Lecture 6:
Adaptive Mesh Refinment

Luca Heltai (luca.heltai@sissa.it)

Aims for this module

- Implement adaptive mesh refinement
  - Hanging notes
  - Error-based refinement marking
- Learn about the AffineConstraints

Adaptive mesh refinement (AMR)

Note: The optimal strategy to minimize the error while keeping the problem as small as possible is to *equilibrate* the local contributions

$$e_K = C\, h_K \|u\|_{H^2(K)}$$

That is, we want to choose

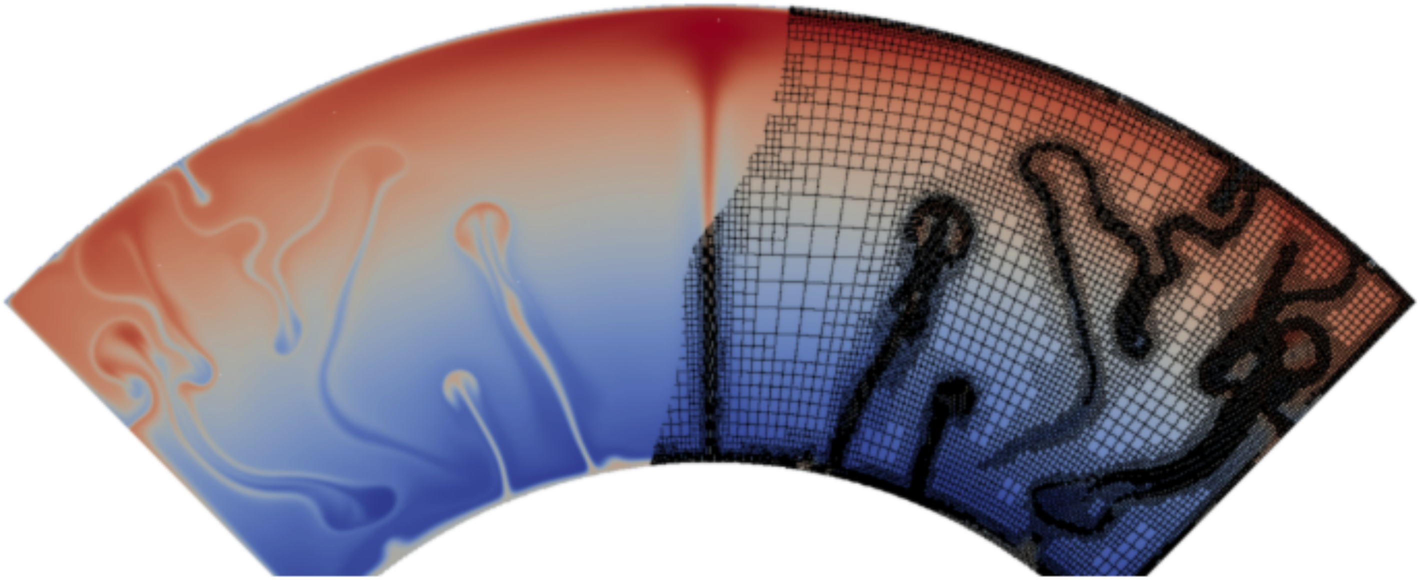
$$h_K \propto \frac{1}{\|u\|_{H^2(K)}}$$

In practice: Exact errors are unknown. Thus, use a local *indicator* of the error η_K and choose h_K so that

$$\sum_K \eta_K = \mathrm{tol}$$

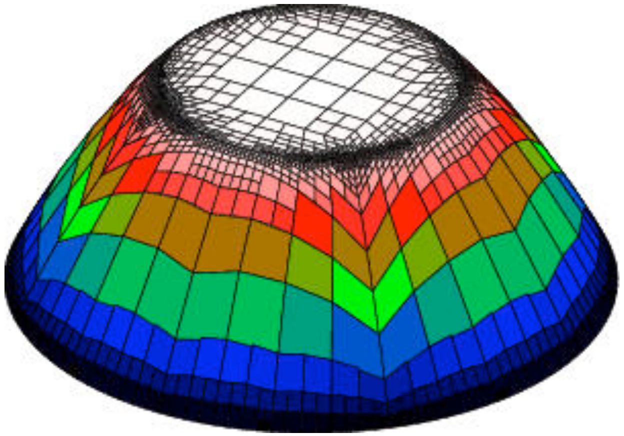
Adaptive mesh refinement (AMR)

Example:



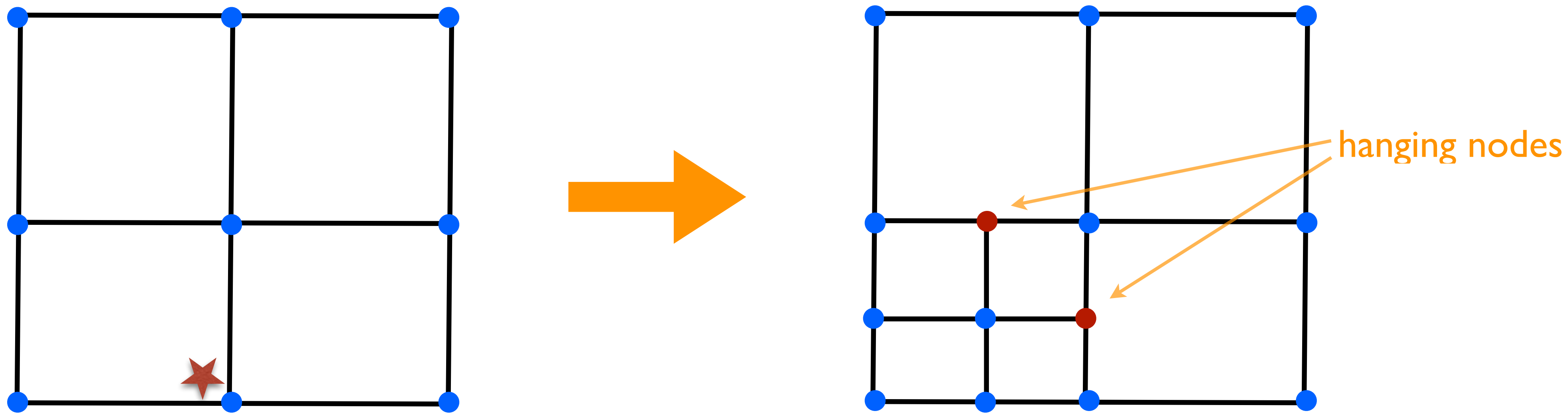
Refine only where “something is happening” (i.e., locally the second derivative of the solution is large).

Adaptive mesh refinement



- Typical loop
  - Solve (non-)linear system
  - Estimate error
  - Mark cells
  - Refine/coarsen
- Error estimate is problem dependent:
  - Approximate gradient jumps: `KellyErrorEstimator` class
  - Approximate local norm of gradient: `DerivativeApproximation` class
  - Or something else
- Cell marking strategy:
  - `GridRefinement::refine_and_coarsen_fixed_number(...)`
  - `GridRefinement::refine_and_coarsen_fixed_fraction(...)`
- Refine/coarsen grid: `triangulation.execute_coarsening_and_refinement ()`
- Transferring the solution: `SolutionTransfer` class (discussed later)

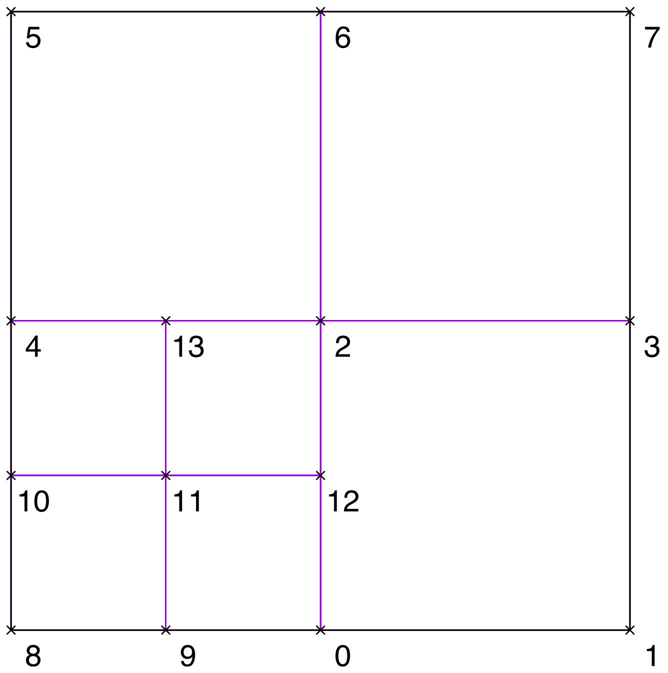
Adaptive mesh refinement (quad/hex)



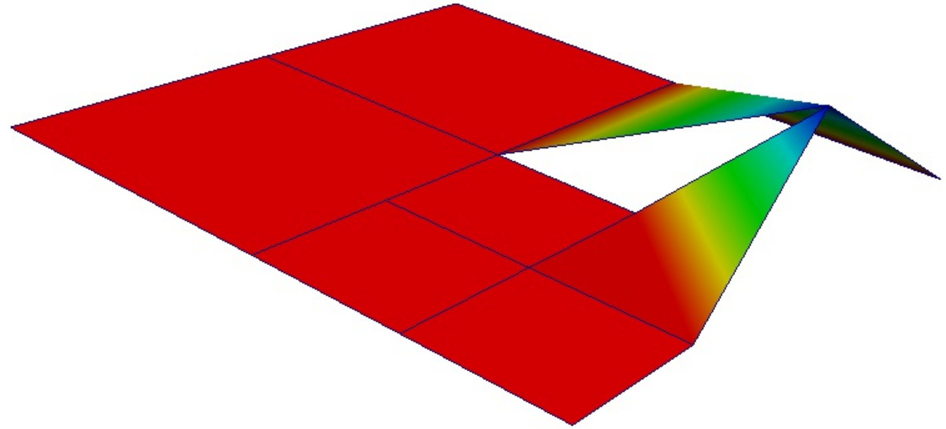
no easy options to keep the children of the top left/bottom right cell as quadrilaterals!

