UPPSALA
UNIVERSITET

Flow/Transport

Problem setting
Applications

Program design
Modularity
Linear algebra

Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly

Summary

Generic and efficient solvers for coupled flow/transport problems

Martin Kronbichler

Division of Scientific Computing
Department of Information Technology
Uppsala University
Sweden

August 24, 2010

UPPSALA UNIVERSITET

Flow/Transport

Problem setting
Applications
Program design
Modularity
Linear algebra
Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly
Summary

Problem setting

- Applications

Program design

- Modularity
- Linear algebra for flow solvers

What I did to make my programs run (a bit) faster

- Parallelization
- Trilinos
- Constraint matrix
- Matrix assembly

Summary

UPPSALA
UNIVERSITET

Flow/Transport

Problem setting
Applications

Program design
Modularity
Linear algebra

Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly

Summary

Simulation of coupled flow/transport problems

- Incompressible flow, described by incompressible Stokes/Navier–Stokes systems in velocities $\mathbf{u}$ and pressure p
- Equation of transport type (convection, convection–diffusion) due to flow field
- Transported variable induces force in fluid's momentum equation

UPPSALA
UNIVERSITET

Flow/Transport

Problem setting
Applications
Program design
Modularity
Linear algebra
Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly
Summary

Simulation of coupled flow/transport problems

- Incompressible flow, described by incompressible Stokes/Navier–Stokes systems in velocities $\mathbf{u}$ and pressure p
- Equation of transport type (convection, convection–diffusion) due to flow field
- Transported variable induces force in fluid's momentum equation
- Two-way coupling (active, not just passive transport)

UPPSALA
UNIVERSITET

Flow/Transport

Problem setting
Applications

Program design
Modularity
Linear algebra

Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly

Summary

Example problem: Boussinesq model (step-31 & step-32)

Stokes system driven by buoyancy:

$$\begin{aligned} -\nabla \cdot (2\eta \nabla^s(\mathbf{u})) + \nabla p &= \rho(T)\mathbf{g} \\ \nabla \cdot \mathbf{u} &= 0 \end{aligned}$$

Temperature field transported by fluid

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T - \kappa \nabla^2 T = \gamma$$

UPPSALA UNIVERSITET

Flow/Transport

Problem setting
Applications

Program design
Modularity
Linear algebra

Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly

Summary

Example problem: Boussinesq model (step-31 & step-32)

Stokes system driven by buoyancy:

$$\begin{aligned} -\nabla \cdot (2\eta \nabla^s(\mathbf{u})) + \nabla p &= \rho(T)\mathbf{g} \\ \nabla \cdot \mathbf{u} &= 0 \end{aligned}$$

Temperature field transported by fluid

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T - \kappa \nabla^2 T = \gamma$$

Coupling:

- T in momentum equation
- $\mathbf{u}$ in transport equation

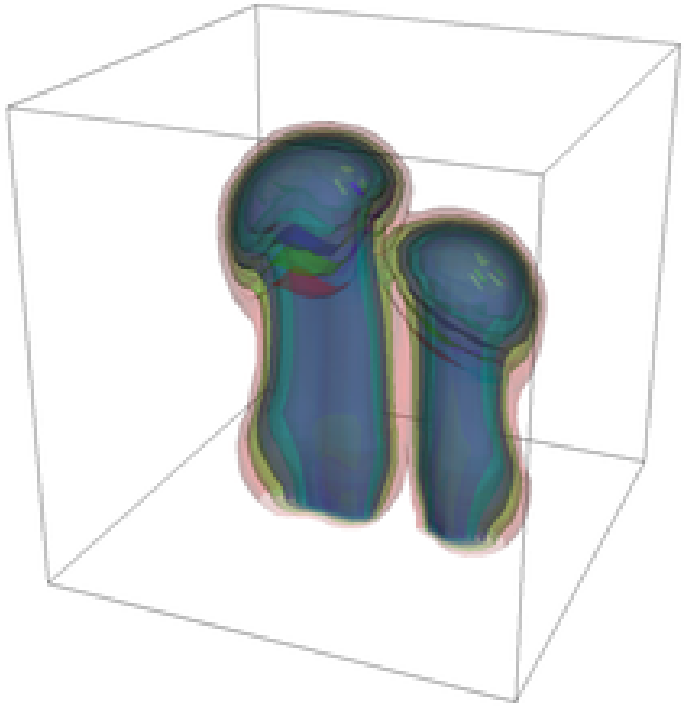
UPPSALA UNIVERSITET

Flow/Transport

Problem setting
Applications
Program design
Modularity
Linear algebra
Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly
Summary

Applications of flow/transport type I

- Motion of magma in earth mantle: Boussinesq problem (`step-31` & `step-32`) — Stokes coupled to temperature convection–diffusion equation, buoyancy drives flow by force term of form $T\mathbf{g}$



Temperature field in earth mantle convection, `step-31`

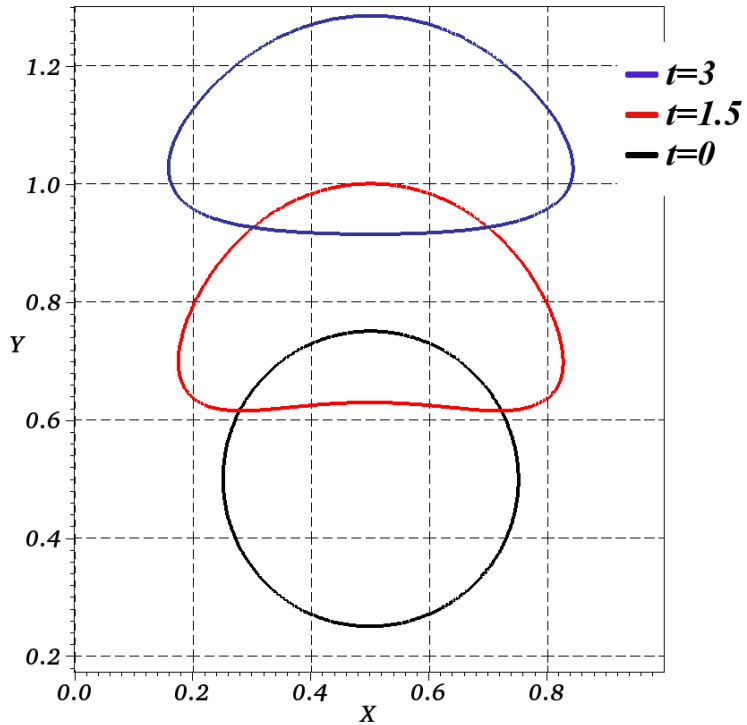
UPPSALA UNIVERSITET

Flow/Transport

Problem setting
Applications
Program design
Modularity
Linear algebra
Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly
Summary

Applications of flow/transport type II

- Two-phase flow modeled by level set models: Navier–Stokes coupled to level set convection equation in ϕ, interface force $\delta_h \mathbf{n}(\phi)\kappa(\phi)$



