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

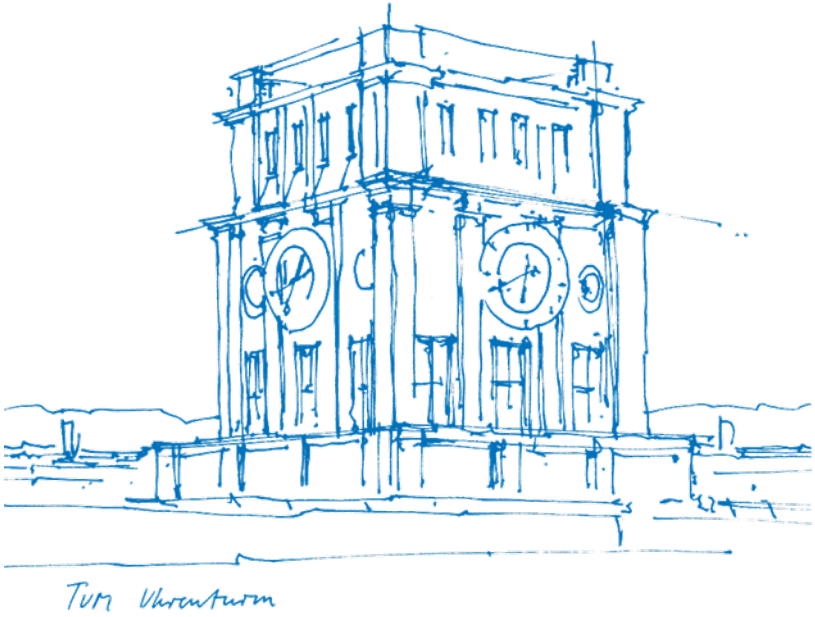
Benjamin Rüth[1], Peter Meisrimel[2], Philipp Birken[2], Gerasimos Chourdakis[1], Benjamin Uekermann[3]

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

[2]Lund University
Mathematics (Faculty of Sciences)
Numerical Analysis

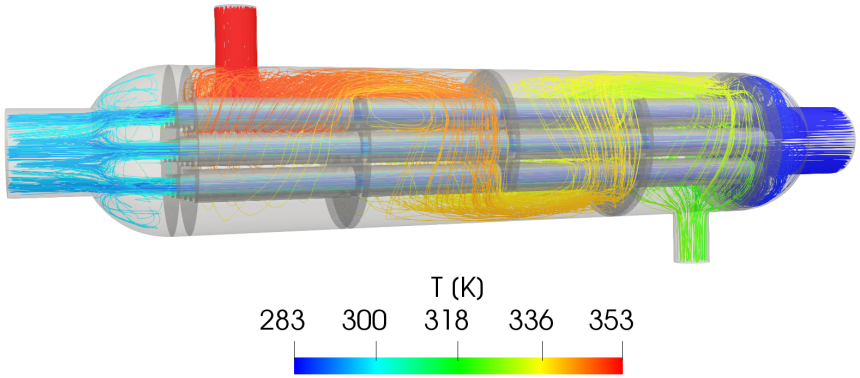
[3]Eindhoven University of Technology
Department of Mechanical Engineering
Energy Technology

GAMM Annual 2019
Vienna, Austria
February 19, 2019



Partitioned Approach

Coupled Problem

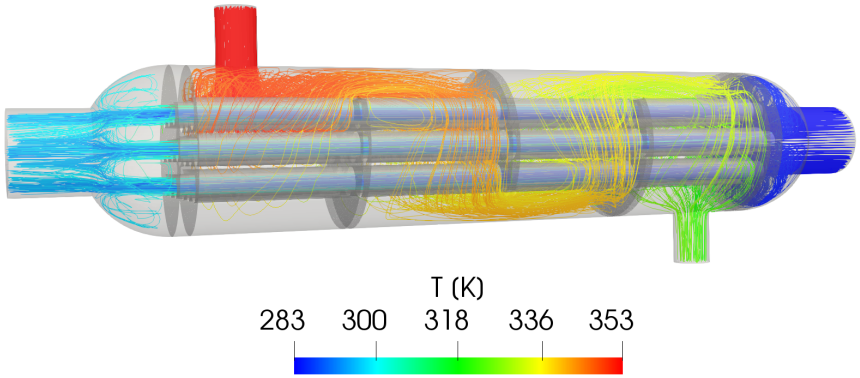


shell and tube heat exchanger using OpenFOAM and CalculiX[1]

[1] Figure from *Rusch, A., Uekermann, B. Comparing OpenFOAM's Intrinsic Conjugate Heat Transfer Solver with preCICE-Coupled Simulations. Technical Report, 2018.*

Partitioned Approach

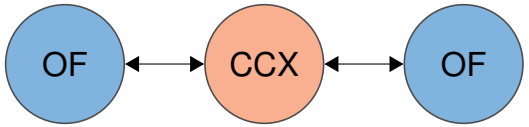
Coupled Problem



shell and tube heat exchanger using OpenFOAM and CalculiX[1]

Basic idea:

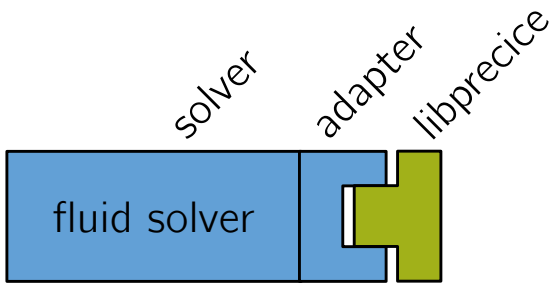
- reuse existing solvers
- combine single-physics to solve multi-physics
- only exchange "black-box" information



