid-structure interaction problems: mathematical modelling and numerical approximation

deal.II LAB — the Triangulation class

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)
King Abdullah University of Science and Technology (kaust.edu.sa)

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

The Abdus Salam International Centre for Theoretical Physics

SISSA

Aims for this week

- Gain familiarity with three core classes
  - **Triangulation**
  - **DoFHandler**
  - **FiniteElement**
- Create and interrogate meshes
- Create and interrogate sparsity patterns

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

The Abdus Salam
International Centre
for Theoretical Physics

SISSA

Reference material

- Main page
  https://dealii.org/current/doxygen/deal.II/index.html
- Tutorials
  - Step-1
    https://dealii.org/current/doxygen/deal.II/step_1.html
  - Step-49
    https://dealii.org/current/doxygen/deal.II/step_49.html
  - Step-2
    https://dealii.org/current/doxygen/deal.II/step_2.html

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

The Abdus Salam
International Centre
for Theoretical Physics

SISSA

First and **BIGGEST** tip

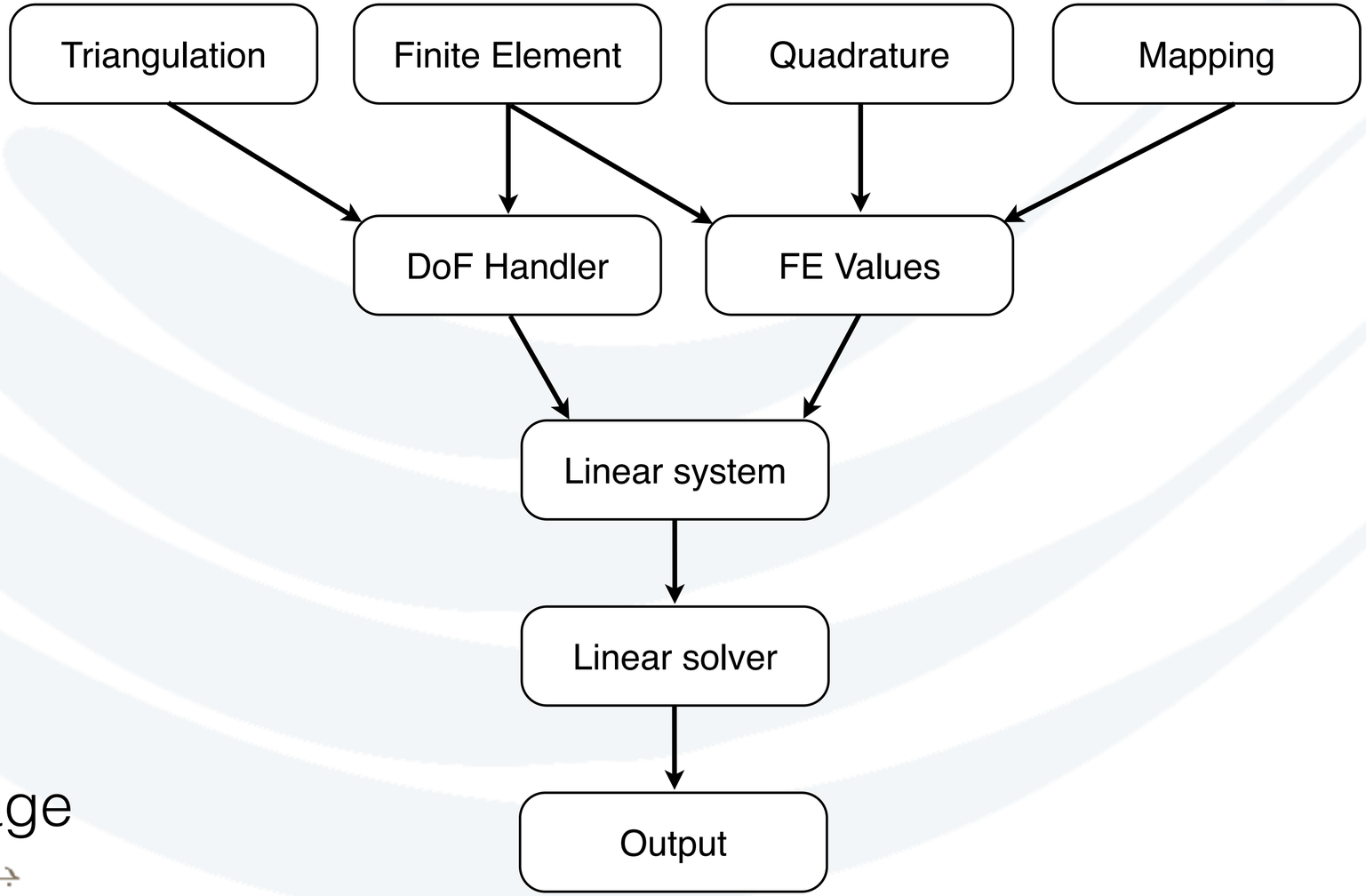
- Program **defensively**
  - Program and test in **debug** mode
    - Additional compiler warnings
    - Add assertions
  - Create papers' results in **release** mode

جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

The Abdus Salam International Centre for Theoretical Physics

SISSA

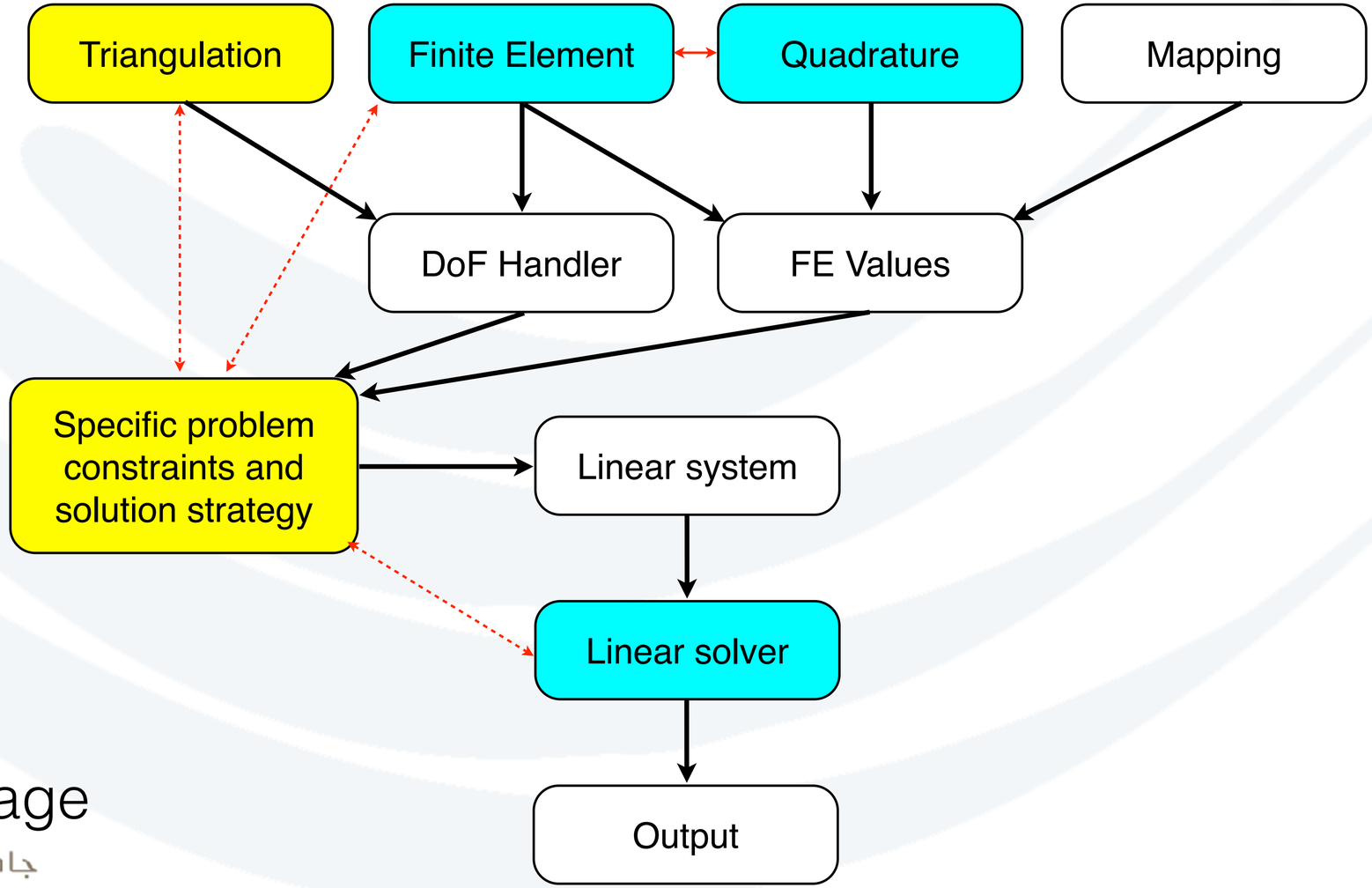
Structure of a prototypical FE problem



Main page

https://dealii.org/current/doxygen/deal.II/index.html

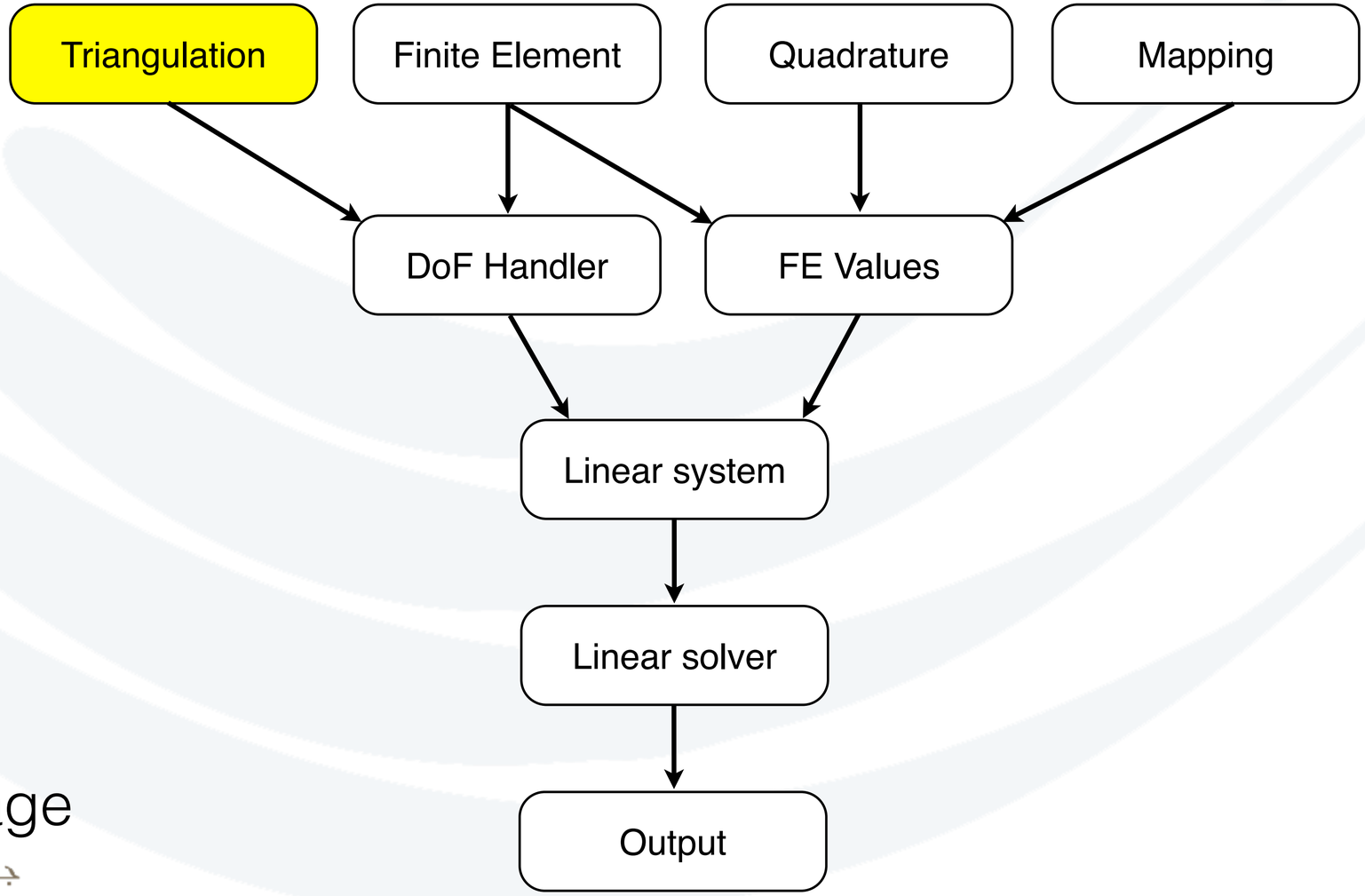
