Seamless Coupling of PDE-based Simulations with the Coupling Library preCICE

SimulTech 2022

Benjamin Uekermann et al.

University of Stuttgart

Keynote Presentations from Previous Years

- 2021, **Peter Fritzson**: *The OpenModelica Environment and Its Use for Development of Sustainable Cyber-physical Systems and Digital Twins*
- 2019, **Hans Vangheluwe**: *Co-Simulation – A Reasearch Agenda*
- More buzzwords: coupling, co-simulation, interoperability, . . .

Keynote Presentations from Previous Years

- 2021, **Peter Fritzson**: *The OpenModelica Environment and Its Use for Development of Sustainable Cyber-physical Systems and Digital Twins*
- 2019, **Hans Vangheluwe**: *Co-Simulation – A Reasearch Agenda*
- More buzzwords: coupling, co-simulation, interoperability, ...

In my community:

Simulation = continuous-time (numerical) simulation of physical systems

Outline

1. Introduction: FMI vs. Coupling Libraries
2. Coupling Library preCICE

1. Introduction: FMI vs. Coupling Libraries

Functional Mock-up Interface (. . . as I see it)


```python
import pyfmi
model1 = load_fmu('Model1.fmu')
model2 = load_fmu('Model2.fmu')

# my own simple master algorithm
while (not done())
    model1.set('u', u1)
    model1.do_step()
    y1 = model1.get('y')
    u2 = y1
    model2.set('u', u2)
    model2.do_step()
    y2 = model2.get('y')
    u1 = accelerate(y2)
```

Functional Mock-up Interface (. . . as I see it)


```python
import pyfmi
model1 = load_fmu('Model1.fmu')
model2 = load_fmu('Model2.fmu')

# my own simple master algorithm
while (not done())
    model1.set('u', u1)
    model1.do_step()
    y1 = model1.get('y')
    u2 = y1
    model2.set('u', u2)
    model2.do_step()
    y2 = model2.get('y')
    u1 = accelerate(y2)
```

- FMUs regarded as black boxes, Goals: flexibility + protect IP

Functional Mock-up Interface (. . . as I see it)


```python
import pyfmi
model1 = load_fmu('Model1.fmu')
model2 = load_fmu('Model2.fmu')

# my own simple master algorithm
while (not done())
    model1.set('u', u1)
    model1.do_step()
    y1 = model1.get('y')
    u2 = y1
    model2.set('u', u2)
    model2.do_step()
    y2 = model2.get('y')
    u1 = accelerate(y2)
```

- FMUs regarded as black boxes,
  Goals: flexibility + protect IP
- Framework approach: coupling calls models

Functional Mock-up Interface (...as I see it)


```
import pyfmi
model1 = load_fmu('Model1.fmu')
model2 = load_fmu('Model2.fmu')

# my own simple master algorithm
while (not done())
    model1.set('u', u1)
    model1.do_step()
    y1 = model1.get('y')
    u2 = y1
    model2.set('u', u2)
    model2.do_step()
    y2 = model2.get('y')
    u1 = accelerate(y2)
```

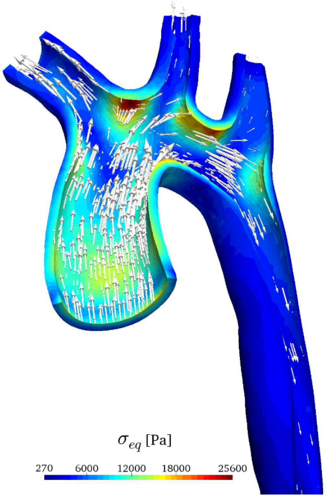
- FMUs regarded as black boxes, Goals: flexibility + protect IP
- Framework approach: coupling calls models
- Units are (for example) ODEs, typically relatively simple components

Functional Mock-up Interface (. . . as I see it)


```python
import pyfmi
model1 = load_fmu('Model1.fmu')
model2 = load_fmu('Model2.fmu')

# my own simple master algorithm
while (not done())
    model1.set('u', u1)
    model1.do_step()
    y1 = model1.get('y')
    u2 = y1
    model2.set('u', u2)
    model2.do_step()
    y2 = model2.get('y')
    u1 = accelerate(y2)
```

- FMUs regarded as black boxes, Goals: flexibility + protect IP
- Framework approach: coupling calls models
- Units are (for example) ODEs, typically relatively simple components
- Typically, communicated data not too large $\rightsquigarrow$ HPC not sooo important