Bubble of low density rises due to gravity

UPPSALA
UNIVERSITET

Flow/Transport

Problem setting
Applications

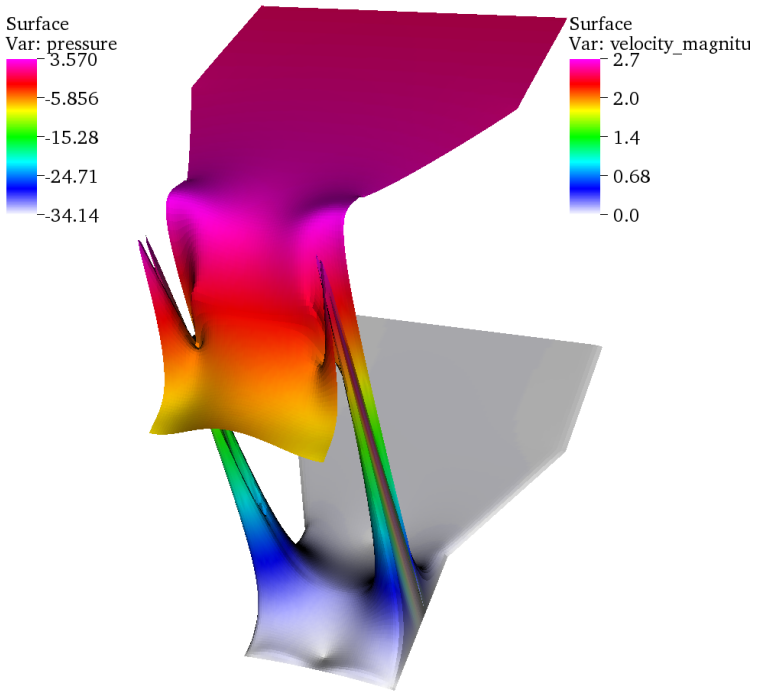
Program design
Modularity
Linear algebra

Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly

Summary

Applications of flow/transport type III

- Solidification processes, liquid phase sintering, etc.: Navier–Stokes coupled to phase field model (Cahn–Hilliard equation) and possibly temperature, interface force from Cahn–Hilliard variables



Pressure and velocity magnitude of phase field flow in a channel

UPPSALA
UNIVERSITET

Flow/Transport

Problem setting
Applications
Program design
Modularity
Linear algebra
Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly
Summary

Programming challenges with this kind of problems

- Different models (e.g. level set, phase field) for same problem and same fluid solver
- Different fluid models (Stokes or Navier–Stokes) for same transport equation

UPPSALA UNIVERSITET

Flow/Transport

Problem setting
Applications
Program design
Modularity
Linear algebra
Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly
Summary

Programming challenges with this kind of problems

- Different models (e.g. level set, phase field) for same problem and same fluid solver
- Different fluid models (Stokes or Navier–Stokes) for same transport equation
- Different flow solvers (projection/fractional-step solver, fully coupled implicit/explicit solver, fully coupled solver with Newton iteration)

All this combined with adaptive mesh refinement, 3D simulations (large scale problems), etc.

UPPSALA
UNIVERSITET

Flow/Transport

Problem setting
Applications
Program design
Modularity
Linear algebra
Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly
Summary

How to design modular programs for this problem type

Requirements:

- Want to use one implementation of fluid solver for different application types and transport evolution equations (keep all variants with most current fluid solver, reduce duplicated code)
- Make change of fluid solver simple