Structure of a prototypical FE problem



Main page

https://dealii.org/current/doxygen/deal.II/index.html

Structure of a prototypical FE problem

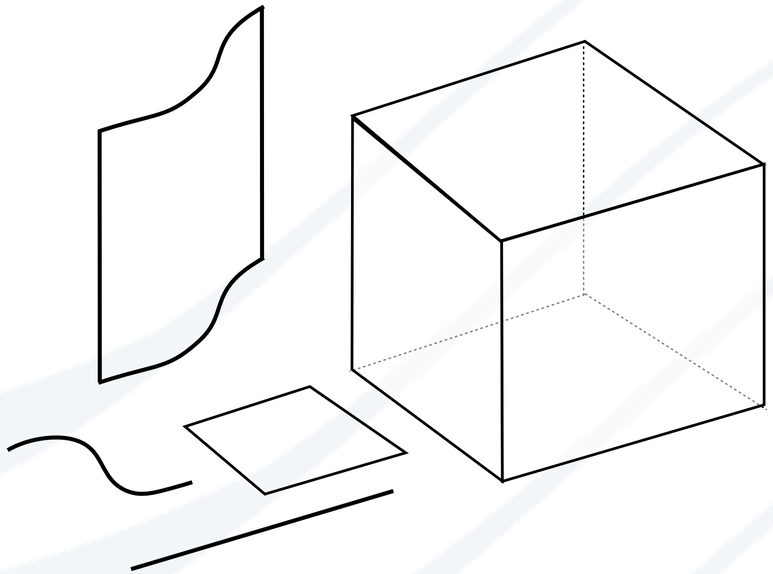


Main page

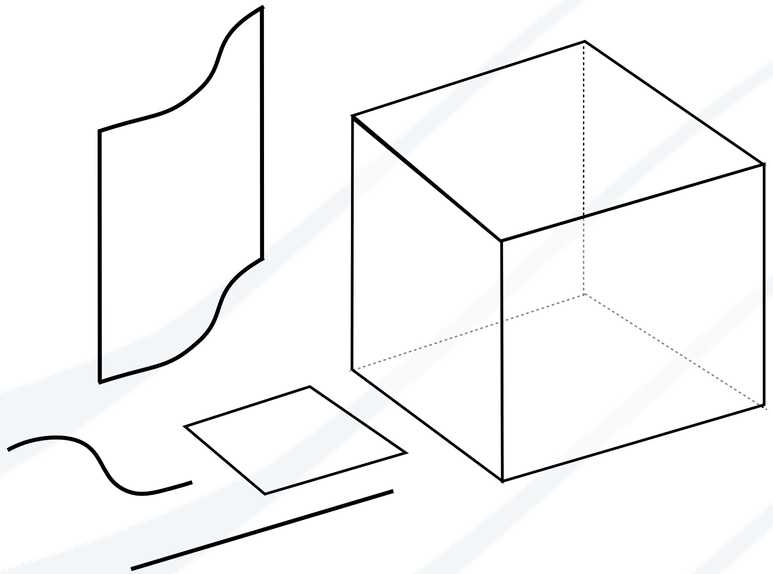
https://dealii.org/current/doxygen/deal.II/index.html

Interaction with geometry: the Triangulation class

- Describes problem geometry
  - Support for lines, quad, hex elements
  - Conceptually even higher order!
  - Structured/unstructured meshes
  - Co-dimension 1 or 2 case
- Grid creation
  - Built-in basic grid generation and manipulation tools
  - Can read in grids