PDE Simulation Units

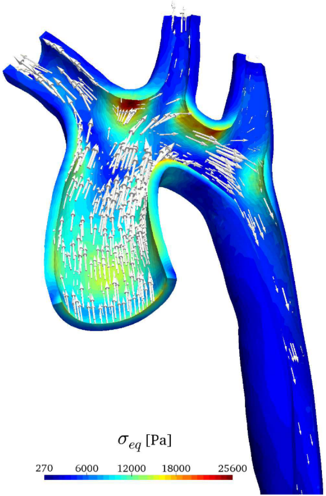


Blood flow through human aorta.
Picture: Totounferoush et al. 2021

Fluid Model $\longleftrightarrow$ Solid Model

- Space resolution and more accuracy needed $\rightsquigarrow$ PDEs

PDE Simulation Units

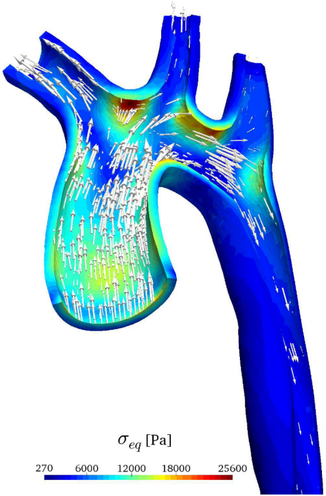


Blood flow through human aorta.
Picture: Totounferoush et al. 2021

Fluid Model $\longleftrightarrow$ Solid Model

- Space resolution and more accuracy needed $\rightsquigarrow$ PDEs
- Coupling data space-dependent $\rightsquigarrow$ data mapping required

PDE Simulation Units

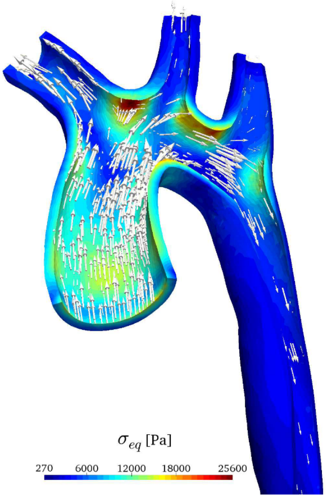


Blood flow through human aorta.
Picture: Totounferoush et al. 2021

Fluid Model $\longleftrightarrow$ Solid Model

- Space resolution and more accuracy needed $\rightsquigarrow$ PDEs
- Coupling data space-dependent $\rightsquigarrow$ data mapping required
- Models (aka. solvers): sophisticated software

PDE Simulation Units

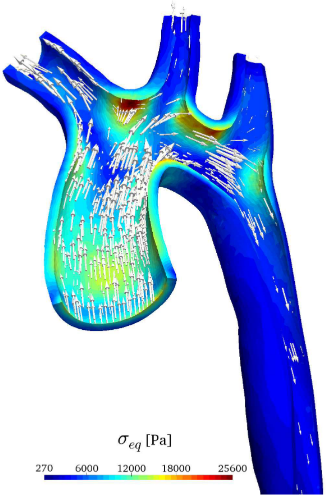


Blood flow through human aorta.
Picture: Totounferoush et al. 2021

Fluid Model ⟷ Solid Model

- Space resolution and more accuracy needed $\rightsquigarrow$ PDEs
- Coupling data space-dependent $\rightsquigarrow$ data mapping required
- Models (aka. solvers): sophisticated software
- Regard models as black boxes,
  Goals: flexibility + to be minimally invasive

PDE Simulation Units

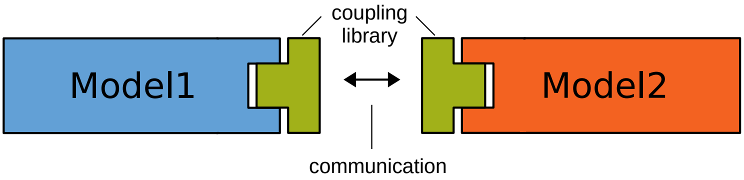


Blood flow through human aorta.
Picture: Totounferoush et al. 2021

Fluid Model $\longleftrightarrow$ Solid Model

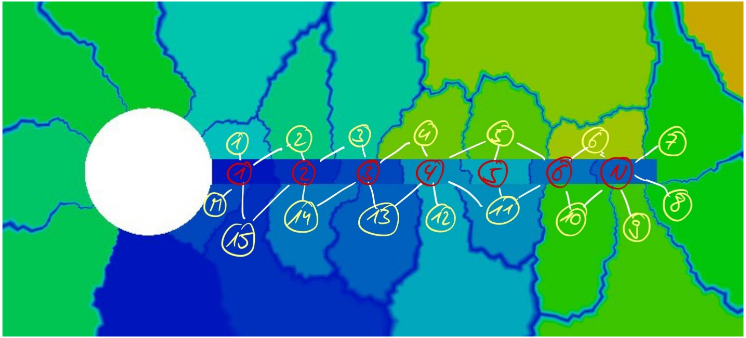
- Space resolution and more accuracy needed $\rightsquigarrow$ PDEs
- Coupling data space-dependent $\rightsquigarrow$ data mapping required
- Models (aka. solvers): sophisticated software
- Regard models as black boxes,
  Goals: flexibility + to be minimally invasive
- Costly simulations, large amounts of distributed coupling data $\rightsquigarrow$ HPC a must

Seamless Coupling of PDE Units: Library Approach



- Library approach: the models call the coupling
- Minimally invasive
- Start models separately
- Coupling logic more complicated

Seamless Coupling of PDE Units: Peer-to-Peer Approach



- No central server-like entity
- Peer-to-peer communication between ranks whose coupling interfaces overlap
- $\Rightarrow$ No scaling issues
- $\Rightarrow$ Coupling numerics (data mapping, acceleration, ...) need to run on distributed data

