HPSF Community Summit
TU Braunschweig | Germany
February 25-27, 2026

der Bundeswehr Universität München

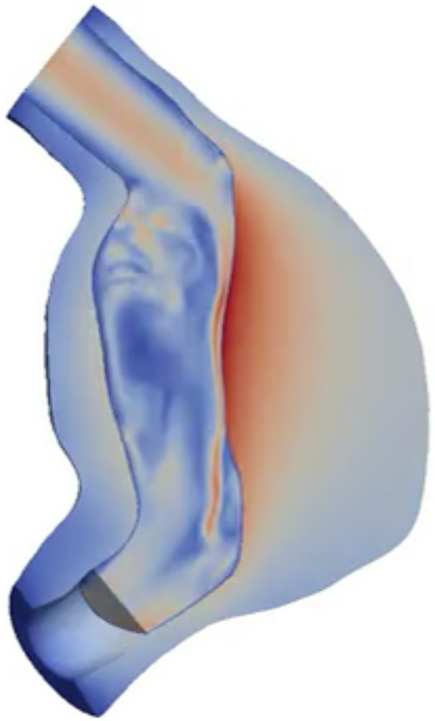
Modernizing the 4C linear algebra backend: from Epetra to Tpetra

Matthias Mayr[1,2], Max Firmbach[1], Martin Frank[1], Vladimir Ivannikov[3]

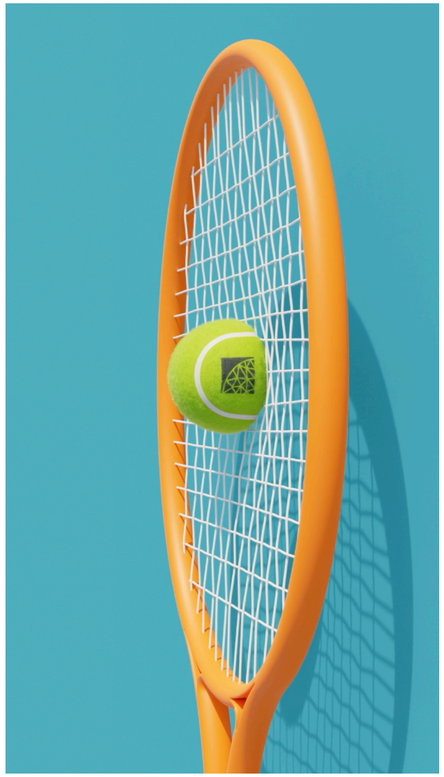
[1]Institute for Mathematics and Computer-Based Simulation | University of the Bundeswehr Munich | Germany
[2]Data Science & Computing Lab | University of the Bundeswehr Munich | Germany
[3]InInstitute of Material Systems Modeling | Helmholtz-Zentrum Hereon | Germany

Comprehensive Computational Community Code (4C)

4C is a parallel multiphysics research code to analyze and solve a plethora of physical problems by means of numerical simulation and computational science.



4C COMPREHENSIVE COMPUTATIONAL COMMUNITY CODE



… enabled by Trilinos.

Guiding principle

Software as a tool to develop and implement innovative numerical methods for research questions in coupled multiphysics systems driven by the needs of applications

Science

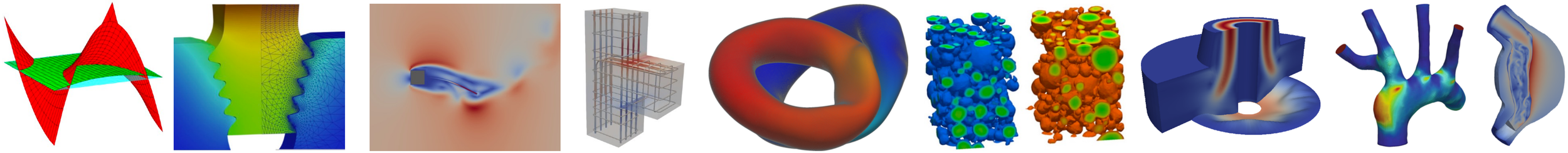
- Numerical methods & algorithms
- Material modeling
- Biopolymer networks
- Fluid/solid interaction
- Mixed-dimensional modeling
- …

