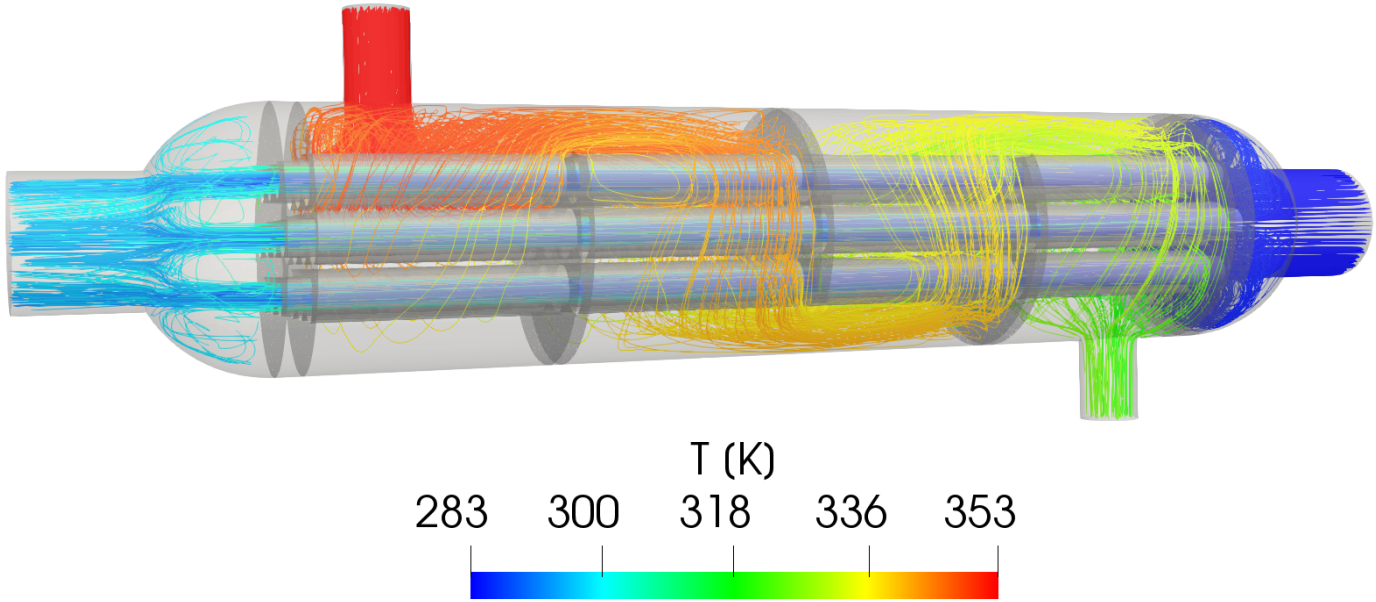
A Coupling Library for Partitioned Multi-Physics Simulations

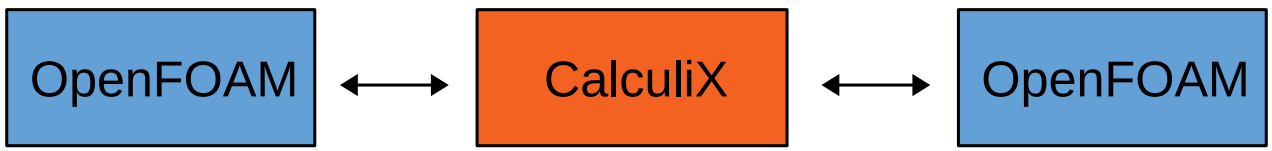
1. What is preCICE?
2. How to get started?
3. How can I couple my own code?

1. What is preCICE?

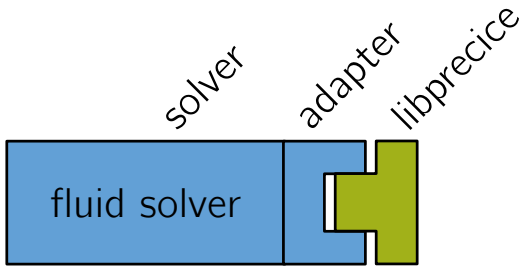
Example: Shell-And-Tube Heat Exchanger



- Partitioned coupling: Usage of three independent solvers
- Reuse of existing solvers