[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



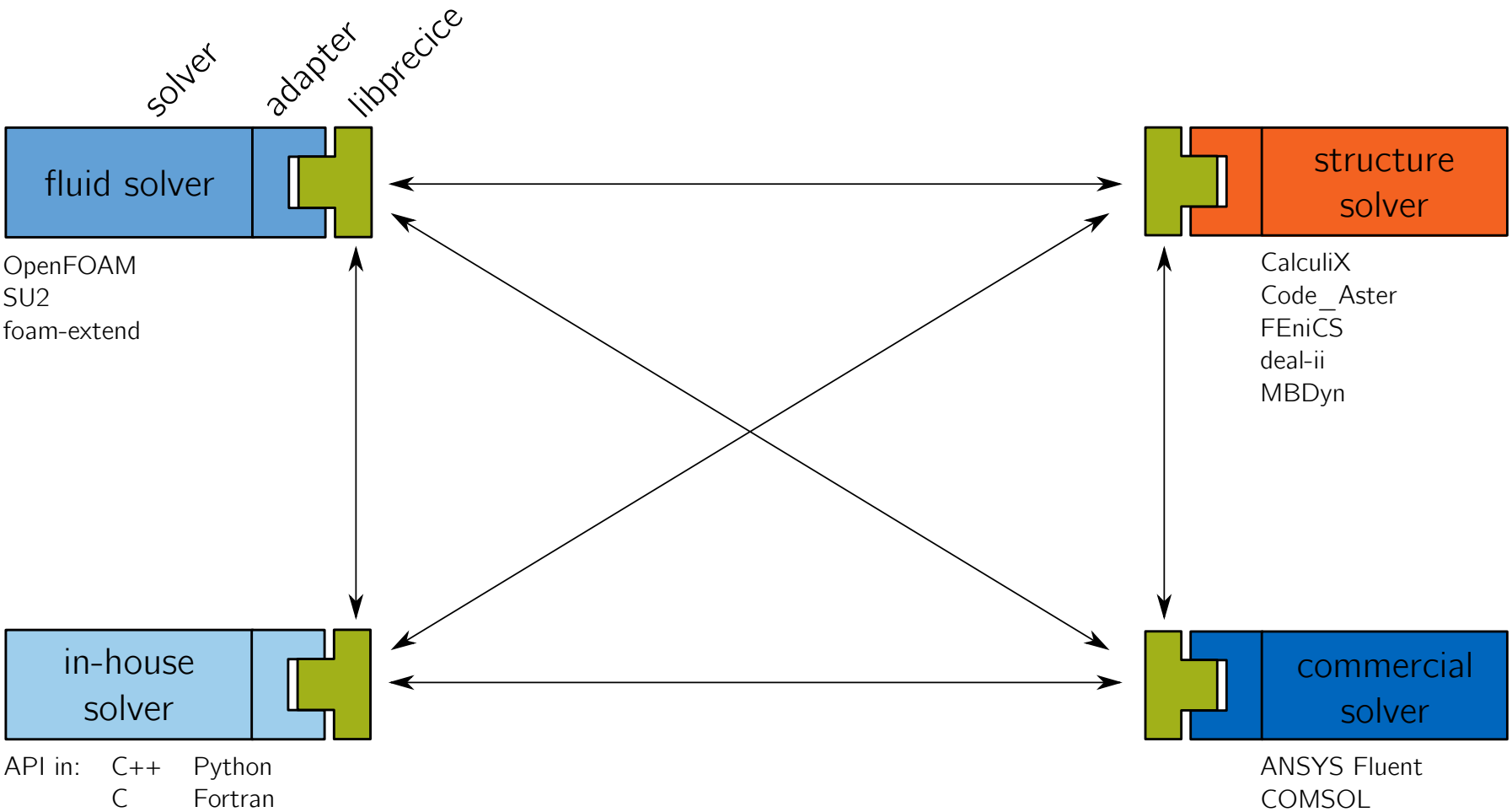
OpenFOAM
SU2
foam-extend

preCICE

A Plug-and-Play Coupling Library

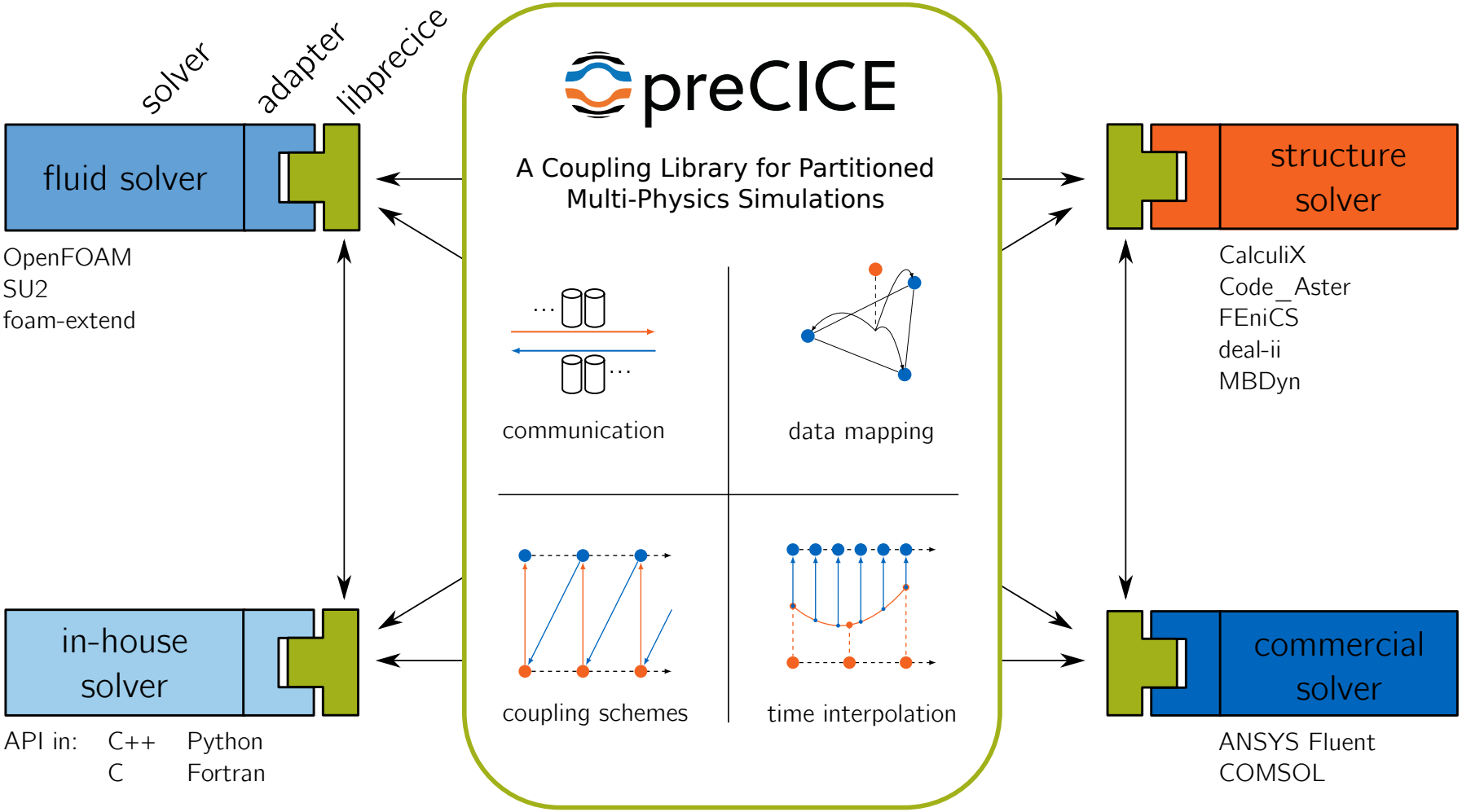
preCICE

A Plug-and-Play Coupling Library



preCICE

A Plug-and-Play Coupling Library



preCICE

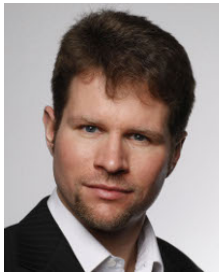
Unique Selling Points (USPs)

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

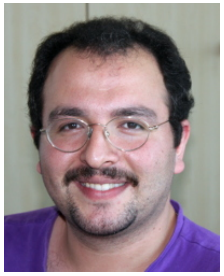
preCICE

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 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 . . .

