Using FEniCS and OpenFOAM for the simulation of conjugate heat transfer in a partitioned fashion

Benjamin Rüth¹, Peter Meisrimel², Philipp Birken², Gerasimos Chourdakis¹, Benjamin Uekermann³

¹Technical University of Munich
Department of Informatics
Chair of Scientific Computing

²Lund University
Mathematics (Faculty of Sciences)
Numerical Analysis

³Eindhoven University of Technology
Department of Mechanical Engineering
Energy Technology

GAMM Annual 2019
Vienna, Austria
February 19, 2019
Partitioned Approach

Coupled Problem

shell and tube heat exchanger using OpenFOAM and CalculiX\textsuperscript{1}

\textsuperscript{1}Figure from Rusch, A., Uekermann, B. Comparing OpenFOAM's Intrinsic Conjugate Heat Transfer Solver with preCICE-Coupled Simulations. Technical Report, 2018.
Partitioned Approach

Coupled Problem

![Diagram of shell and tube heat exchanger using OpenFOAM and CalculiX](image)

Basic idea:
- reuse existing solvers
- combine single-physics to solve multi-physics
- only exchange "black-box" information

---

1 Figure from Rusch, A., Uekermann, B. Comparing OpenFOAM’s Intrinsic Conjugate Heat Transfer Solver with preCICE-Coupled Simulations. Technical Report, 2018.
preCICE
A Plug-and-Play Coupling Library

OpenFOAM
SU2
foam-extend

Benjamin Rüth (TUM) | Using FEniCS and OpenFOAM for CHT in a partitioned fashion
3
precICE
A Plug-and-Play Coupling Library

solver  adapter  libprecice

fluid solver
OpenFOAM
SU2
foam-extend

structure solver
CalciX
Code_Aster
FEniCS
deal-II
MBDyn
precICE
A Plug-and-Play Coupling Library

precICE
A Coupling Library for Partitioned Multi-Physics Simulations

solver
adapter
libprecice

fluid solver
OpenFOAM
SU2
foam-extend

in-house
solver
API in:
C++
Python
Fortran

structure
solver
CalculiX
Code_Aster
FEniCS
deal-ii
MBDyn

commercial
solver
ANSYS Fluent
COMSOL

Benjamin Rüth (TUM) | Using FEniCS and OpenFOAM for CHT in a partitioned fashion
preCICE
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
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 additional 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

Upcoming:

- IAG, University of Stuttgart, Germany
- CTTC UPC, Barcelona, Spain
- Amirkabir U. of Technology, Iran
- 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
Plug and play?

- OpenFOAM and FEniCS
- Test case: flow over plate
- Literature results: Vynnycky
The Solvers
OpenFOAM®

Software

- open-source (GPLv3)
- widely used for CFD
- ready-to-use solvers
- C++, Libraries with C++ API
- can be used for HPC
- main: openfoam.com (ESI/OpenCFD)
- also popular: openfoam.org (The OpenFOAM Foundation)

An FVM framework for PDEs

- CFD, Heat Transfer, ...
- Meshing
- Solving
- Post-Processing
- ...

---

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

The Solvers
preCICE Tutorials using OpenFOAM

On www.precice.org/resources (step-by-step):

Flow above a heated plate
  - Demo in precice/openfoam-adapter
  - buoyantPimpleFoam + laplacianFoam
  - Learn how to use the OpenFOAM adapter

Shell-and-Tube Heat Exchanger
  - Larger case in precice/tutorials
  - buoyantSimpleFoam (x2) + CalculiX
  - Learn how to do multi-coupling
The Solvers
FEniCS

Software

- open-source (LGPLv3)
- extensive documentation
- Python and C++ API
- can be used for HPC
- www.fenicsproject.org

Computing platform for solving PDEs

- Definition of weak forms
- Finite Element basis functions
- Meshing
- Solving
- ...

---


Benjamin Rüth (TUM) | Using FEniCS and OpenFOAM for CHT in a partitioned fashion
The Solvers
Toy problem: Partitioned Heat Equation

Partitioned heat equation / transmission problem already discussed in literature (e.g.\(^1\) or \(2\)).

- in precice/tutorials
- today: only use left half of the domain + FEniCS adapter
- for details see \(3\).

