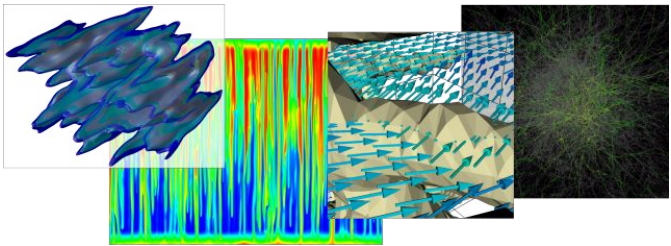
DUNE PDELab Tutorial 04

Finite Elements for the Wave Equation



Speaker:

Dominic Kempf
Scientific Software Center
Heidelberg University

Example Problem

In this tutorial we solve the wave equation formulated as a first order in time system. This way the example serves as a model for the treatment of systems of partial differential equations in PDELab.

$$\partial_{tt} u - c^2 \Delta u = 0 \quad \text{in } \Omega \times \Sigma, \tag{1a}$$
$$u = 0 \quad \text{on } \partial\Omega, \tag{1b}$$
$$u = q \quad \text{at } t = 0, \tag{1c}$$
$$\partial_t u = w \quad \text{at } t = 0, \tag{1d}$$

where c is the speed of sound.

Renaming $u_0 = u$ and introducing $u_1 = \partial_t u_0 = \partial_t u$ we can write the wave equation as a system of two equations:

$$\partial_t u_1 - c^2 \Delta u_0 = 0 \qquad \text{in } \Omega \times \Sigma, \tag{2a}$$

$$\partial_t u_0 - u_1 = 0 \qquad \text{in } \Omega \times \Sigma, \tag{2b}$$

$$u_0 = 0 \qquad \text{on } \partial\Omega, \tag{2c}$$

$$u_1 = 0 \qquad \text{on } \partial\Omega, \tag{2d}$$

$$u_0 = q \qquad \text{at } t = 0, \tag{2e}$$

$$u_1 = w \qquad \text{at } t = 0. \tag{2f}$$

Since $u_0 = u = 0$ on the boundary we also have $\partial_t u = u_1 = 0$ on the boundary. Alternatively, omit the boundary condition on u_1.

Weak Formulation

Multiplying (2a) with the test function v_0 and (2b) with the test function v_1 and using integration by parts we arrive at the weak formulation: Find $(u_0(t), u_1(t)) \in U_0 \times U_1$ s.t.

$$\begin{aligned} d_t(u_1, v_0)_{0,\Omega} + c^2(\nabla u_0, \nabla v_0)_{0,\Omega} = 0 \quad \forall v_0 \in U_0 \\ d_t(u_0, v_1)_{0,\Omega} - (u_1, v_1)_{0,\Omega} = 0 \quad \forall v_1 \in U_1 \end{aligned} \tag{3}$$

where we used the notation of the L^2 inner product $(u, v)_{0,\Omega} = \int_\Omega uv \, dx$.

An equivalent formulation to (3) that hides the system structure reads as follows:

$$\begin{aligned} &d_t\left[(u_0, v_1)_{0,\Omega} + (u_1, v_0)_{0,\Omega}\right] \\ &\qquad + \left[c^2(\nabla u_0, \nabla v_0)_{0,\Omega} - (u_1, v_1)_{0,\Omega}\right] = 0 \quad \forall (v_0, v_1) \in U_0 \times U_1 \end{aligned} \tag{4}$$

With the latter we readily identify the temporal and spatial residual forms:

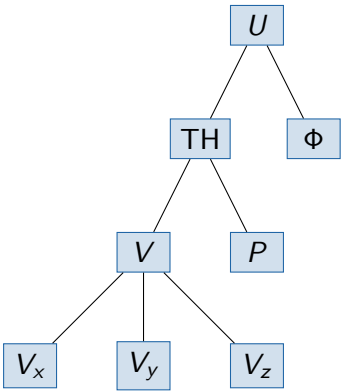
$$m^{\mathrm{WAVE}}((u_0, u_1), (v_0, v_1)) = (u_0, v_1)_{0,\Omega} + (u_1, v_0)_{0,\Omega}, \tag{5}$$

$$r^{\mathrm{WAVE}}((u_0, u_1), (v_0, v_1)) = c^2(\nabla u_0, \nabla v_0)_{0,\Omega} - (u_1, v_1)_{0,\Omega}\,. \tag{6}$$

Trees of Function spaces

$U = (V(\Omega_S))^d \times P(\Omega_S) \times \Phi(\Omega_D)$

- Computer science way of representing mathematical expressions: **Trees**
- Expose internal nodes to users
  - Enable recursive bottom-up construction
  - Extract subtrees to pass to legacy subproblem code
- Tree structure mostly static after construction
  - Nodes are C++ templates with children as template arguments
  - Allows extensive compiler optimizations, including inlining of tree traversals



Linear Algebra

[fragile] Given an assembled residual $r = \mathcal{R}(\vec{u_0})$ and its Jacobian $A = \nabla \mathcal{R}_h$, we have to solve the linear problem

$$Az = r$$

to obtain a correction and calculate $u = u_0 - z$.

