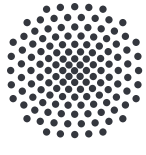
Seminar at NTUA Chemical Engineering, Athens, Greece - September 29, 2025

Black-box coupling of simulation codes using preCICE

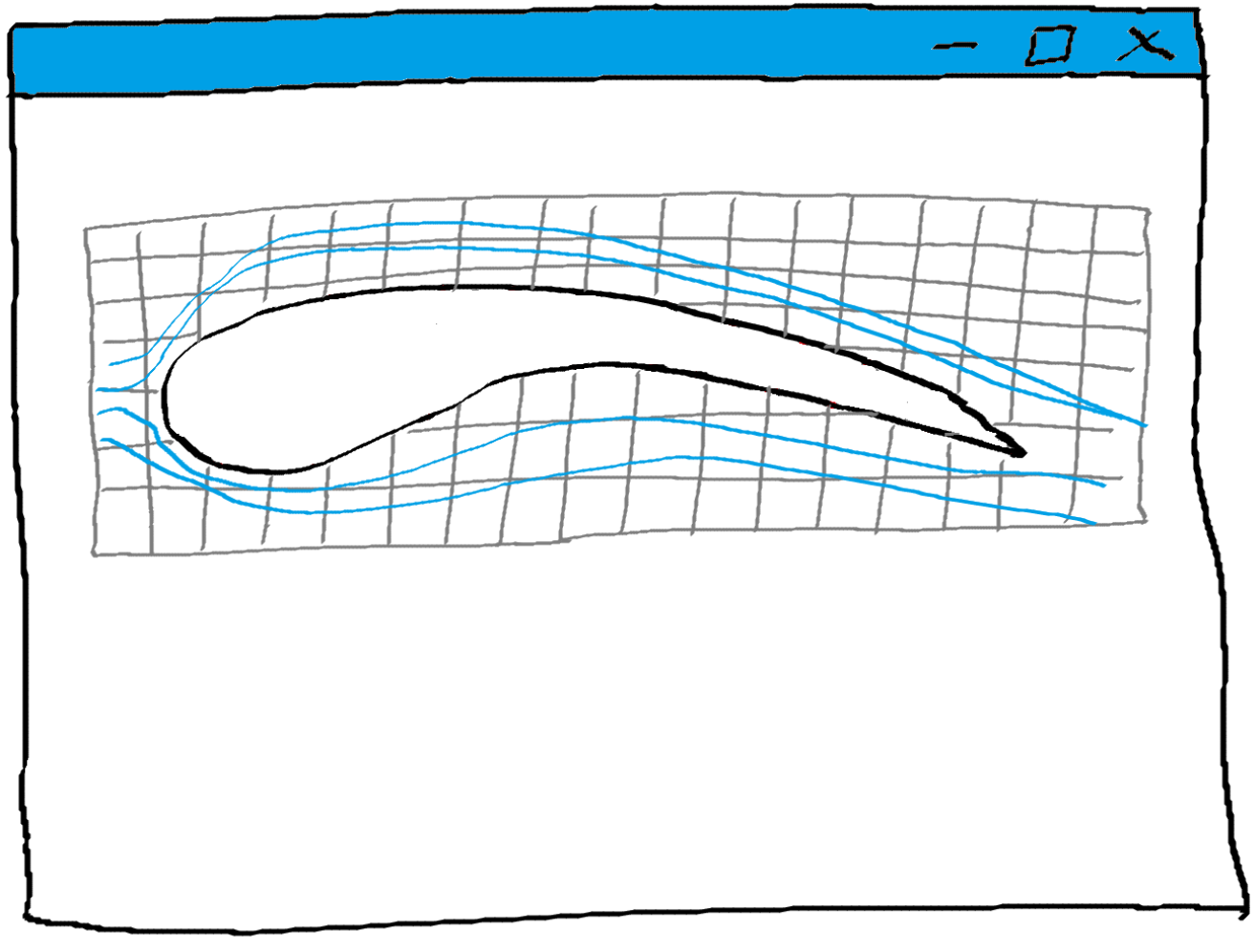
Gerasimos Chourdakis

Institute for Parallel and Distributed Systems
University of Stuttgart
gerasimos.chourdakis@ipvs.uni-stuttgart.de

