PAUL SCHERRER INSTITUT

Three Week Presentation

Building blocks for Finite Element computations in IPPL

Lukas Bühler

Tuesday 24th October, 2023

PAUL SCHERRER INSTITUT

PSI

1. Problem
2. FEM Introduction
3. My Task and Timeline
4. Implementation
   - Software Architecture
   - Finite Element Space class: Mesh mapping and Basis Functions
   - Element Classes and Transformations
5. Future

PAUL SCHERRER INSTITUT

Problem

- Usecase for FEM in IPPL
  - Goal: Solving Maxwell equations
  - FDTD is 2nd order, FEM can be higher accuracy
- Arguments against external libraries
  - Avoiding data copies
  - Less coupling to external libraries, more modularity
  - More control: exascale, performant, portable FEM in IPPL

Finite Element Method

- Formulate the PDE in variational formulation:
  Example: Poisson Equation

$$\begin{cases} -\Delta \boldsymbol{u} = \boldsymbol{f} & \text{in } \Omega \\ \boldsymbol{u} = \boldsymbol{0} & \text{on } \partial\Omega \end{cases} \quad \Rightarrow \quad \int_\Omega \nabla \boldsymbol{u} \cdot \nabla \boldsymbol{v} = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \quad \forall \boldsymbol{v} \in V$$

- Mesh the domain with Elements: $\Omega = \bigcup_{e=1}^{N_E} K_e$
- Each element has a corresponding local system of equations: $\mathbf{A}_K \boldsymbol{x} = \boldsymbol{b}_K$
- Assemble the global system of equations

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$$

This system of equations can be solved matrix-free!

Conjugate Gradient (CG) Method

- Matrix-free, avoids building the global FEM matrix
- Iterative algorithm, terminates when remainder is sufficiently small
- Relies on the "action" of the matrix on a vector.

CG evaluates $\mathbf{A}\boldsymbol{x}$, with a given $\boldsymbol{x}$, every iteration.
Requires $\boldsymbol{b}$ at the beginning of the algorithm.

PAUL SCHERRER INSTITUT

Building blocks

- Evaluate basis functions (and their gradient) on elements
- Transformation from local to global coordinate system for element
- Numerical integration rule using polynomial interpolation on element
- Functions to interface FEM with CG.

