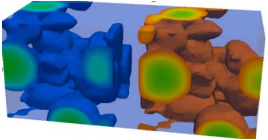
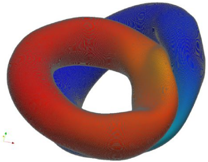
Trilinos User-Developer Group Meeting

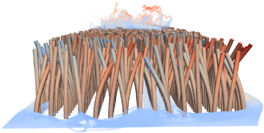
CSRI, Albuquerque | October 22nd - 24th, 2024

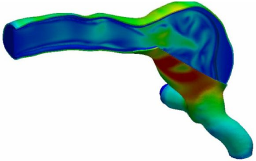
der Bundeswehr
Universität München

Trilinos use in 4C-multiphysics









Max Firmbach [1] **Matthias Mayr** [1,2]

[1] Institute for Mathematics and Computer-Based Simulation, University of the Bundeswehr Munich

[2] Data Science & Computing Lab, University of the Bundeswehr Munich

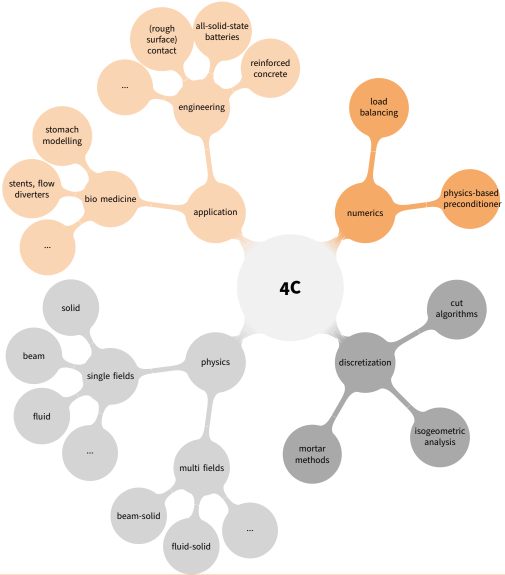
What is 4C-multiphysics?

Comprehensive Computational Community Code (4C)

4C is a parallel multi-physics research code to analyze and solve a plethora of physical problems by means of computational mechanics.

- provides simulation capabilities for a variety of physical models, including
  - single fields such as solids and structures, fluids, scalar transport, or porous media
  - multiphysics coupling and interactions between several fields
- mostly based on finite element methods (FEM, CutFEM)
- leverages the TRILINOS project for sparse linear algebra, nonlinear solvers, linear solvers & preconditioners, domain partitioning & rebalancing, automatic differentiation, ...
- parallelized with MPI for distributed memory hardware architectures

Research areas in 4C



Guiding principle

Application-motivated fundamental research.

All parts of the code are in one form or another related to current or former research projects.

Jointly developed by several reasearch groups across Germany!

Universität der Bundeswehr München
Institut für Mathematik und Computergestützte Simulation

RUHR UNIVERSITÄT BOCHUM RUB

Helmholtz-Zentrum hereon

Technische Universität München

Some facts and statistics about 4C

Repository

- First commit on Jan 9, 2002 (author: unknown)
- More than 30k commits
- 126 contributors (+students)

Research output

- >400 peer-reviewed publications
- >57 PhD theses

Code base

- 1911 files (89% C++)
- >1.16 mio lines of code (incl. 35% comments)
- Code coverage: 71.6 %

TRILINOS CMake SuperLU ARBORX SuiteSparse ...

Takeaway

- Trilinos is by far the most important third-party library
- Trilinos' develop branch is checked and tested against the 4C main-branch on a weekly basis

Trilinos is an integral part of 4C

Currently $\approx$ 20 packages are in active use.