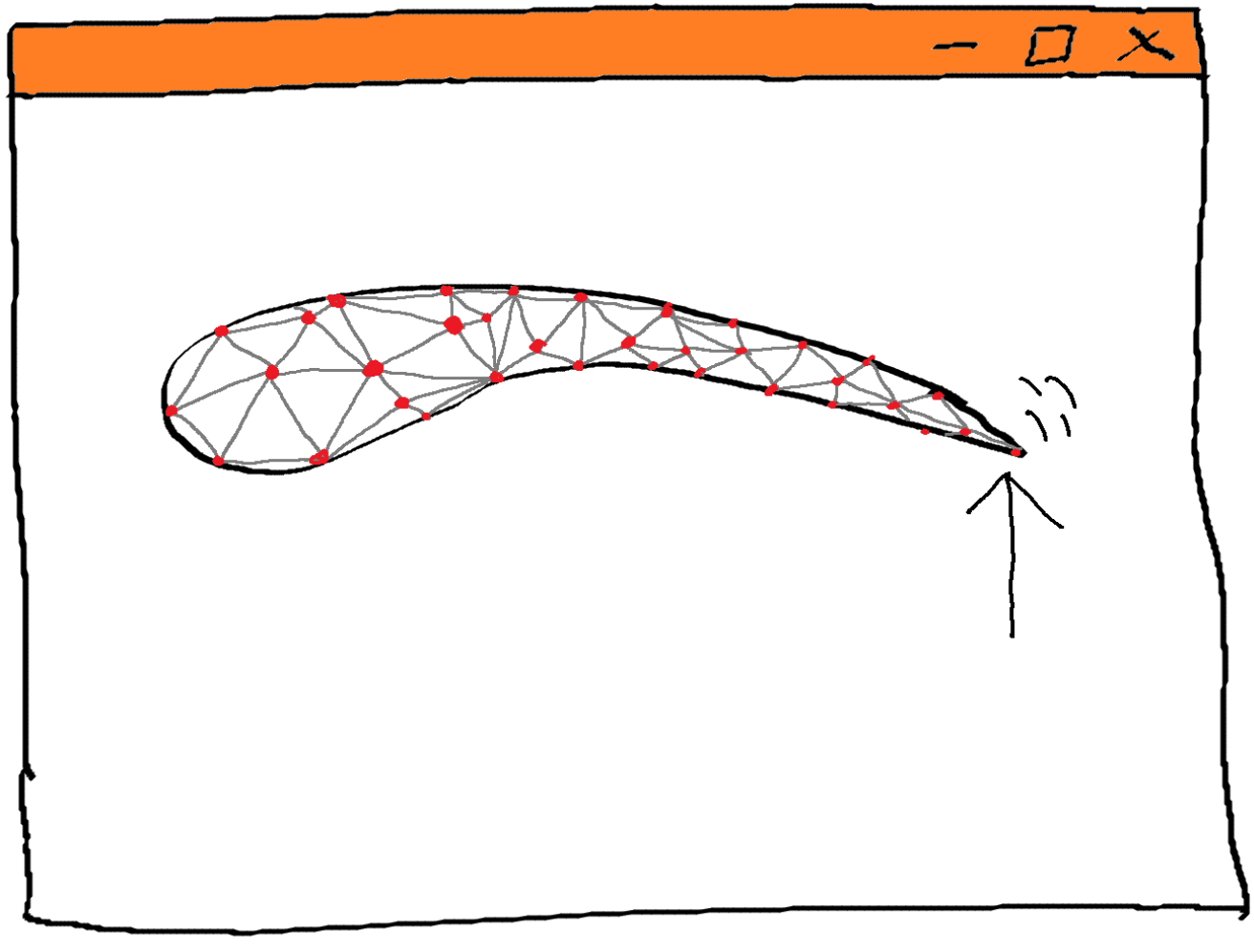


University of Stuttgart
Germany

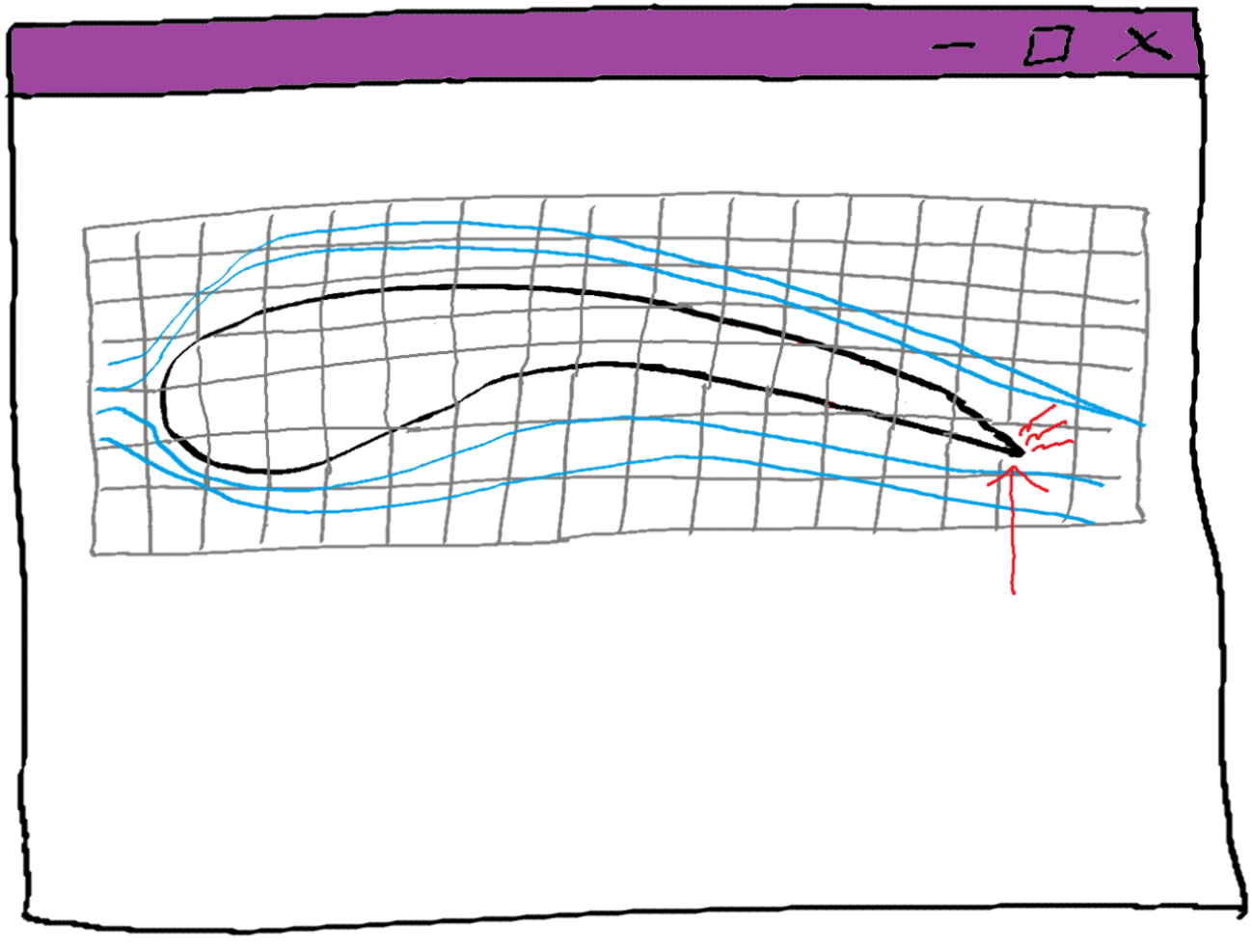
CFD: Simulating flow around a wing



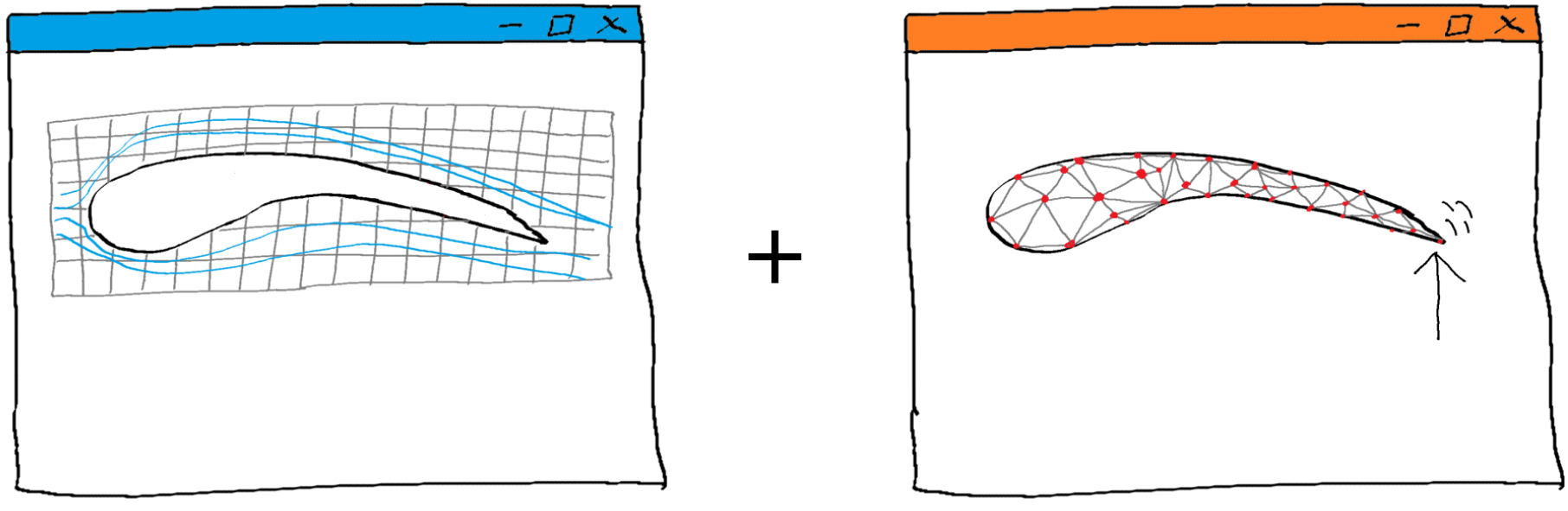
FEM: Simulating stresses on a wing



Can we simulate both at the same time?