PAUL SCHERRER INSTITUT

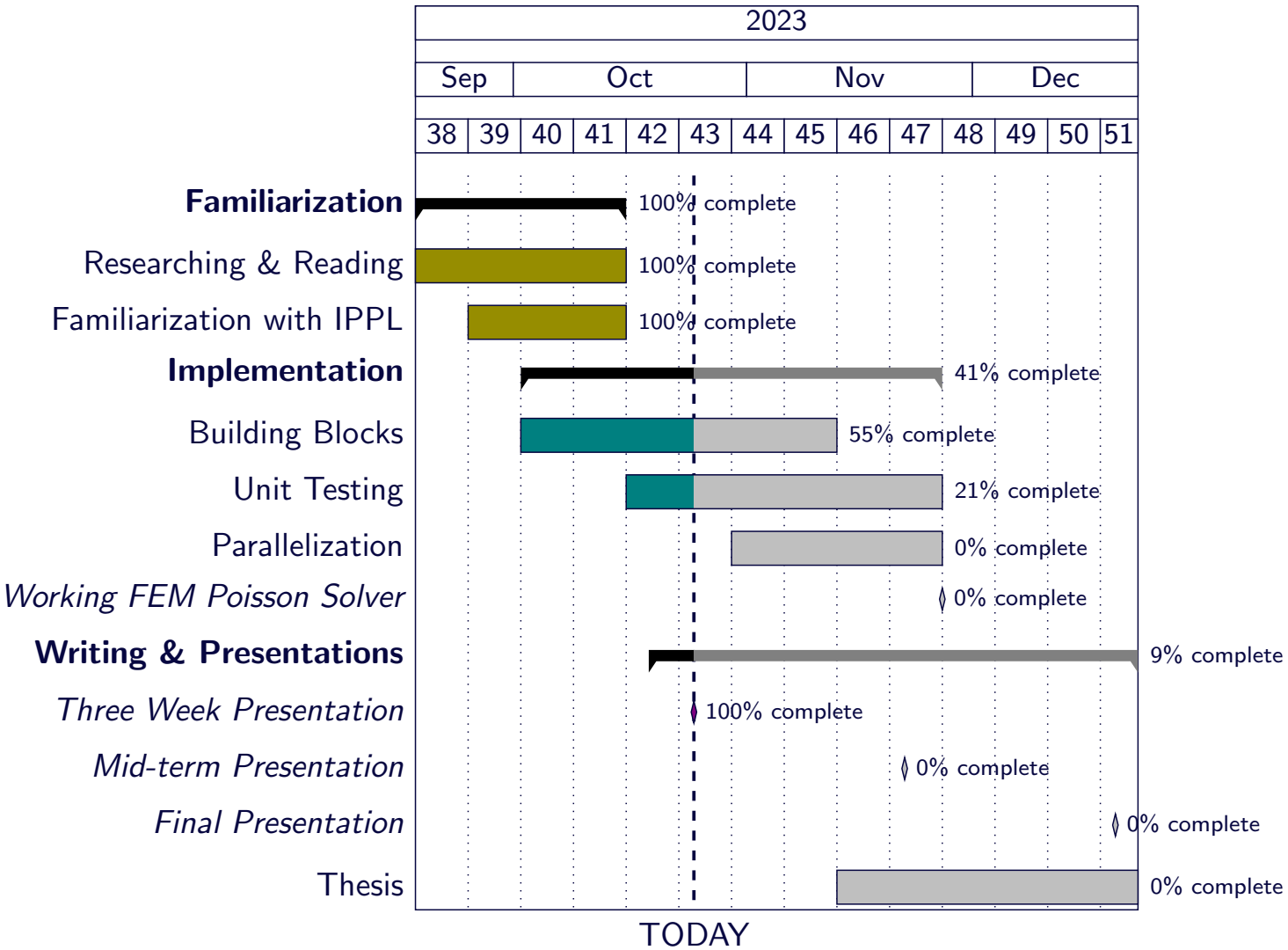
My Task

Task: Implementation of the building blocks and corresponding unit tests in IPPL.

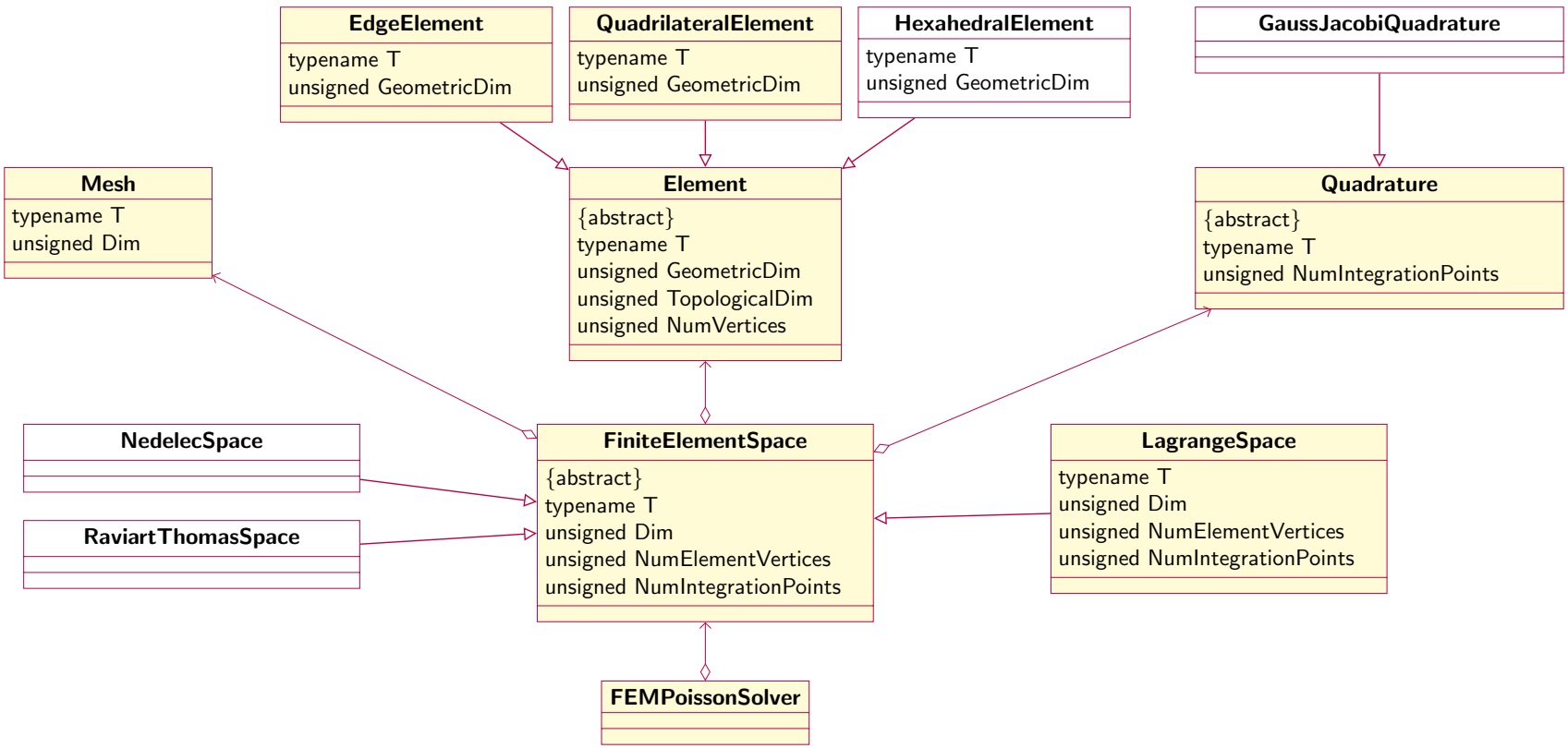
Building blocks:

1. Reference element base class and child element classes with local to global transformations for
   - Edge (1D)
   - Quadrilateral (2D)
   - Hexahedral (3D)
2. Finite element space base class
   - Mesh (and vertices) to elements mapping
   - Child classes with corresponding basis functions for the FEM spaces:
     - Lagrange
     - Nédélec
     - Raviart-Thomas
3. Quadrature rule base class and Gauss-Jacobi quadrature rule