Software Solutions

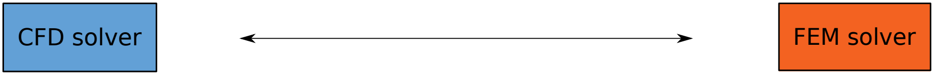
- **OpenPALM** / **CWIPI** (Coupling With Interpolation Parallel Interface) from CERFACS + ONERA (F)
- **MUI** (Multiscale Universal Interface) from UKRI-STFC (UK) + Brown + Berkeley + IBM (US)
- **DTK** (Data Transfer Kit) from Sandia + ORNL (US)
- **preCICE** from U Stuttgart + TUM (D)
- (MpCCI from Fraunhofer SCAI (D), commercial, limited HPC capabilities)
- ... many more

Selling Points of preCICE

- Free software (LGPL3)
- Extensive user documentation (when converted to PDF, more than 250 pages)
- Vivid and quickly growing community
- Industry-ready support program
- Ready-to-use adapters for OpenFOAM, FEniCS, deal.II, CalculiX, Nutils, SU2, ...
- API in C++, C, Fortran, Python, Matlab, Julia
- Scalability up to complete supercomputers
- xSDK member, pre-installed on more and more supercomputers
- Robust quasi-Newton coupling (and soon waveform iteration)
- Coupling of arbitrary many components (*arbitrary many = more than two*)
- CI with tests on all levels (unit, integration, bindings, adapters, system)

