

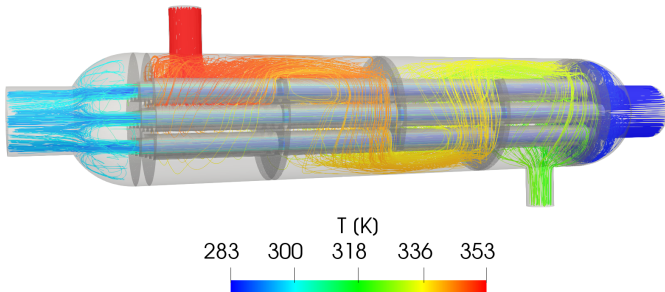


**A Coupling Library  
for Partitioned Multi-Physics  
Simulations**

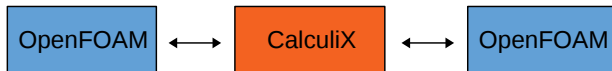
1. What is preCICE?
2. How to get started?
3. How can I couple my own code?

# 1. What is preCICE?

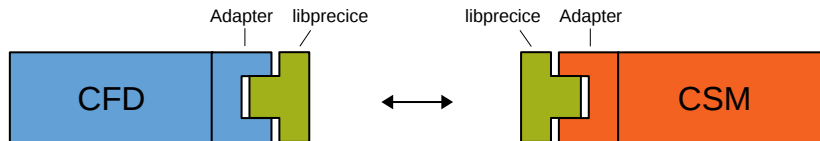
## Example: Shell-And-Tube Heat Exchanger



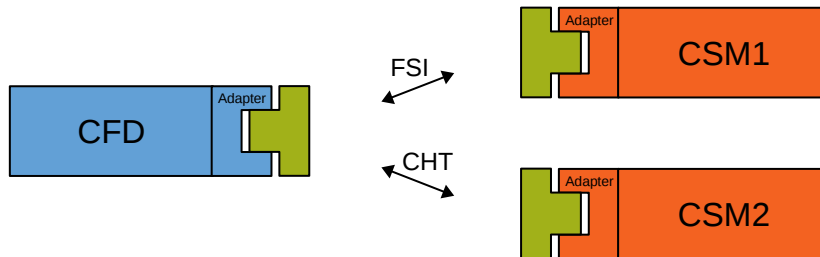
- ▶ Partitioned coupling: Usage of three independent solvers
- ▶ Reuse of existing solvers



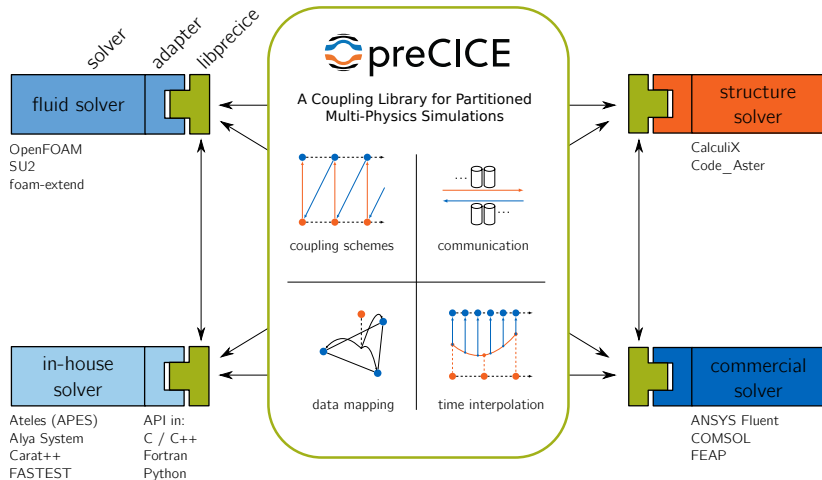
# preCICE – A Plug-and-Play Coupling Library