Timeline

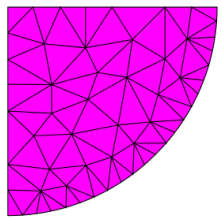

2023
Sep
Oct
Nov
Dec
38
39
40
41
42
43
44
45
46
47
48
49
50
51
Familiarization
100% complete
Researching & Reading
100% complete
Familiarization with IPPL
100% complete
Implementation
41% complete
Building Blocks
55% complete
Unit Testing
21% complete
Parallelization
0% complete
Working FEM Poisson Solver
0% complete
Writing & Presentations
9% complete
Three Week Presentation
100% complete
Mid-term Presentation
0% complete
Final Presentation
0% complete
Thesis
0% complete
TODAY

Software Architecture



Mesh types

- Unstructured Mesh



- Structured Mesh

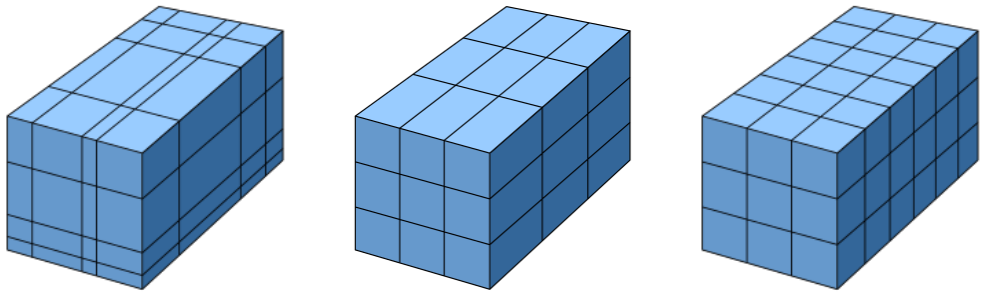


Figure: Rectilinear mesh, Regular mesh, Cartesian mesh[1]

$\Rightarrow$ IPPL Only has structured meshes

[1] Source: Wikimedia Commons. Drawn by Slffea, vectorized by Mysid.

Mesh: Element and Vertex mapping

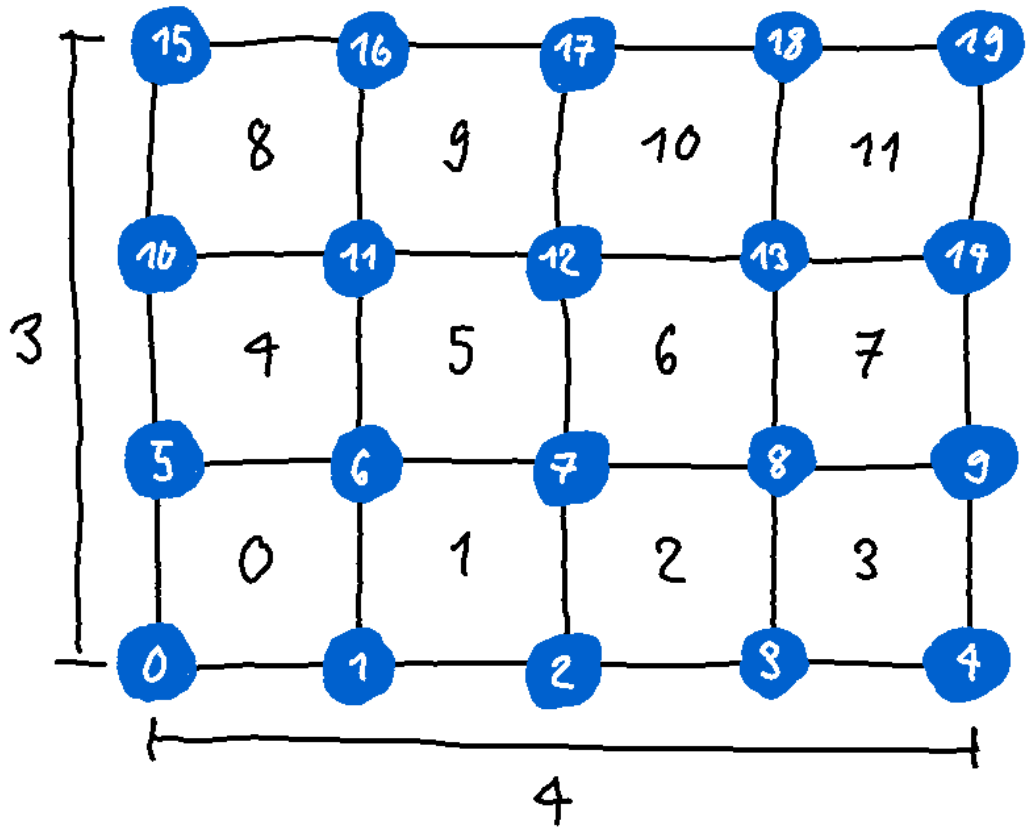


Figure: Example 4×3 2D regular mesh

Basis functions

$$b_h^j(x_i) = \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

Lagrange 1st order shape functions:

- 1D: $x \in [0, 1]$

$$b_h^1 = 1 - x$$
$$b_h^2 = x$$

- 2D: $(x, y) \in [0, 1]^2$

$$b_h^1 = (1 - x)(1 - y)$$
$$b_h^2 = x(1 - y)$$
$$b_h^3 = (1 - x)y$$
$$b_h^4 = xy$$