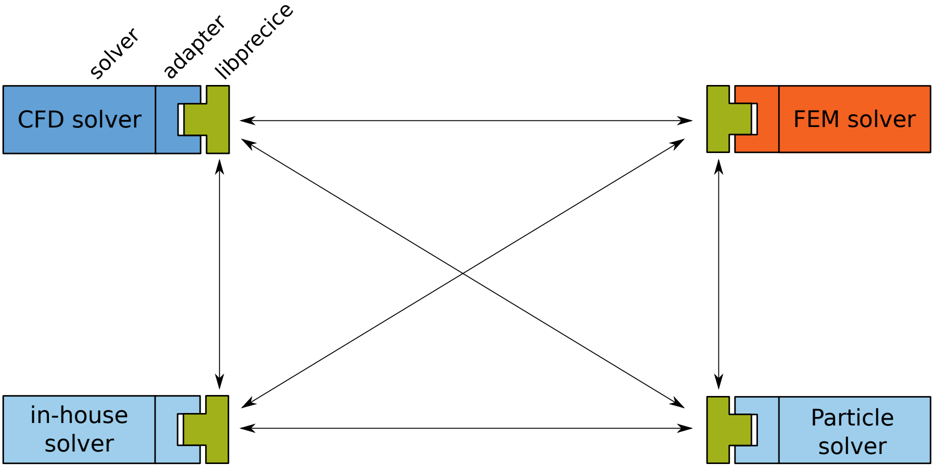
2. Coupling Library preCICE

Overview



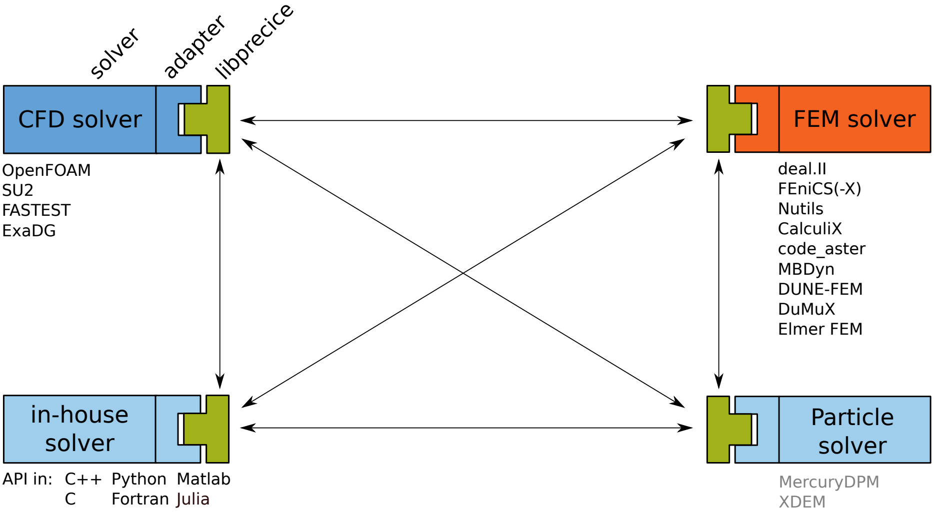
Overview

Overview

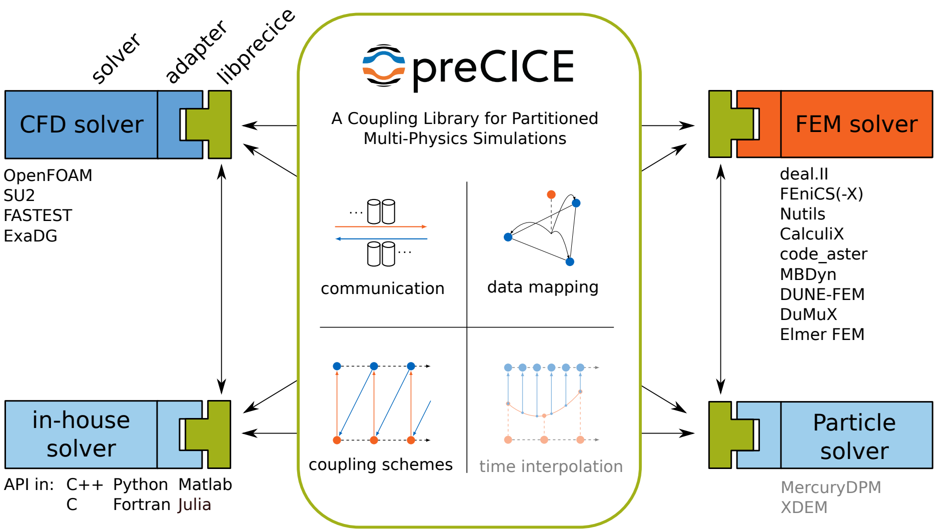


solver
adapter
libprecice
CFD solver
FEM solver
in-house solver
Particle solver

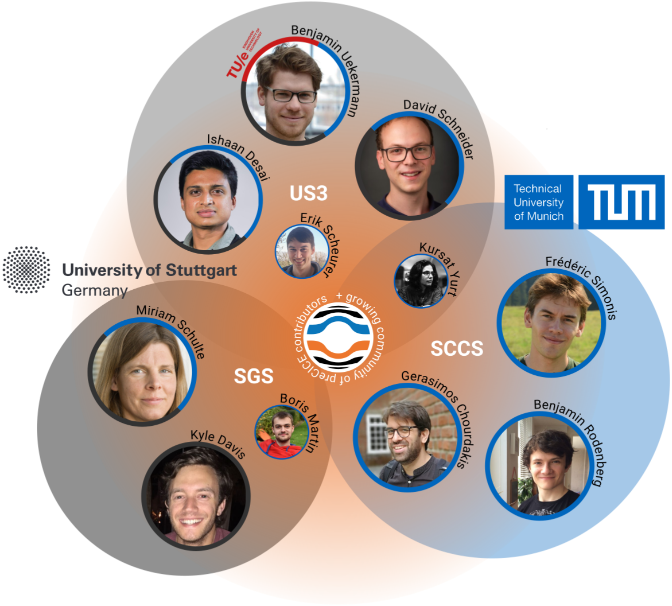
Overview


solver
adapter
libprecice
CFD solver
FEM solver
in-house solver
Particle solver
OpenFOAM
SU2
FASTEST
ExaDG
deal.II
FEniCS(-X)
Nutils
CalculiX
code_aster
MBDyn
DUNE-FEM
DuMuX
Elmer FEM
API in: C++ Python Matlab
C Fortran Julia
MercuryDPM
XDEM

Overview


solver
adapter
libprecice
CFD solver
OpenFOAM
SU2
FASTEST
ExaDG
preCICE
A Coupling Library for Partitioned Multi-Physics Simulations
communication
data mapping
coupling schemes
time interpolation
FEM solver
deal.II
FEniCS(-X)
Nutils
CalculiX
code_aster
MBDyn
DUNE-FEM
DuMuX
Elmer FEM
in-house solver
API in: C++ Python Matlab C Fortran Julia
Particle solver
MercuryDPM
XDEM

Maintainers



Application Programming Interface

```python
import precice
interface = precice.Interface('FluidSolver', 'precice-config.xml', rank, size)
```

Some details omitted. Follow preCICE Course to learn the API step-by-step.

Application Programming Interface

```
import precice
interface = precice.Interface('FluidSolver', 'precice-config.xml', rank, size)

positions = ... #define interface mesh, 2D array with shape (n, dim)
vertex_ids = interface.set_mesh_vertices('Fluid-Mesh', positions)
```

Some details omitted. Follow preCICE Course to learn the API step-by-step.

Application Programming Interface

```python
import precice
interface = precice.Interface('FluidSolver', 'precice-config.xml', rank, size)

positions = ... #define interface mesh, 2D array with shape (n, dim)
vertex_ids = interface.set_mesh_vertices('Fluid-Mesh', positions)

interface.initialize()
```

Some details omitted. Follow preCICE Course to learn the API step-by-step.

Application Programming Interface

```python
import precice
interface = precice.Interface('FluidSolver', 'precice-config.xml', rank, size)

positions = ... #define interface mesh, 2D array with shape (n, dim)
vertex_ids = interface.set_mesh_vertices('Fluid-Mesh', positions)

interface.initialize()

while interface.is_coupling_ongoing(): # main time loop









```

Some details omitted. Follow preCICE Course to learn the API step-by-step.

Application Programming Interface

```python
import precice
interface = precice.Interface('FluidSolver', 'precice-config.xml', rank, size)

positions = ... #define interface mesh, 2D array with shape (n, dim)
vertex_ids = interface.set_mesh_vertices('Fluid-Mesh', positions)

interface.initialize()

while interface.is_coupling_ongoing(): # main time loop




    u = solve_time_step() # returns new solution




```

Some details omitted. Follow preCICE Course to learn the API step-by-step.