# preCICE – A Plug-and-Play Coupling Library



# preCICE – A Plug-and-Play Coupling Library



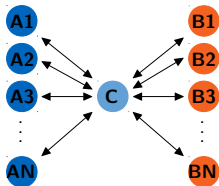
# USPs

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



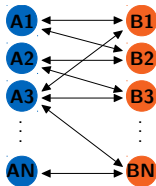
# USP 1: Scalability

## 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

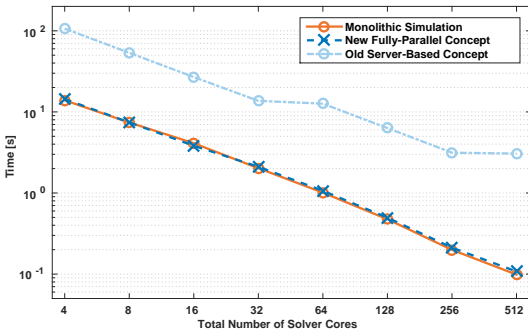
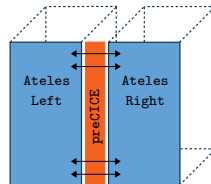
## Our Peer-To-Peer Concept



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

# USP 1: Scalability

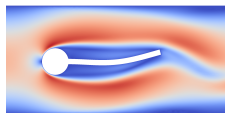
- ▶ Travelling density pulse (Euler equations) through artificial coupling interface
- ▶ DG solver Ateles (U Siegen),  $7.1 \cdot 10^6$  dofs
- ▶ Nearest neighbor mapping and communication



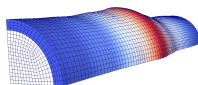
## USP 2: Quasi-Newton Coupling

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

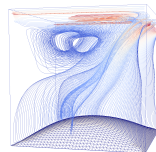
FSI3



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



Alexander Rusch  
ETH Zürich



Hans Bungartz  
TUM



Benjamin Rüh  
TUM



Gerasimos  
Chourdakis  
TUM



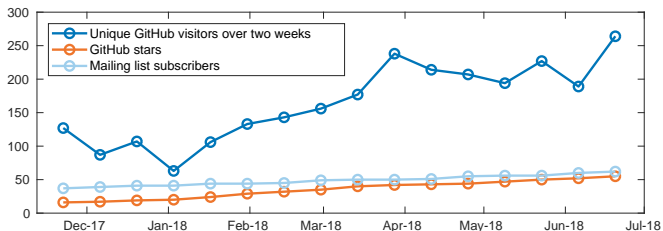
Benjamin  
Uekermann  
TUM

Previous contributors:

- ▶ Bernhard Gatzhammer, Klaudius Scheufele, Lucia Cheung, Alexander Shukaev, Peter Vollmer, Georg Abrams, ...

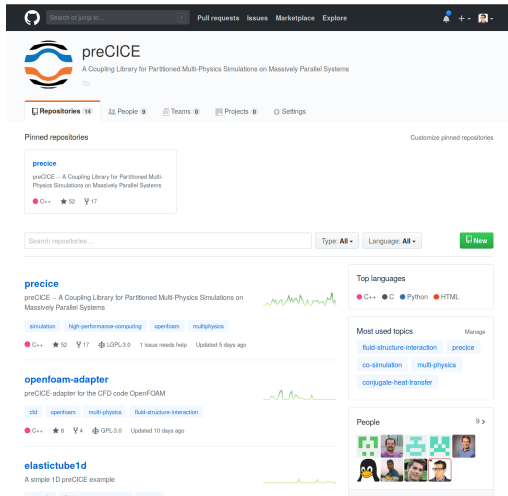
# Users

- ▶ Dr.-Ing. Fettah Aldudak et al., LSM, University of Siegen, Germany
- ▶ Davide Cinquegrana, PhD et al., SCpA, CIRA, Italy
- ▶ Kyle Davis et al., Cardiothoracic Surgery, University of the Free State, South Africa
- ▶ Dr. Lee Margetts et al., Mechanical and Aeronautical Engineering, University of Manchester
- ▶ Vinh-Tan Nguyen, PhD et al., A\*STAR, Singapore
- ▶ Dr. Thorsten Reimann et al., SC, TU Darmstadt, Germany
- ▶ Prof. Sabine Roller et al., STS, University of Siegen, Germany
- ▶ Prof. Michael Schäfer et al., FNB, TU Darmstadt, Germany
- ▶ Dr. Qing Xiao et al., CFD & FSI-RG, University of Strathclyde, UK
- ▶ Prof. Alexander van Zuijlen et al., Aerospace Engineering, TU Delft
- ▶ Global Research for Safety (GRS), Garching (upcoming)
- ▶ ...