Finite Element Spaces: Lagrange (1st order) 1D

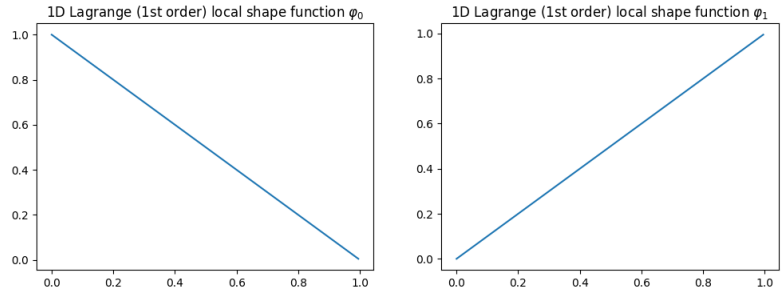


Figure: Local Shape functions for 1D 1st order Lagrange

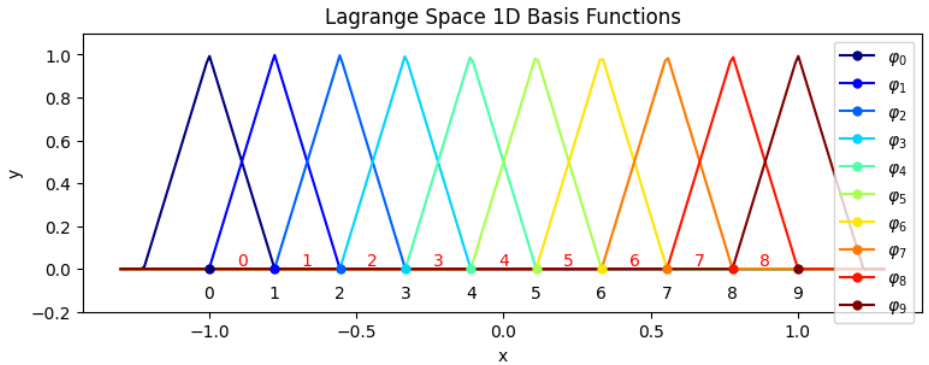


Figure: Shape functions of 1D uniform mesh with ten DoFs. (1D 1st order Lagrange)