Application Programming Interface

```python
import precice
interface = precice.Interface('FluidSolver', 'precice-config.xml', rank, size)

positions = ... #define interface mesh, 2D array with shape (n, dim)
vertex_ids = interface.set_mesh_vertices('Fluid-Mesh', positions)

interface.initialize()

while interface.is_coupling_ongoing(): # main time loop



    displacements = interface.read_block_vector_data('Displacement', vertex_ids)
    u = solve_time_step(displacements) # returns new solution
    forces = compute_forces(u) # returns 2D array with shape (n, dim)
    interface.write_block_vector_data('Force', vertex_ids, forces)


```

Some details omitted. Follow preCICE Course to learn the API step-by-step.

Application Programming Interface

```python
import precice
interface = precice.Interface('FluidSolver', 'precice-config.xml', rank, size)

positions = ... #define interface mesh, 2D array with shape (n, dim)
vertex_ids = interface.set_mesh_vertices('Fluid-Mesh', positions)

interface.initialize()

while interface.is_coupling_ongoing(): # main time loop



    displacements = interface.read_block_vector_data('Displacement', vertex_ids)
    u = solve_time_step(displacements) # returns new solution
    forces = compute_forces(u) # returns 2D array with shape (n, dim)
    interface.write_block_vector_data('Force', vertex_ids, forces)

    interface.advance()
```

Some details omitted. Follow preCICE Course to learn the API step-by-step.

Application Programming Interface

```python
import precice
interface = precice.Interface('FluidSolver', 'precice-config.xml', rank, size)

positions = ... #define interface mesh, 2D array with shape (n, dim)
vertex_ids = interface.set_mesh_vertices('Fluid-Mesh', positions)

precice_dt = interface.initialize()

while interface.is_coupling_ongoing(): # main time loop
    dt = compute_adaptive_dt()
    dt = min(precice_dt, dt)

    displacements = interface.read_block_vector_data('Displacement', vertex_ids)
    u = solve_time_step(dt, displacements) # returns new solution
    forces = compute_forces(u) # returns 2D array with shape (n, dim)
    interface.write_block_vector_data('Force', vertex_ids, forces)

    precice_dt = interface.advance(dt)
```

Some details omitted. Follow preCICE Course to learn the API step-by-step.

Demo: Conjugate Heat Transfer with OpenFOAM and Nutils

Conjugate Heat Transfer

- Coupled heat transfer between a fluid (convection and conduction) and a solid (only conduction)

OpenFOAM[1], ESI/OpenCFD

- Open-source C++ finite-volume framework
- Widely used for CFD (industry and academia)
- We use `buoyantPimpleFoam`, transient compressible NSE

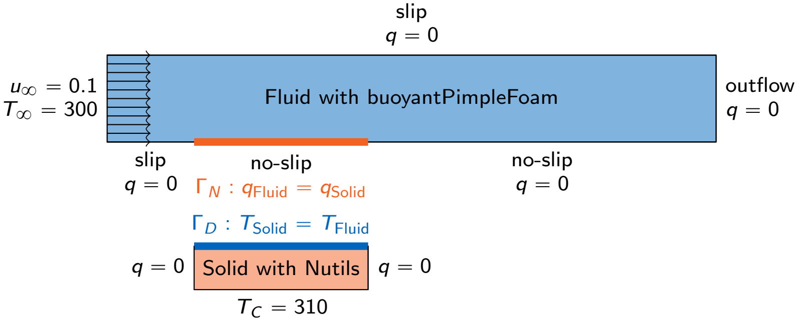
Nutils

- Open-source Python programming library for Finite Element Method computations
- Developed by Evalf Computing, spin-off of TU Eindhoven
- We use it here for the simple heat equation (conduction in the solid)

[1] OPENFOAM is a registered trade mark of OpenCFD Limited, producer and distributor of the OpenFOAM software via www.openfoam.com.

Demo: Conjugate Heat Transfer with OpenFOAM and Nutils

Boundary conditions and geometry



Additional parameters

- $\text{Pr} = 0.01$
- $\text{Re} = \rho u_\infty d/\mu = 500$ (use characteristic length d = plate length)
- $k_s = 100 \cdot k_f$ (thermal conductivities)