## 2. How to get started?

We are on GitHub: <https://github.com/precice>



- ▶ LGPL3 license
- ▶ User documentation in the wiki



# Building

## Dependencies

- ▶ Eigen, Boost (version  $\geq 1.60$ ), 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

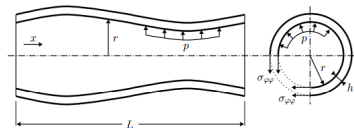
- ▶ macOS
- ▶ Other Linux distributions

## Experimental

- ▶ Conda, Docker, Debian package
- ▶ Windows

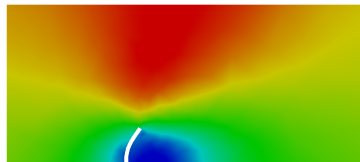
## 1D Elastic Tube

- ▶ Simple provided solvers
- ▶ Learn about API and configuration



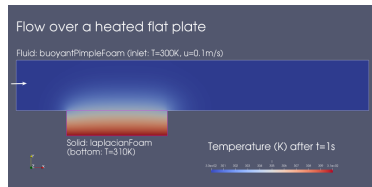
## Flexible beam

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



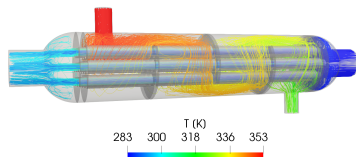
## Flow over a Heated Plate

- ▶ Conjugate-heat transfer
- ▶ Couple two OpenFOAM solvers
- ▶ Learn about OpenFOAM adapter

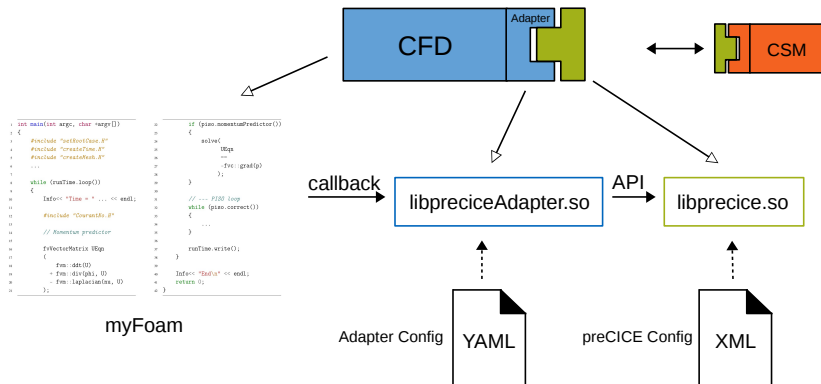


## Heat exchanger

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



# The OpenFOAM Adapter



## Flow over a Heated Plate

Load adapter at runtime in system/controlDict:

```
1 functions
2 {
3     preCICE_Adapter
4     {
5         type preciceAdapterFunctionObject;
6         libs ("libpreciceAdapterFunctionObject.so");
7     }
8 }
```

Define coupling boundary in system/blockMeshDict:

```
1 interface
2 {
3     type wall;
4     faces
5     (
6         (4 0 1 5)
7     );
8 }
```

# Flow over a Heated Plate

Configure adapter in precice-adapter-config.yml:

```
1 participant: Fluid
2
3 precice-config-file: /path/to/precice-config.xml
4
5 interfaces:
6 - mesh: Fluid-Mesh
7   patches: [interface]
8   write-data: Temperature
9   read-data: Heat-Flux
```