Engineering

- Contact & tribology
- Aerospace engineering
- All-solid-state batteries
- Additive manufacturing
- Structural health monitoring of bridges
- …

Biomedicine

- Respiratory system
- Cardiovascular system
- Musculoskeletal system
- Tumors
- Stomach
- Stents & stent grafts
- Growth & remodeling
- …



Single fields

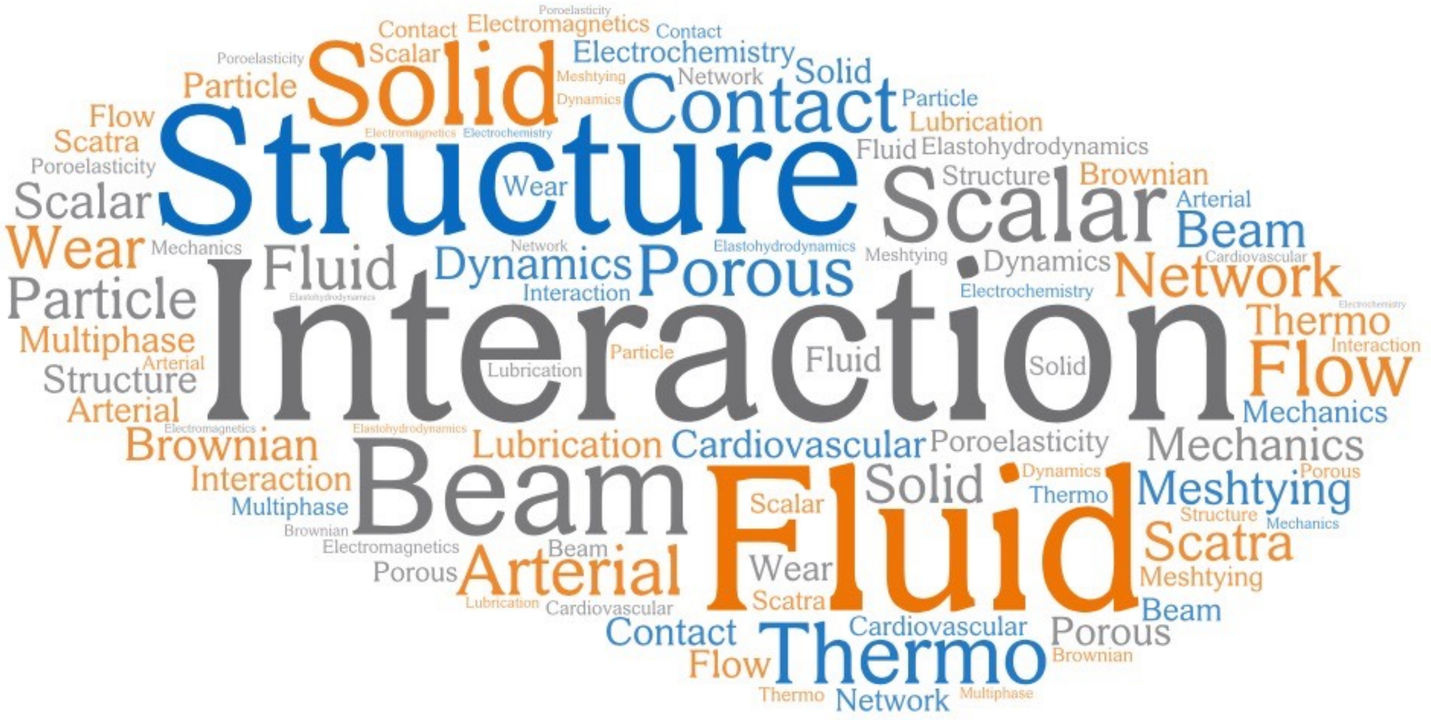
- Solid & structural mechanics
- Fluid flow
- Scalar transport & reactions
- Heat conduction

Coupled multiphysics

- Coupling algorithms (FBI, FSI, SSI, STI, TSI, ...)
- Mortar methods
- Block preconditioning

Numerical methods

- Finite Elements, IGA & CutFEM
- Particle algorithms
- Time integration
- Nonlinear solver
- Linear solvers & preconditioners