Demo: Conjugate Heat Transfer with OpenFOAM and Nutils

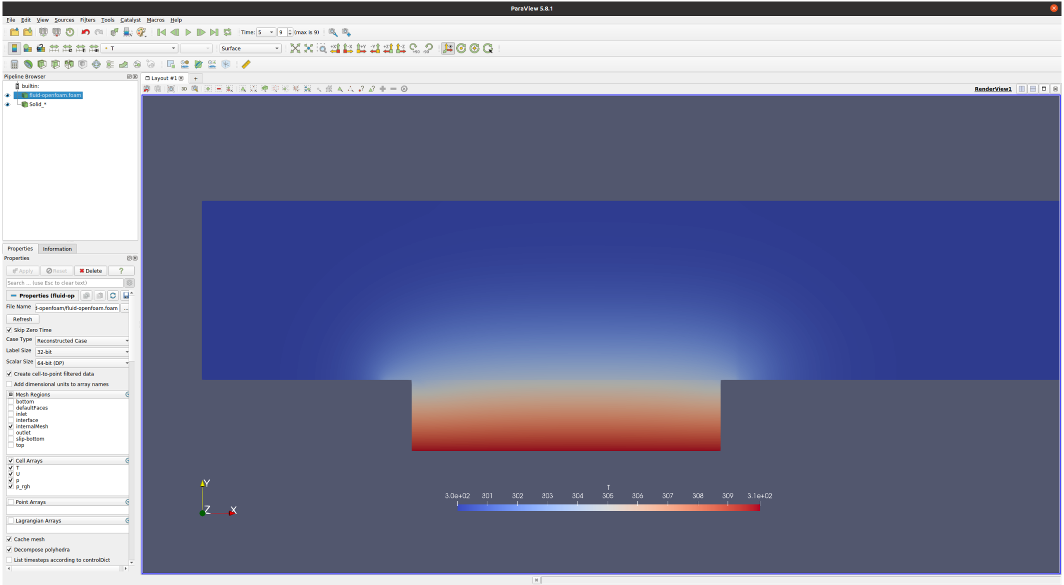
uekermbn@lapus301: ~/Repos/tutorials/flow-over-heated-plate/fluid-openfoam

```
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

---[preciceAdapter] Loaded the OpenFOAM-preCICE adapter - v1.1.0 + unreleased changes.
---[preciceAdapter] Reading preciceDict...
---[precice]  This is preCICE version 2.4.0
---[precice]  Revision info: v2.4.0-61-gc9095ee9
---[precice]  Build type: Debug
---[precice]  Configuring preCICE with configuration "../precice-config.xml"
---[precice]  I am participant "Fluid"
---[precice]  Setting up primary communication to coupling partner/s
---[precice]  Primary ranks are connected
---[precice]  Setting up preliminary secondary communication to coupling partner/s
---[precice]  Prepare partition for mesh Fluid-Mesh
---[precice]  Gather mesh Fluid-Mesh
---[precice]  Send global mesh Fluid-Mesh
---[precice]  Receive global mesh Solid-Mesh
---[precice]  Setting up secondary communication to coupling partner/s
---[precice]  Secondary ranks are connected
---[precice]  iteration: 1 of 30, time-window: 1, time: 0 of 1, time-window-size: 0.01, max-times
tep-length: 0.01, ongoing: yes, time-window-complete: no, write-iteration-checkpoint
---[precice]  initializeData is skipped since no data has to be initialized.
---[preciceAdapter] preCICE was configured and initialized
---[preciceAdapter] Setting the solver's endTime to infinity to prevent early exits. Only preCICE
 will control the simulation's endTime. Any functionObject's end() method will be triggered by th
e adapter. You may disable this behavior in the adapter's configuration.
Courant Number mean: 0.0837143 max: 0.403668
Time = 0.01

diagonal:  Solving for rho, Initial residual = 0, Final residual = 0, No Iterations 0
PIMPLE: iteration 1
DILUPBiCGStab:  Solving for Ux, Initial residual = 1, Final residual = 1.17224e-07, No Iterations
 2
DILUPBiCGStab:  Solving for Uy, Initial residual = 1, Final residual = 9.27306e-07, No Iterations
 2
DILUPBiCGStab:  Solving for h, Initial residual = 1, Final residual = 8.8372e-07, No Iterations 1
5
DICPCG:  Solving for p_rgh, Initial residual = 0.826826, Final residual = 0.00778075, No Iteratio
ns 87
diagonal:  Solving for rho, Initial residual = 0, Final residual = 0, No Iterations 0
time step continuity errors : sum local = 6.94593e-05, global = -1.44839e-07, cumulative = -1.448
```