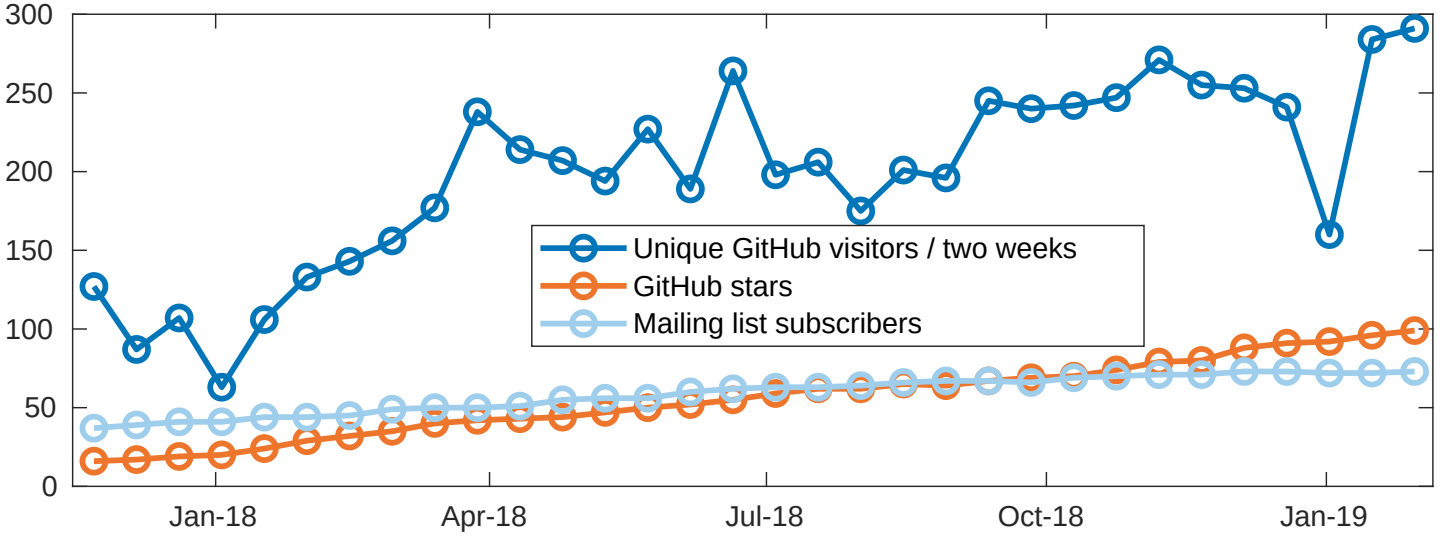
preCICE

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



preCICE

Plug and play?

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

The Solvers

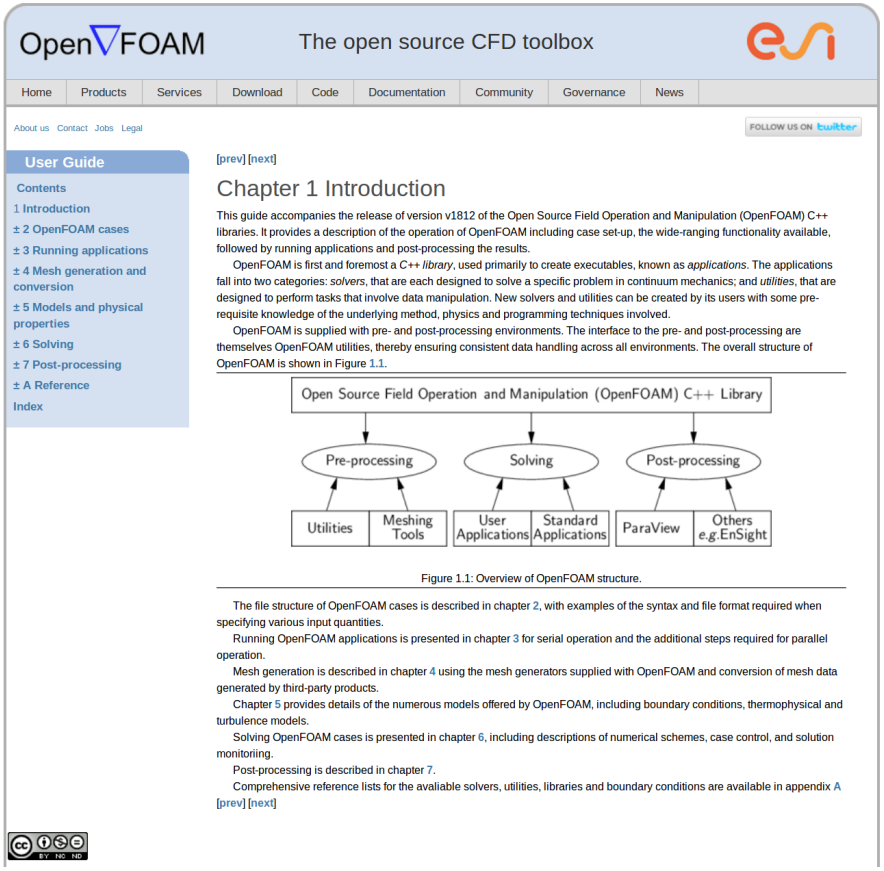
OpenFOAM®[1]

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
- ...



OpenFOAM User Guide[2]