4C and its dependencies

4C: Some technical aspects

- Leverages the Trilinos project for sparse linear algebra, nonlinear solvers, and linear solvers and preconditioners
- Fully implemented in C++
- Parallelized with MPI for distributed memory hardware architectures
- Open-source development process on GitHub (https://github.com/4C-multiphysics/4C)
  - jointly developed by several groups (TUM, UniBw M, Hereon, RUB)

Tool	Use case
Trilinos	Sparse linear algebra, (non-)linear solvers
ArborX	Search algorithms
CMake & Crest	Build & test system
MPI	Parallelization
GitHub / git	Collaboration, review, CI, version control
Doxygen	Source code documentation
sphinx	Documentation & user guide

Since the beginning (~2005)

- Sparse linear algebra (Epetra)
- Linear solvers and preconditioners (Amesos, AztecOO, Ifpack, ML)
- Utilities: Teuchos, Zoltan

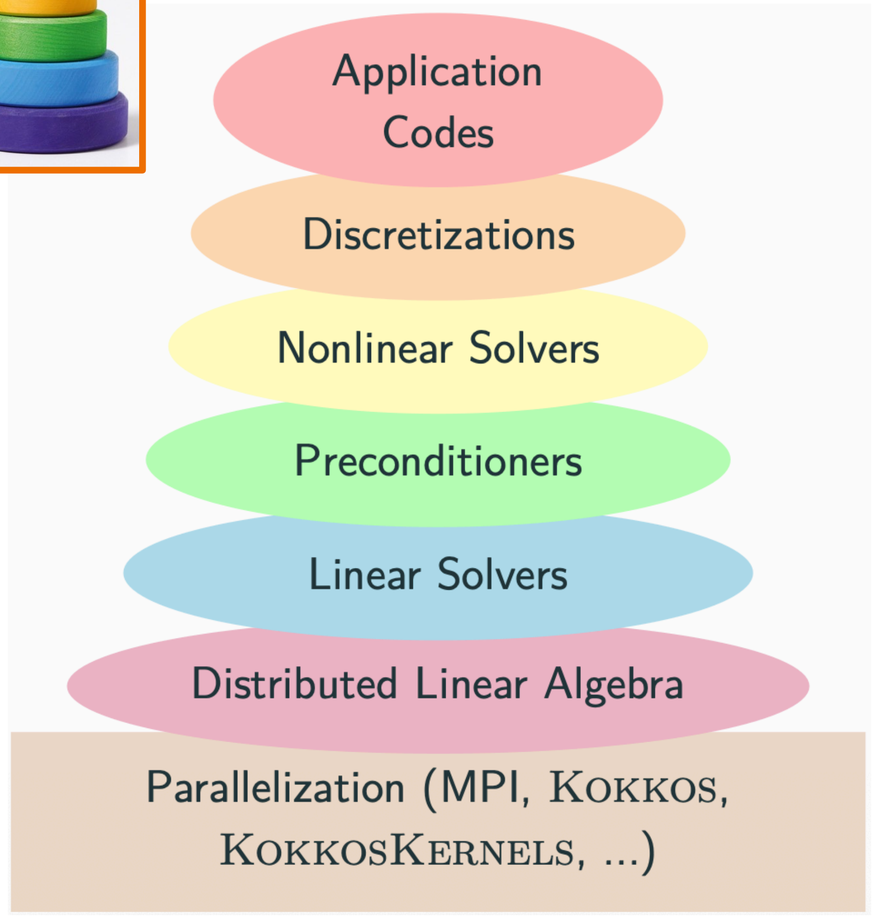
Updates and changes over time

- Linear solvers & preconditioners (Teko, Amesos ➡ Amesos2, AztecOO ➡ Belos, ML ➡ MueLu,)
- Update to utilities: Isoropia ➡ Zoltan2
- Finite Element Technology: Intrepid2
- Nonlinear solvers (NOX)