Core

- Epetra
- EpetraExt
- Isorropia
- Kokkos
- Tpetra
- Teuchos
- Thyra
- Zoltan
- Zoltan2

Solvers

- Amesos
- Belos
- Ifpack
- MueLu
- Stratimikos
- Teko
- NOX
- Xpetra

Discretizations and Analysis

- Intrepid2
- Shards
- Sacado

4C & Linear solvers / Preconditioners

- Prototyping with direct solvers from **Amesos** (Umfpack, SuperLU)
- Production runs with iterative solvers from **Belos** (mostly GMRES) and respective preconditioners:
  - **Ifpack** for incomplete factorizations (RILUK, ILUT)
  - **MueLu** for algebraic multigrid (Unsmoothed, Smoothed, Petrov-Galerkin)
  - **Teko** for block preconditioning (Block Gauss-Seidel, Block LU, SIMPLE)
    Implementing own Block LU Strategy for mixed-dimensional preconditioning (e.g. for beam-solid interaction)
- Exploring and starting to use the **Stratimikos** interface with xml-files ...
  - ... to provide users easy access to example linear solver configs
  - ... to simplify the linear solver interface to Trilinos and ease maintenance

Current state in 4C

Amesos, **Belos**, **Ifpack** and **MueLu** run very stable for already a long time in 4C. Recently introduced **Teko** to replace self-implementations of block methods and add special features for block preconditioning $\rightarrow$ so far works great!

4C & Linear solvers / Preconditioners

Mixed-dimensional beam-solid problem:

$$\begin{pmatrix} A & B_1^T \\ B_2 & C \end{pmatrix} = \begin{pmatrix} K^B+\epsilon D^T \kappa^{-1} D & -\epsilon D^T \kappa^{-1} M \\ -\epsilon M^T \kappa^{-1} D & K^S+\epsilon M^T \kappa^{-1} M \end{pmatrix}$$

- large ϵ results in high condition number
- A is block diagonal
- block system might be nonsymmetric

Implementation based on **Teko**:

- `LU2x2PreconditionerFactory`
- derived `LU2x2Strategy`

In-house methods conveniently added to **Stratimikos** linear solver builder.

Block preconditioner

1. Pre-compute SPAI of A and form explicit approximate Schur complement:

$$\tilde{A}^{-1} = SPAI(A) \quad \text{and} \quad S = C - B_1 \tilde{A}^{-1} B_1^T$$

2. Calculate residual:

$$\begin{pmatrix} r_1 \\ r_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} - \begin{pmatrix} A & B_1^T \\ B_1 & C \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

3. Solve prediction of beam equation with SPAI smoother:

$$x_1^{k+1} = x_1^k + \tilde{A}^{-1} r_1$$

4. Solve Schur complement equation with AMG:

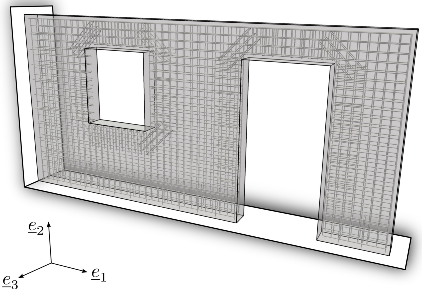
$$\tilde{S} x_2 = r_2 - B_2 x_1$$

5. Solve correction of beam equation with SPAI smoother:

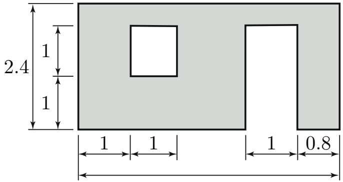
$$x_1^{k+1} = x_1^k + \tilde{A}^{-1}(r_1 - B_1^T x_2)$$

Comparison of linear solvers

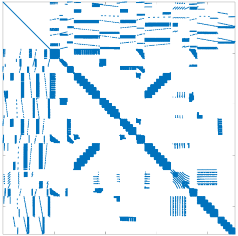
- **Amesos** direct solver not feasible for problem size
- Incomplete factorization as preconditioner in **Ifpack** leads to no convergence
- Very special block LU as **Teko** preconditioner is scalable and fast (≈ 25 iterations)