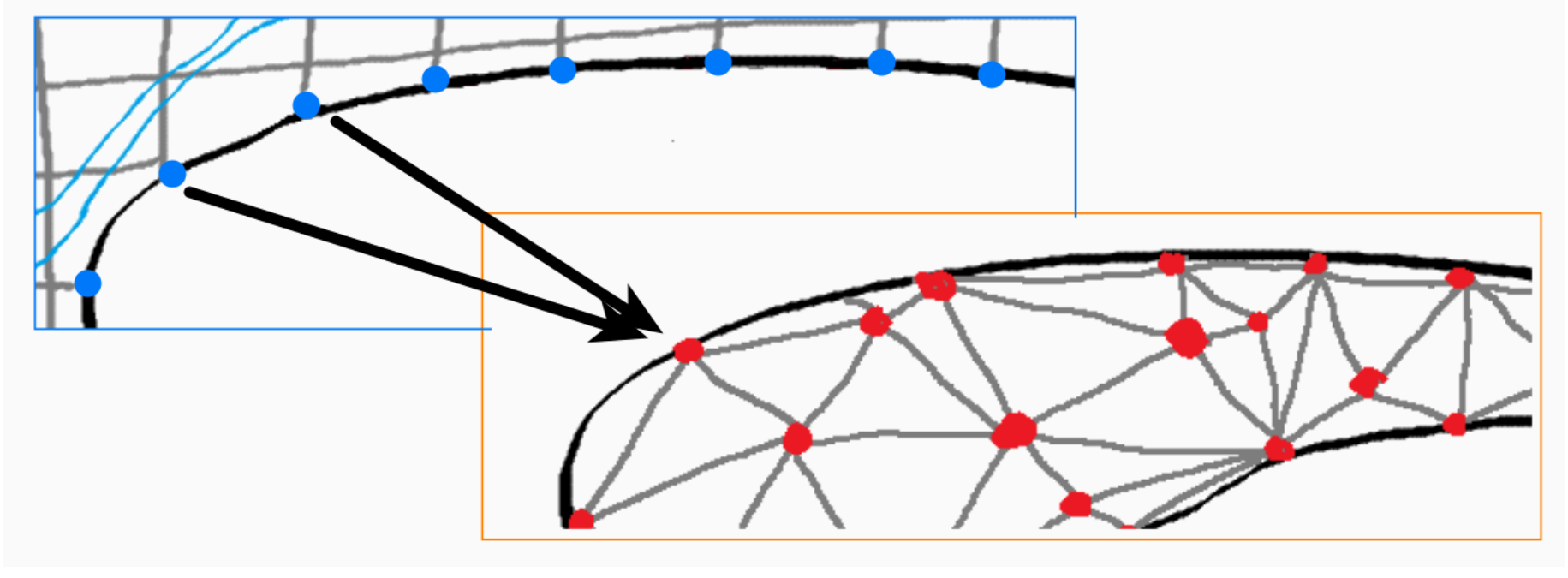


...but reusing the CFD and FEM codes?