# Flow over a Heated Plate

```

/bin/bash B3X50
[...][preCiceAdapter][DEBUG] write-data :
[...][preCiceAdapter][DEBUG] Heat-Flux
[...][preCiceAdapter][DEBUG] read-data :
[...][preCiceAdapter][DEBUG] Temperature
[...][preCiceAdapter][DEBUG] subcycling : 1
[...][preCiceAdapter][DEBUG] prevent early exit : 1
[...][preCiceAdapter][DEBUG] evaluate boundaries : 1
[...][preCiceAdapter][DEBUG] disable checkpointing : 0
[...][preCiceAdapter][DEBUG] CHT module enabled : 1
[...][preCiceAdapter][DEBUG] Configuring the CHT module...
[...][preCiceAdapter][DEBUG] user-defined solver type : name
[...][preCiceAdapter][DEBUG] temperature field name : T
[...][preCiceAdapter][DEBUG] transportProperties name : transportProperties
[...][preCiceAdapter][DEBUG] conductivity name for basic solvers : k
[...][preCiceAdapter][DEBUG] density name for incompressible solvers : rho
[...][preCiceAdapter][DEBUG] heat capacity name for incompressible solvers : Cp
[...][preCiceAdapter][DEBUG] Prandtl number name for incompressible solvers : Pr
[...][preCiceAdapter][DEBUG] Turbulent thermal diffusivity field name for incompressible solvers : alphaT
[...][preCiceAdapter][DEBUG] Determining the solver type...
[...][preCiceAdapter][DEBUG] Found the transportProperties dictionary.
[...][preCiceAdapter][DEBUG] Did not find the turbulenceProperties dictionary.
[...][preCiceAdapter][DEBUG] Did not find the thermophysicalProperties dictionary.
[...][preCiceAdapter][DEBUG] This is a basic solver, as transport properties are provided, while turbulence or transport properties are not provided.
[...][preCiceAdapter][DEBUG] Checking the timestep type (fixed vs adjustable)...
[...][preCiceAdapter][DEBUG] Timestep type: fixed.
[...][preCiceAdapter][DEBUG] Creating the preCICE solver interface...
[...][preCiceAdapter][DEBUG] Number of processes: 1
[...][preCiceAdapter][DEBUG] MPI rank: 0
[...][preCiceAdapter][DEBUG] preCICE solver interface was created.
[...][preCiceAdapter][DEBUG] Configuring preCICE...
(0) 16:10:32 [impl:SolverInterfaceImpl]:119 in configure: Configuring preCICE with configuration: "preCice-config.xml"
(0) 16:10:32 [impl:SolverInterfaceImpl]:153 in configure: Run in coupling mode
[...][preCiceAdapter][DEBUG] preCICE was configured.
[...][preCiceAdapter][DEBUG] Creating interfaces.
[...][preCiceAdapter][DEBUG] Interface created on meshSolid-Mesh
[...][preCiceAdapter][DEBUG] Adding coupling data writers...
[...][preCiceAdapter][DEBUG] Constructed KappaEff Basic.
[...][preCiceAdapter][DEBUG] Name of transportProperties: transportProperties
[...][preCiceAdapter][DEBUG] Name of conductivity: k
[...][preCiceAdapter][DEBUG] k = 100.000000
[...][preCiceAdapter][DEBUG] Added writer: Heat Flux for basic solvers.
[...][preCiceAdapter][DEBUG] Adding coupling data readers...
[...][preCiceAdapter][DEBUG] Added reader: Temperature.
[...][preCiceAdapter][DEBUG] Initializing the preCICE solver interface...
(0) 16:10:32 [impl:SolverInterfaceImpl]:216 in initialize: Setting up master communication to coupling partner/s

/bin/bash B3X50
Create mesh for time = 0

PIMPLE: Operating solver in PISO mode

Reading thermophysical properties

Selecting thermodynamics package
{
    type            heRhoThermo;
    mixture          pureMixture;
    transport        const;
    thermo            hConst;
    equationOfState   perfectGas;
    specie            specie;
    energy            sensibleEnthalpy;
}

Reading field U

Reading/calculating face flux field phi

Creating turbulence model

Selecting turbulence model type laminar
Selecting laminar stress model Stokes

Reading g

Reading hRef
Calculating field g.h

Reading field p_rgh

Creating field dpdt

Creating field kinetic energy K

No MRF models present

Radiation model not active: radiationProperties not found
Selecting radiationModel none
No finite volume options present

Courant Number mean: 0.0837143 max: 0.403668

Starting time loop
```

### 3. How can I couple my own code?



## API 1: Steering

```
1  turnOnSolver(); //e.g. setup and partition mesh
2
3
4
5  double dt; // solver timestep size
6
7
8
9
10 while (not simulationDone()){
11     dt = beginTimeStep(); // e.g. compute adaptive dt
12
13     computeTimeStep(dt);
14
15     endTimeStep(); // e.g. update variables, increment time
16 }
17
18 turnOffSolver();
```

## API 1: Steering

```
1  turnOnSolver(); //e.g. setup and partition mesh
2  precice::SolverInterface precice("FluidSolver",rank,size);
3
4
5  double dt; // solver timestep size
6
7
8
9
10 while (not simulationDone()){
11     dt = beginTimeStep(); // e.g. compute adaptive dt
12
13     computeTimeStep(dt);
14
15     endTimeStep(); // e.g. update variables, increment time
16 }
17
18 turnOffSolver();
```