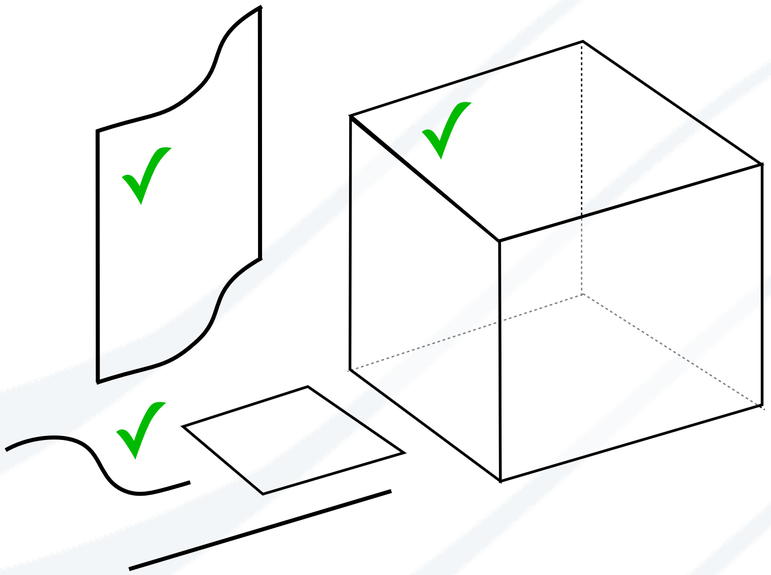
Interaction with geometry: the Triangulation class

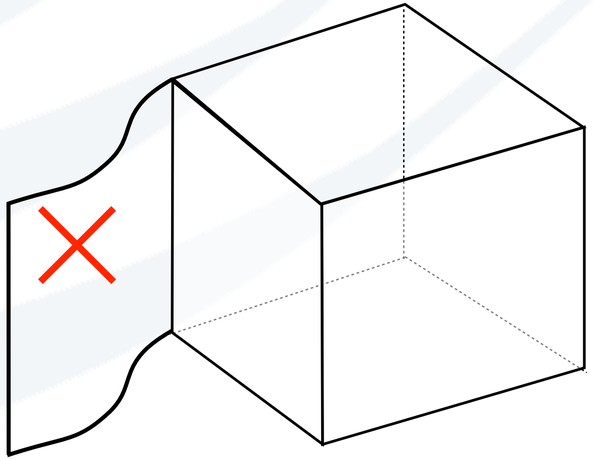


- Assign helper ID's
  - Materials
  - Boundaries
  - Manifolds
- Allows storage of custom data-structure attached to each cell/face
- Cells know about neighbour cells
  - Useful for DG methods

Interaction with geometry: the Triangulation class

- Can enforce topologies
  - Manifolds on boundary
  - Internal manifolds
- Disadvantage
  - Cannot mix triangulation types
  - e.g. Volumetric body with extended manifold surface