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

[2] OpenFOAM User Guide: `https://www.openfoam.com/documentation/user-guide/`

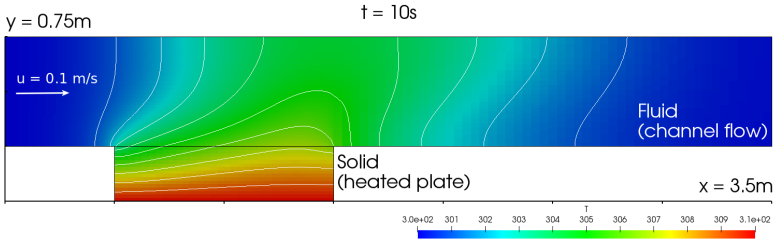
The Solvers

preCICE Tutorials using OpenFOAM

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

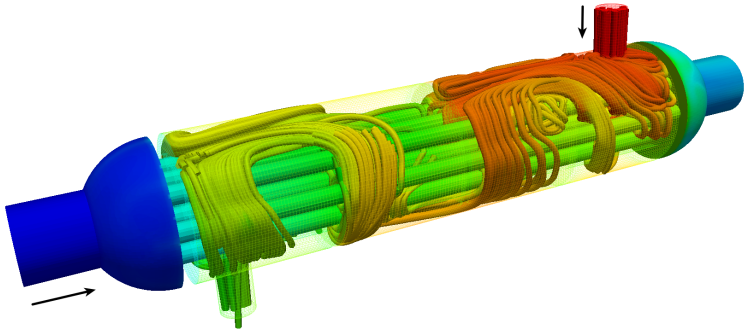
Flow above a heated plate

- 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

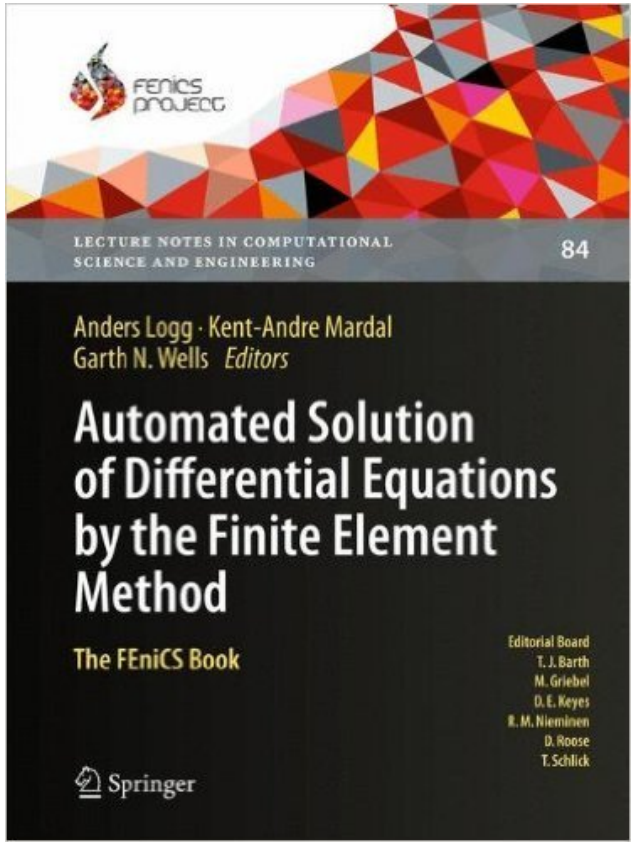
FEniCS[1]

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
- ...



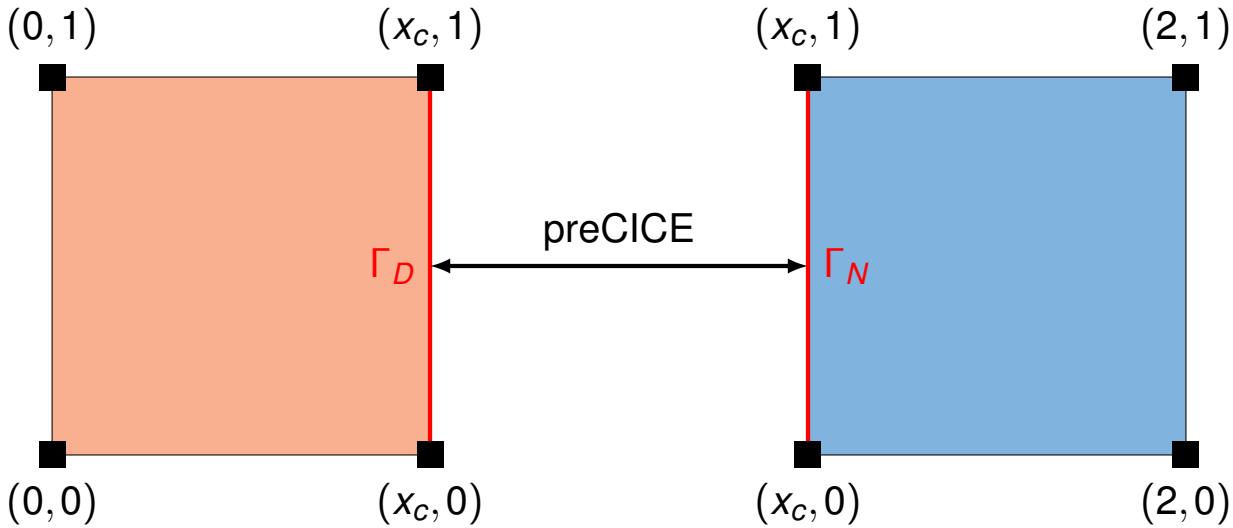
FEniCS book[2]

[1] *Alnaes, M. S., et al. (2015). The FEniCS Project Version 1.5.*

[2] *Logg, A., Mardal, K. A., & Wells, G. N. (2012). Automated solution of differential equations by the finite element method.*

The Solvers

Toy problem: Partitioned Heat Equation



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

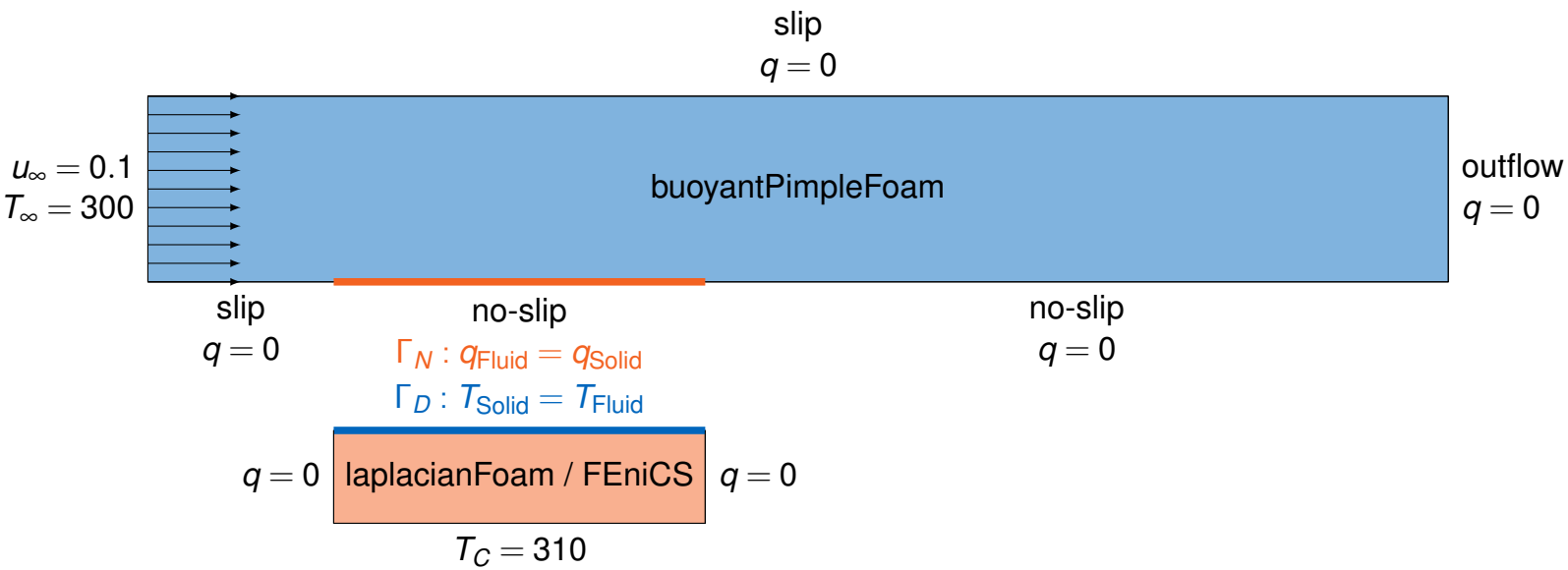
- in `precice/tutorials`
- today: only use left half of the domain + FEniCS adapter
- for details see [3].

[1] Monge, A. (2018). Partitioned methods for time-dependent thermal fluid-structure interaction. Lund University.
[2] Toselli, A., & Widlund, O. (2005). Domain Decomposition Methods - Algorithms and Theory (1st ed.).
[3] Rüth, B. et al. (2018). Solving the Partitioned Heat Equation Using FEniCS and preCICE. GAMM CSE Workshop 2018.