Finite Element Spaces: Lagrange (1st order) 2D

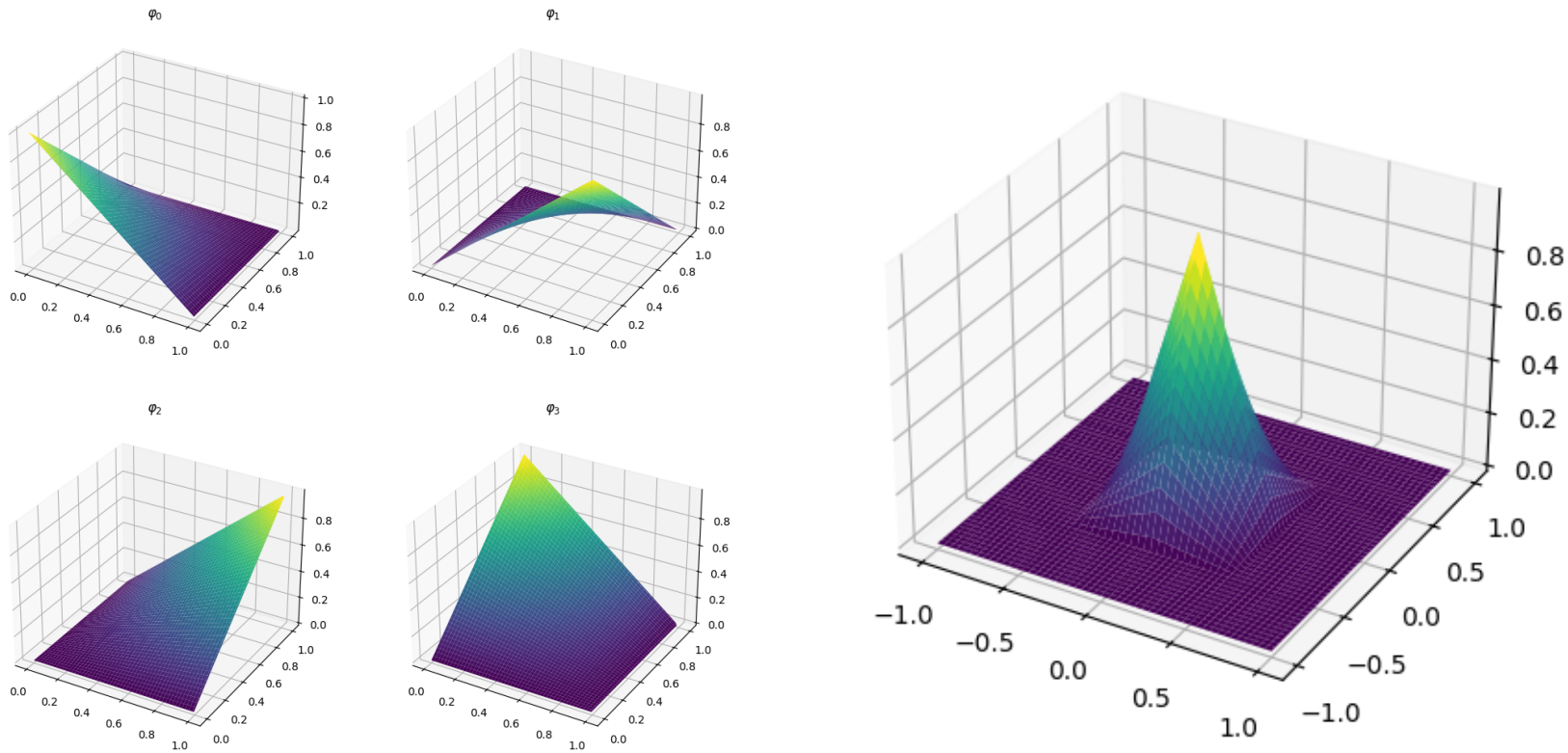
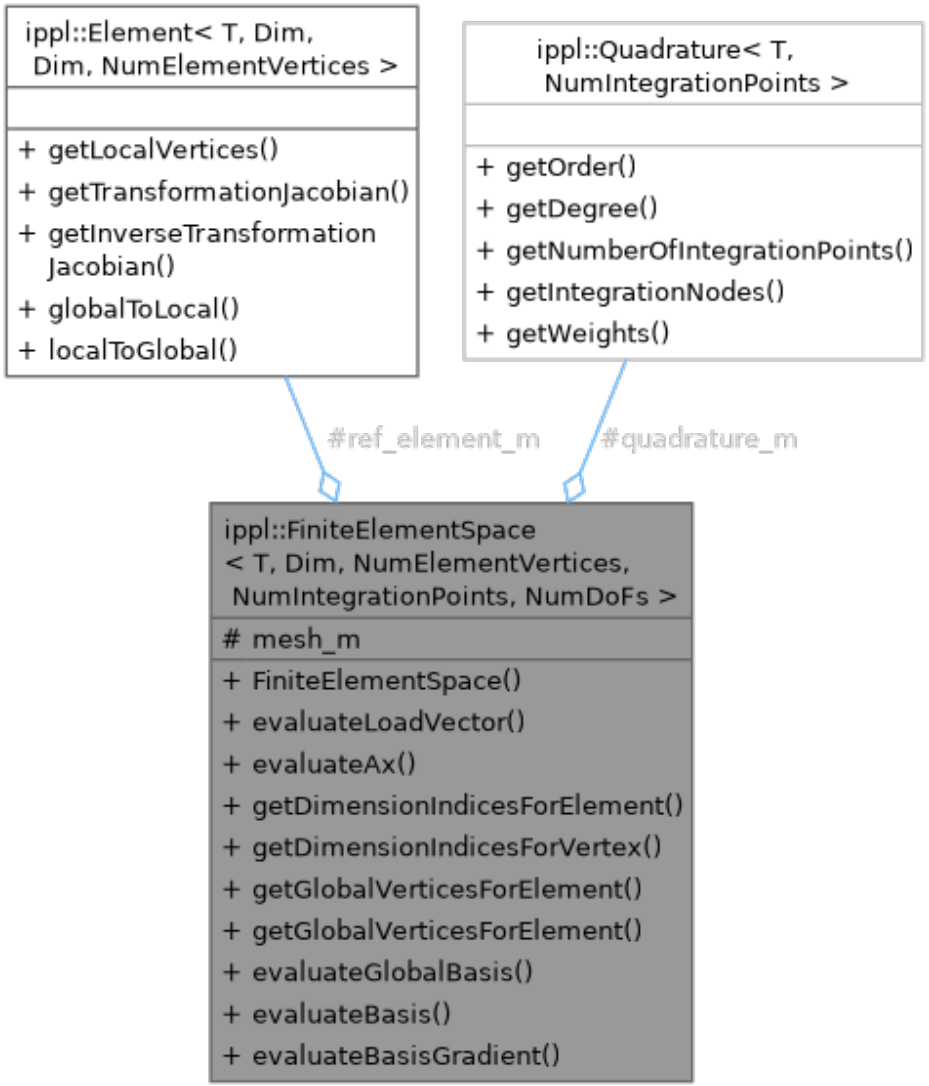


Figure: Left: local shape functions, Right: Basis of Node 12 on 5x5 mesh (2D 1st order Lagrange)

FiniteElementSpace Class



Elements: Local mapping

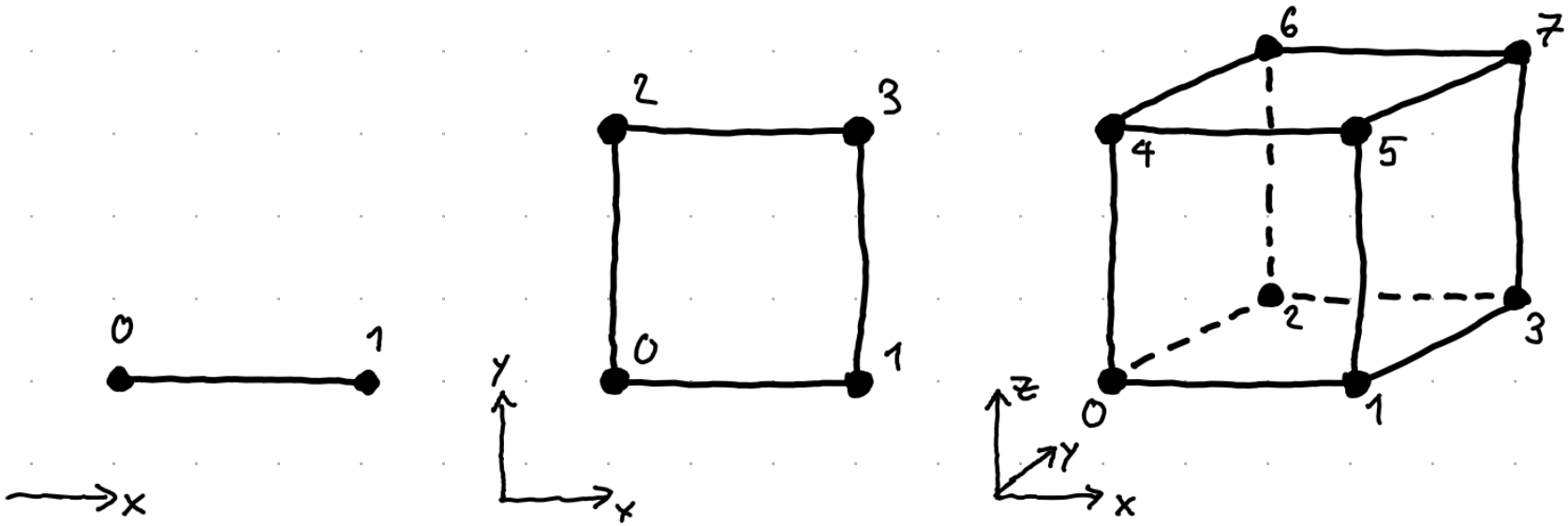
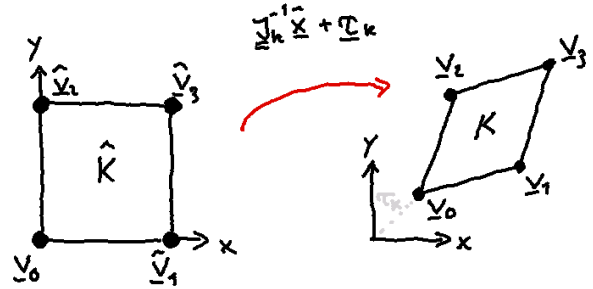