Solution: keep hanging nodes and “deal” with them in another way

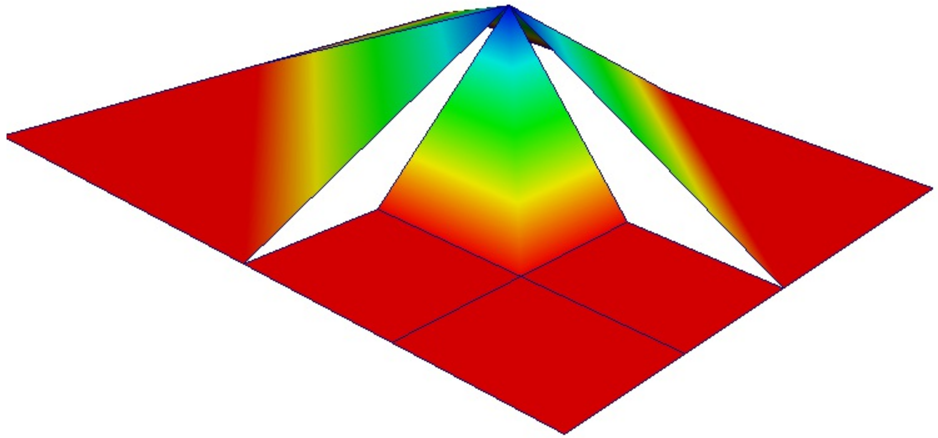
Hanging nodes



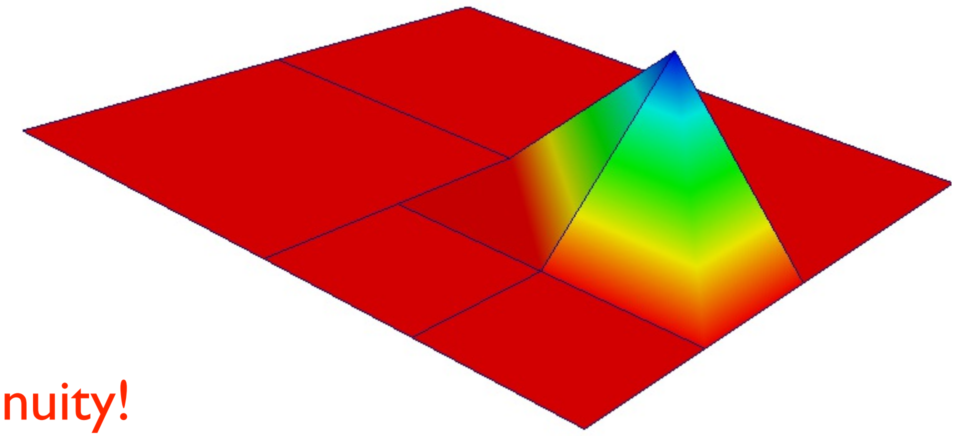
$N_0(\mathbf{x})$:



$N_2(\mathbf{x})$:



$N_{12}(\mathbf{x})$:



Discontinuous FE space!

Not a subspace of H^1

Bilinear forms would require special treatment as gradients are not defined everywhere

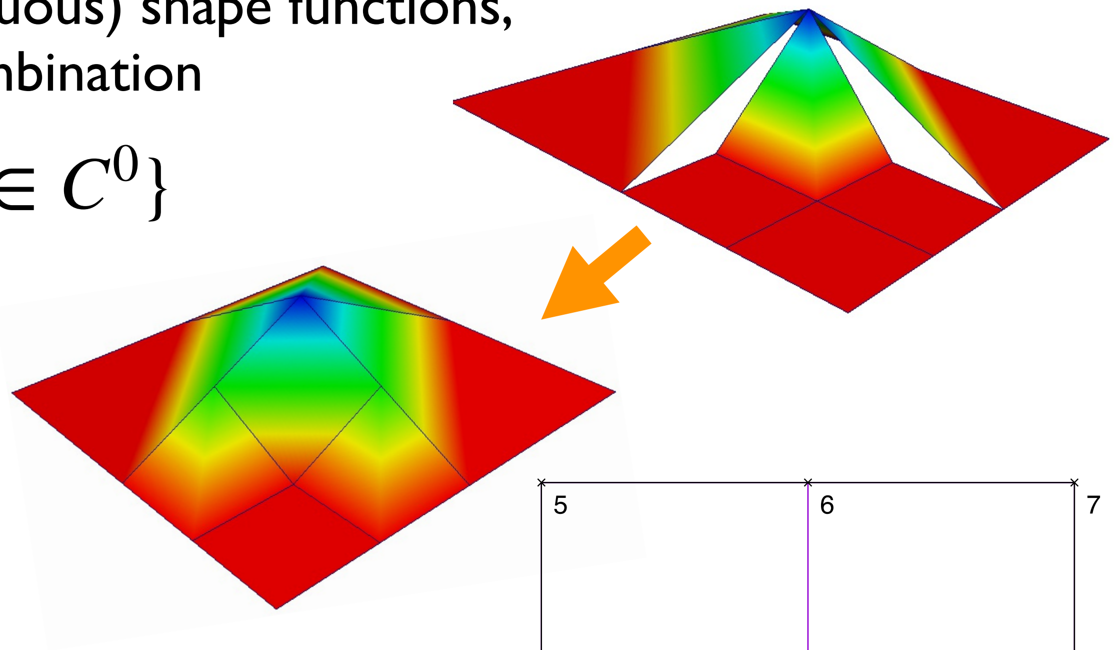
$$a(\phi_i, \phi_j) = \int_\Omega \nabla N_i(\mathbf{x}) \cdot \nabla N_j(\mathbf{x}) \mathrm{dv}$$