Flow over plate

preCICE tutorial [1]

Boundary conditions and geometry



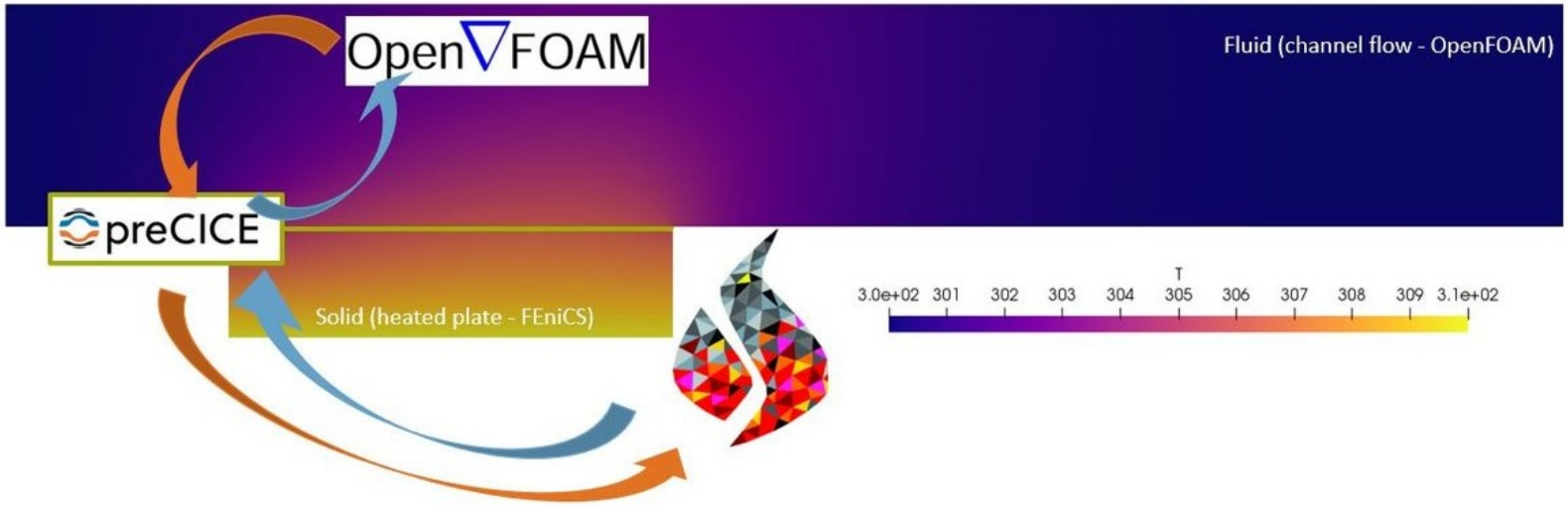
Additional parameters

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

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

Flow over plate

OpenFOAM-OpenFOAM vs. OpenFOAM-FEniCS



OpenFOAM-OpenFOAM

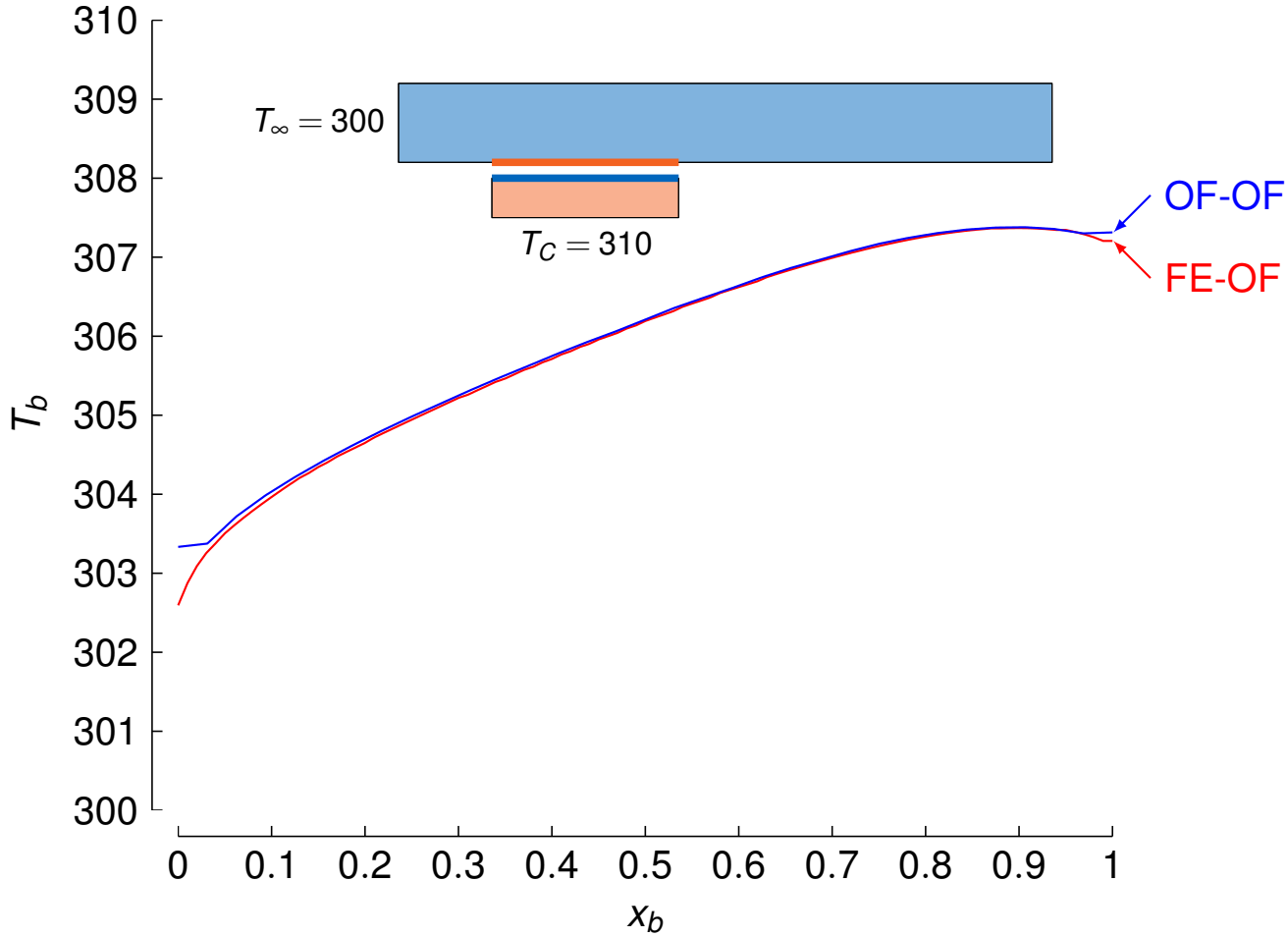
- `Fluid` from `openfoam-adapter/tutorials`
- `Solid` from `openfoam-adapter/tutorials`
- `precice-config.xml` from `openfoam-adapter/tutorials`

OpenFOAM-FEniCS

- `Fluid` from `openfoam-adapter/tutorials`
- `Solid/heat.py` from `precice/tutorials`
- `precice-config.xml` from `openfoam-adapter/tutorials`