Figure: Local vertex mapping for EdgeElement, QuadrilateralElement, HexahedralElement (from left to right)

Element Transformations

- Transformation to and from local and global coordinates using affine transfomations



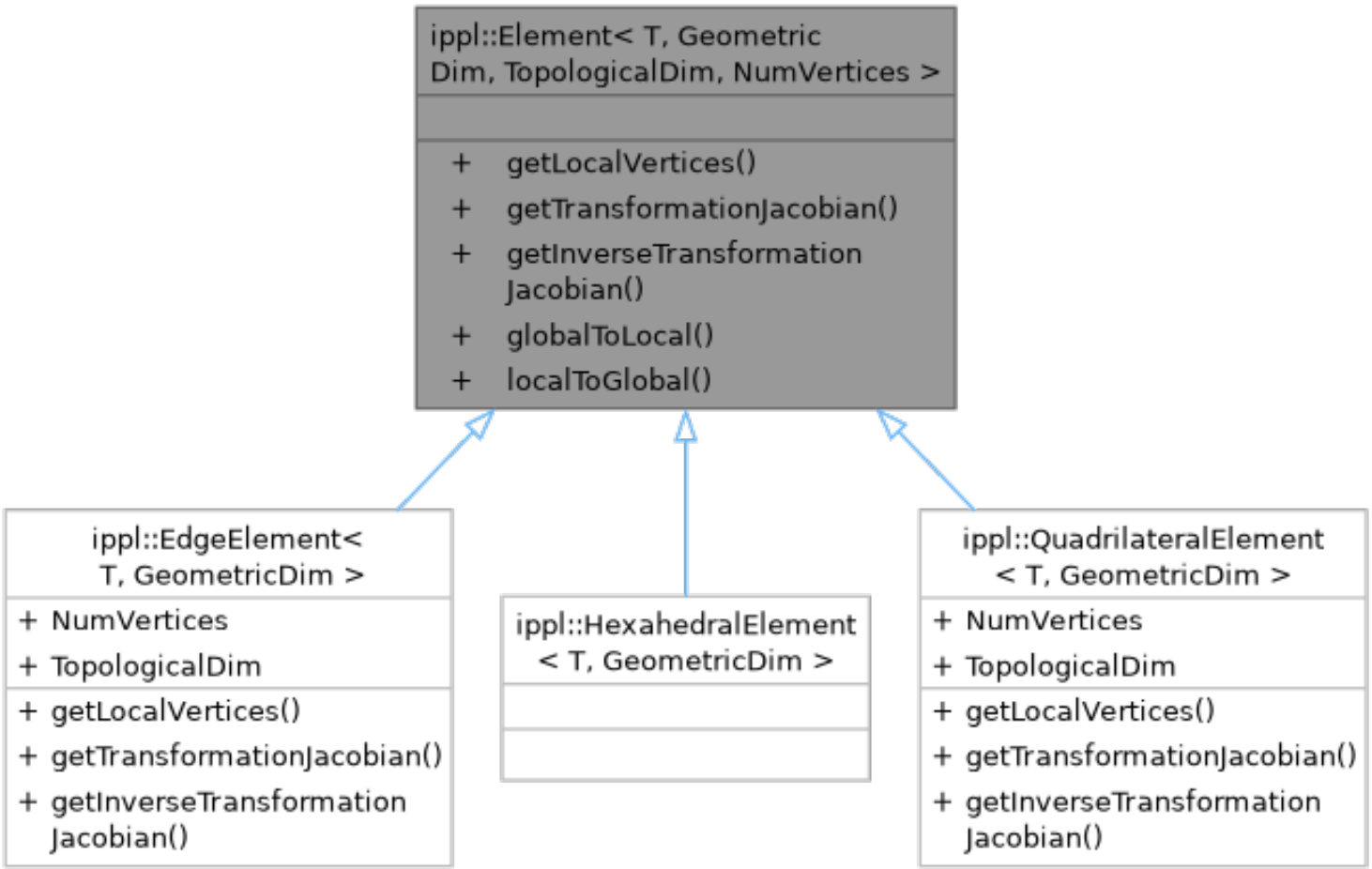
- Local to Global: $\boldsymbol{x} = \mathbf{J}_K^{-1}\hat{\boldsymbol{x}} + \boldsymbol{\tau}_K$,with $\boldsymbol{\tau}_K = \boldsymbol{v}_0$