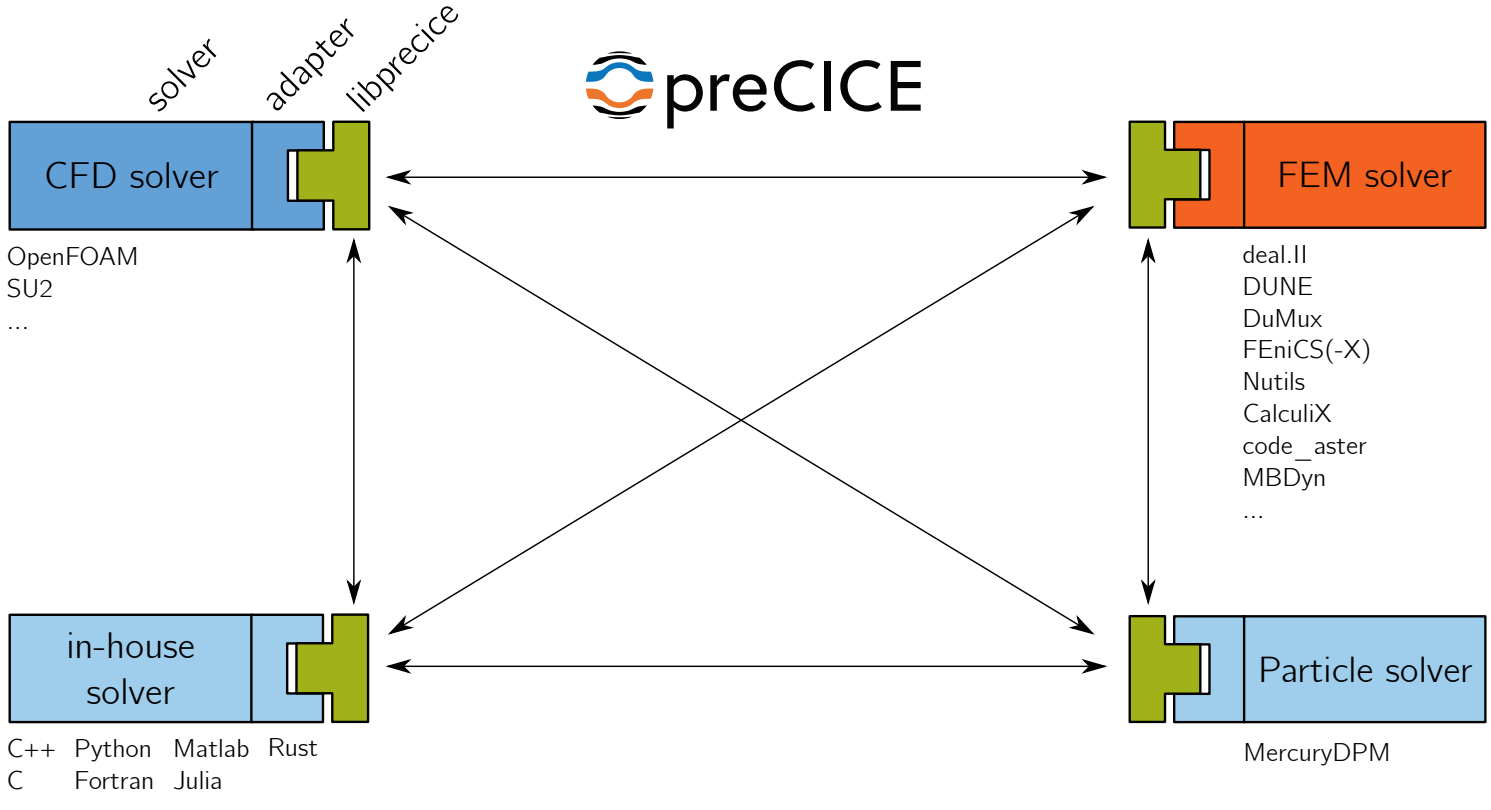


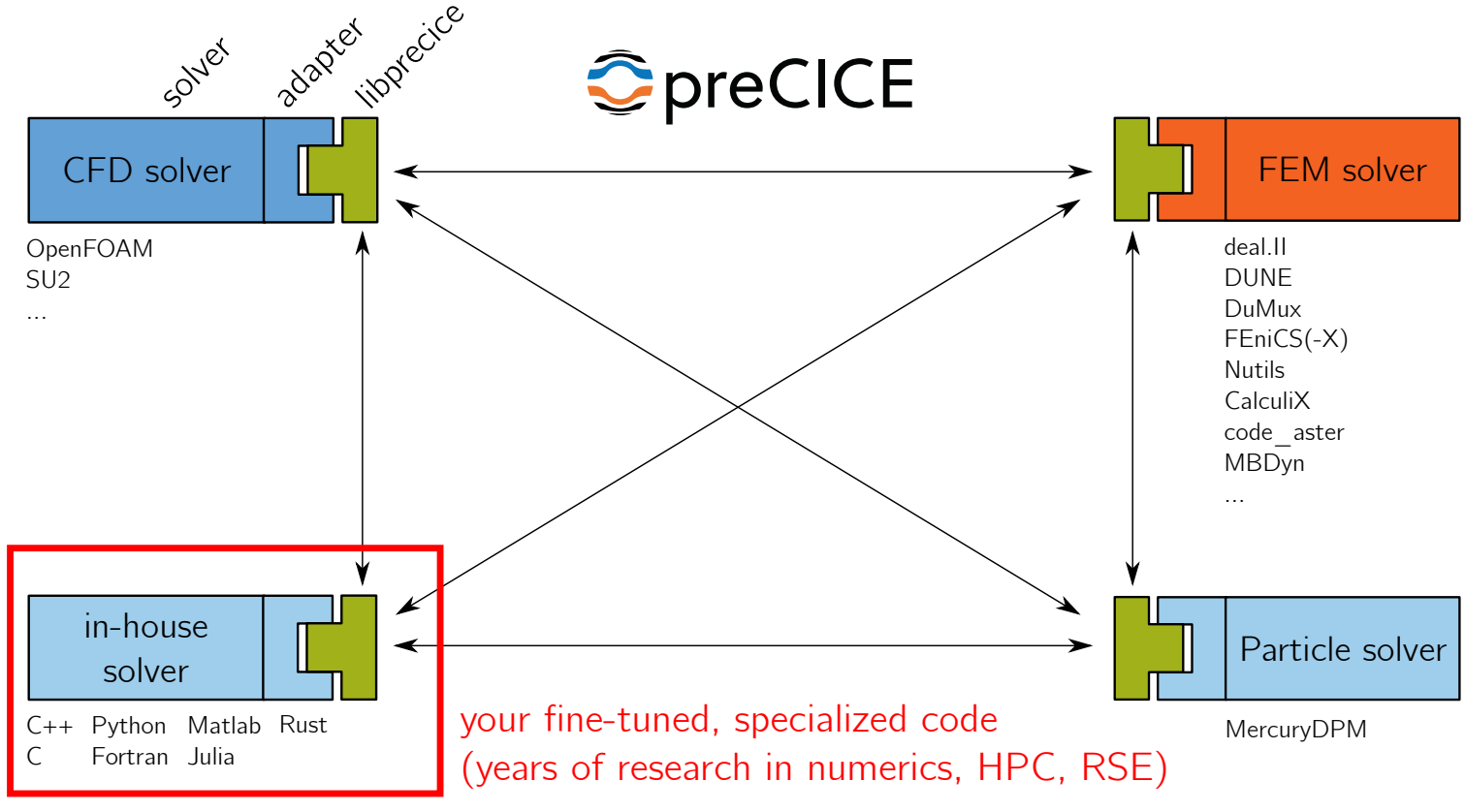
preCICE

From the CFD world to the FEM world

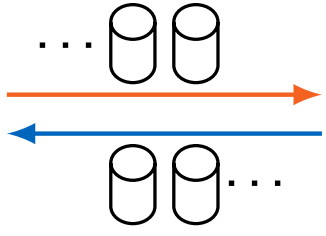



solver
adapter
libprecice
preCICE
CFD solver
FEM solver


solver
adapter
libprecice
preCICE
CFD solver
OpenFOAM
SU2
...
FEM solver
deal.II
DUNE
DuMux
FEniCS(-X)
Nutils
CalculiX
code_aster
MBDyn
...
in-house solver
C++ Python Matlab Rust
C Fortran Julia
Particle solver
MercuryDPM


solver
adapter
libprecice
preCICE
CFD solver
OpenFOAM
SU2
...
FEM solver
deal.II
DUNE
DuMux
FEniCS(-X)
Nutils
CalculiX
code_aster
MBDyn
...
in-house solver
C++ Python Matlab Rust
C Fortran Julia
Particle solver
MercuryDPM
your fine-tuned, specialized code
(years of research in numerics, HPC, RSE)

Where preCICE helps

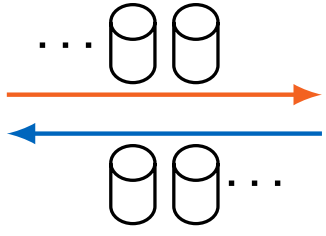


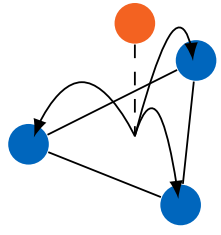
Communication

Options:
- MPI ports (fast)
- TCP sockets (robust)

Fully-parallel, peer-to-peer

Where preCICE helps





Communication

Options:
- MPI ports (fast)
- TCP sockets (robust)

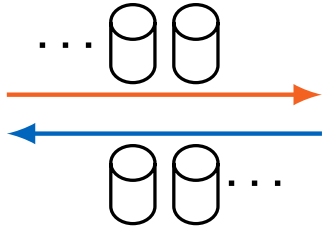
Fully-parallel, peer-to-peer

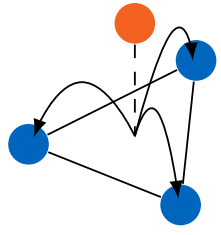
Data mapping

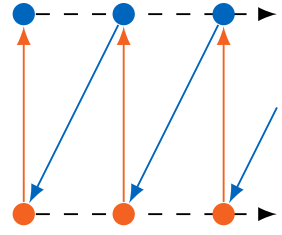
Options:
- radial-basis functions
- projection-based
- conservative/consistent
- direct mesh access

Compute on any side

Where preCICE helps







Communication

Options:
- MPI ports (fast)
- TCP sockets (robust)

Fully-parallel, peer-to-peer

Data mapping

Options:
- radial-basis functions
- projection-based
- conservative/consistent
- direct mesh access

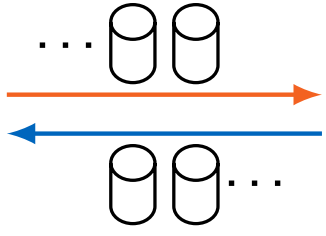
Compute on any side

Coupling schemes

Options:
- serial / parallel
- explicit / implicit
- compositional, multi
- IQN, Aitken, ...