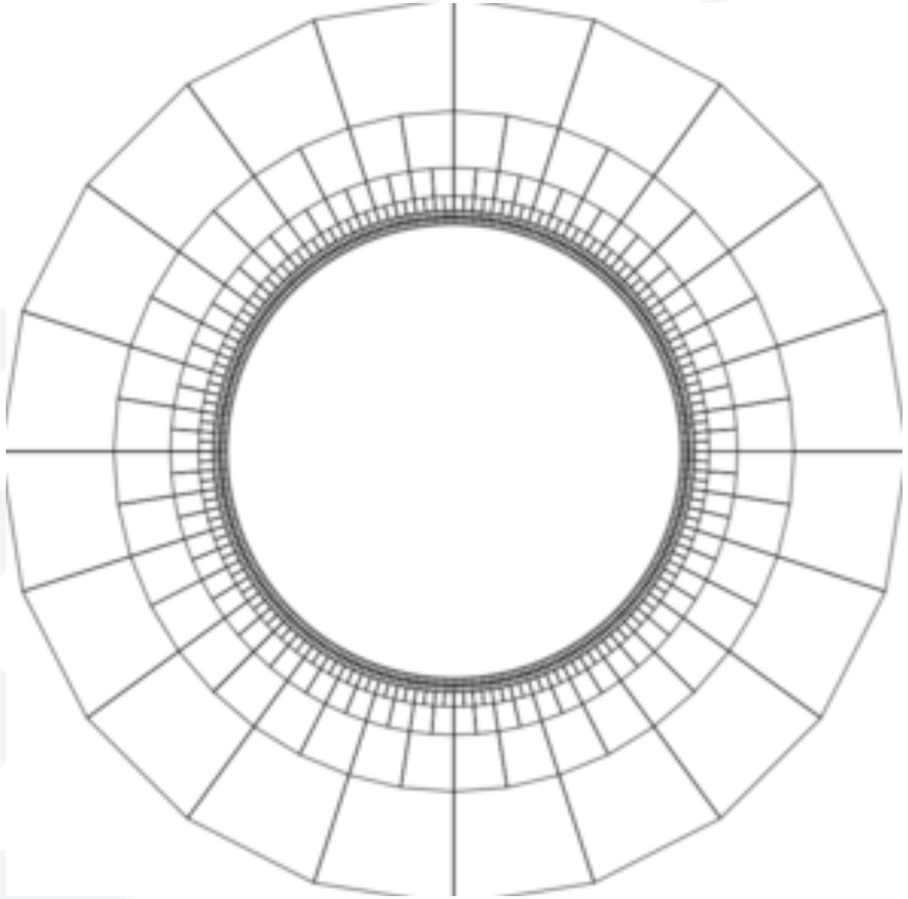




Interaction with geometry: the Triangulation class

- Demonstration: Step-1, step-49
  https://www.dealii.org/current/doxygen/deal.II/step_1.html
  https://www.dealii.org/current/doxygen/deal.II/step_49.html
  http://www.math.colostate.edu/~bangerth/videos.676.5.html
  http://www.math.colostate.edu/~bangerth/videos.676.6.html

- Key points
  - deal.II headers
  - Creating a triangulation
  - Boundary topology
  - Traversing a triangulation
  - Querying geometric information
  - Manipulating a triangulation
  - Aspects of grid refinement
  - Visualising a triangulation



Triangulation

$$\mathcal{T}_h := \cup_{m=0}^{\text{n\_active\_cells}} \{K_m\}$$

- Mesh/**Triangulation**/Grid: a collection of **cells, faces,** and **vertices**
- if the mesh has been refined (possibly in an adaptive way):
  - hierarchy of refinement levels
- **Triangulation** in deal.II only stores the geometry
- Main Reference: https://www.dealii.org/current/doxygen/deal.II/group__grid.html

Iterators and Accessors

$$\mathcal{T}_h := \cup_{m=0}^{\text{n\_active\_cells}} \{K_m\}$$

- An **iterator** in **C++** is an object that enables programmers to traverse a container,
- deal.II gives access to cells via **iterators:** they are pointers to objects that give you **access** to information
  - increment operation: `operator ++`
  - decremented operation: `operator--`
  - **`begin()`**/**`begin_*()`**: first element of a collection
  - **`end()`**: one-past-the-end iterator
  - **access** operator: the **->** command, i.e., **cell->center()**

Iterators and Accessors

$$\mathcal{T}_h := \cup_{m=0}^{\text{n\_active\_cells}} \{K_m\}$$

- An **accessor** in **C++** is an object that *hides* the internal details of a (possibly complex, hierarchical, distributed, etc.) container class, and gives you *access* to *information*

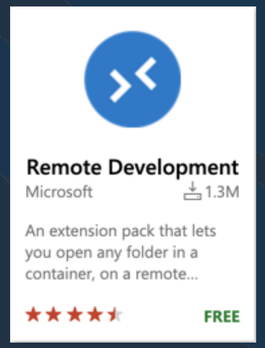
- deal.II **iterators** acts as pointers to **accessors:**

  - Accessors in deal.II: **lines, quads, tria, hexes, tetra, etc.**

  - `Ref:` https://www.dealii.org/developer/doxygen/deal.II/group__Iterators.html

Setting up VSCode

- Download and install **Docker**: https://www.docker.com/products/docker-desktop
  - Read some doc: https://www.docker.com/get-started
- Download and install: https://code.visualstudio.com/download
  - Read some doc: https://code.visualstudio.com/docs
  - Install the following extension:



Open the course repository

- Clone the repository of the course to a directory of your liking
- Open the directory containing the repository with Visual Studio Code (the directory contains a hidden folder, called “.**devcontainer”,** used by VSCode to understand the
- VSCode should ask you if you want to reopen the folder within a container. Say yes.
- VSCode will now download a docker image. The first time around, this will take some time.