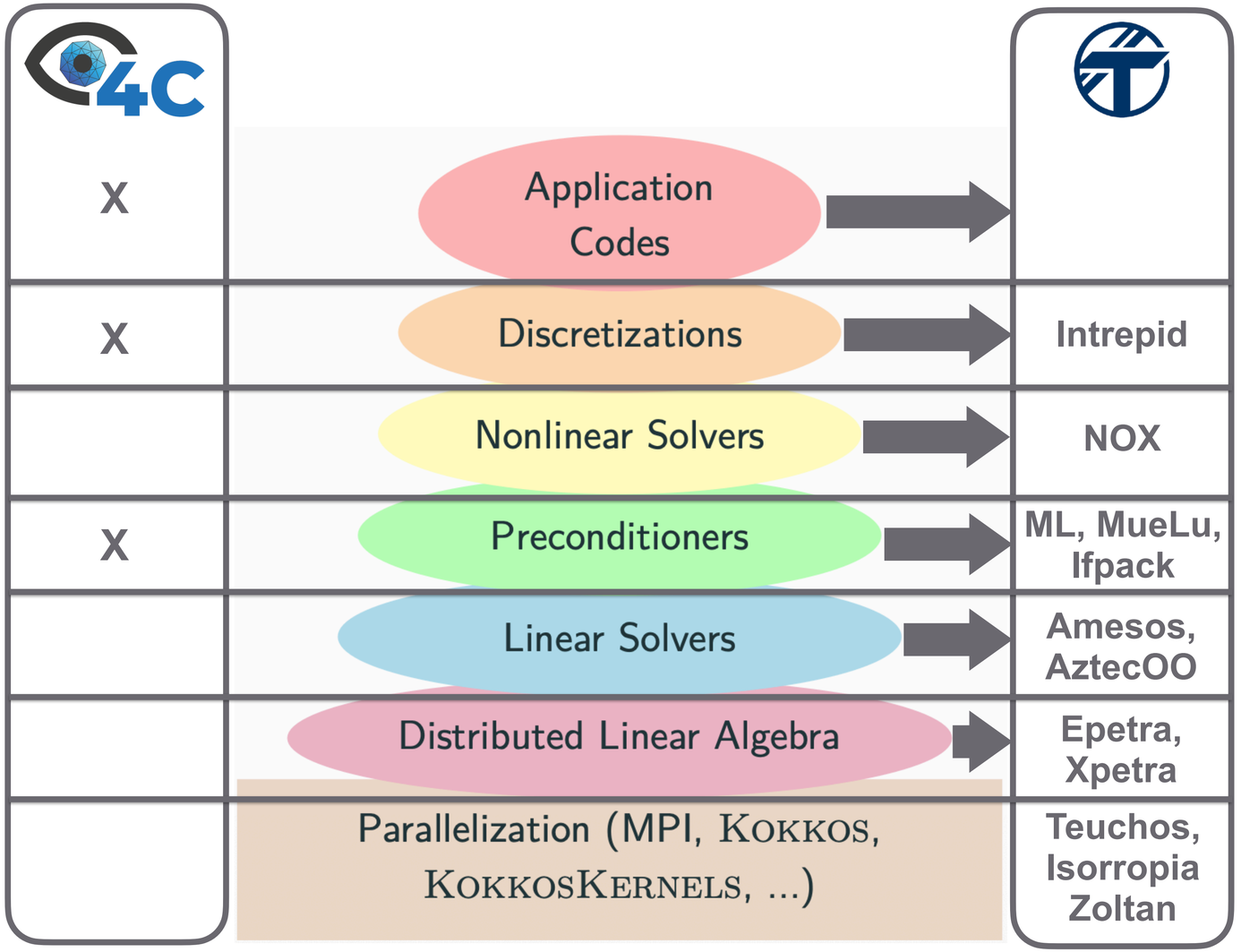
In progress

- Epetra ➡ Tpetra
- Ifpack ➡ Ifpack2
- Stratimikos

Organization of Trilinos



Trilinos in 4C



Co-development of 4C and Trilinos

Some characteristics of multiphysics systems

- Block matrices
- Properties vary across fields
- Ill-conditioned linear systems

Co-development

- Block preconditioning (Teko, MueLu)
- Algebraic multigrid methods (MueLu)

Utilities for block matrices

From 4C to Trilinos

- Xpetra::Blocked[...]
- Xpetra::MapExtractor
- Some MueLu factories

Use cases

Fluid/solid interaction | **Contact mechanics** | **Beam/solid interaction** | **…**

4C: How do we integrate Trilinos?