Flow over plate

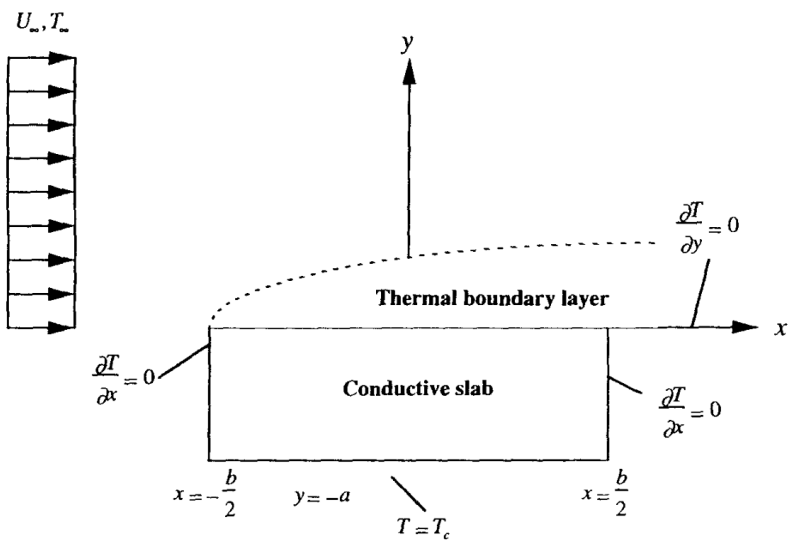
OpenFOAM-OpenFOAM vs. OpenFOAM-FEniCS



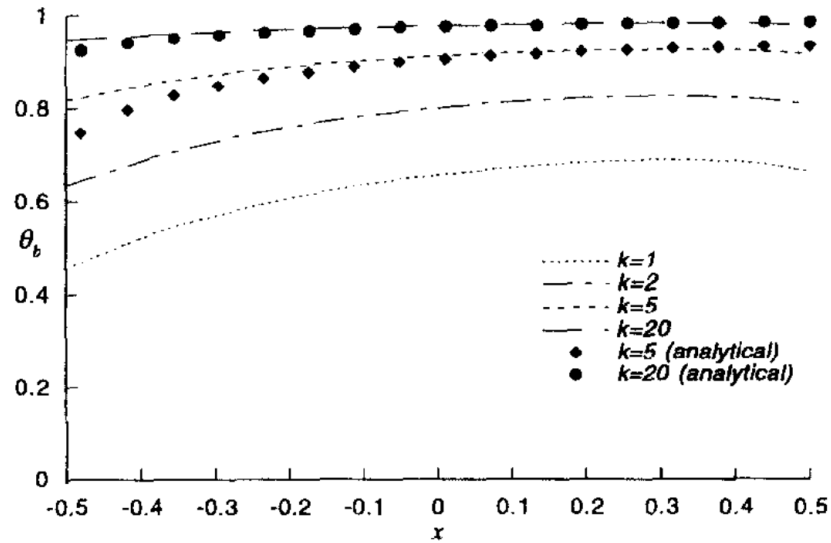
T_b on Solid domain for both setups.

Comparison to literature

Vynnycky [1]



Problem setup from [1]



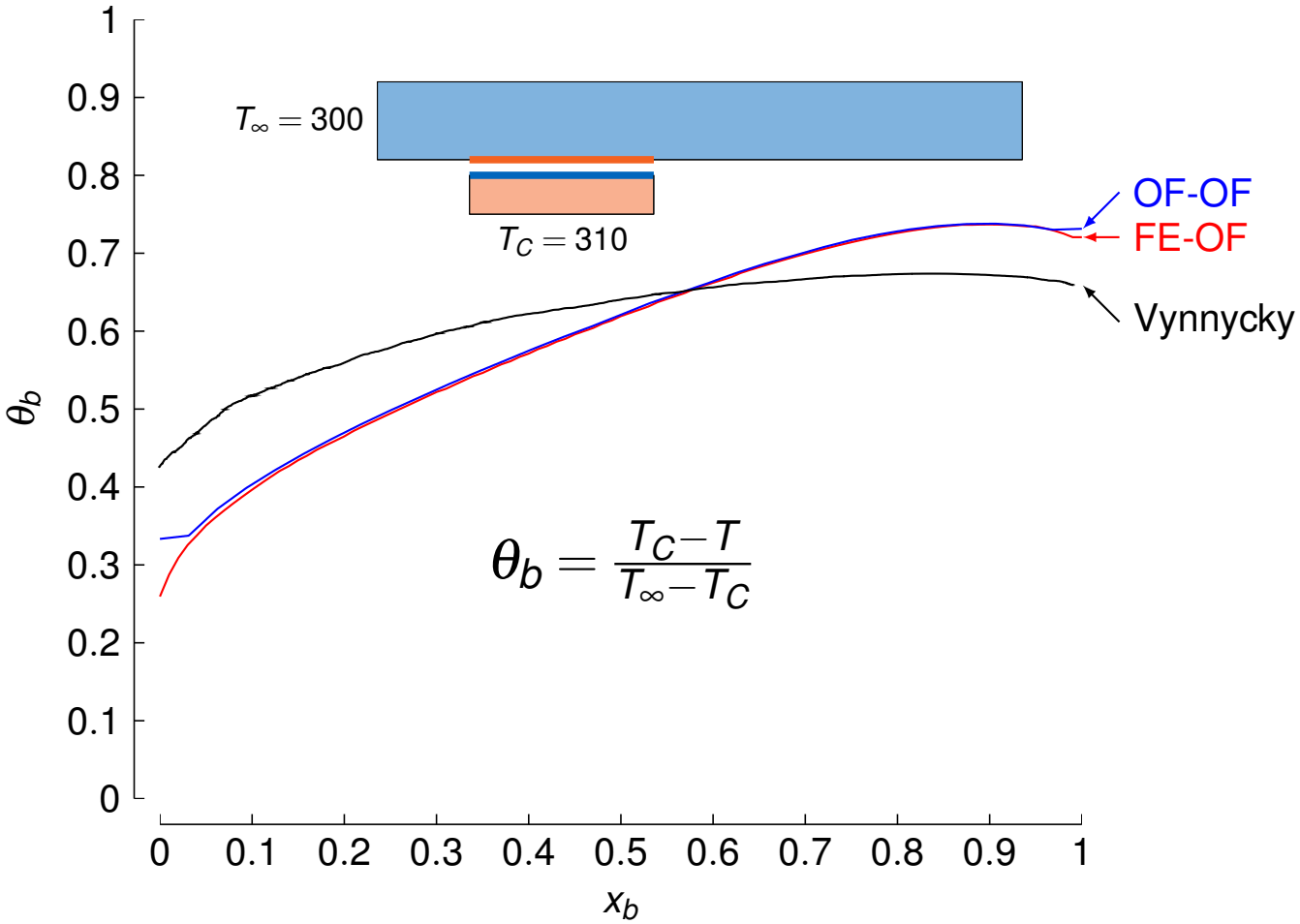
Results from [1]

$Re = 500$, $Pr = 0.01$, $\lambda = 0.25$

[1] Vynnycky, M., et al. (1998). Forced convection heat transfer from a flat plate: the conjugate problem.