Solution: introduce constraints to require continuity!

Hanging nodes

Use standard (possibly globally discontinuous) shape functions, but require continuity of their linear combination

$$\mathcal{S}^h = \{u^h = \sum_i u_i N_i(\mathbf{x}) : u^h(\mathbf{x}) \in C^0\}$$



Note, that we encounter discontinuities along edges 0-12-2 and 2-13-4.

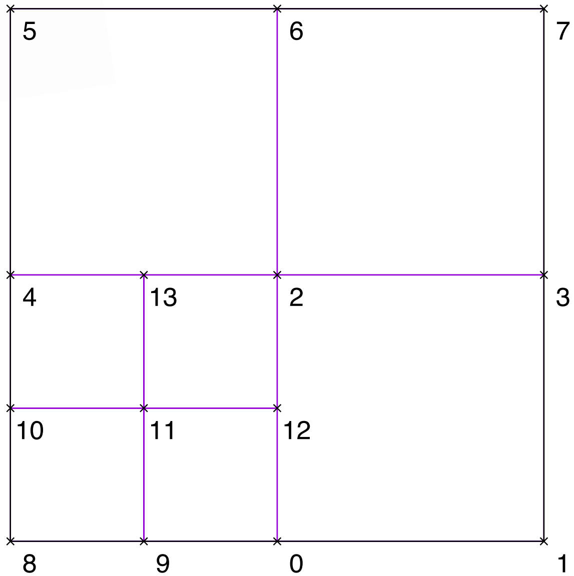
We can make the function continuous by making it continuous at vertices 12 and 13:

$$u_{12} = \frac{1}{2}u_0 + \frac{1}{2}u_2$$

$$u_{13} = \frac{1}{2}u_2 + \frac{1}{2}u_4$$

define a subset of all DoFs to be constrained

$$\mathcal{N}_C \subset \mathcal{N}$$



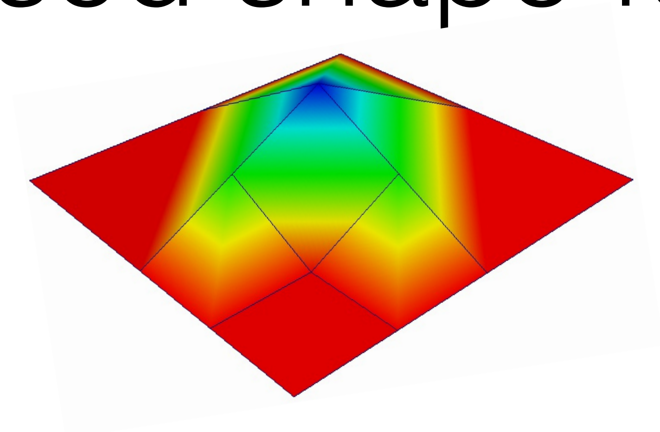
The general form:

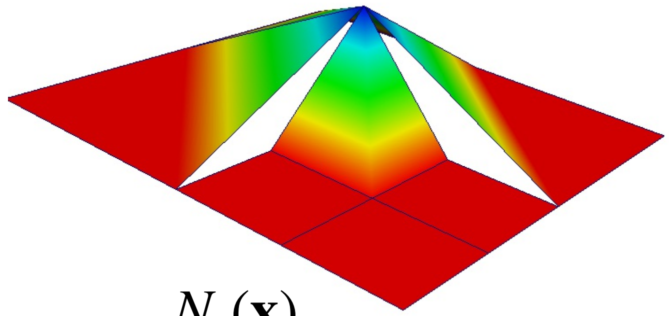
$$u_i = \sum_{j \in \mathcal{N}} c_{ij} u_j + b_i \quad \forall i \in \mathcal{N}_C$$

similar constraints arise from boundary conditions (normal/tangential component) or hp-adaptive FE

Condensed shape functions

The alternative viewpoint is to construct a set of conforming shape functions:

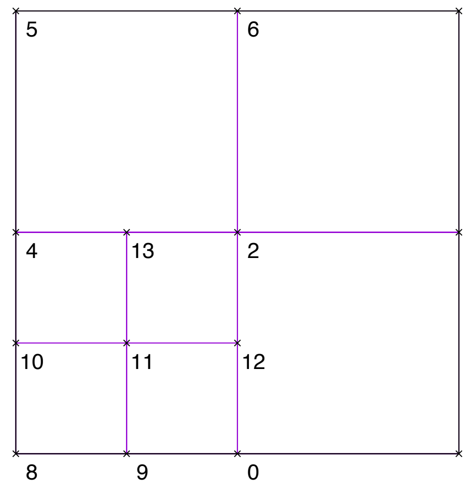




$N_2(\mathbf{x})$

$$\widetilde{N}_2 := N_2 + \frac{1}{2}N_{13} + \frac{1}{2}N_{12}$$

$$\mathcal{S}^h = \{u^h = \sum_{i \in \mathcal{N}/\mathcal{N}_c} u_i \widetilde{N}_i(\mathbf{x})\}$$



$$[\boldsymbol{K}]_{ij} = \begin{cases} a(\widetilde{N}_i, \widetilde{N}_j) & \text{if } i \in \mathcal{N} \setminus \mathcal{N}_c \text{ and } j \in \mathcal{N} \setminus \mathcal{N}_c \\ 1 & \text{if } i \equiv j \text{ and } j \in \mathcal{N}_c \\ 0 & \text{otherwise} \end{cases}$$

$$[\boldsymbol{F}]_i = \begin{cases} (f, \widetilde{N}_i) & \text{if } i \in \mathcal{N} \setminus \mathcal{N}_c \\ 0 & \text{otherwise} \end{cases}$$

The beauty of the approach is that we can assemble local matrix and RHS as usual and then obtain condensed forms in a separate step, i.e

$$\forall i \in \mathcal{N} \setminus \mathcal{N}_c : \quad [\mathbf{F}]_i = (f, \widetilde{N}_i) = (f, N_i + \sum_{j \in \mathcal{N}_c} c_{ji} N_j) = (f, N_i) + \sum_{j \in \mathcal{N}_c} c_{ji}(f, N_j) = \left[\widetilde{\mathbf{F}}\right]_i + \sum_{j \in \mathcal{N}_c} c_{ji} \left[\widetilde{\mathbf{F}}\right]_j$$

Algebraic form

$$\boldsymbol{x} =: \begin{bmatrix} \boldsymbol{x}_u \\ \boldsymbol{x}_m \\ \boldsymbol{x}_c \end{bmatrix}$$

not involved in constraints

master dofs in constraints

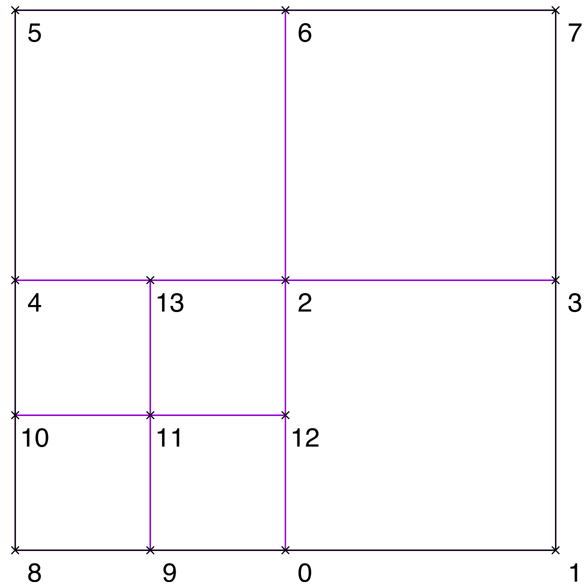
constrained dofs

$$\begin{bmatrix} u_{12} \\ u_{13} \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} u_0 \\ u_2 \\ u_4 \end{bmatrix}$$

$$\boldsymbol{x}_c = \boldsymbol{C}\boldsymbol{x}_m$$

$$\boldsymbol{\Gamma} := \begin{bmatrix} \boldsymbol{I}_u & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_m \\ \boldsymbol{0} & \boldsymbol{C} \end{bmatrix}$$

transformation matrix



The condensed matrix and vector can be written as

$$\boldsymbol{K} = \boldsymbol{\Gamma}^T \widetilde{\boldsymbol{K}} \boldsymbol{\Gamma} \qquad \boldsymbol{F} = \boldsymbol{\Gamma}^T \widetilde{\boldsymbol{F}}$$