Same high-level API
Configurable at runtime

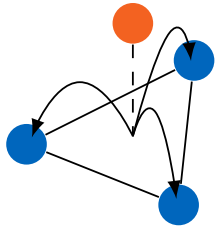
Where preCICE helps



Communication

Options:
- MPI ports (fast)
- TCP sockets (robust)

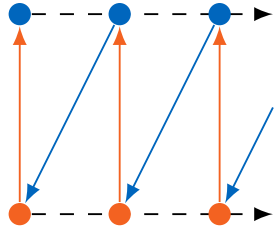
Fully-parallel, peer-to-peer



Data mapping

Options:
- radial-basis functions
- projection-based
- conservative/consistent
- direct mesh access

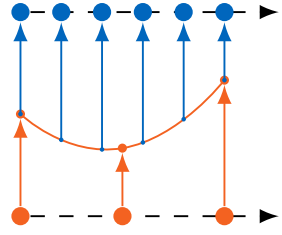
Compute on any side



Coupling schemes

Options:
- serial / parallel
- explicit / implicit
- compositional, multi
- IQN, Aitken, ...

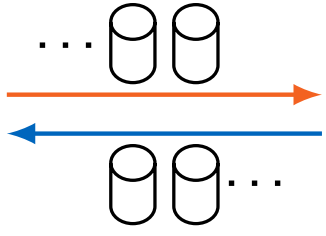
Same high-level API
Configurable at runtime



Time interpolation

Options:
- waveform iteration

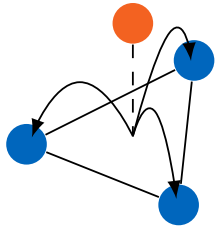
Where preCICE helps



Communication

Options:
- MPI ports (fast)
- TCP sockets (robust)

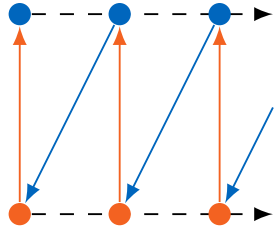
Fully-parallel, peer-to-peer



Data mapping

Options:
- radial-basis functions
- projection-based
- conservative/consistent
- direct mesh access

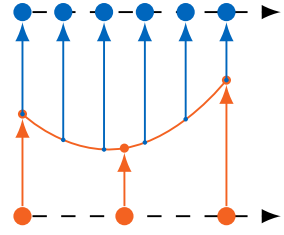
Compute on any side



Coupling schemes

Options:
- serial / parallel
- explicit / implicit
- compositional, multi
- IQN, Aitken, ...