Comparison to literature

preCICE vs. Vynnycky

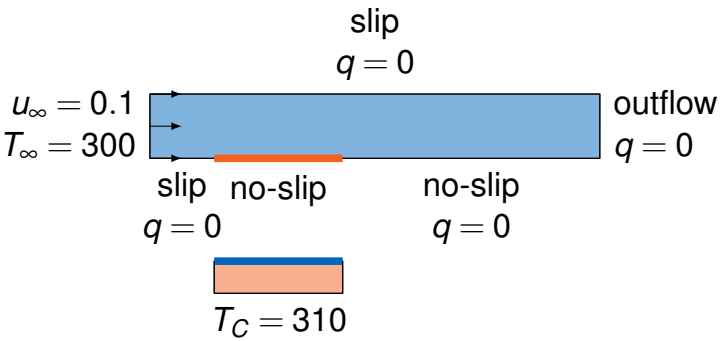


θ_b on Solid domain for both setups.

Comparison to literature

Different Setups

Cheung[1] vs. Vynnycky[2]



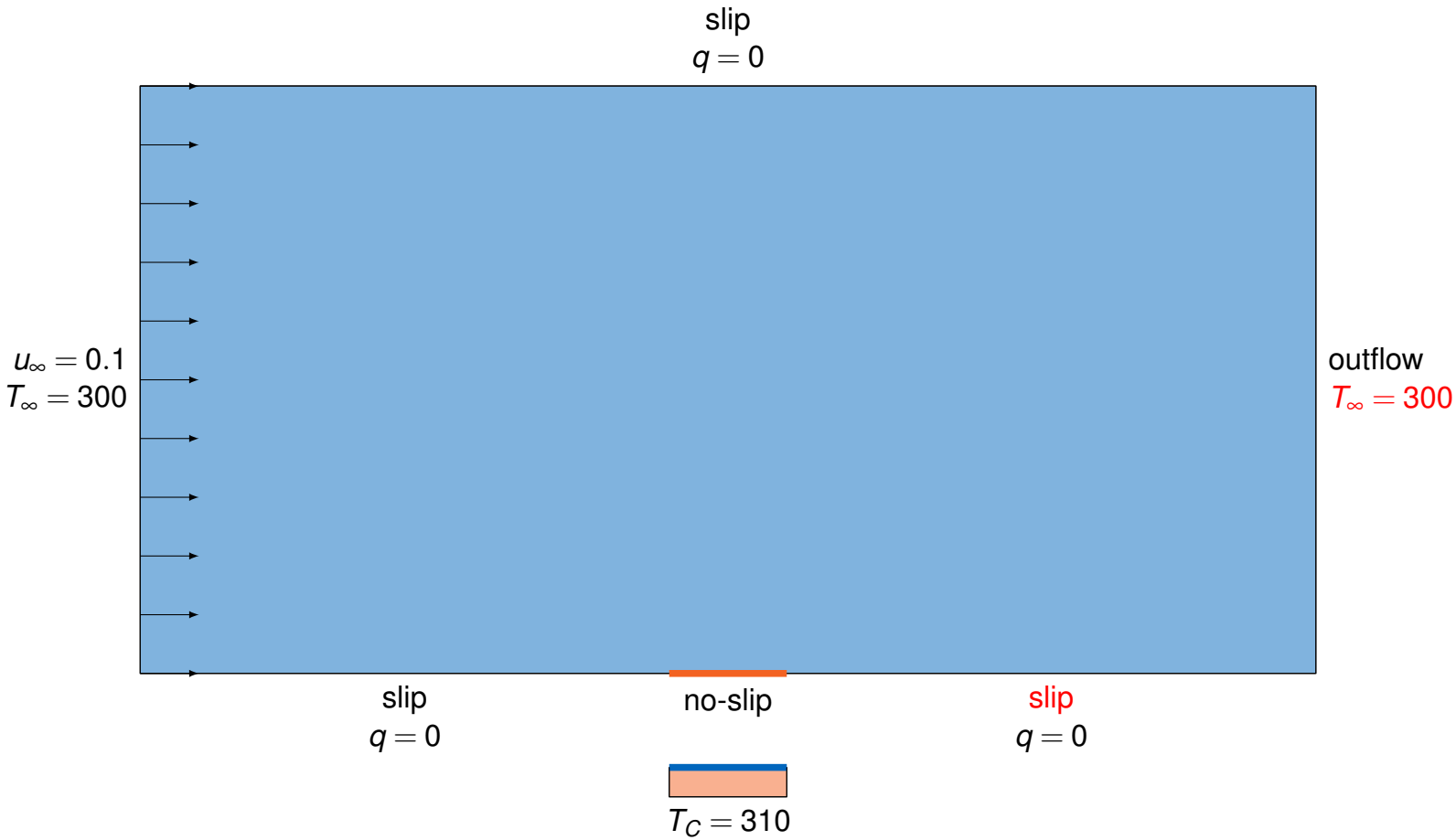
[1] Cheung Yau, L. (2016). Conjugate Heat Transfer with the Multiphysics Coupling Library preCICE.

[2] Vynnycky, M., et al. (1998). Forced convection heat transfer from a flat plate: the conjugate problem.

Comparison to literature

Different Setups

Cheung[1] vs. **Vynnycky**[2]



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

[2] Vynnycky, M., et al. (1998). Forced convection heat transfer from a flat plate: the conjugate problem.

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`

[1] *Logg, A., Mardal, K. A., & Wells, G. N. (2012). Automated solution of differential equations by the finite element method. Lecture Notes in Computational Science and Engineering.*

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

[1] *Logg, A., Mardal, K. A., & Wells, G. N. (2012). Automated solution of differential equations by the finite element method. Lecture Notes in Computational Science and Engineering.*

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[1]
- Reproducing Vynnycky's results (`github.com/precice/tutorials/issues/22`)

[1] *Logg, A., Mardal, K. A., & Wells, G. N. (2012). Automated solution of differential equations by the finite element method. Lecture Notes in Computational Science and Engineering.*