To look closer at what condensation does, write the original matrix/vector in block form:

$$\widetilde{\boldsymbol{K}} =: \begin{bmatrix} \boldsymbol{K}_{uu} & \boldsymbol{K}_{um} & \boldsymbol{K}_{uc} \\ \boldsymbol{K}_{mu} & \boldsymbol{K}_{mm} & \boldsymbol{K}_{mc} \\ \boldsymbol{K}_{cu} & \boldsymbol{K}_{cm} & \boldsymbol{K}_{cc} \end{bmatrix} \qquad \widetilde{\boldsymbol{F}} =: \begin{bmatrix} \boldsymbol{F}_u \\ \boldsymbol{F}_m \\ \boldsymbol{F}_c \end{bmatrix} \qquad \boldsymbol{F} =: \begin{bmatrix} \boldsymbol{F}_u \\ \boldsymbol{F}_m + \boldsymbol{C}^T \boldsymbol{F}_c \end{bmatrix}$$

$$\boldsymbol{K} =: \begin{bmatrix} \boldsymbol{K}_{uu} & \boldsymbol{K}_{um} + \boldsymbol{K}_{uc}\boldsymbol{C} \\ \boldsymbol{K}_{mu} + \boldsymbol{C}^T \boldsymbol{K}_{cu} & \boldsymbol{K}_{mm} + \boldsymbol{C}^T \boldsymbol{K}_{cm} + \boldsymbol{K}_{mc}\boldsymbol{C} + \boldsymbol{C}^T \boldsymbol{K}_{cc}\boldsymbol{C} \end{bmatrix}$$

Using constraints:

- The beauty of the FEM is that we do exactly the same thing on every cell
- In other words: assembly on cells with hanging nodes should work exactly as on cells without

Approach 1 (step-3):

$$\widetilde{\mathcal{S}}^h = \{u^h = \sum_i u_i N_i(x)\}$$

this is not a continuous space, but we may still use it as an intermediate step for matrices!

$$\mathcal{S}^h = \{u^h = \sum_i u_i N_i(x) : u^h(x) \in C^0\}$$

Step 1: Build matrix/rhs $\widetilde{\mathbf{K}}, \widetilde{\mathbf{F}}$ with all DoFs as if there were no constraints.

Step 2: Modify $\widetilde{\mathbf{K}}, \widetilde{\mathbf{F}}$ to get $\mathbf{K}, \mathbf{F}$

i.e. eliminate the rows and columns of the matrix that correspond to constrained degrees of freedom

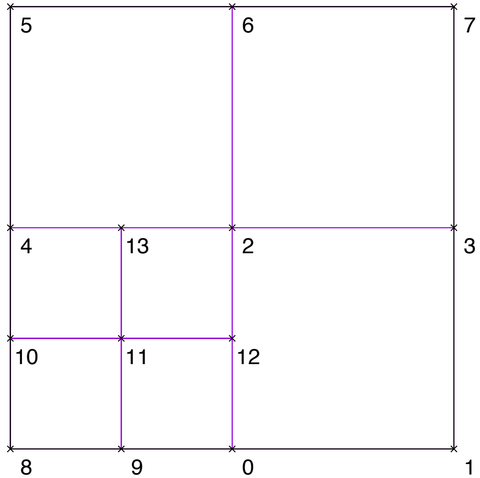
Step 3: Solve $\mathbf{K} \cdot \mathbf{u} = \mathbf{F}$

Step 4: Fill in the constrained components of $\mathbf{u}$ to use $\widetilde{\mathcal{S}}^h$ for evaluation of the field.

Disadvantages: (i) bottleneck for 3d or higher order/hp FEM; (ii) hard to implement in parallel where a process may not have access to all elements of the matrix; (iii) two matrices may have different sparsity pattern.

Approach 1 (example):