UPPSALA
UNIVERSITET

Flow/Transport

Problem setting
Applications

Program design
Modularity
Linear algebra

Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly

Summary

How to design modular programs for this problem type

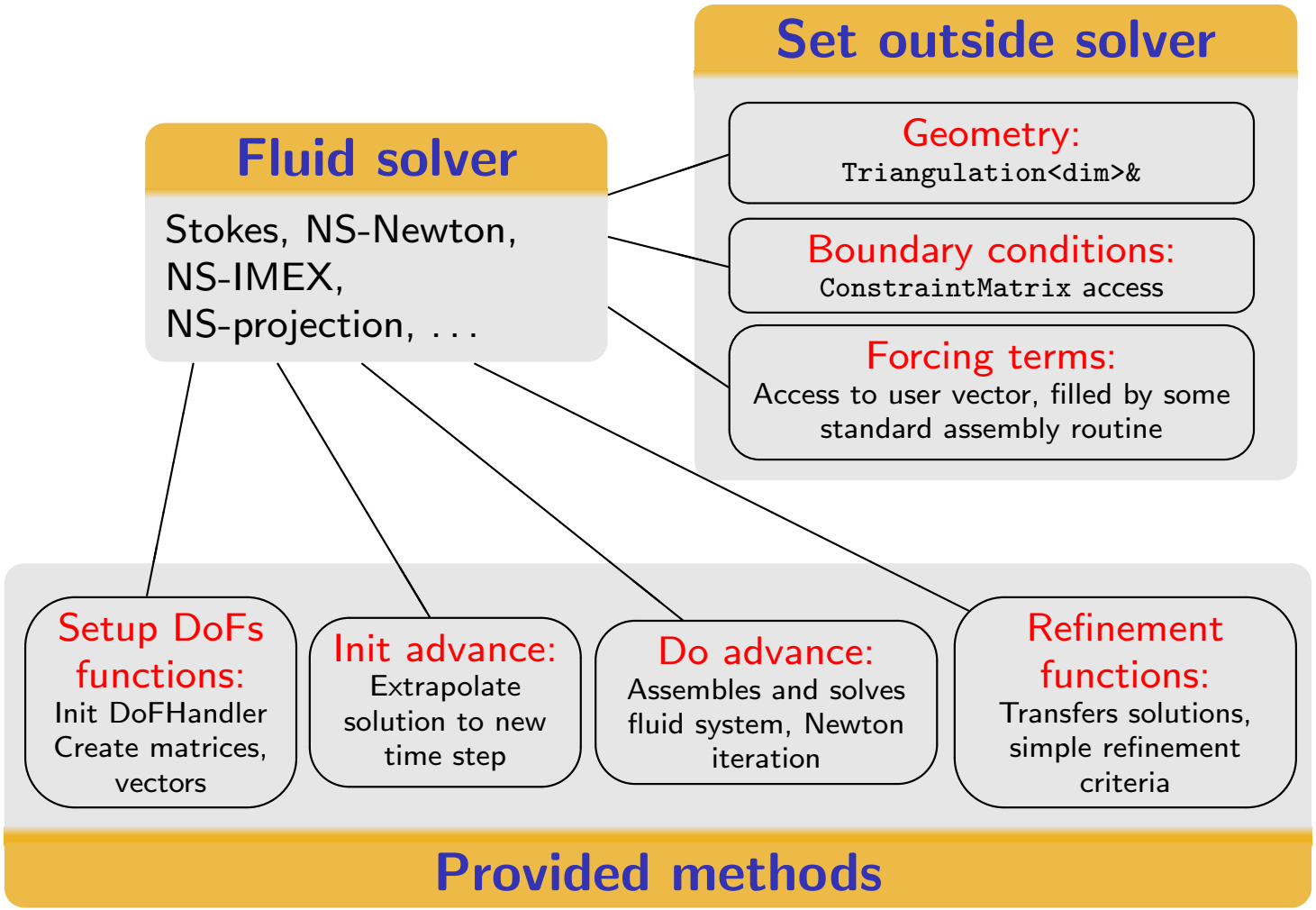
Requirements:

- Want to use one implementation of fluid solver for different application types and transport evolution equations (keep all variants with most current fluid solver, reduce duplicated code)
- Make change of fluid solver simple

Algorithm outline

Discretize in time by a time lag scheme: first update transport equation based on extrapolated velocity, then flow equations based on new right hand side

Generic flow solver interface



UPPSALA
UNIVERSITET

Flow/Transport

Problem setting
Applications
Program design
Modularity
Linear algebra
Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly
Summary

Linear algebra for flow solvers

For blocked systems use GMRES with Schur complement preconditioners

- Stokes:

$$P^{-1} = \begin{pmatrix} A_{\text{AMG}}^{-1} & A_{\text{AMG}}^{-1} B M_{\text{p}}^{-1} \\ 0 & M_{\text{p}}^{-1} \end{pmatrix}$$

with pressure mass matrix M_p as Schur complement approximation

---

Collaboration with Wolfgang Bangerth and Timo Heister

UPPSALA UNIVERSITET

Flow/Transport

Problem setting
Applications

Program design
Modularity
Linear algebra

Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly

Summary

Linear algebra for flow solvers

For blocked systems use GMRES with Schur complement preconditioners

- Navier–Stokes:

$$P^{-1} = \begin{pmatrix} \hat{A}^{-1} & \hat{A}^{-1} B \hat{S}^{-1} \\ 0 & \hat{S}^{-1} \end{pmatrix}$$

with $\hat{A}^{-1}$ approximated by ILU or AMG, $\hat{S}^{-1} = \frac{1}{\eta} M_{\mathrm{p}}^{-1} + \frac{1}{\Delta t} L_{\mathrm{p}}^{-1}$ (pressure mass and Laplace matrix)

Collaboration with Bärbel Janssen

UPPSALA
UNIVERSITET

Flow/Transport

Problem setting
Applications

Program design
Modularity
Linear algebra

Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly

Summary

What I did to make my programs run (a bit) faster

- Parallelization
- Trilinos linear algebra
- Improvements in ConstraintMatrix
- Matrix assembly

UPPSALA
UNIVERSITET

Flow/Transport

Problem setting
Applications

Program design
Modularity
Linear algebra

Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly

Summary

Parallelization

For large scale problems (3D), need to distribute work and data structures (matrices, triangulations) over several compute nodes. Introduced some of that in above framework (not yet all of it).

- I have worked with enabling deal.II to use Trilinos parallel linear algebra
- Enabling massive parallelism with deal.II $\Rightarrow$ Timo Heister's talk

UPPSALA
UNIVERSITET

Flow/Transport

Problem setting
Applications
Program design
Modularity
Linear algebra
Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly
Summary

Trilinos structures

Trilinos provides framework for distributed calculations very similar to deal.II's own structures and PETSc wrappers

- Matrices: `TrilinosWrappers::SparseMatrix`, `TrilinosWrappers::BlockSparseMatrix`, distributed over processors row-wise
- Vectors: `TrilinosWrappers::Vector`, `TrilinosWrappers::MPI::Vector` and block versions, former is not distributed (or full size in distributed environments), the latter is distributed as matrices

UPPSALA
UNIVERSITET

Flow/Transport

Problem setting
Applications
Program design
Modularity
Linear algebra
Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly
Summary

Trilinos structures

Trilinos provides framework for distributed calculations very similar to deal.II's own structures and PETSc wrappers

- Matrices: `TrilinosWrappers::SparseMatrix`, `TrilinosWrappers::BlockSparseMatrix`, distributed over processors row-wise
- Vectors: `TrilinosWrappers::Vector`, `TrilinosWrappers::MPI::Vector` and block versions, former is not distributed (or full size in distributed environments), the latter is distributed as matrices
- Preconditioners: Jacobi, SSOR, ILU, Chebyshev, AMG

UPPSALA UNIVERSITET

Flow/Transport

Problem setting
Applications
Program design
Modularity
Linear algebra
Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly
Summary

Trilinos AMG preconditioner

`TrilinosWrappers::PreconditionAMG` is a very convenient preconditioner that works well for many PDE operators

Properties:

- Works with Trilinos sparse matrices and deal.II sparse matrices (creates a copy to a Trilinos matrix internally)
- Works well for linear and quadratic elements in 2D and 3D, around 20 iterations to solve Laplace problems
- Works less well for higher order elements

UPPSALA UNIVERSITET

Flow/Transport

Problem setting
Applications
Program design
Modularity
Linear algebra
Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly
Summary

Trilinos AMG preconditioner

`TrilinosWrappers::PreconditionAMG` is a very convenient preconditioner that works well for many PDE operators

Properties:

- Works with Trilinos sparse matrices and deal.II sparse matrices (creates a copy to a Trilinos matrix internally)
- Works well for linear and quadratic elements in 2D and 3D, around 20 iterations to solve Laplace problems
- Works less well for higher order elements

To make things more efficient. . .