Wrappers

- Currently mix of wrappers and direct use
- Transition to wrap EVERYTHING to
  - Control the interface
  - Ease transition to Tpetra

Releases (since Trilinos v16.2)

- We cannot update anymore, since we still rely on Epetra.

Installation

- Install other required software via spack
- Trilinos: manual build & installation w/ our selection of packages
- CMake integration into 4C

Migration from Epetra to Tpetra

From Epetra to Tpetra: starting point

- We used from the Epetra stack
  - AztecOO
  - Ifpack
  - ML (and some MueLu)
  - Isorropia & Zoltan
  - Epetra, EpetraExt

Integration of Trilinos in 4C

- Wrappers for some matrices (Epetra_CrsMatrix ➡ Core::LinAlg::SparseMatrix)
- Self-written matrices built upon Epetra (Core::LinAlg::BlockSparseMatrix as collection of Epetra_CrsMatrix objects with block matrix capabilites)
- Direct integration of Epetra_Vector / Epetra_MultiVector / Epetra_IntVector / Epetra_FEMultiVector
- Many linear algebra objects derive from Epetra objects (e.g. Core::LinAlg::SparseOperator & Epetra_Operator)

Consequences

- Total lack of encapsulation: Epetra headers included in almost every source code file
- Uncontrolled use of Epetra features

- Some statistics (02/2023, commit 7aee9e67d58b76b)
  - 4C source code: 1298656 lines of C++
  - Number of Epetra include statements: 661
  - Number of “{E,e}petra” search results: 33308

Transition strategy

- Requirements
  - We need a working code at all times.
  - Chunks of changes need to be small or at least managable.
- Approach: exploit layered structure of Trilinos-based applications

Phase 1

- Some packages work with Epetra and Tpetra.
- Tackle them first:
  - Amesos ➡ Amesos2
  - AztecOO ➡ Belos
  - Isorropia ➡ Zoltan2
  - ML ➡ MueLu

Phase 2

- Wrap Epetra objects into our own linear algebra implementation
  - Epetra_CrsMatrix ➡ Core::LinAlg::SparseMatrix
  - Epetra_MultiVector ➡ Core::LinAlg::MultiVector
  - Epetra_CrsGraph ➡ Core::LinAlg::Graph
  - …

Phase 3

- Delay Epetra-only packages
  - Ifpack ➡ Ifpack2
  - Epetra ➡ Tpetra
- Unify interface to Trilinos packages
  - Stratimikos

Nonlinear solvers: NOX

- Capabilities
  - Implements a variety of Newton-based globalization techniques including Line Search and Trust Region algorithms
  - Provides higher and lower order models using Broyden and Tensor algorithms
  - Special algorithms for use with inexact linear solvers such as Krylov subspace techniques.

4C: Before

- NOX for nonlinear solvers (Newton)
- Callback to linear algebra through NOX::Epetra::

4C: Now

- Work with NOX::Abstract:: and our own linear algebra wrappers (Core::LinAlg::)
- Implement our own interface to NOX

- Options
  - NOX::Thyra:: ➡ powerful, but (too) complex
  - NOX::Tpetra:: ➡ no working implementation, yet.
  - Derive from NOX::Abstract:: ➡ full control, but implement everything yourself

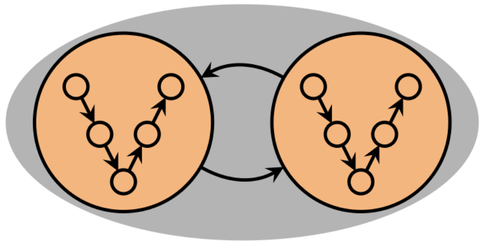
Current state

Capability	Package	Comment	Status
Misc	Teuchos	Teuchos::BLAS, use MPI_COMM directly	✓
Discretization	Intrepid > Intrepid2		✓
Nonlinear solvers	NOX	Add fixes to NOX, work with our own linear algebra objects	✓
Preconditioners	ML, MueLu > MueLu	MueLu template xml files for convenience	✓
Preconditioners	Teko	Replace many in-house block preconditioners	⚙⚙
Linear solvers	AztecOO > Belos		✓
Linear solvers	Amesos > Amesos2	SuperLU_DIST v7.2.0, add global reindexing to Amesos2	✓
Distributed linear algebra	Epetra, EpetraExt > Tpetra	Wrap Epetra into Core::Linalg::	⚙⚙
Serial linear algebra	Epetra > Teuchos	Teuchos::SerialDense* (future: Kokkos::View, std::mdspan)	✓
Parallelism	Isorropia, Zoltan > Zoltan2	Different results for ParMETIS due to apparently different default values	✓