$$\begin{bmatrix} u_{12} \\ u_{13} \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} u_0 \\ u_2 \\ u_4 \end{bmatrix}$$




```
============================================
Number of active cells: 7
Number of degrees of freedom: 14
=============== constraints ===============
    12 0:  0.5
    12 2:  0.5
    13 2:  0.5
    13 4:  0.5
============== un-condensed ===============
================= matrix ==================
 1.333e+00 -1.667e-01 -1.667e-01 -3.333e-01  0.000e+00                                             -1.667e-01            -3.333e-01 -1.667e-01
-1.667e-01  6.667e-01 -3.333e-01 -1.667e-01
-1.667e-01 -3.333e-01  2.667e+00 -3.333e-01 -1.667e-01 -3.333e-01 -3.333e-01 -3.333e-01             0.000e+00  0.000e+00 -3.333e-01 -1.667e-01 -1.667e-01
-3.333e-01 -1.667e-01 -3.333e-01  1.333e+00                       -3.333e-01 -1.667e-01
 0.000e+00            -1.667e-01             1.333e+00 -1.667e-01 -3.333e-01                                  -1.667e-01 -3.333e-01            -1.667e-01
                      -3.333e-01            -1.667e-01  6.667e-01 -1.667e-01
                      -3.333e-01 -3.333e-01 -3.333e-01 -1.667e-01  1.333e+00 -1.667e-01
                      -3.333e-01 -1.667e-01                       -1.667e-01  6.667e-01
                                                                                         6.667e-01 -1.667e-01 -1.667e-01 -3.333e-01
-1.667e-01             0.000e+00                                                        -1.667e-01  1.333e+00 -3.333e-01 -3.333e-01 -3.333e-01
                       0.000e+00            -1.667e-01                                  -1.667e-01 -3.333e-01  1.333e+00 -3.333e-01            -3.333e-01
-3.333e-01            -3.333e-01            -3.333e-01                                  -3.333e-01 -3.333e-01 -3.333e-01  2.667e+00 -3.333e-01 -3.333e-01
-1.667e-01            -1.667e-01                                                                   -3.333e-01            -3.333e-01  1.333e+00 -3.333e-01
                      -1.667e-01            -1.667e-01                                                        -3.333e-01 -3.333e-01 -3.333e-01  1.333e+00
```

```
=============== condensed ================
================= matrix ==================
 1.500e+00 -1.667e-01 -8.333e-02 -3.333e-01 -8.333e-02                                             -3.333e-01            -5.000e-01  0.000e+00
-1.667e-01  6.667e-01 -3.333e-01 -1.667e-01
-8.333e-02 -3.333e-01  2.833e+00 -3.333e-01 -8.333e-02 -3.333e-01 -3.333e-01 -3.333e-01            -1.667e-01 -1.667e-01 -6.667e-01  0.000e+00  0.000e+00
-3.333e-01 -1.667e-01 -3.333e-01  1.333e+00                       -3.333e-01 -1.667e-01
-8.333e-02            -8.333e-02             1.500e+00 -1.667e-01 -3.333e-01                                  -3.333e-01 -5.000e-01             0.000e+00
                      -3.333e-01            -1.667e-01  6.667e-01 -1.667e-01
                      -3.333e-01 -3.333e-01 -3.333e-01 -1.667e-01  1.333e+00 -1.667e-01
                      -3.333e-01 -1.667e-01                       -1.667e-01  6.667e-01
                                                                                         6.667e-01 -1.667e-01 -1.667e-01 -3.333e-01
-3.333e-01            -1.667e-01                                                        -1.667e-01  1.333e+00 -3.333e-01 -3.333e-01  0.000e+00
                      -1.667e-01            -3.333e-01                                  -1.667e-01 -3.333e-01  1.333e+00 -3.333e-01             0.000e+00
-5.000e-01            -6.667e-01            -5.000e-01                                  -3.333e-01 -3.333e-01 -3.333e-01  2.667e+00  0.000e+00  0.000e+00
 0.000e+00             0.000e+00                                                                    0.000e+00             0.000e+00  1.333e+00  0.000e+00
                       0.000e+00             0.000e+00                                                         0.000e+00  0.000e+00  0.000e+00  1.333e+00
```

Approach 2 (step-6):

$$\widetilde{\mathscr{S}}^h = \{u^h = \sum_i u_i N_i(x)\}$$

$$\mathscr{S}^h = \{u^h = \sum_i u_i N_i(x) : u^h(x) \ \in C^0\}$$

Step 1: Build local matrix/rhs $\widetilde{\mathbf{K}}_K, \widetilde{\mathbf{F}}_K$ with all DoFs as if there were no constraints.

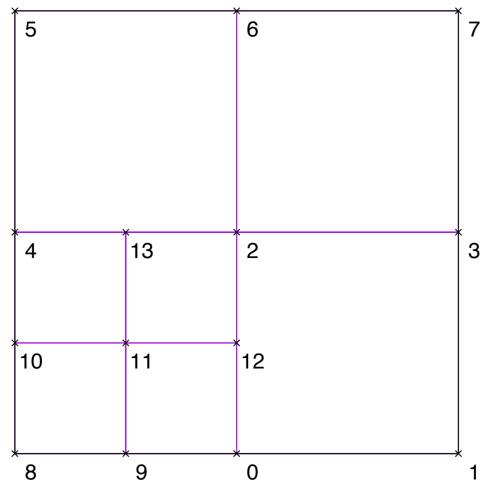
Step 2: Apply constraints during assembly operation (local-to-global) $\mathbf{K}_K, \mathbf{F}_K$

Step 3: Solve $\mathbf{K} \cdot \mathbf{u} = \mathbf{F}$

Step 4: Fill in the constrained components of $\mathbf{u}$ to use $\widetilde{\mathscr{S}}^h$ for evaluation of the field.

Approach 2 (example):