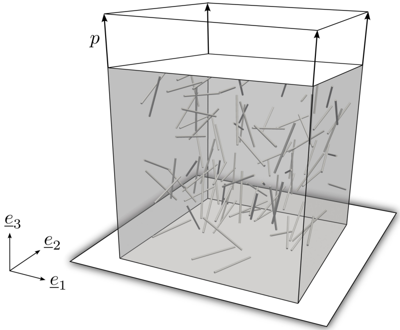
Reinforced concrete wall model setup:



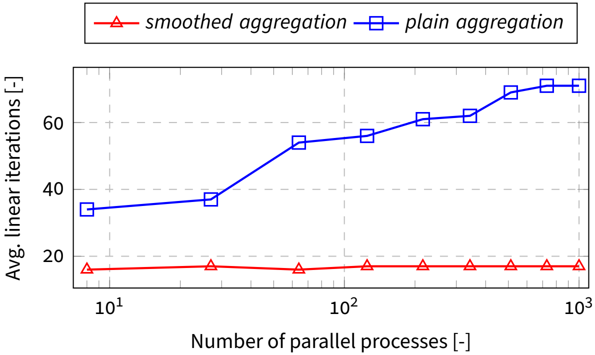
Sparsity pattern of the stiffness operator:



Weak scaling study based on minimal example:
Solid cube randomly filled with fibers



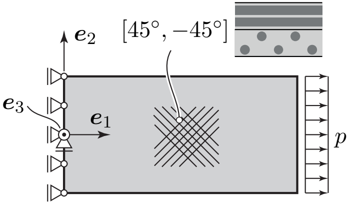
Scaling from $\approx$ 50.000 DOFs to $\approx$ 50.000.000 DOFs.

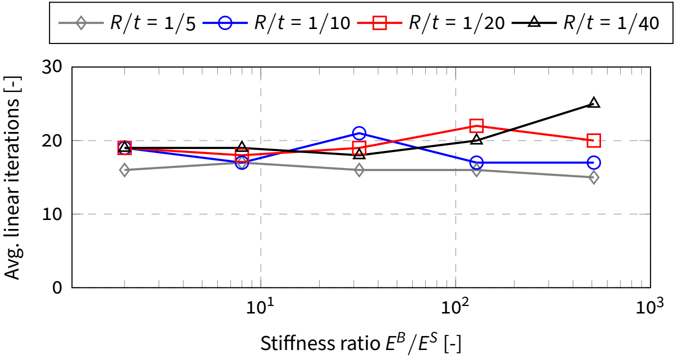


Regarding scalability

Special care has to be taken regarding the AMG method for the Schur complement!

Parameter robustness study on a composite plate:





Varying stiffnes ratio E^B/E^S and beam radius to plate thickness ratio R/t.

Regarding robustness

Preconditioner is considered to be robust in all relevant parameters!

Monolithic fluid-structure interaction problem:

$$A = \begin{pmatrix} S & & C_{SF} \\ & G & C_{GF} \\ C_{FS} & C_{FG} & F \end{pmatrix}$$

Construct block Gauss-Seidel preconditioner:

$$M^{-1} = \begin{pmatrix} S & & \\ & G & \\ C_{FS} & C_{FG} & F \end{pmatrix}^{-1}$$

Implementation based on **Teko** and **MueLu**:

- `GaussSeidelPreconditionerFactory`
- approximate sub-block inverses with AMG

Again **Stratimikos** makes it easy to build the preconditioner.

"all-in-one" algebraic multigrid method

1. Build segregated transfer operators:

   $$R = \begin{pmatrix} R^{\mathcal{S}} & & \\ & R^{\mathcal{G}} & \\ & & R^{\mathcal{F}} \end{pmatrix} \text{ and } P = \begin{pmatrix} P^{\mathcal{S}} & & \\ & P^{\mathcal{G}} & \\ & & P^{\mathcal{F}} \end{pmatrix}$$

   Coarsen individual physical fields separately.