uekermbn@lapus301: ~/Repos/tutorials/flow-over-heated-plate/solid-nutils

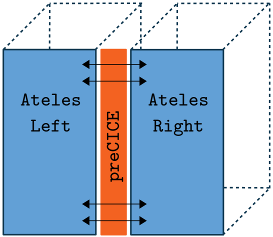
```
uekermbn@lapus301 ..ls/flow-over-heated-plate/solid-nutils (git)-[develop] % ./run.sh
opened log at file:///home/uekermbn/public_html/solid.py/log-35.html
nutils v6.3 "garak-guksu"
start Mon Jul 11 18:57:25 2022
Running nutils
optimize > solve > solving 21 dof system to machine precision using direct solver
optimize > solve > solver returned with residual 0e+00
optimize > constrained 21/126 dofs
optimize > optimum value -1.46e-11
---[precice]  This is preCICE version 2.4.0
---[precice]  Revision info: v2.4.0-61-gc9095ee9
---[precice]  Build type: Debug
---[precice]  Configuring preCICE with configuration "../precice-config.xml"
---[precice]  I am participant "Solid"
---[precice]  Setting up primary communication to coupling partner/s
---[precice]  Primary ranks are connected
---[precice]  Setting up preliminary secondary communication to coupling partner/s
---[precice]  Receive global mesh Fluid-Mesh
---[precice]  Prepare partition for mesh Solid-Mesh
---[precice]  Gather mesh Solid-Mesh
---[precice]  Send global mesh Solid-Mesh
---[precice]  Setting up secondary communication to coupling partner/s
---[precice]  Secondary ranks are connected
---[precice]  Compute "read" mapping from mesh "Fluid-Mesh" to mesh "Solid-Mesh".
---[precice]  Mapping distance min:4.19134e-05 max:0.00539908 avg: 0.00173304 var: 1.55644e-06 cn
t: 40
---[precice]  iteration: 1 of 30, time-window: 1, time: 0 of 1, time-window-size: 0.01, max-times
tep-length: 0.01, ongoing: yes, time-window-complete: no, write-iteration-checkpoint
optimize > solve > solving 126 dof system to machine precision using direct solver
optimize > solve > solver returned with residual 7e-13
optimize > optimum value 0.00e+00
Solid_0.vtk
optimize > solve > solving 21 dof system to machine precision using direct solver
optimize > solve > solver returned with residual 2e-13
optimize > constrained 42/126 dofs
optimize > optimum value 2.33e-10
solve > solving 84 dof system to machine precision using direct solver
solve > solver returned with residual 7e-11
solve > solving 21 dof system to machine precision using direct solver
solve > solver returned with residual 1e-13
---[precice]  relative convergence measure: relative two-norm diff of data "Temperature" = 1.00e+
00, limit = 1.00e-05, normalization = 3.81e+03, conv = false
---[precice]  iteration: 2 of 30, time-window: 1, time: 0 of 1, time-window-size: 0.01, max-times
tep-length: 0.01, ongoing: yes, time-window-complete: no, read-iteration-checkpoint
optimize > solve > solving 21 dof system to machine precision using direct solver
optimize > solve > solver returned with residual 2e-13
optimize > constrained 42/126 dofs
optimize > optimum value 3.57e-05
solve > solving 84 dof system to machine precision using direct solver
solve > solver returned with residual 6e-11
solve > solving 21 dof system to machine precision using direct solver
```

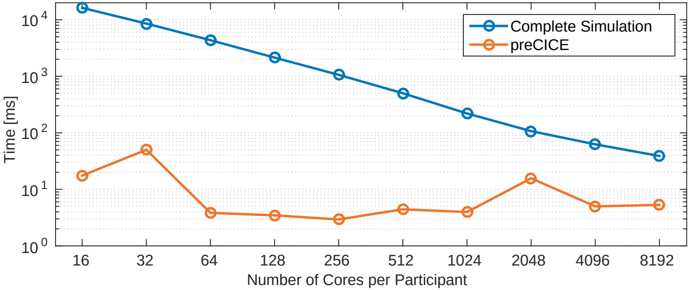
Demo: Conjugate Heat Transfer with OpenFOAM and Nutils



Scalability: Time per Timestep

- Travelling density pulse (Euler equations) through artificial coupling interface
- DG solver Ateles (U Siegen), $5.7 \cdot 10^7$ dofs
- Nearest neighbor mapping and communication
- *Uekermann. FSI on Massively Parallel Systems (2016)*
- SuperMUC Thin Nodes





Users

Oden Institute
USA

CIRA
Italy

Fluid Mechanics
Germany

Applied Mechanics
Germany

DHCAE
Germany

EuroCFD
France

Global Research for Safety
Germany

MPI-IPP
Germany

Scientific Computing
Germany

University of the Free State
South Africa

A*STAR
Singapore

NRG
Netherlands

CFD & FSI
United Kingdom

Bitron
Italy

AIT
Austria

Fluid Mechanics
Luxembourg

Helicopter Technology
Germany

FNB
Germany

IWS
Germany

Wind Energy
Netherlands

University of Split
Croatia

IIT
India

Nuclear Engineering
Sweden

IAG
Germany

FAST
Germany

Noise & Vibration
Belgium

Aerodynamics
United Kingdom

MTU Aero Engines
Germany

FERMI Fusion Project
USA

Corvid Technologies
USA

Space Transportation System Laboratory
Japan

Aeroacoustics and Flow Physics
USA

STS
Germany

Aerodynamics
Netherlands

Heat and Mass Transfer TC
Spain

IFL
Germany

ATA Engineering
USA

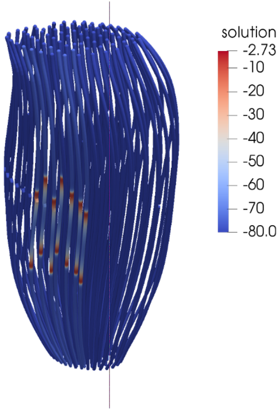
Korea Atomic Energy Research Institute
South Korea