Same high-level API
Configurable at runtime

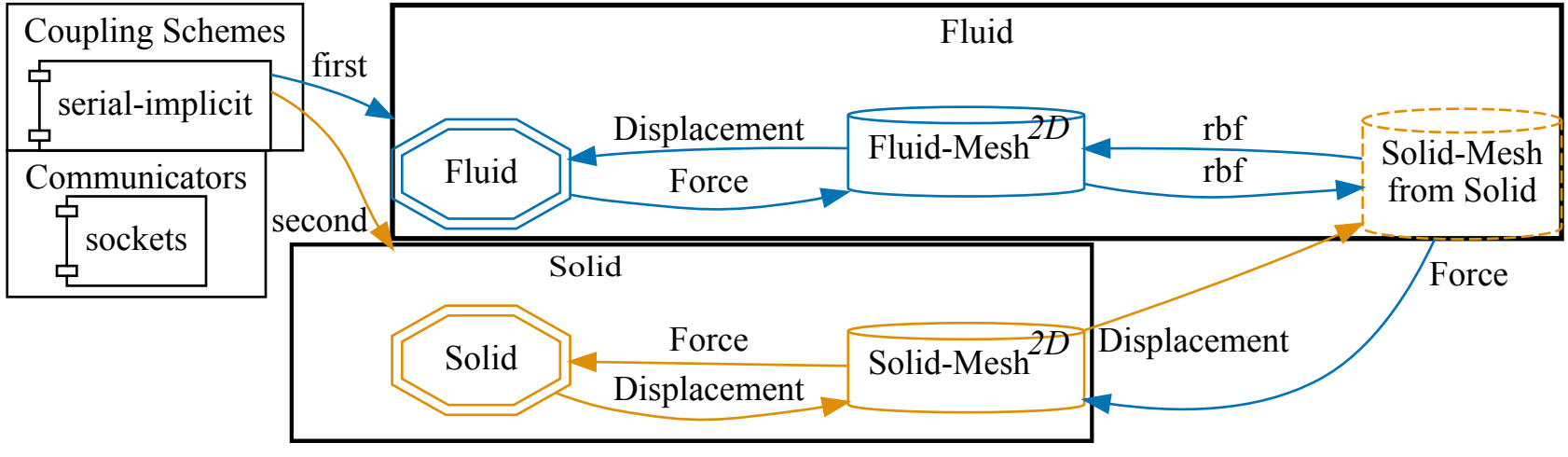


Time interpolation

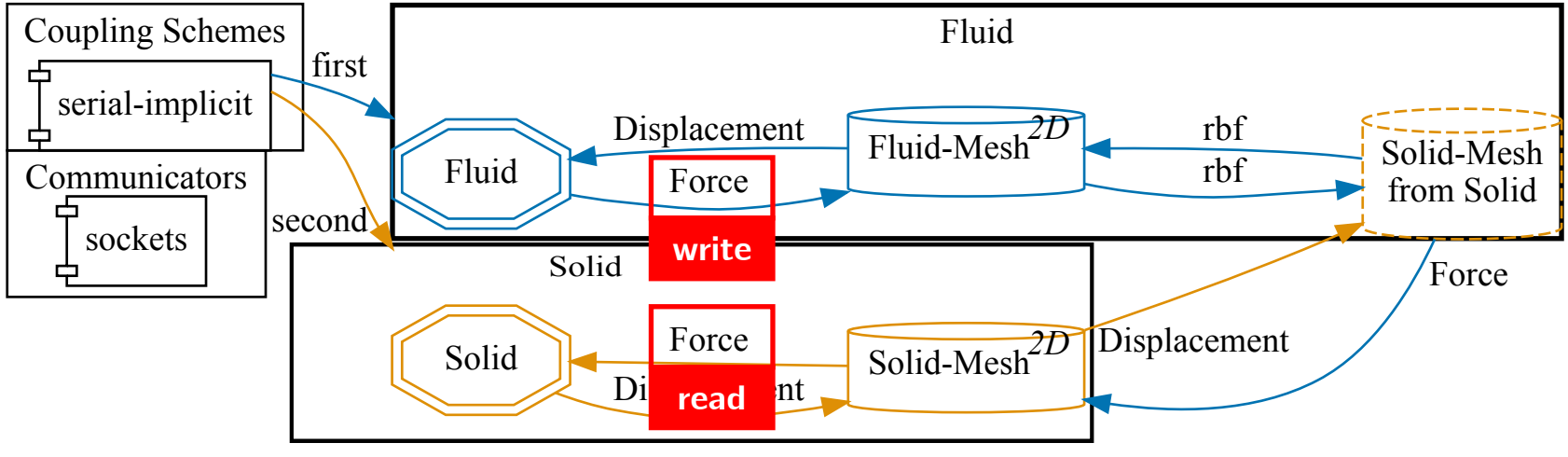
Options:
- waveform iteration

http://doi.org/
10/m9d8

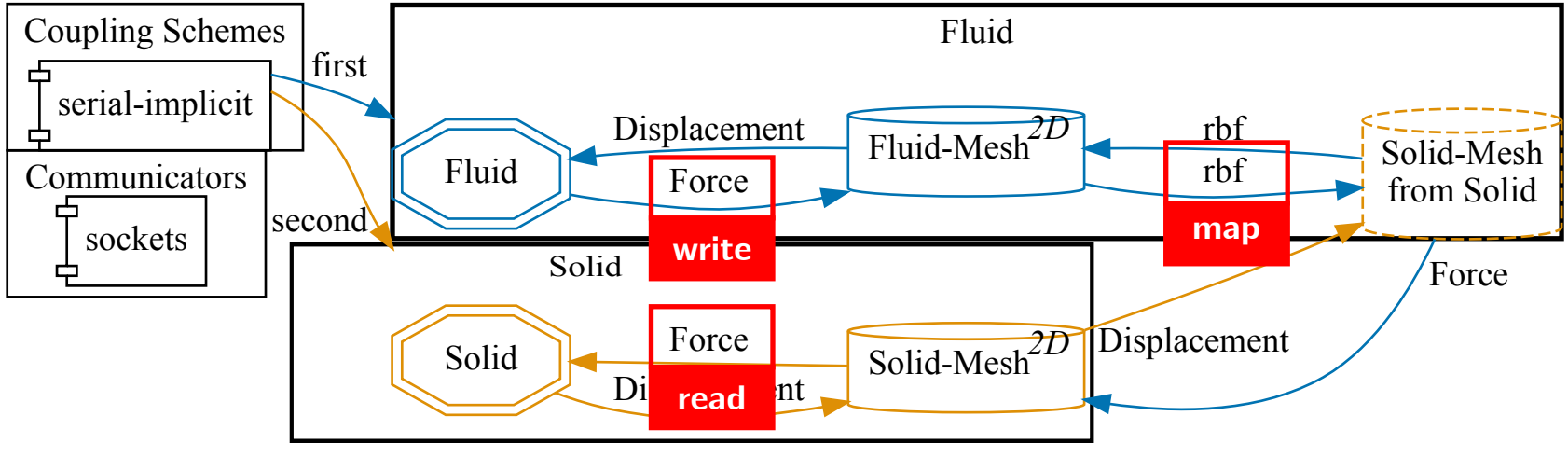
An FSI example setup



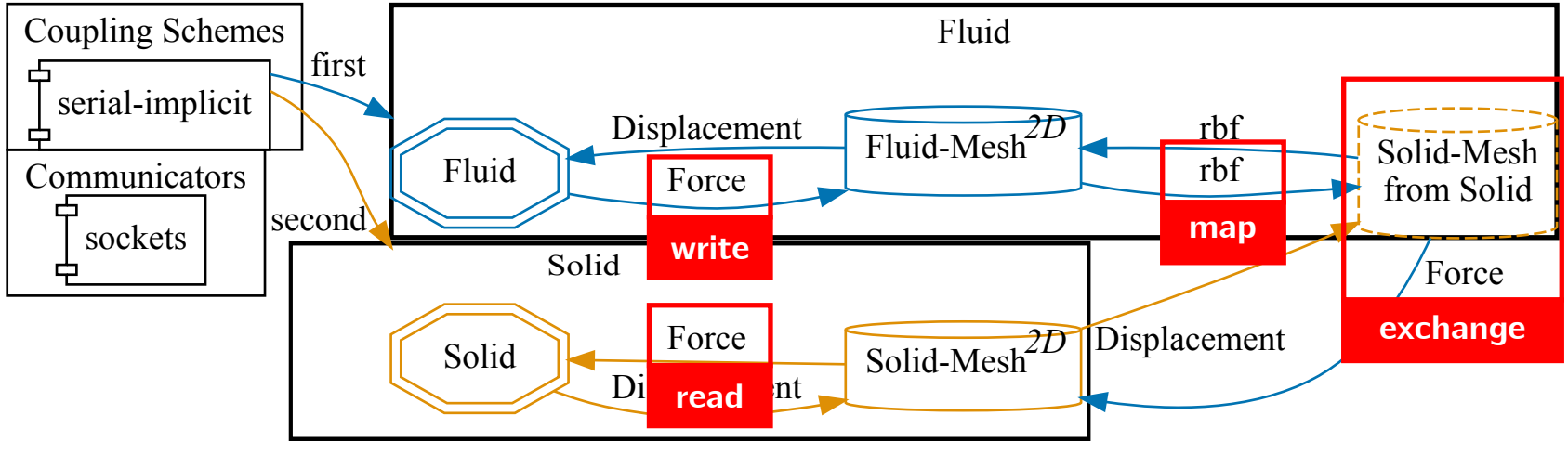
An FSI example setup



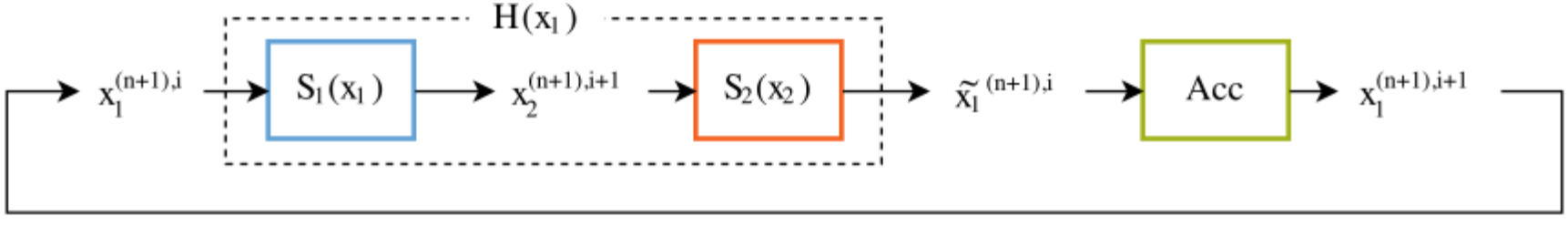
An FSI example setup



An FSI example setup



Implicit coupling algorithm (serial)