$$\mathbf{J}_K^{-1} := \begin{bmatrix} \boldsymbol{v}_x^1 - \boldsymbol{v}_x^0 & \boldsymbol{v}_x^2 - \boldsymbol{v}_x^0 \\ \boldsymbol{v}_y^1 - \boldsymbol{v}_y^0 & \boldsymbol{v}_y^2 - \boldsymbol{v}_y^0 \end{bmatrix} \quad \text{(General)}$$

$$\mathbf{J}_K^{-1} := \begin{bmatrix} \boldsymbol{v}_x^1 - \boldsymbol{v}_x^0 & 0 \\ 0 & \boldsymbol{v}_y^2 - \boldsymbol{v}_y^0 \end{bmatrix} \quad \text{(Grid, only scaling)}$$

- Global to Local: $\hat{\boldsymbol{x}} = \mathbf{J}_K(\boldsymbol{x} - \boldsymbol{\tau}_K)$

Element Classes



Plan for the (near) future

- Currently working on: Element transformations.
- Next steps:
  1. Implement `evaluateAx`, `evaluateLoadVector`
  2. Implement Gauss-Jacobi Quadrature
  3. Going from 1D and 2D to 3D
  4. Thread- and Process-level Parallelization

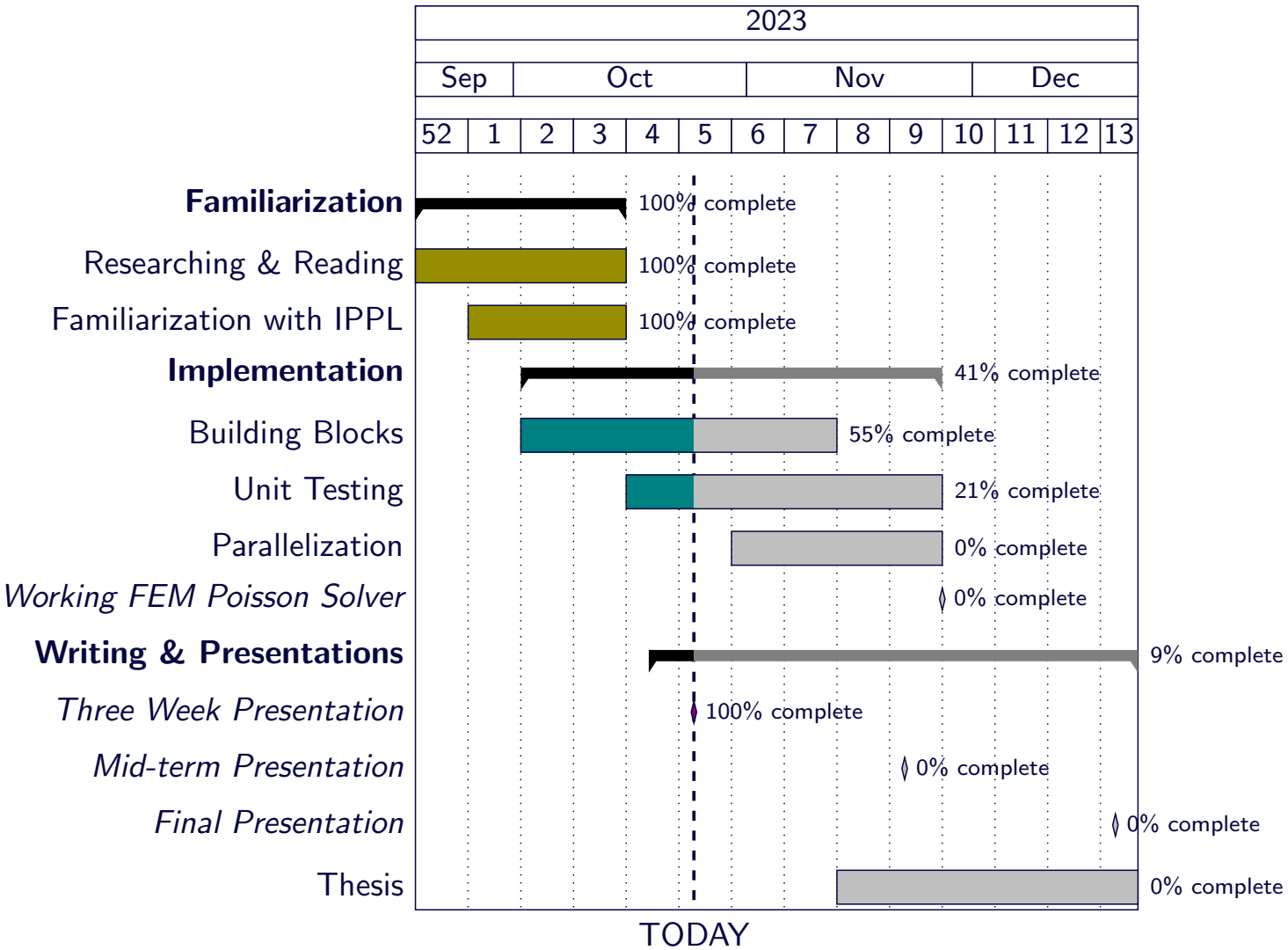
Add unit tests continually to test completed implementations.