## API 1: Steering

```
1  turnOnSolver(); //e.g. setup and partition mesh
2  precice::SolverInterface precice("FluidSolver",rank,size);
3  precice.configure("precice-config.xml");
4
5  double dt; // solver timestep size
6
7
8
9
10 while (not simulationDone()){
11     dt = beginTimeStep(); // e.g. compute adaptive dt
12
13     computeTimeStep(dt);
14
15     endTimeStep(); // e.g. update variables, increment time
16 }
17
18 turnOffSolver();
```

## API 1: Steering

```
1  turnOnSolver(); //e.g. setup and partition mesh
2  precice::SolverInterface precice("FluidSolver",rank,size);
3  precice.configure("precice-config.xml");
4
5  double dt; // solver timestep size
6  double maxDt; // maximum precice timestep size
7
8
9
10 while (not simulationDone()){
11     dt = beginTimeStep(); // e.g. compute adaptive dt
12
13     computeTimeStep(dt);
14
15     endTimeStep(); // e.g. update variables, increment time
16 }
17
18 turnOffSolver();
```

## API 1: Steering

```
1  turnOnSolver(); //e.g. setup and partition mesh
2  precice::SolverInterface precice("FluidSolver",rank,size);
3  precice.configure("precice-config.xml");
4
5  double dt; // solver timestep size
6  double maxDt; // maximum precice timestep size
7
8  maxDt = precice.initialize()
9
10 while (not simulationDone()){
11     dt = beginTimeStep(); // e.g. compute adaptive dt
12
13     computeTimeStep(dt);
14
15     endTimeStep(); // e.g. update variables, increment time
16 }
17
18 turnOffSolver();
```

## API 1: Steering

```
1  turnOnSolver(); //e.g. setup and partition mesh
2  precice::SolverInterface precice("FluidSolver",rank,size);
3  precice.configure("precice-config.xml");
4
5  double dt; // solver timestep size
6  double maxDt; // maximum precice timestep size
7
8  maxDt = precice.initialize()
9
10 while (not simulationDone() && precice.isCouplingOngoing()){
11     dt = beginTimeStep(); // e.g. compute adaptive dt
12
13     computeTimeStep(dt);
14
15     endTimeStep(); // e.g. update variables, increment time
16 }
17
18 turnOffSolver();
```

## API 1: Steering

```
1  turnOnSolver(); //e.g. setup and partition mesh
2  precice::SolverInterface precice("FluidSolver",rank,size);
3  precice.configure("precice-config.xml");
4
5  double dt; // solver timestep size
6  double maxDt; // maximum precice timestep size
7
8  maxDt = precice.initialize()
9
10 while (not simulationDone() && precice.isCouplingOngoing()){
11     dt = beginTimeStep(); // e.g. compute adaptive dt
12     dt = min(maxDt, dt);
13     computeTimeStep(dt);
14
15     endTimeStep(); // e.g. update variables, increment time
16 }
17
18 turnOffSolver();
```

## API 1: Steering

```
1  turnOnSolver(); //e.g. setup and partition mesh
2  precice::SolverInterface precice("FluidSolver",rank,size);
3  precice.configure("precice-config.xml");
4
5  double dt; // solver timestep size
6  double maxDt; // maximum precice timestep size
7
8  maxDt = precice.initialize()
9
10 while (not simulationDone() && precice.isCouplingOngoing()){
11     dt = beginTimeStep(); // e.g. compute adaptive dt
12     dt = min(maxDt, dt);
13     computeTimeStep(dt);
14     maxDt = precice.advance(dt); // communication, data mapping, ...
15     endTimeStep(); // e.g. update variables, increment time
16 }
17
18 turnOffSolver();
```



## API 1: Steering

```
1  turnOnSolver(); //e.g. setup and partition mesh
2  precice::SolverInterface precice("FluidSolver",rank,size);
3  precice.configure("precice-config.xml");
4
5  double dt; // solver timestep size
6  double maxDt; // maximum precice timestep size
7
8  maxDt = precice.initialize()
9
10 while (not simulationDone() && precice.isCouplingOngoing()){
11     dt = beginTimeStep(); // e.g. compute adaptive dt
12     dt = min(maxDt, dt);
13     computeTimeStep(dt);
14     maxDt = precice.advance(dt); // communication, data mapping, ...
15     endTimeStep(); // e.g. update variables, increment time
16 }
17 precice.finalize();
18 turnOffSolver();
```