Several options

Monolithic solve of $Az = r$

- No stability problems
- Often very difficult with standard iterative solvers

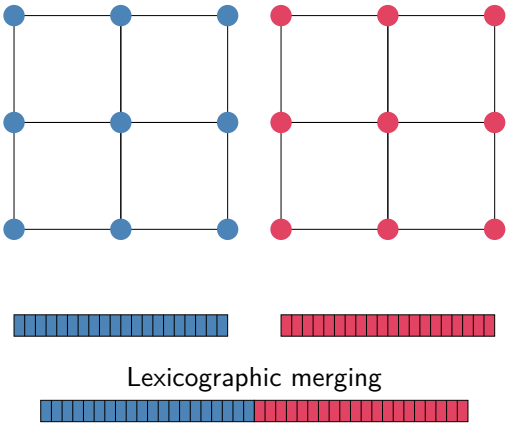
Exploiting structure of the coupling

- Does not require monolithic code base
- Matrix / vector data structures must contain structure for good performance
- Advanced preconditioners enable iterative solvers

Index Merging – Example

$$U = U_1 \times U_2$$

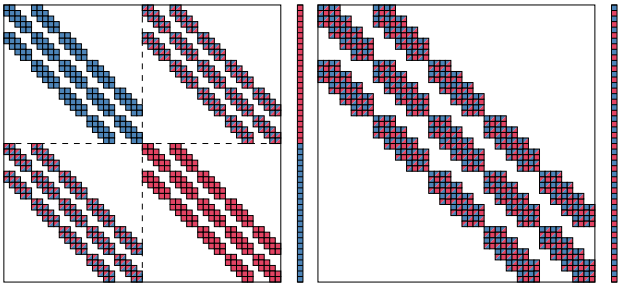
- Two Q_1 spaces on common mesh
- Each space has canonical order defined by vertex iteration
- Two merging strategies
  Lexicographic: Preserve structure of individual problems, separate matrix blocks for coupling
  Interleaved: Regard problem as vector-valued version of scalar problem



Merging + Blocking (I)

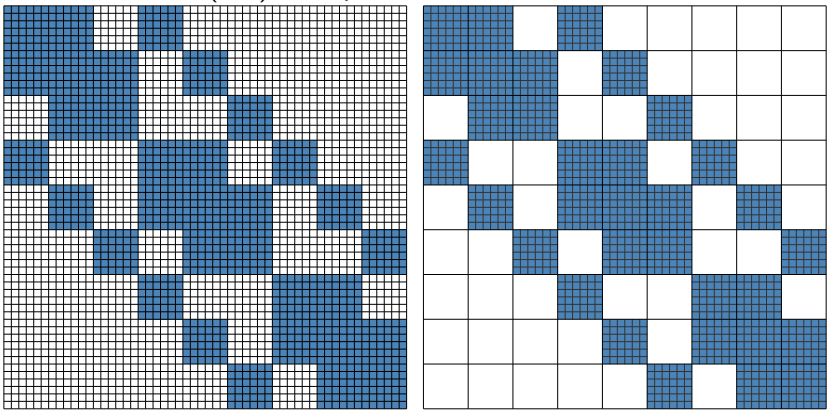
- Merging can be repeated at every tree node

  $\Rightarrow$ recursive construction of index structure from function space structure
- Also support **blocking** during merging
  - Large blocks for extracting subproblem matrices
  - Small blocks for block-aware preconditioners and reduced memory usage



Merging + Blocking (II)

A Discontinuous Galerkin (DG) example:



All the DoF per entity have dense coupling and are blocked together

Realization in PDELab

1) The ini-file `tutorial04.ini` holds parameters controlling the execution.
2) Main file `tutorial04.cc` includes the necessary C++, DUNE and PDELab header files; contains `main` function; instantiates DUNE grid objects and calls the `driver` function
3) Function `driver` in file `driver.hh` instantiates the necessary PDELab classes and finally solves the problem.
4) File `wavefem.hh` contains the local operator classes `WaveFEM` and `WaveL2` realizing the spatial and temporal residual forms.