References

[Hiptmair NumCSE] R. Hiptmair. Numerical Methods for Computational Science and Engineering. 2019. URL: https://www.sam.math.ethz.ch/ grsam/NCSE20/NumCSE_Lecture_Document.pdf

[Hiptmair NumPDE] R. Hiptmair. Numerical Methods for (Partial) Differential Equations. 2022. URL: https://www.sam.math.ethz.ch/ grsam/NUMPDEFL/NUMPDE.pdf

Timeline


2023
Sep
Oct
Nov
Dec
52
1
2
3
4
5
6
7
8
9
10
11
12
13
Familiarization
100% complete
Researching & Reading
100% complete
Familiarization with IPPL
100% complete
Implementation
41% complete
Building Blocks
55% complete
Unit Testing
21% complete
Parallelization
0% complete
Working FEM Poisson Solver
0% complete
Writing & Presentations
9% complete
Three Week Presentation
100% complete
Mid-term Presentation
0% complete
Final Presentation
0% complete
Thesis
0% complete
TODAY

Appendix A: Quadrature

m-point quadrature rule on $[a, b]$, $m \in \mathbb{N}$

$$\int_a^b f(t)\ \mathrm{d}t \approx \sum_{j=1}^{m} w_j\, f(p_j) \tag{1}$$

where $w_j \in \mathbb{R}$ are the quadrature weights and $\boldsymbol{p}_j$ quadrature nodes $\in [a, b]$.

Appendix B: Pseudo-code

`evaluateAx`$(\boldsymbol{x})$

$z \leftarrow 0$

for $i \leftarrow 0$ **to** N_{DoFs}; $i = i + 1$ **do**

  $\mathbf{J}^{-1}, det\mathbf{J} \leftarrow$ `getInverseTransformationJacobian`(element)

  `evaluatePDE`$(i, j, \boldsymbol{q}_k) \leftarrow \mathbf{J}^{-1}\nabla\boldsymbol{\varphi}_i(\boldsymbol{q}_k) \cdot \mathbf{J}^{-1}\nabla\boldsymbol{\varphi}_j(\boldsymbol{q}_k)$

  for $j \leftarrow 0$ **to** N_{DoFs}; $j = j + 1$ **do**

    $\mathbf{A}_{ij}^K \leftarrow \sum_{k=0}^{N_I} w_k \cdot$ `evaluatePDE`$(i, j, \boldsymbol{q}_k)$

    $z \leftarrow z + A_{ij}^K \boldsymbol{x}_j$

  end

end

Algorithm 1: `evaluateAx` (Poisson equation)

Appendix C: Example: Poisson Equation - Strong to weak formulation

- Poisson equation in strong form, with Dirichlet boundary conditions and given source term $\boldsymbol{f}$.

$$\begin{aligned} -\Delta \boldsymbol{u} &= \boldsymbol{f} \quad \text{in } \Omega \\ \boldsymbol{u} &= \boldsymbol{0} \quad \text{on } \partial\Omega \end{aligned} \tag{2}$$

- Integral equation

$$-\int_\Omega \Delta \boldsymbol{u} \cdot \boldsymbol{v} = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \quad \forall \boldsymbol{v} \in V$$

- Weak form of the Poisson equation

$$\int_\Omega \nabla \boldsymbol{u} \cdot \nabla \boldsymbol{v} = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \quad \forall \boldsymbol{v} \in V \tag{3}$$

Appendix C: Example: Poisson Equation - Discretization

- Restrict spaces: $v, u \in V_h \subset V$ (Galerkin finite element method)
- Space discretization (Meshing) $\quad \Omega = \bigcup_{e=1}^{N_E} K_e$

$$\sum_{e=1}^{N_E} \int_{K_e} \nabla \boldsymbol{u} \cdot \nabla \boldsymbol{v} = \sum_{e=1}^{N_E} \int_{K_e} \boldsymbol{f} \cdot \boldsymbol{v} \quad \forall \boldsymbol{v} \in V_h$$

- Write functions in terms of the basis functions

$$\boldsymbol{u} = \sum_i u_i \boldsymbol{\varphi}_i, \quad \boldsymbol{v} = \sum_j v_j \boldsymbol{\varphi}_j$$

$$\sum_i u_i \sum_{e=1}^{N_E} \int_{K_e} \nabla \boldsymbol{\varphi}_i \cdot \nabla \boldsymbol{\varphi}_j = \sum_{e=1}^{N_E} \int_{K_e} \boldsymbol{f} \cdot \boldsymbol{\varphi}_j \quad \forall \boldsymbol{\varphi}_j \tag{4}$$

Appendix C: Example: Linear System of Equations

$$\sum_i u_i \underbrace{\sum_{e=1}^{N_E} \int_{K_e} \nabla \varphi_i \cdot \nabla \varphi_j}_{\mathbf{A}_{ij}} = \underbrace{\sum_{e=1}^{N_E} \int_{K_e} \boldsymbol{f} \cdot \varphi_j}_{\boldsymbol{b}_j} \quad \forall \varphi_j \tag{4}$$

Solving the finite element method consists of solving a linear system of equations:

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$$