preCICE – A Plug-and-Play Coupling Library



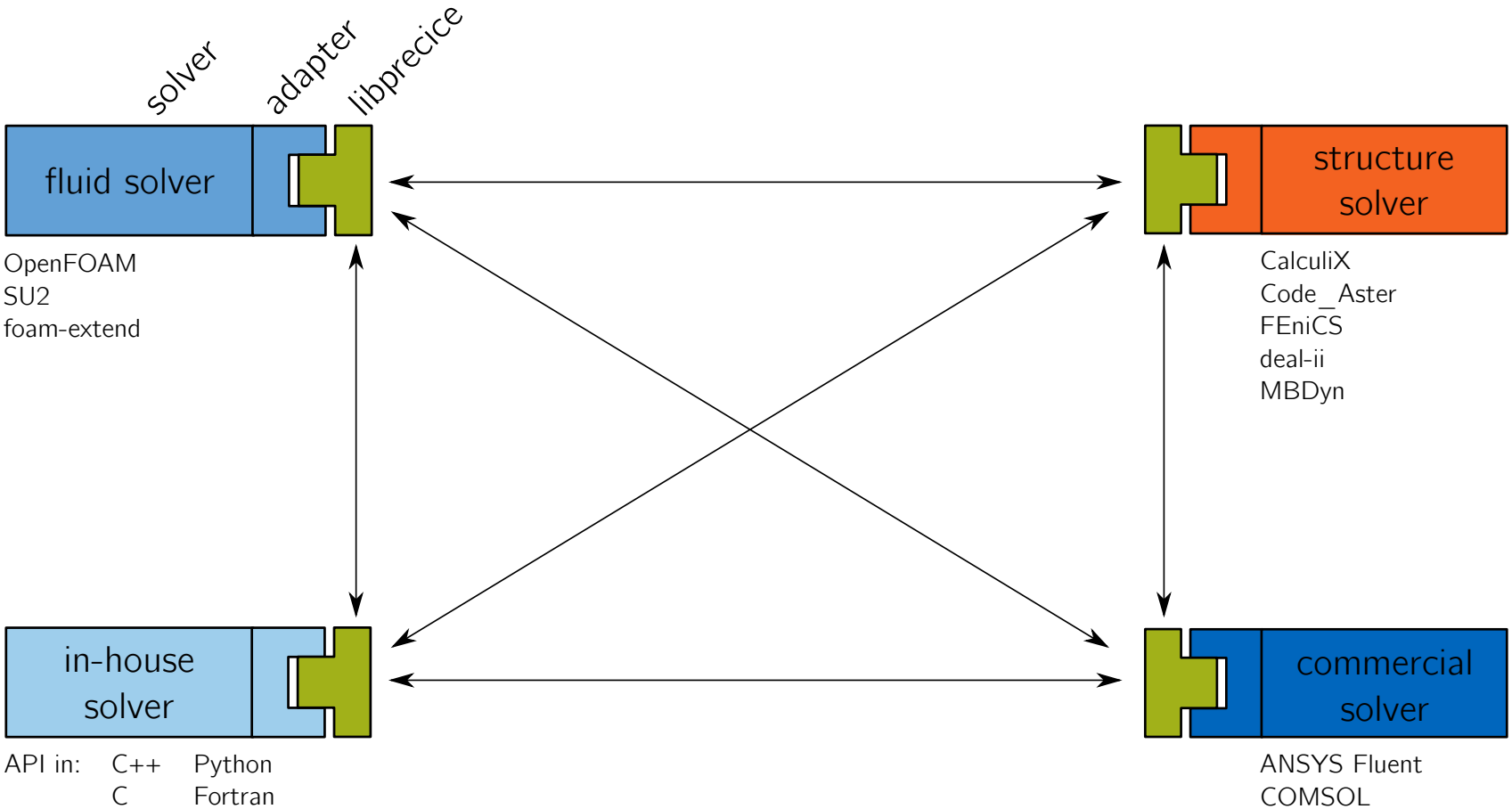
OpenFOAM
SU2
foam-extend

preCICE – A Plug-and-Play Coupling Library

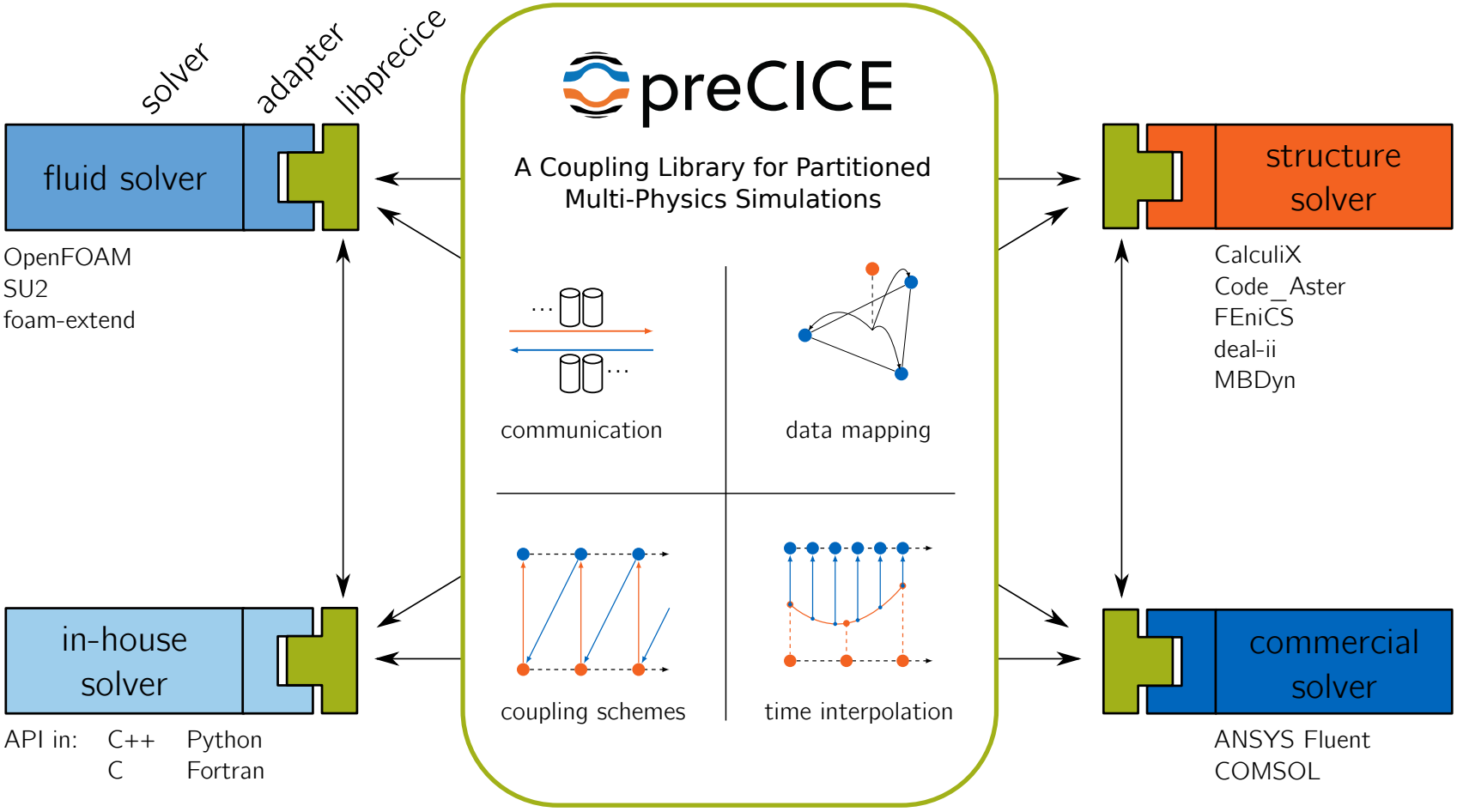
OpenFOAM
SU2
foam-extend

CalculiX
Code_Aster
FEniCS
deal-ii
MBDyn

preCICE – A Plug-and-Play Coupling Library



preCICE – A Plug-and-Play Coupling Library

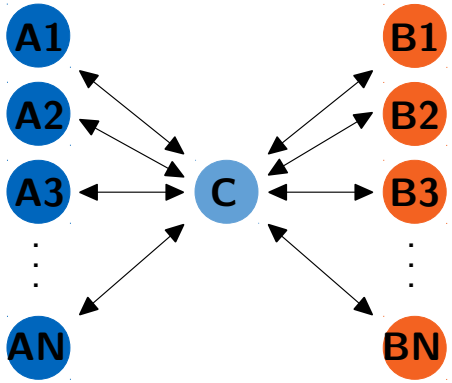


Unique Selling Points (USPs)

1. Scalability
2. Robust quasi-Newton coupling
3. Coupling of arbitrary many components
   (*arbitrary many = more than two*)
4. Minimally-invasive coupling
5. Open-source, community