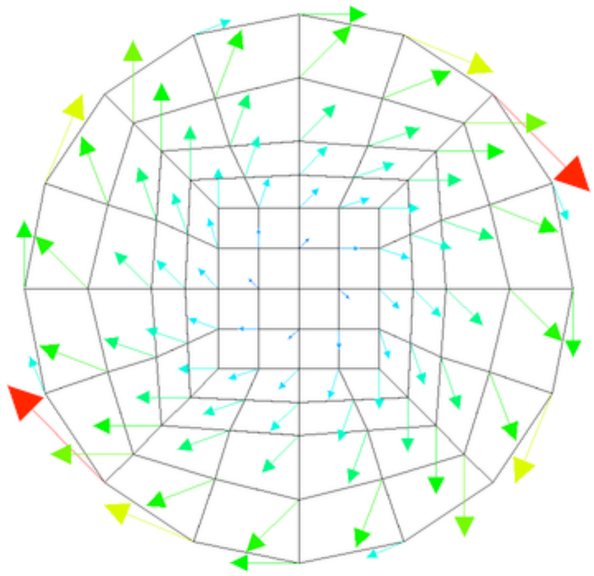
$$\begin{bmatrix} u_{12} \\ u_{13} \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0 & 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} u_0 \\ u_2 \\ u_4 \end{bmatrix}$$




```
=============================================
Number of active cells: 7
Number of degrees of freedom: 14
=============== constraints ===============
    12 0:  0.5
    12 2:  0.5
    13 2:  0.5
    13 4:  0.5
================ condensed ================
================= matrix ==================
 1.500e+00 -1.667e-01 -8.333e-02 -3.333e-01 -8.333e-02                                             -3.333e-01            -5.000e-01  0.000e+00
-1.667e-01  6.667e-01 -3.333e-01 -1.667e-01
-8.333e-02 -3.333e-01  2.833e+00 -3.333e-01 -8.333e-02 -3.333e-01 -3.333e-01 -3.333e-01            -1.667e-01 -1.667e-01 -6.667e-01  0.000e+00  0.000e+00
-3.333e-01 -1.667e-01 -3.333e-01  1.333e+00                       -3.333e-01 -1.667e-01
-8.333e-02            -8.333e-02             1.500e+00 -1.667e-01 -3.333e-01                                  -3.333e-01 -5.000e-01             0.000e+00
                      -3.333e-01            -1.667e-01  6.667e-01 -1.667e-01
                      -3.333e-01 -3.333e-01 -3.333e-01 -1.667e-01  1.333e+00 -1.667e-01
                      -3.333e-01 -1.667e-01                       -1.667e-01  6.667e-01
                                                                                         6.667e-01 -1.667e-01 -1.667e-01 -3.333e-01
-3.333e-01            -1.667e-01                                                        -1.667e-01  1.333e+00 -3.333e-01 -3.333e-01  0.000e+00
                      -1.667e-01            -3.333e-01                                  -1.667e-01 -3.333e-01  1.333e+00 -3.333e-01             0.000e+00
-5.000e-01            -6.667e-01            -5.000e-01                                  -3.333e-01 -3.333e-01 -3.333e-01  2.667e+00  0.000e+00  0.000e+00
 0.000e+00             0.000e+00                                                                   0.000e+00             0.000e+00  1.333e+00  0.000e+00
                       0.000e+00             0.000e+00                                                        0.000e+00  0.000e+00  0.000e+00  1.333e+00
```

Applying constraints: the AffineConstraints class

- This class is used for
  - Hanging nodes
  - Dirichlet and periodic constraints
  - Other constraints
- Linear constraints of the the form



Applying constraints: the AffineConstraints class

- System setup
  - Hanging node constraints created using
    `DoFTools::make_hanging_node_constraints()`
  - Will also use for boundary values from now on:
    `VectorTools::interpolate_boundary_values(..., constraints);`
  - Need different SparsityPattern creator
    `DoFTools::make_sparsity_pattern (..., constraints, …)`
    - Can remove constraints from linear system
      `DoFTools::make_sparsity_pattern (..., constraints, / *keep_constrained_dofs = * / false)`
  - Sorted, rearrange, optimise constraints
    `constraints.close()`

Applying constraints: the AffineConstraints class

- Assembly
  - Assemble local matrix and vector as normal
  - Eliminate while transferring to global matrix:
```
constraints.distribute_local_to_global (
  cell_matrix, cell_rhs,
  local_dof_indices,
  system_matrix, system_rhs);
```
  - Solve and then set all constraint values correctly:
```
ConstraintMatrix::distribute(...)
```

deal.II methods and classes for AMR

- Error estimate is problem dependent:
  - Approximate gradient jumps: `KellyErrorEstimator` class
  - Approximate local norm of gradient: `DerivativeApproximation` class
  - … or something else
- Cell marking strategy:
  - `GridRefinement::refine_and_coarsen_fixed_number(...)`
  - `GridRefinement::refine_and_coarsen_fixed_fraction(...)`
  - `GridRefinement::refine_and_coarsen_optimize(...)`
- Refine/coarsen grid:
  `triangulation.execute_coarsening_and_refinement ()`
- Transferring the solution: `SolutionTransfer` class (discussed later)

Reference material

- Tutorials
  - https://dealii.org/current/doxygen/deal.II/step_6.html
  - http://www.math.colostate.edu/~bangerth/videos.676.15.html
  - http://www.math.colostate.edu/~bangerth/videos.676.16.html
  - http://www.math.colostate.edu/~bangerth/videos.676.17.html
  - http://www.math.colostate.edu/~bangerth/videos.676.17.25.html
  - http://www.math.colostate.edu/~bangerth/videos.676.17.5.html
  - http://www.math.colostate.edu/~bangerth/videos.676.17.75.html
- Documentation
  - https://dealii.org/current/doxygen/deal.II/group__constraints.html
  - https://dealii.org/current/doxygen/deal.II/group__grid.html
  - https://dealii.org/current/doxygen/deal.II/namespaceGridRefinement.html
  - https://dealii.org/current/doxygen/deal.II/namespaceDerivativeApproximation.html