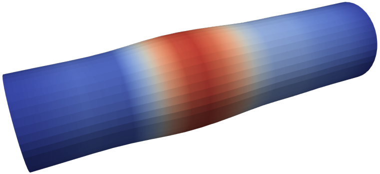
2. Construct block Gauss-Seidel smoother with:

   $$L^{-1} = \begin{pmatrix} S & & \\ & G & \\ C_{FS} & C_{FG} & F \end{pmatrix}^{-1}$$

$\rightarrow$ Proper represenation of the multi-physics problem on coarse levels and thus efficient smoothing of the error frequencies related to the coupling.

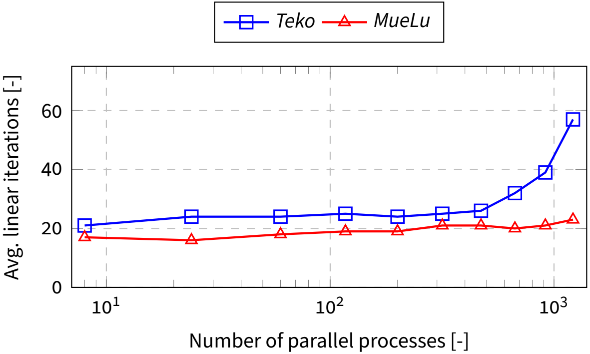
Block Gauss-Seidel components could be reused from **Teko**.

Weak scaling study based on the pressure wave benchmark problem:



Assuming matching meshes for fluid, solid and ALE!

Scaling from ≈ 60.000 DOFs to ≈ 60.000.000 DOFs.



Regarding scalability

Only the "all-in-one" multigrid method based on **MueLu** shows scalability!

Still using the **Epetra** backend ...

... thus bound to MPI only without the use of OpenMP or similar ...

... but currently in the process of switching to **Tpetra**!

Current strategy:

- Replace **Epetra** based packages with ones, which can do both: **Epetra** and **Tpetra**
- Reduce **Epetra** based self-implementations and use Trilinos functionality for it (e.g. block preconditioning with **Teko**)
- Introduce wrapper classes for **Epetra** based objects

Long road ahead ...

Transition of the linear solver stack almost complete! Transition of nonlinear solver still not clear.

4C & Epetra to Tpetra Transition

Decided to use **Thyra** and the respective framework, due to **Stratimikos** (and most likely **NOX**).

Additional challenges:

- **Thyra** vs. **Xpetra** situation going on in the code → especially with our block matrix implementation and it's wrapping to `Thyra::PhysicallyBlockedLinearOp` vs. `Xpetra::BlockedCrsMatrix` (most cumbersome point is the GID numbering)
- Actively removing of almost all `Teuchos::RCP` as they pollute 4C (tend to be overused when not necessary) → trying to avoid them
- Internal handling of `Teuchos::ParameterList` and the recent changes made to it (`Teuchos_MODIFY_DEFAULTS_DURING_VALIDATION`)

Keep continuous integration running ...

Always guarantee that 4C builds with the current **Trilinos** develop branch!

Thank you!

4C has in the past, is currently and will in the future heavily build on **Trilinos** and uses a lot of it's features to do application-driven research! Excited what's to come with **Tpetra** (and **Kokkos**)!

Collaborators:

- Matthias Mayr, UniBw M
- ... all 4C developers!

References: For more information and use-cases have a look at the 4C-multiphysics website: https://www.4c-multiphysics.org/

Financial support:

dtec.bw
Funded by the European Union
NextGenerationEU

dtec.bw: Digitalization and Technology Research Center of the Bundeswehr through the project *hpc.bw: Competence Platform for High Performance Computing*

Contact:

- max.firmbach@unibw.de (https://www.unibw.de/imcs-en)