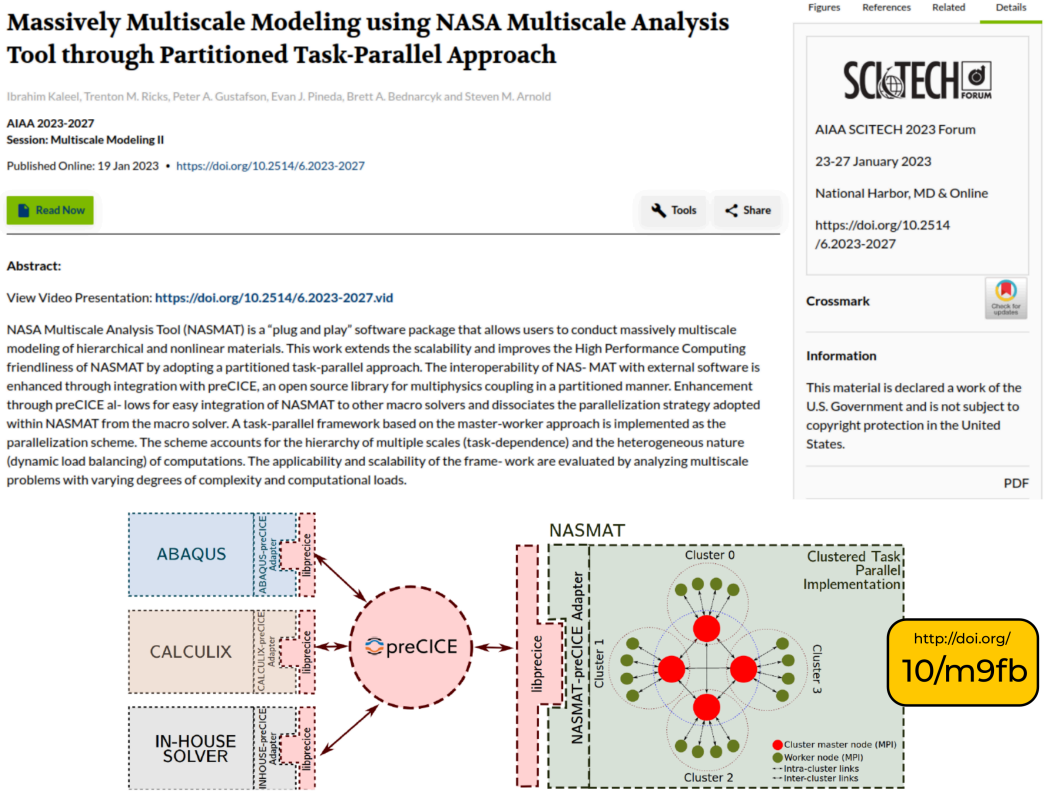
$$i \leftarrow i + 1, \quad S_1 \leftarrow S_1^{(n)}, \quad S_2 \leftarrow S_2^{(n)}$$

Acc: constant / Aitken under-relaxation, Anderson acceleration (IQN).
Also explicit, parallel-implicit, and multi-coupling schemes.

What people do with preCICE

From fluid-structure interaction to new directions

Application example: Material science



Massively Multiscale Modeling using NASA Multiscale Analysis Tool through Partitioned Task-Parallel Approach

Ibrahim Kaleel, Trenton M. Ricks, Peter A. Gustafson, Evan J. Pineda, Brett A. Bednarcyk and Steven M. Arnold

AIAA 2023-2027
Session: Multiscale Modeling II

Published Online: 19 Jan 2023 • https://doi.org/10.2514/6.2023-2027

Read Now | Tools | Share

Abstract:

View Video Presentation: **https://doi.org/10.2514/6.2023-2027.vid**

NASA Multiscale Analysis Tool (NASMAT) is a "plug and play" software package that allows users to conduct massively multiscale modeling of hierarchical and nonlinear materials. This work extends the scalability and improves the High Performance Computing friendliness of NASMAT by adopting a partitioned task-parallel approach. The interoperability of NAS- MAT with external software is enhanced through integration with preCICE, an open source library for multiphysics coupling in a partitioned manner. Enhancement through preCICE al- lows for easy integration of NASMAT to other macro solvers and dissociates the parallelization strategy adopted within NASMAT from the macro solver. A task-parallel framework based on the master-worker approach is implemented as the parallelization scheme. The scheme accounts for the hierarchy of multiple scales (task-dependence) and the heterogeneous nature (dynamic load balancing) of computations. The applicability and scalability of the frame- work are evaluated by analyzing multiscale problems with varying degrees of complexity and computational loads.

Figures | References | Related | Details

SCITECH FORUM

AIAA SCITECH 2023 Forum

23-27 January 2023

National Harbor, MD & Online

https://doi.org/10.2514/6.2023-2027

Crossmark

Information

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

PDF

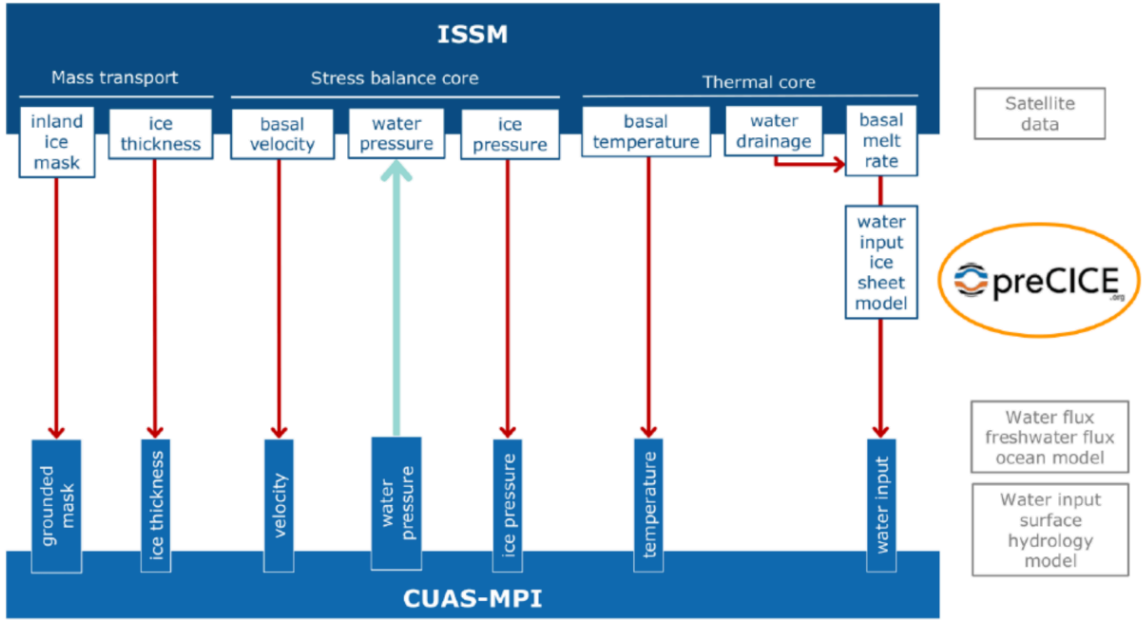
Hierarchical materials, NASA Glenn Research Center (USA)