---


Flow over plate
preCICE tutorial

Boundary conditions and geometry

\[ u_\infty = 0.1 \]
\[ T_\infty = 300 \]

buoyantPimpleFoam

\[ \Gamma_D: T_{\text{Solid}} = T_{\text{Fluid}} \]

\[ q = 0 \]

\[ q = 0 \]

\[ u_\infty = 0 \]

\[ T_\infty = 300 \]

\[ \Gamma_N: q_{\text{Fluid}} = q_{\text{Solid}} \]

\[ q = 0 \]

\[ q = 0 \]

\[ q = 0 \]

\[ T_C = 310 \]

Additional parameters

- \[ \lambda = 0.25 \] = plate width/plate length
- \[ Pr = 0.01 \]
- \[ Re = \rho u_\infty d/\mu = 500 \] (use characteristic length \( d = \) plate length)
- \[ k_s = k_f \] (thermal conductivities)

---

Cheung Yau, L. (2016). Conjugate Heat Transfer with the Multiphysics Coupling Library preCICE.

Benjamin Rüth (TUM) | Using FEniCS and OpenFOAM for CHT in a partitioned fashion
Flow over plate
OpenFOAM-OpenFOAM vs. OpenFOAM-FEniCS

OpenFOAM-OpenFOAM
- Fluid from openfoam-adapter/tutorials
- Solid from openfoam-adapter/tutorials
- precice-config.xml from openfoam-adapter/tutorials

OpenFOAM-FEniCS
- Fluid from openfoam-adapter/tutorials
- Solid/heat.py from precice/tutorials
- precice-config.xml from openfoam-adapter/tutorials
Flow over plate
OpenFOAM-OpenFOAM vs. OpenFOAM-FEniCS

$T_\infty = 300$

$T_C = 310$

$T_b$ on Solid domain for both setups.
Comparison to literature

Vynnycky

Problem setup from

\[ \text{Results from} \]

\[ \text{Re} = 500, \; \text{Pr} = 0.01, \; \lambda = 0.25 \]

---

Comparison to literature
preCICE vs. Vynnycky

\[ \theta_b = \frac{T_C - T}{T_\infty - T_C} \]

\( T_\infty = 300 \)
\( T_C = 310 \)

\( \theta_b \) on Solid domain for both setups.
Comparison to literature

Different Setups

Cheung\textsuperscript{1} vs. Vynnycky\textsuperscript{2}

\begin{align*}
\text{Cheung} & : \quad u_\infty = 0.1, \quad T_\infty = 300, \quad T_C = 310, \\
\text{Vynnycky} & : \quad q = 0
\end{align*}

\textsuperscript{1}Cheung Yau, L. (2016). Conjugate Heat Transfer with the Multiphysics Coupling Library preCICE.
Comparison to literature

Different Setups

Cheung\textsuperscript{1} vs. Vynnycky\textsuperscript{2}

\textsuperscript{1}Cheung Yau, L. (2016). Conjugate Heat Transfer with the Multiphysics Coupling Library preCICE.

Summary & Outlook

FEniCS + OpenFOAM

- FEniCS or OpenFOAM are used for heat equation in solid domain
  - `heat.py` is only a proof-of-concept
  - `laplacianFoam` more advanced
- OpenFOAM's `buoyantPimpleFoam` solves flow + energy transport in fluid domain.
- `preCICE` couples the solvers with identical `precice-config.xml`
- Tutorial can be found at
  github.com/precice/tutorials/CHT/flow-over-plate/buoyantPimpleFoam-fenics

---

**Summary & Outlook**

**FEniCS + OpenFOAM**

- FEniCS or OpenFOAM are used for heat equation in solid domain
  - heat.py is only a proof-of-concept
  - laplacianFoam more advanced
- OpenFOAM’s buoyantPimpleFoam solves flow + energy transport in fluid domain.
- preCICE couples the solvers with identical precice-config.xml
- Tutorial can be found at
  github.com/precice/tutorials/CHT/flow-over-plate/buoyantPimpleFoam-fenics

**Quantitative assessment**

- good agreement of FEniCS + OpenFOAM with OpenFOAM + OpenFOAM
- analytic solution and simulation do not match

---

Summary & Outlook

FEniCS + OpenFOAM

- FEniCS or OpenFOAM are used for heat equation in solid domain
  - heat.py is only a proof-of-concept
  - laplacianFoam more advanced
- OpenFOAM’s buoyantPimpleFoam solves flow + energy transport in fluid domain.
- preCICE couples the solvers with identical precice-config.xml
- Tutorial can be found at github.com/precice/tutorials/CHT/flow-over-plate/buoyantPimpleFoam-fenics

Quantitative assessment

- good agreement of FEniCS + OpenFOAM with OpenFOAM + OpenFOAM
- analytic solution and simulation do not match

Outlook

- more FEniCS tutorials (FEniCS + X)
- FEniCS-based solvers as CBC.Block, CBC.RANS and CBC.Solve
  Reproducing Vynnycky’s results (github.com/precice/tutorials/issues/22)