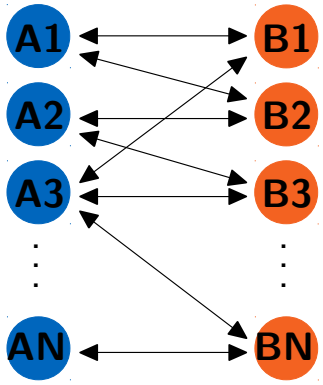
USP 1: Scalability

Server-Based Concept



- Complete communication through central server process
- Interface computations on server (in sequential)
- $\Rightarrow$ Coupling becomes bottleneck for overall simulation already on moderate parallel systems

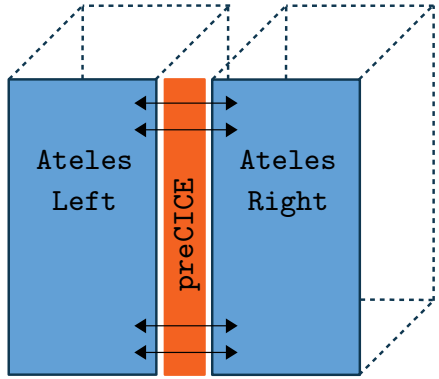
Our Peer-To-Peer Concept

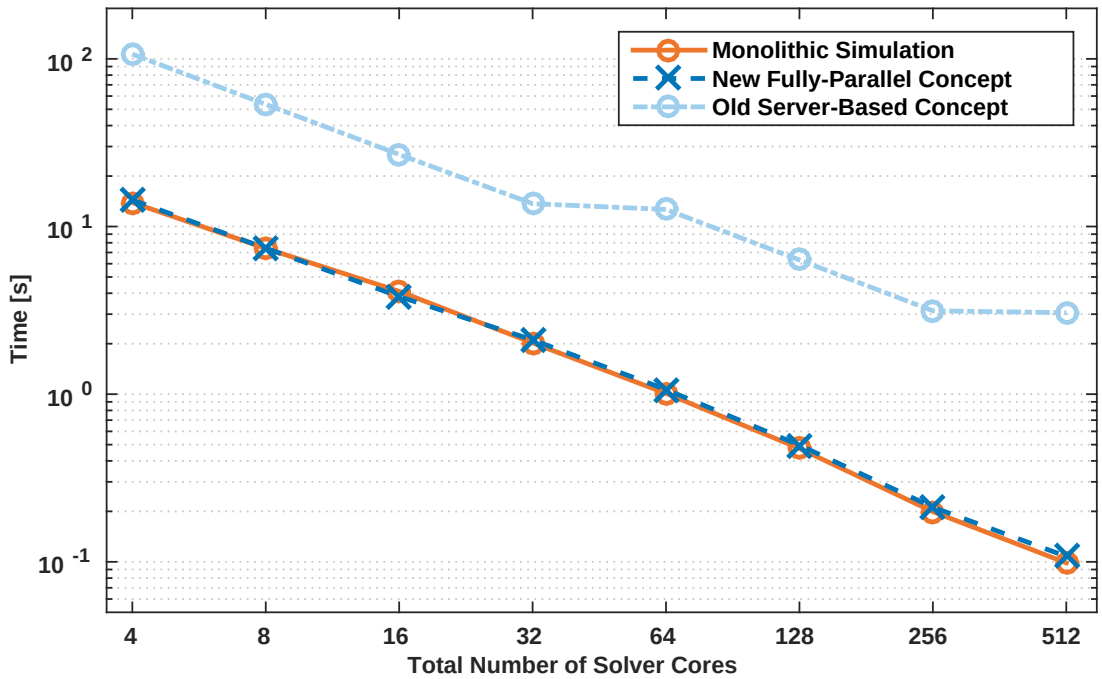


- No central entity
- $\Rightarrow$ Easier to handle (user does not need to care about server)
- $\Rightarrow$ No scaling issues

USP 1: Scalability

- Travelling density pulse (Euler equations) through artificial coupling interface
- DG solver Ateles (U Siegen), $7.1 \cdot 10^6$ dofs
- Nearest neighbor mapping and communication

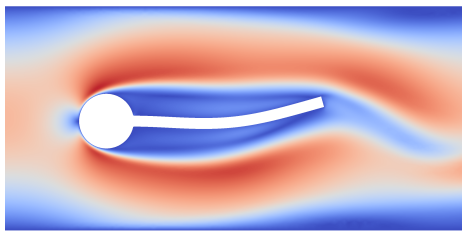




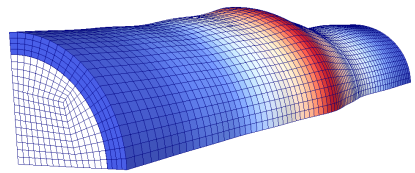
USP 2: Quasi-Newton Coupling

Coupled problem: $F : d \mapsto f,\ S : f \mapsto d \quad \rightsquigarrow \quad (S \circ F)(d) \overset{!}{=} d$