## API 2: Mesh and Data Access

```
1  precice::SolverInterface precice("FluidSolver",rank,size);
2  precice.configure("precice-config.xml");
3
4
5
6
7
8
9
10
11
12  precice.initialize()
13
14
15
16
17
18  [...]
```

## API 2: Mesh and Data Access

```
1  precice::SolverInterface precice("FluidSolver",rank,size);
2  precice.configure("precice-config.xml");
3
4  int meshID = precice.getMeshID("FluidMesh");
5
6
7
8
9
10
11
12  precice.initialize()
13
14
15
16
17
18  [...]
```

## API 2: Mesh and Data Access

```
1  precice::SolverInterface precice("FluidSolver",rank,size);
2  precice.configure("precice-config.xml");
3
4  int meshID = precice.getMeshID("FluidMesh");
5  int vertexSize; // number of vertices at coupling interface
6
7
8
9
10
11
12  precice.initialize()
13
14
15
16
17
18  [...]
```

## API 2: Mesh and Data Access

```
1  precice::SolverInterface precice("FluidSolver",rank,size);
2  precice.configure("precice-config.xml");
3
4  int meshID = precice.getMeshID("FluidMesh");
5  int vertexSize; // number of vertices at coupling interface
6  // determine vertexSize
7
8
9
10
11
12  precice.initialize()
13
14
15
16
17
18  [...]
```

## API 2: Mesh and Data Access

```
1  precice::SolverInterface precice("FluidSolver",rank,size);
2  precice.configure("precice-config.xml");
3
4  int meshID = precice.getMeshID("FluidMesh");
5  int vertexSize; // number of vertices at coupling interface
6  // determine vertexSize
7  double* coords = new double[vertexSize*dim]; // coupling mesh (nodes)
8
9
10
11
12  precice.initialize()
13
14
15
16
17
18  [...]
```

## API 2: Mesh and Data Access

```
1  precice::SolverInterface precice("FluidSolver",rank,size);
2  precice.configure("precice-config.xml");
3
4  int meshID = precice.getMeshID("FluidMesh");
5  int vertexSize; // number of vertices at coupling interface
6  // determine vertexSize
7  double* coords = new double[vertexSize*dim]; // coupling mesh (nodes)
8  // determine coordinates
9
10
11
12  precice.initialize()
13
14
15
16
17
18  [...]
```

## API 2: Mesh and Data Access

```
1  precice::SolverInterface precice("FluidSolver",rank,size);
2  precice.configure("precice-config.xml");
3
4  int meshID = precice.getMeshID("FluidMesh");
5  int vertexSize; // number of vertices at coupling interface
6  // determine vertexSize
7  double* coords = new double[vertexSize*dim]; // coupling mesh (nodes)
8  // determine coordinates
9  int* vertexIDs = new int[vertexSize];
10
11
12  precice.initialize()
13
14
15
16
17
18  [...]
```



## API 2: Mesh and Data Access

```
1  precice::SolverInterface precice("FluidSolver",rank,size);
2  precice.configure("precice-config.xml");
3
4  int meshID = precice.getMeshID("FluidMesh");
5  int vertexSize; // number of vertices at coupling interface
6  // determine vertexSize
7  double* coords = new double[vertexSize*dim]; // coupling mesh (nodes)
8  // determine coordinates
9  int* vertexIDs = new int[vertexSize];
10 precice.setMeshVertices(meshID, vertexSize, coords, vertexIDs);
11
12 precice.initialize()
13
14
15
16
17
18 [...]
```

## API 2: Mesh and Data Access

```
1  precice::SolverInterface precice("FluidSolver",rank,size);
2  precice.configure("precice-config.xml");
3
4  int meshID = precice.getMeshID("FluidMesh");
5  int vertexSize; // number of vertices at coupling interface
6  // determine vertexSize
7  double* coords = new double[vertexSize*dim]; // coupling mesh (nodes)
8  // determine coordinates
9  int* vertexIDs = new int[vertexSize];
10 precice.setMeshVertices(meshID, vertexSize, coords, vertexIDs);
11
12 precice.initialize()
13
14 [...]
15
16 int forceID = precice.getDataID("Forces", meshID);
17
18 [...]
```