---

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
Study on Vynnycky setup
Different geometry & boundary conditions

\[ \theta_b = \frac{T_C - T}{T_\infty - T_C} \]

Where does the shift come from?
Study on Vynnycky setup

Different $T_\infty$

$T_C = 310, T_\infty = 300$

$T_C = 310, T_\infty = 25$

How to set $T_C$ and $T_\infty$?
Heat Equation in FEniCS

Heat Equation

\[
\frac{\partial u}{\partial t} = \Delta u + f \text{ in } \Omega \\
u = u_0(t) \text{ on } \partial \Omega
\]

Analytical Solution, if \( f = \beta - 2 - 2\alpha \) we get
\( u = 1 + x^2 + \alpha y^2 + \beta t \).

Discretization

- implicit Euler:
\[
\frac{u^k - u^{k-1}}{dt} = \Delta u^k + f^k
\]

- trial space:
\[
u \in V_h \subset V = \{ v \in H^1(\Omega) : v = u_0 \text{ on } \partial \Omega \}\]

- test space:
\[
\tilde{V}_h \subset V = \{ v \in H^1(\Omega) : v = 0 \text{ on } \partial \Omega \}\]

- weak form:
\[
\int_{\Omega} (u^k v + dt \nabla u^k \cdot \nabla v) dx = \int_{\Omega} (u^{k-1} + dt f^k) v dx
\]

Remark: Tutorial from the FEniCS book\(^1\)

---

\(^1\)Langtangen, H. P., & Logg, A. (2016). Solving PDEs in Python - The FEniCS Tutorial I (1st ed.).
Heat Equation in FEniCS

**Geometry:** \( \Omega, \partial \Omega, \Gamma_D, \Gamma_N \)

```python
class RightBoundary(SubDomain):
    def inside(self, x, on_boundary):
        tol = 1E-14
        if on_boundary
            and near(x[0], x_r, tol):
                return True
        else:
            return False

class Boundary(SubDomain):
    def inside(self, x, on_boundary):
        if on_boundary:
            return True
        else:
            return False
```

\( p_0 = \text{Point}(0, 0) \)

\( p_1 = \text{Point}(1, 1) \)

**Mesh:** \( \Omega_h \)

\( nx = 5 \)

\( ny = 5 \)

```python
mesh = RectangleMesh(p0, p1, nx, ny)
```

Mesh created with FEniCS
Heat Equation in FEniCS

Function Space: $V_h \subset V = \{ v \in H^1(\Omega) \}$

$V = \text{FunctionSpace}(\text{mesh}, \ 'P', \ 1)$

Expressions: $u = 1 + x^2 + \alpha y^2 + \beta t$ and $f = \beta - 2 - 2\alpha$

$u_D = \text{Expression}(\ '1 + x[0]*x[0] + alpha*x[1]*x[1] + beta*t', \ ..., \ t=0)$

$f = \text{Constant}(\beta - 2 - 2 * \alpha)$

Boundary Conditions: $u \in V_h \subset V = \{ v \in H^1(\Omega) : v = u_D \text{ on } \partial \Omega \}$ and $v \in \hat{V}_h \subset V = \{ v \in H^1(\Omega) : v = 0 \text{ on } \partial \Omega \}$

$bc = \text{DirichletBC}(V, u_D, \text{Boundary})$

$u = \text{TrialFunction}(V)$

$v = \text{TestFunction}(V)$

Initial Condition: $u^0 = u(t = 0)$

$u_n = \text{interpolate}(u_D, V)$
Heat Equation in FEniCS

**Variational Problem:** \[ \int_{\Omega} (u^k v + dt \nabla u^k \cdot \nabla v) dx = \int_{\Omega} (u^{k-1} + dt f^k) v dx \]

\[ F = u * v * dx + dt * \text{dot} (\text{grad}(u), \text{grad}(v)) * dx - (u_n + dt * f) * v * dx \]

\[ a, L = \text{lhs}(F), \text{rhs}(F) \]

**Time-stepping and simulation loop:** \[ \frac{u^k - u^{k-1}}{dt} = \Delta u^k + f^k \]

\[ u_{np1} = \text{Function}(V) \]
\[ t = 0 \]
\[ T = 1 \]
\[ dt = .1 \]
\[ u_D.t = t + dt \]

**while** \( t < T: \)

\[ \text{solve}(a == L, u_{np1}, bc) \]
\[ t += dt \]
\[ u_D.t = t + dt \]
\[ u_n.assign(u_{np1}) \]