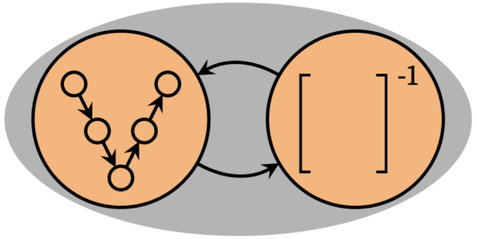
Background

- Multi-physics systems often yield block matrices
- Efficient preconditioning essential for solution process

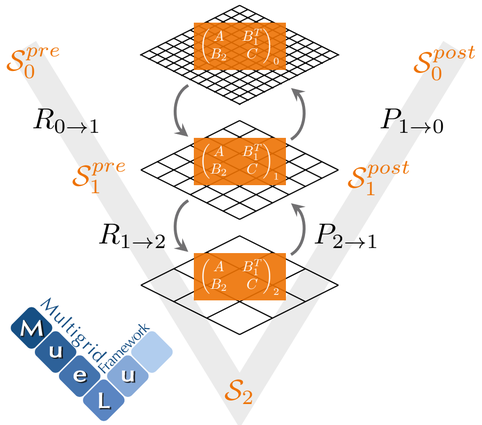
- Block-iterative approach
  - Coupling on fine level
  - AMG for each block



  - AMG for some blocks



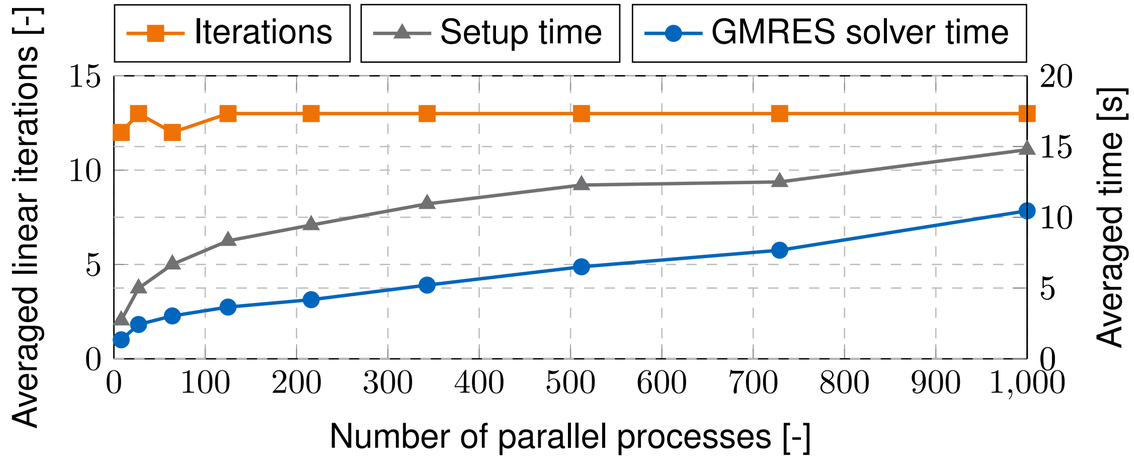
- Fully coupled AMG



  - Segregated transfer operators
  - Block level smoothers
  - Coupling on all levels

Where are we?

- Fully coupled AMG through MueLu
- Block-iterative approach for
  - Fluid/solid (3x3 block matrix) ✓
  - Beam/solid interaction (2x2 or 3x3 block matrix) ✓



Open questions

- Talk by Christoph Schmidt

Work in progress: Epetra

Where are we?