- Would need to create patches of low-order (linear) elements for higher-order or *hp* approximations and base preconditioner on matrices from those elements

UPPSALA UNIVERSITET

Flow/Transport

Problem setting
Applications
Program design
Modularity
Linear algebra
Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly
Summary

Writing into matrices with the `ConstraintMatrix` class

For writing into matrices (and vectors), I improved `ConstraintMatrix::distribute_local_to_global` functions.

Purpose of function:

- Resolves constraints and writes into matrix
- Works for deal.II and distributed matrices (where usual condense functions and apply_boundary_conditions do not work as neatly)

UPPSALA UNIVERSITET

Flow/Transport

Problem setting
Applications

Program design
Modularity
Linear algebra

Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly

Summary

Writing into matrices with the `ConstraintMatrix` class: Speed

- Timing for 3D problem, $\mathcal{Q}_2$ elements, 274k DoFs with `dealii::SparseMatrix<double>`, only boundary constraints:

  SP::add(i,j,value) (one by one) **2.8 s**

  CM function **0.83 s**, of which 0.61 s sparse matrix read/write access (unavoidable)

UPPSALA UNIVERSITET

Flow/Transport

Problem setting
Applications

Program design
Modularity
Linear algebra

Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly

Summary

Writing into matrices with the `ConstraintMatrix` class: Speed

- Timing for 3D problem, $\mathcal{Q}_2$ elements, 274k DoFs with `dealii::SparseMatrix<double>`, only boundary constraints:

  SP::add(i,j,value) (one by one) **2.8 s**
  CM function **0.83 s**, of which 0.61 s sparse matrix read/write access (unavoidable)
- Complexity per row: `n_entries_in_matrix_row` * const, (constant about 4 in 2D, 8 in 3D)
- traditional algorithm: `n_entries_in_matrix_row` * log(`n_entries_in_matrix_row`) * const

UPPSALA UNIVERSITET

Flow/Transport

Problem setting
Applications

Program design
Modularity
Linear algebra

Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly

Summary

Writing into matrices with the `ConstraintMatrix` class: Speed

- Timing for 3D problem, $\mathcal{Q}_2$ elements, 274k DoFs with `dealii::SparseMatrix<double>`, only boundary constraints:

  SP::add(i,j,value) (one by one) **2.8 s**
  CM function **0.83 s**, of which 0.61 s sparse matrix read/write access (unavoidable)
- Complexity per row: `n_entries_in_matrix_row` * const, (constant about 4 in 2D, 8 in 3D)
- traditional algorithm: `n_entries_in_matrix_row` * log(`n_entries_in_matrix_row`) * const
- Much more efficient than adding elements one by one with SP::add $\Rightarrow$ always do this — empty CM to write local matrices into global ones!

UPPSALA UNIVERSITET

Flow/Transport

Problem setting
Applications
Program design
Modularity
Linear algebra
Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly
Summary

Writing into matrices with the `ConstraintMatrix` class: Implementation

- Sorts indices, then traverses matrix row and adds data to present elements (avoids several accesses in case of constraints)
- Resolves constraints and collects data before writing into sparse matrix

UPPSALA UNIVERSITET

Flow/Transport

Problem setting
Applications

Program design
Modularity
Linear algebra

Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly

Summary

Matrix assembly with constant coefficients

Implemented function `FEValues::get_cell_similarity` which checks for translations of cells. If one cell is translation of the previous one (as e.g. for Cartesian meshes):

- FEValues does not need to recompute gradients or Jacobian transformations for simple mappings (implemented in library)
- If assembling a constant-coefficient matrix (e.g. Laplace matrix), do not need to recompute matrix (user's responsibility)

UPPSALA
UNIVERSITET

Flow/Transport

Problem setting
Applications
Program design
Modularity
Linear algebra
Efficiency
Parallelization
Trilinos
Constraint matrix
Matrix assembly
Summary

Summary

- Implemented framework for dealing with coupled flow/transport problems
- Work on efficient (parallel) numerical linear algebra
- A few things that might also help others to speed up their programs