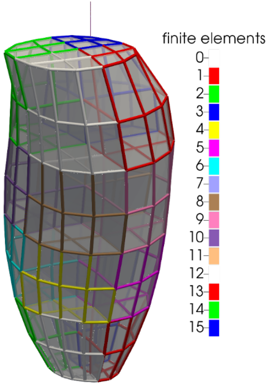
Energy Technology
Netherlands

Computational Fluid Dynamics
United Kingdom

What Our Users Do

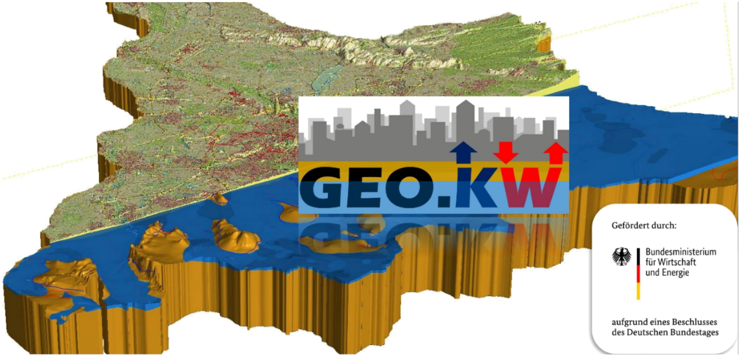
- Coupling mechanics and electrophysiology in skeletal muscles
- Benjamin Maier, IPVS, U Stuttgart
- More information on `www.precice.org/testimonials`





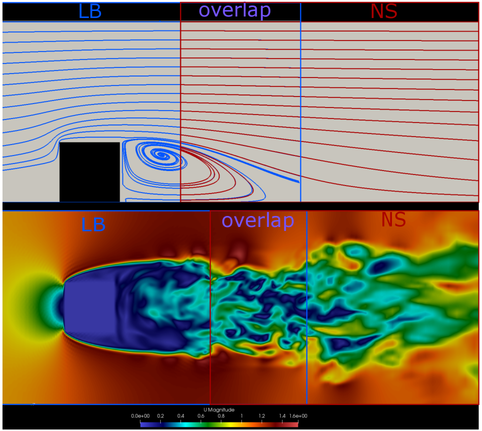
What Our Users Do

- Optimization of thermal groundwater heat pump usage in Munich
- GEO.KW project
- More information on `www.precice.org/testimonials`



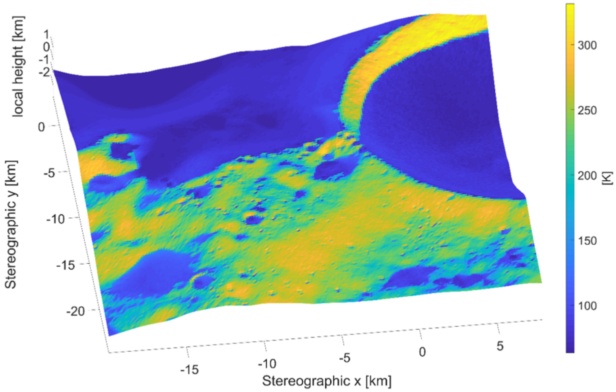
What Our Users Do

- Hybrid simulation methods for wind modelling in urban areas
- Aerodynamics research group, MACE, The University of Manchester
- More information on `www.precice.org/testimonials`

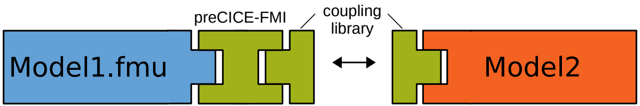


What Our Users Do

- Heat simulation on the moon
- Chair of Astronautics, Technical University of Munich
- More information on `www.precice.org/testimonials`



Couple FMUs via preCICE?




```python
# preCICE-FMI script
import precice, pyfmi
interface = precice.Interface('Model1', 'precice-config.xml', rank, size)
model = load_fmu('Model1.fmu')

vertex_ids = interface.set_mesh_vertices('Model1_Mesh', positions)
interface.initialize()

while interface.is_coupling_ongoing():

    u = interface.read_data('u', vertex_ids)
    model.set('u', u)
    model.do_step()
    y = model.get('y')
    interface.write_data('y', vertex_ids, y)

    interface.advance(dt)
```

Summary

- Sometimes, you need PDE-based models $\rightsquigarrow$ sophisticated, legacy, stand-alone software packages
- Easier (minimally invasive) if we turn things around: the models call the coupling
- For performance reasons: peer-to-peer coupling
- Coupling libraries can help, e.g. preCICE
- Why preCICE? Community, many adapters, documentation, HPC, ...
- You could still couple FMUs with preCICE

Funding

SimTech

SPPEXA

DFG

EuroTech Universities

Bundesministerium
für Wirtschaft
und Energie

H2020 grant 754462

KONWIHR

TUM

TU/e

Universität
Stuttgart

Resources

- benjamin.uekermann@ipvs.uni-stuttgart.de
- precice.org
- github.com/precice/
- precice.discourse.group
- @preCICE_org
- Chourdakis G, Davis K, Rodenberg B et al.
  preCICE v2: A sustainable and user-friendly coupling library
  [version 1; peer review: 2 approved]
  Open Res Europe 2022, **2**:51 doi:10.12688/openreseurope.14445.1
- Cluster of Excellence EXC 2075 – 390740016 – *Data-Integrated Simulation Science*
  University of Stuttgart