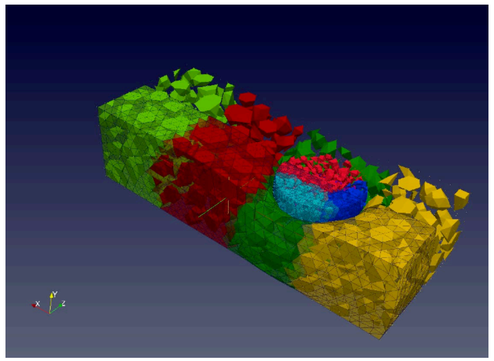
- Core::LinAlg:: provides wrappers for distributed Epetra objects such as MultiVector, Graph, CrsMatrix, …
- Localization of Epetra includes into a handful of wrappers
- Proper assertion for Epetra-internal errors w/o additional user interaction

Current activities

- Clean-up wrappers
  - Unify duplicated implementations
  - Remove unused functions
- Prepare for Tpetra
  - Remove methods without a Tpetra counterpart (e.g. replace (e.g. MinValue, MaxValue, ReciprocalMultipy)
  - Use std::span instead of raw pointers to arrays

Open questions (selection)

- Inheritance from Epetra_Operator
- How to handle FEVector
  - Epetra: two distinct classes
  - Tpetra: one class with two modi operandi
- Allocation of sparse matrices
  - Tpetra requires static patter, i.e. no dynamic memory allocation after creation of object
  - Single-field problem ✓
  - Multi-field problem with fixed coupling (such as fluid/solid interaction) ✓
  - Problem with changing interactions (e.g. contact) ✗



From Epetra to Tpetra: current state

- We still use from the Epetra stack
  - ~~AztecOO~~ ➡ Belos
  - Ifpack
  - ~~ML (and some MueLu)~~ ➡ MueLu
  - ~~Isorropia & Zoltan~~ ➡ Zoltan2
  - Epetra, EpetraExt

Integration of Trilinos in 4C

- Wrappers for all sparse matrices
- Self-written matrices built upon Epetra (Core::LinAlg::BlockSparseMatrix as collection of Epetra_CrsMatrix objects with block matrix capabilites)
- Wrappers for all distributed vectors

Consequences

- Good encapsulation: Epetra headers included in just a handful of source code file
- Controlled use of Epetra features

- Some statistics (02/2026, commit b937fce7423ed62e)
  - 4C source code: 1522109 lines of C++
  - Number of Epetra include statements: 29
  - Number of “{E,e}petra” search results: 1100

- There is only one way: forward

Milestones / Migration to Tpetra

Edit | Close Milestone | New issue

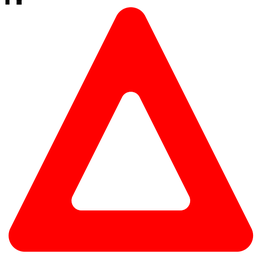
Migration to Tpetra

Open · Overdue by 2 month(s) · Due by November 24, 2025 Last updated 5 days ago

87% complete

Pain points and challenges

- Migration requires a certain level of software engineering skills.
- Migration itself does not advance research.
- No updates of Trilinos anymore



Important

- Finish internal restructuring and wrapping of sparse linear algebra objects
- Clarify open questions around static patterns for sparse matrix allocation

- Timeline: unclear

Concluding remarks

- 4C ...
  - ... has used Trilinos from the beginning
  - ... is in the process of migrating to Tpetra
  - ... will remain a Trilinos user & contributor

Collaborators:

The 4C Development team, in particular

- Max Firmbach (UniBw M)
- Martin Frank (UniBw M)
- Vladimir Ivannikov (Hereon)

Resources

- https://4c-multiphysics.org
- https://github.com/4C-multiphysics/4C

Funding:

dtec.bw - Digitalization and Rechnology Research Center of the Bundeswehr. dtec.bw is funded by the European Union – NextGenerationEU.

dtec.bw
Zentrum für Digitalisierungs- und Technologieforschung der Bundeswehr

Finanziert von der Europäischen Union
NextGenerationEU

Contact:

- matthias.mayr@unibw.de
- https://www.unibw.de/imcs-en
- GitHub: @mayrmt

Thank you.