Summary

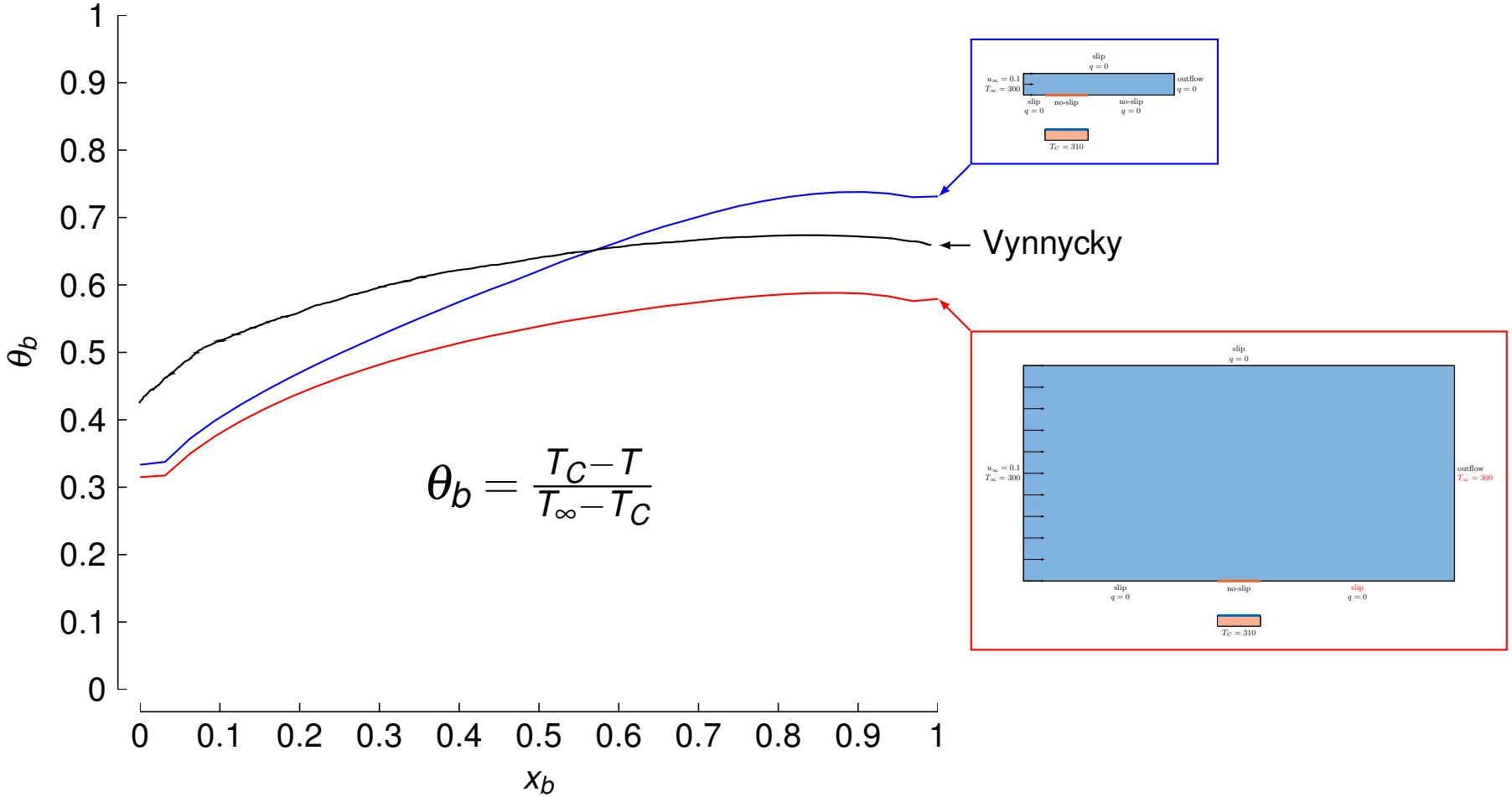
preCICE

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

Study on Vynnycky setup

Different geometry & boundary conditions



Where does the shift come from?

Study on Vynnycky setup

Different T_∞

$T_C = 310, T_\infty = 300$

$T_C = 310, T_\infty = 25$

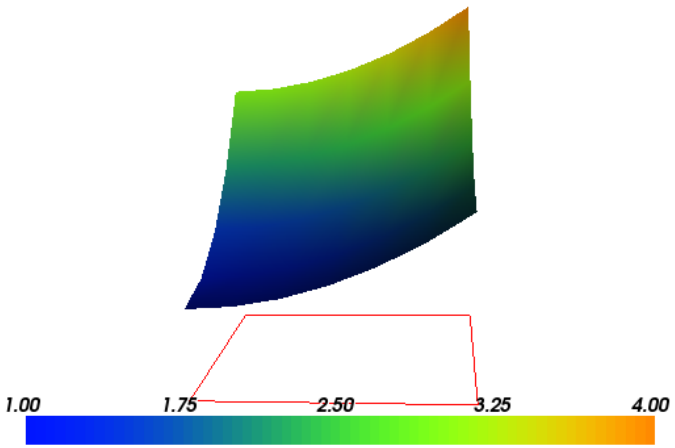
How to set T_C and T_∞?

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 $u = 1 + x^2 + \alpha y^2 + \beta t$.



Solution of Poisson equation. Figure from [1].

Discretization

- **implicit Euler:**

$$\frac{u^k - u^{k-1}}{dt} = \Delta u^k + f^k$$

- **trial space:**

$$u \in V_h \subset V = \left\{ v \in H^1(\Omega) : v = u_0 \text{ on } \partial\Omega \right\}$$

- **test space:**

$$v \in \hat{V}_h \subset V = \left\{ v \in H^1(\Omega) : v = 0 \text{ on } \partial\Omega \right\}$$

- **weak form:**

$$\int_\Omega (u^k v + \mathrm{d}t \nabla u^k \cdot \nabla v) \mathrm{d}x = \int_\Omega \left(u^{k-1} + \mathrm{d}t\, f^k \right) v \mathrm{d}x$$

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: $\Omega, \partial\Omega, \Gamma_D, \Gamma_N$

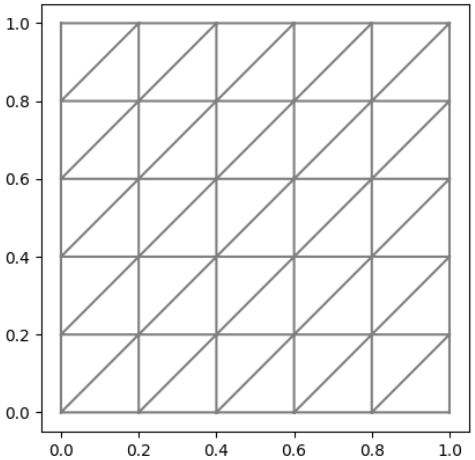
```
class RightBoundary(SubDomain):
    def inside(self, x, on_boundary):
        tol = 1E-14
        if on_boundary
           and near(x[0], x_r, tol):
            return True
        else:
            return False

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

p0 = Point(0, 0)
p1 = Point(1, 1)
```

Mesh: Ω_h

```
nx = 5
ny = 5
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 = FunctionSpace(mesh, 'P', 1)
```

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

```
u_D = Expression('1 + x[0]*x[0] + alpha*x[1]*x[1] + beta*t', ..., t=0)
f = 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 = DirichletBC(V, u_D, Boundary)
u = TrialFunction(V)
v = TestFunction(V)
```

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

```
u_n = interpolate(u_D, V)
```

Heat Equation in FEniCS

Variational Problem: $\int_\Omega (u^k v + \mathrm{d}t \nabla u^k \cdot \nabla v)\mathrm{d}x = \int_\Omega \left(u^{k-1} + \mathrm{d}t\, f^k\right) v \mathrm{d}x$

```
F = u * v * dx + dt * dot(grad(u), grad(v)) * dx - (u_n + dt * f) * v *
    dx
a, L = lhs(F), rhs(F)
```

Time-stepping and simulation loop: $\frac{u^k - u^{k-1}}{\mathrm{d}t} = \Delta u^k + f^k$

```
u_np1 = Function(V)
t = 0
T = 1
dt = .1
u_D.t = t + dt

while t < T:
    solve(a == L, u_np1, bc)
    t += dt
    u_D.t = t + dt
    u_n.assign(u_np1)
```