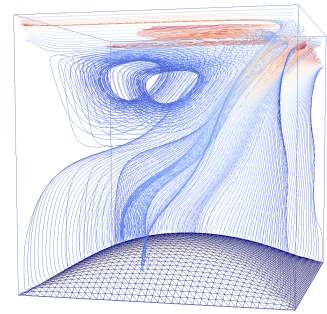
FSI3



3D-Tube



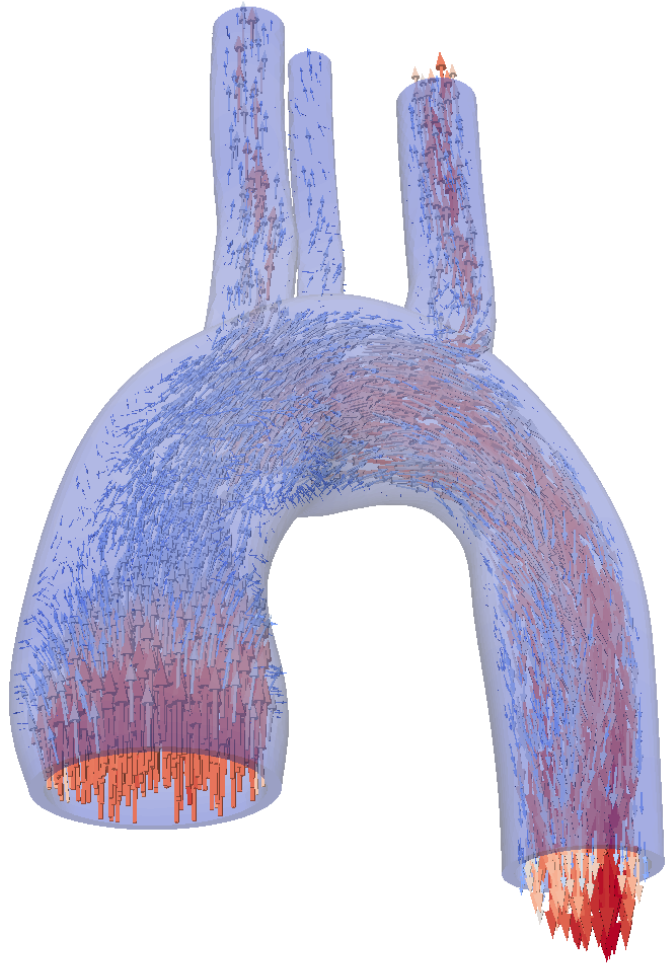
Driven Cavity



Mean Iterations	Aitken	Quasi-Newton
FSI3	17.0	3.3
3D-Tube	Div.	7.5
Driven Cavity	7.4	2.0

USP 2: Quasi-Newton Coupling

- Quasi-Newton can even handle biomedical applications, such as an Aortic bloodflow
- Stable coupling (no added-mass instabilities)
- Six times less iterations than Aitken



- Joint work with Juan-Carlos Cajas (Barcelona Supercomputing Center)
- Geometry by Jordi Martorell

Contributors

Miriam Mehl

U Stuttgart

Florian Lindner

U Stuttgart

Amin Totounferoush

U Stuttgart

Kyle Davis

U Stuttgart

Alexander Rusch

ETH Zürich

Hans Bungartz

TUM

Benjamin Rüth

TUM

Gerasimos Chourdakis

TUM

Frédéric Simonis

TUM

Benjamin Uekermann

TU/e

Previous and minor contributors:

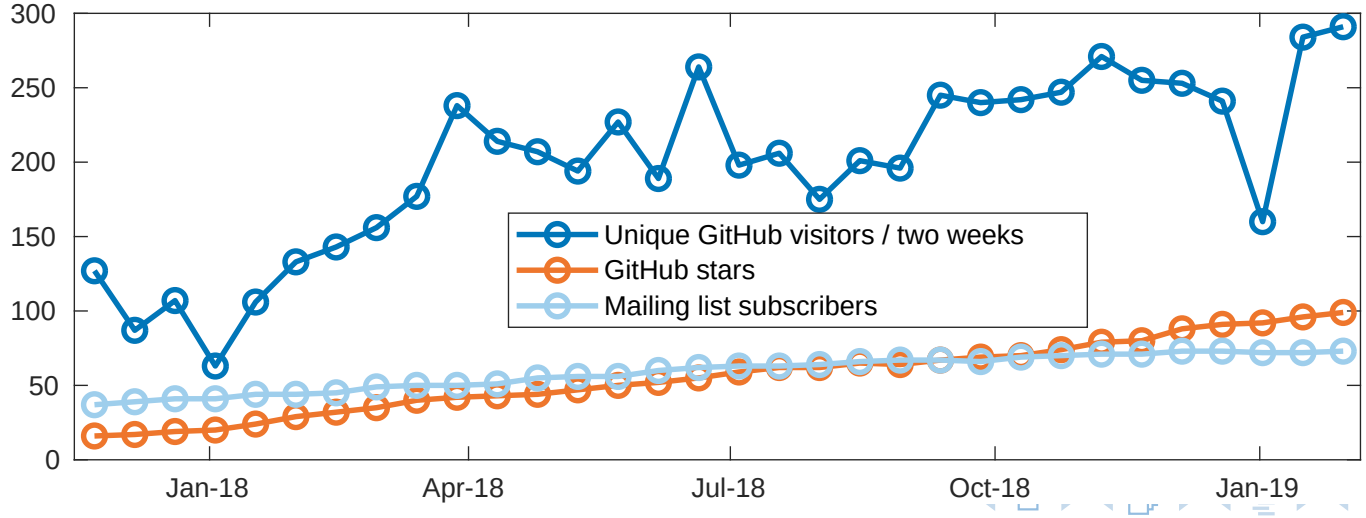
- Bernhard Gatzhammer, Klaudius Scheufele, Lucia Cheung, Alexander Shukaev, Peter Vollmer, Georg Abrams, Alex Trujillo, Dmytro Sashko, David Sommer, David Schneider, Richard Hertrich, Saumitra Joshi, Peter Meisrimel, Derek Risseeuw, Rafal Kulaga, Ishaan Desai ...

Users

- LSM & STS, U Siegen, Germany
- SC & FNB, TU Darmstadt, Germany
- SCpA, CIRA, Italy
- Cardiothoracic Surgery, UFS, South Africa
- A*STAR, Singapore
- NRG, Petten, The Netherlands
- Aerodynamics & Wind Energy (KITE Power), TU Delft, The Netherlands
- Mechanical and Aeronautical Eng., University of Manchester, UK
- University of Strathclyde, Glasgow, UK
- FAST, KIT, Germany
- AIT, Vienna, Austria
- IAG, University of Stuttgart, Germany
- CTTC UPC, Barcelona, Spain
- Amirkabir U. of Technology, Iran

Upcoming:

- GRS, Garching, Germany
- MTU Aero Engines, Munich, Germany
- Numerical Analysis, Lund, Sweden
- Helicopter Technology & Astronautics, TUM, Germany
- ATA Engineering Inc., USA
- BITS Pilani, India
- Aviation, MSU Denver, USA



2. How to get started?

Infrastructure

We are on GitHub: `https://github.com/precice`

- LGPL3 license
- User documentation in the wiki

Building

Dependencies

- Eigen, Boost (version $\geq$ 1.65), libxml2
- Optional: PETSc, Python (incl. Numpy), MPI

The easy way

- Ubuntu 18.04: All dependencies available through distribution
- Ubuntu 16.04: All dependencies available except Boost

