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

OpenFOAM → CalculiX → OpenFOAM
Overview

fluid solver ——— structure solver
Overview

fluid solver adapter libprecice

structure solver
Overview

Solver (in-house):
- OpenFOAM
- SU2
- foam-extend

Fluid solver adapter:
- libprecice

Structure solver:
- CalculiX
- Code_Aster
- FEniCS
- deal-II
- MBDyn

Commercial solver:
- ANSYS Fluent
- COMSOL

API in:
- C++
- Python
- Fortran
Overview

precICE
A Coupling Library for Partitioned Multi-Physics Simulations

- Communication
- Data mapping
- Coupling schemes
- Time interpolation

solver
adapter
libprecice

fluid solver
in-house
solver

OpenFOAM
SU2
foam-extend

API in:
C++
C
Fortran

structure solver

CalculiX
Code_Aster
FEniCS
deal-II
MBDyn

commercial solver

ANSYS Fluent
COMSOL
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: \( F : d \mapsto f, \quad S : f \mapsto d \quad \implies \quad (S \circ F)(d) \overset{!}{=} d \)

<table>
<thead>
<tr>
<th>Mean Iterations</th>
<th>Aitken</th>
<th>Quasi-Newton</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSI3</td>
<td>17.0</td>
<td>3.3</td>
</tr>
<tr>
<td>3D-Tube</td>
<td>Div.</td>
<td>7.5</td>
</tr>
<tr>
<td>Driven Cavity</td>
<td>7.4</td>
<td>2.0</td>
</tr>
</tbody>
</table>
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 Chouridakis
TUM

Frédéric Simonis
TUM

Benjamin Uekermann
TU/e

Previous and further 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, Dominik Volland, Michel Takken, . . .
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, Ranshofen, Austria
- GRS, Garching, Germany
- MTU Aero Engines, Munich, Germany
- Helicopter Technology & Astronautics, TUM, Germany
- IAG & IWS & MechBau & VISUS, University of Stuttgart, Germany
- CTTC UPC, Barcelona, Spain
- Amirkabir U. of Technology, Iran
- Noise & Vibration Research Group, KU Leuven, Belgium

Upcoming:

- Numerical Analysis, Lund, Sweden
- ATA Engineering Inc., USA
- BITS Pilani, India
- Aviation, MSU Denver, USA
- IMVT, University of Stuttgart
- Engineering Science, U of Luxembourg
- Renewable and Sustainable Energy Systems & Hydrogeology, TUM, Germany
2. How to get started?
Infrastructure

We are on GitHub: github.com/precice

- LGPL3 license
- User documentation in the wiki
- Debian packages, Spack, Docker, cmake
Tutorials

1D Elastic Tube

- Simple provided solvers
- Learn about API and configuration

Flexible beam

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

Flow over a Heated Plate

▸ Conjugate-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

```c
int main(int argc, char *argv[]) {
    #include "setCaseFiles.H"
    #include "createGridBox.H"
    #include "createMesh.H"
    ...
    while (runTime.loop()) {
        Info<< "Time = " << endl;
        #include "CurrentSolution.H"
        // Momentum predictor
        fVectorMatrix UEqn;
        ...
        Info<< "End of UEqn" << endl;
        return 0;
    }
}
if (piso.momentumPredictor()) {
    solve(UEqn
       ...
       fEqn::grad(p);
    );
    // .... PISO loop
    while (piso.correct()) {
        ...
        runTime.write();
        Info<< "End of run" << endl;
        return 0;
    }
}
```

Adapter Config
- YAML
- XML

libpreciceAdapter.so

myFoam

CFD Adapter

libprecice.so

API

CSM

callback
Flow over a Heated Plate

Load adapter at runtime in `system/controlDict`:

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

Define coupling boundary in `system/blockMeshDict`:

```plaintext
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

Solid: waiting...

Fluid: preparing...
3. How can I couple my own code?
How to couple my own code?

```cpp
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
Funding

H2020 grant 754462
Summary

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