## API 2: Mesh and Data Access

```
1  precice::SolverInterface precice("FluidSolver",rank,size);
2  precice.configure("precice-config.xml");
3
4  int meshID = precice.getMeshID("FluidMesh");
5  int vertexSize; // number of vertices at coupling interface
6  // determine vertexSize
7  double* coords = new double[vertexSize*dim]; // coupling mesh (nodes)
8  // determine coordinates
9  int* vertexIDs = new int[vertexSize];
10 precice.setMeshVertices(meshID, vertexSize, coords, vertexIDs);
11
12 precice.initialize()
13
14 [...]
15
16 int forceID = precice.getDataID("Forces", meshID);
17 double* forces = new double[vertexSize*dim];
18 [...]
```

## API 2: Mesh and Data Access

```
1  precice::SolverInterface precice("FluidSolver",rank,size);
2  precice.configure("precice-config.xml");
3
4  int meshID = precice.getMeshID("FluidMesh");
5  int vertexSize; // number of vertices at coupling interface
6  // determine vertexSize
7  double* coords = new double[vertexSize*dim]; // coupling mesh (nodes)
8  // determine coordinates
9  int* vertexIDs = new int[vertexSize];
10 precice.setMeshVertices(meshID, vertexSize, coords, vertexIDs);
11
12 precice.initialize()
13
14 [...]
15
16 int forceID = precice.getDataID("Forces", meshID);
17 double* forces = new double[vertexSize*dim];
18 precice.writeBlockVectorData(forceID, vertexSize, vertexIDs, forces);
```

# preCICE Configuration

```
1 <precice-configuration>
2
3   <data:vector name="Forces" />
4   <data:vector name="Displacements" />
5
6   <mesh name="FluidMesh">
7     <use-data name="Forces" />
8     <use-data name="Displacements" />
9   </mesh>
10
11  <participant name="FluidSolver">
12    <use-mesh name="FluidMesh" provide="yes" />
13    <write-data name="Forces" mesh="FluidMesh" />
14
15    [...]
16
17  </participant>
18
19  [...]
```

## API 3: Implicit Coupling

```
1 while (not simulationDone() && precice.isCouplingOngoing()){
2
3
4
5
6
7     dt = beginTimeStep();
8     computeTimeStep(dt);
9     precice.advance(dt); // communication, data mapping, ...
10
11
12
13
14
15
16     endTimeStep(); // e.g. update variables, increment time
17
18 }
```

## API 3: Implicit Coupling

```
1 while (not simulationDone() && precice.isCouplingOngoing()){
2     if(precice.isActionRequired("WriteIterationCheckpoint")){
3         saveCheckpoint(); // save internal state of solver
4         precice.fulfilledAction("WriteIterationCheckpoint");
5     }
6
7     dt = beginTimeStep();
8     computeTimeStep(dt);
9     precice.advance(dt); // communication, data mapping, ...
10
11
12
13
14
15
16     endTimeStep(); // e.g. update variables, increment time
17
18 }
```

## API 3: Implicit Coupling

```
1 while (not simulationDone() && precice.isCouplingOngoing()){
2     if(precice.isActionRequired("WriteIterationCheckpoint")){
3         saveCheckpoint(); // save internal state of solver
4         precice.fulfilledAction("WriteIterationCheckpoint");
5     }
6
7     dt = beginTimeStep();
8     computeTimeStep(dt);
9     precice.advance(dt); // communication, data mapping, ...
10
11     if(precice.isActionRequired("ReadIterationCheckpoint")){
12         reloadCheckpoint(); // set variables back to checkpoint
13         precice.fulfilledAction("ReadIterationCheckpoint");
14     }
15     else{ // timestep converged
16         endTimeStep(); // e.g. update variables, increment time
17     }
18 }
```



# Roadmap

## Current Developments

- ▶ Debian package
- ▶ FSI module for the OpenFOAM adapter
- ▶ Fluid-fluid module for the OpenFOAM adapter
- ▶ Structure-structure coupling with CalculiX

## Long-term Goals

- ▶ 3D-1D and 3D-2D data mapping
- ▶ Parallel initialization and support of dynamically changing coupling interfaces
- ▶ Consistent time interpolation

## Funding



## More information

- ▶ Open-source project: [www.precice.org](http://www.precice.org)
- ▶ Source code: [github.com/precice](https://github.com/precice)

## Contact us

- ▶ Mailing list
- ▶ Gitter chat room