Still doable

- macOS
- Other Linux distributions

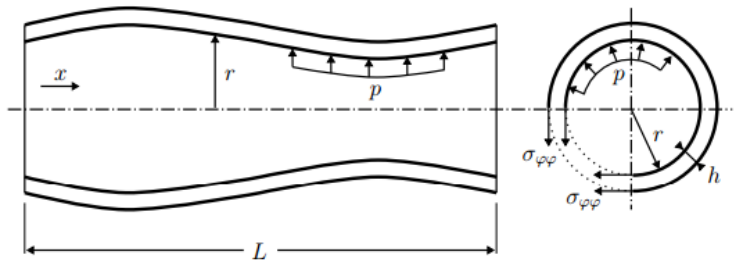
Experimental

- Conda, Docker, Debian package, Spack
- Windows

Tutorials

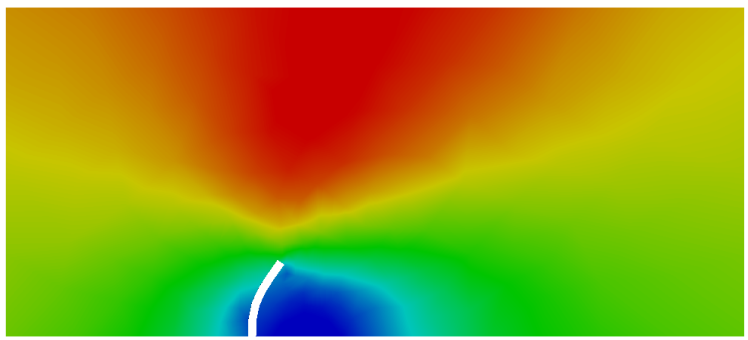
1D Elastic Tube

- Simple provided solvers
- Learn about API and configuration



Flexible beam

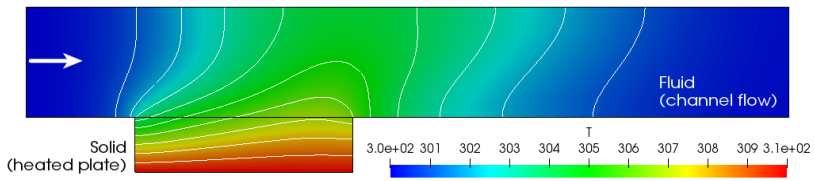
- Fluid-structure interaction
- Couple SU2 or OpenFOAM to CalculiX
- Learn about coupling schemes
- Also interactive version available in browser `http://run.coplon.de/`



Tutorials

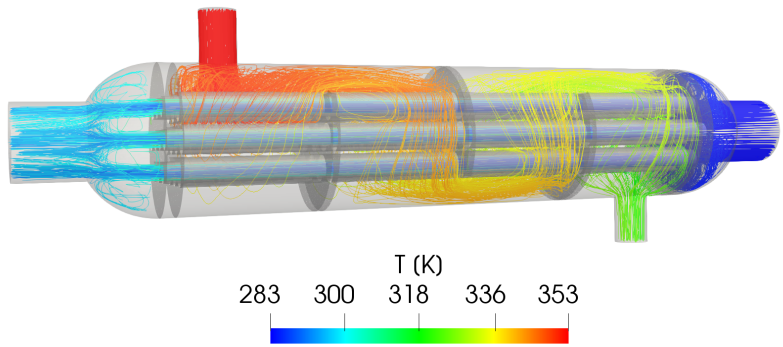
Flow over a Heated Plate

- Conjuagte-heat transfer
- Couple two OpenFOAM solvers
- Learn about OpenFOAM adapter



Heat exchanger

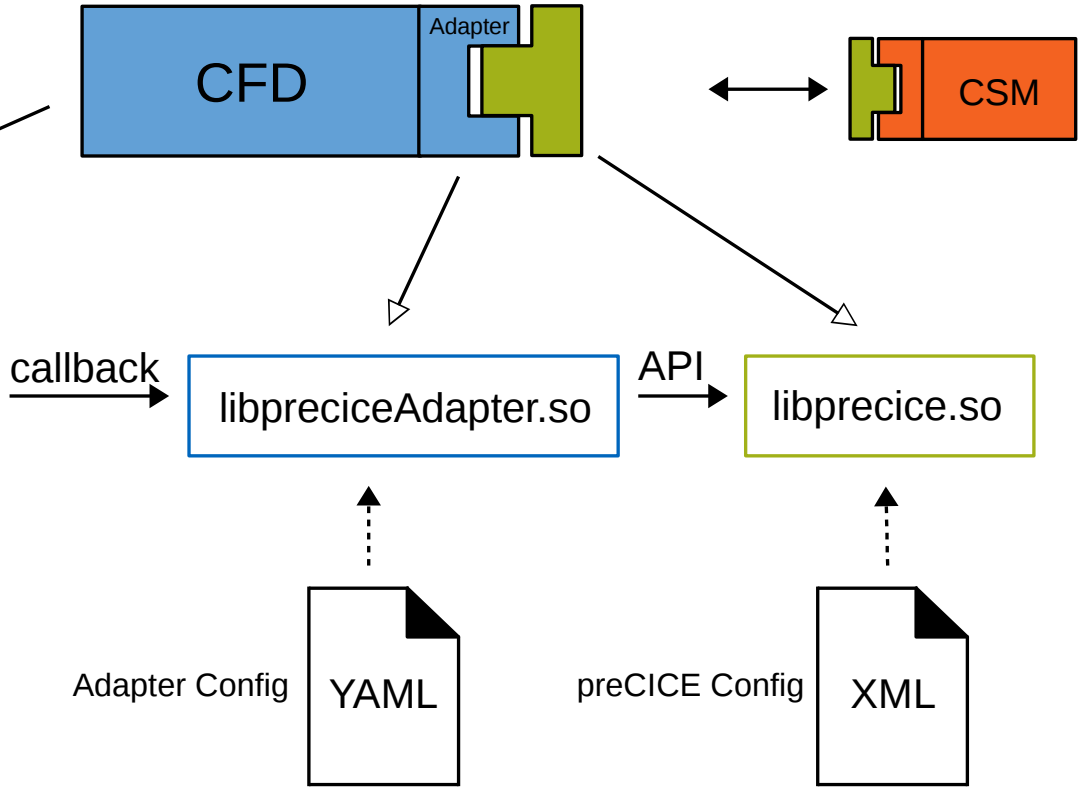
- Conjugate-heat transfer
- Couple two OpenFOAM instances with CalculiX
- Learn about multi coupling



The OpenFOAM Adapter

```
int main(int argc, char *argv[])
{
    #include "setRootCase.H"
    #include "createTime.H"
    #include "createMesh.H"
    ...

    while (runTime.loop())
    {
        Info<< "Time = " ... << endl;

        #include "CourantNo.H"

        // Momentum predictor

        fvVectorMatrix UEqn
        (
            fvm::ddt(U)
          + fvm::div(phi, U)
          - fvm::laplacian(nu, U)
        );

        if (piso.momentumPredictor())
        {
            solve(
                  UEqn
                  ==
                  -fvc::grad(p)
                 );
        }

        // --- PISO loop
        while (piso.correct())
        {
            ...
        }

        runTime.write();
    }

    Info<< "End\n" << endl;
    return 0;
}
```

myFoam



Flow over a Heated Plate

Load adapter at runtime in `system/controlDict`:

```
functions
{
    preCICE_Adapter
    {
        type preciceAdapterFunctionObject;
        libs ("libpreciceAdapterFunctionObject.so");
    }
}
```

Define coupling boundary in `system/blockMeshDict`:

```
interface
{
    type wall;
    faces
    (
        (4 0 1 5)
    );
}
```

Flow over a Heated Plate

Configure adapter in `precice-adapter-config.yml`:

```yaml
participant: Fluid

precice-config-file: /path/to/precice-config.xml

interfaces:
- mesh: Fluid-Mesh
  patches: [interface]
  write-data: Temperature
  read-data: Heat-Flux
```

Flow over a Heated Plate

3. How can I couple my own code?

How to couple my own code?

```
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");
precice.setMeshVertices();
precice.initialize();

while (precice.isCouplingOngoing()) { // main time loop
    solve();

    precice.writeBlockVectorData();
    precice.advance();
    precice.readBlockVectorData();

    endTimeStep(); // e.g. write results, increase time
}

precice.finalize();
```

- Timesteps, most arguments, and less important methods omitted.
- Full example in the wiki.
- API in C++, C, Fortran, and Python

Roadmap

Current Developments v1.4

- Debian package
- Building with cmake

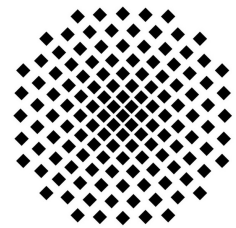
Current Developments Adapters

- Fluid-fluid module for the OpenFOAM adapter
- Update of Fluent adapter
- Official adapters for dealii, FEniCS, and Nutils

Long-term Goals v2.0

- 3D-1D and 3D-2D data mapping
- Parallel initialization $\rightarrow$ support of very large cases
- Support of re-meshing and dynamically changing coupling interfaces
- Consistent time interpolation

Funding



Universität
Stuttgart

Summary

preCICE

Flexible: Couple your own solver with any other
Easy: Add a few lines to your code
Ready: Out-of-the box support for many solvers
Fast: Fully parallel, peer-to-peer, designed for HPC
Stable: Implicit coupling, accelerated with Quasi-Newton
Multi-coupling: Couple more than two solvers
Free: LGPL3, source on GitHub

`www.precice.org`
`github.com/precice`
`@preCICE_org`
Mailing-list, Gitter
Literature Guide on wiki