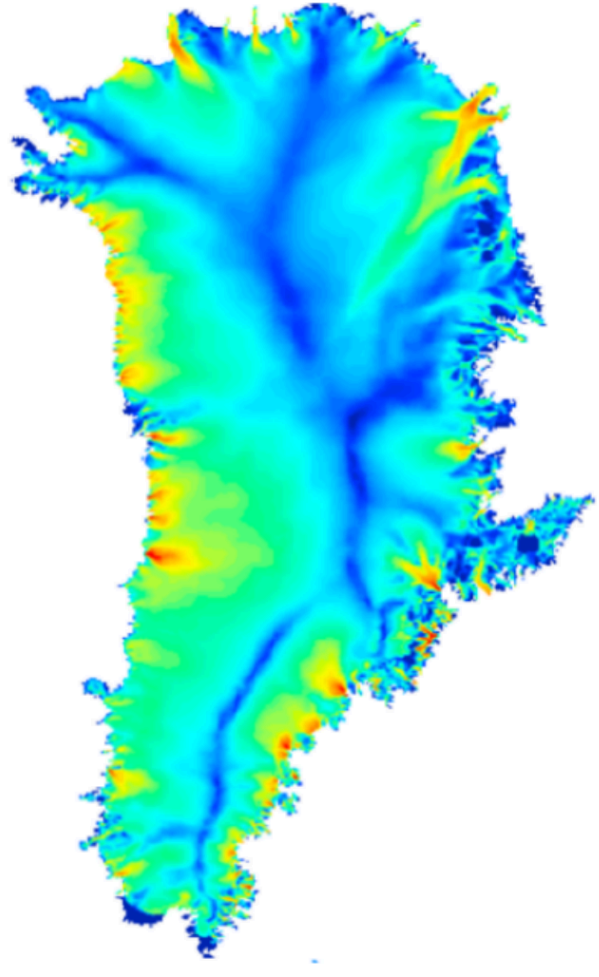
Application example: Glaciology

A preCICE-Interface for the Ice-sheet and Sea-level System Model (ISSM)

Yannic Fischler
yannic.fischler@tu-darmstadt.de
preCICE Workshop 2023

http://doi.org/10/m9fc





Ice-sheets and subglacial hydrology, AWI (Germany)

Application example: Flow control with ML

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

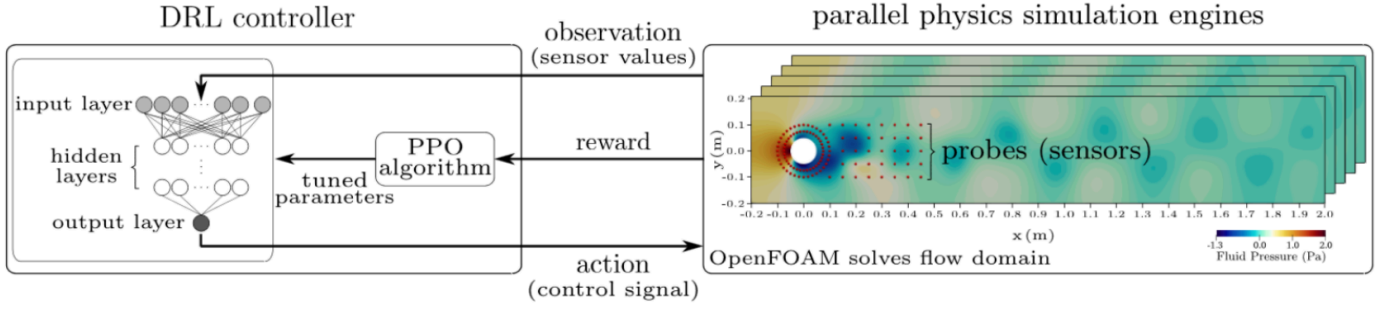
Gym-preCICE: Reinforcement learning environments for active flow control

Mosayeb Shams *, Ahmed H. Elsheikh

Heriot-Watt University, Edinburgh, United Kingdom

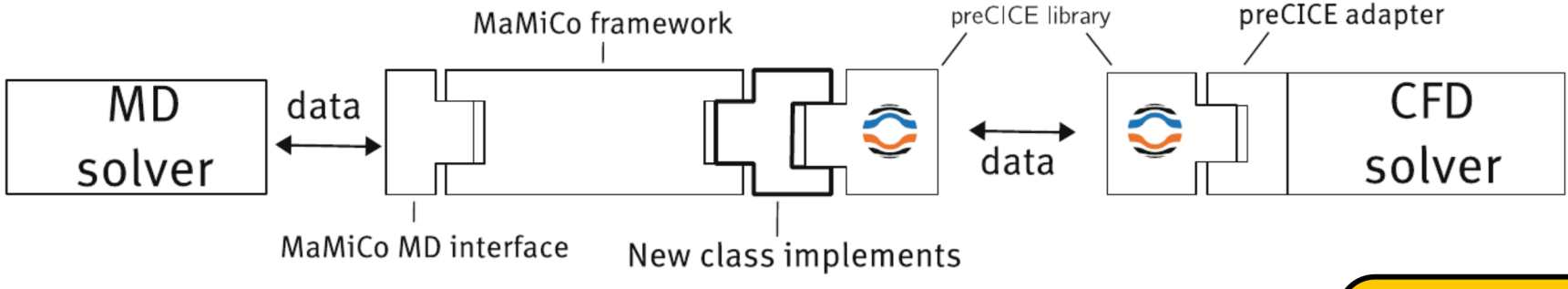
http://doi.org/

10/n3rd



Active flow control with ML, Heriot-Watt University (UK)

Application: Multi-scale flows (MD to CFD)



http://doi.org/
10/p7pk

Further applications

- Porous media and free flow coupling
- Blood flow simulation (e.g., heart valves)
- Nuclear reactor thermohydraulics (1D/3D)
- Metal casting process
- Continuous fiber-reinforced plastics manufacturing
- Muscular systems (e.g., amputation)

https://precice.org/community-projects.html

Using the preCICE library

Coupling two toy Python solvers

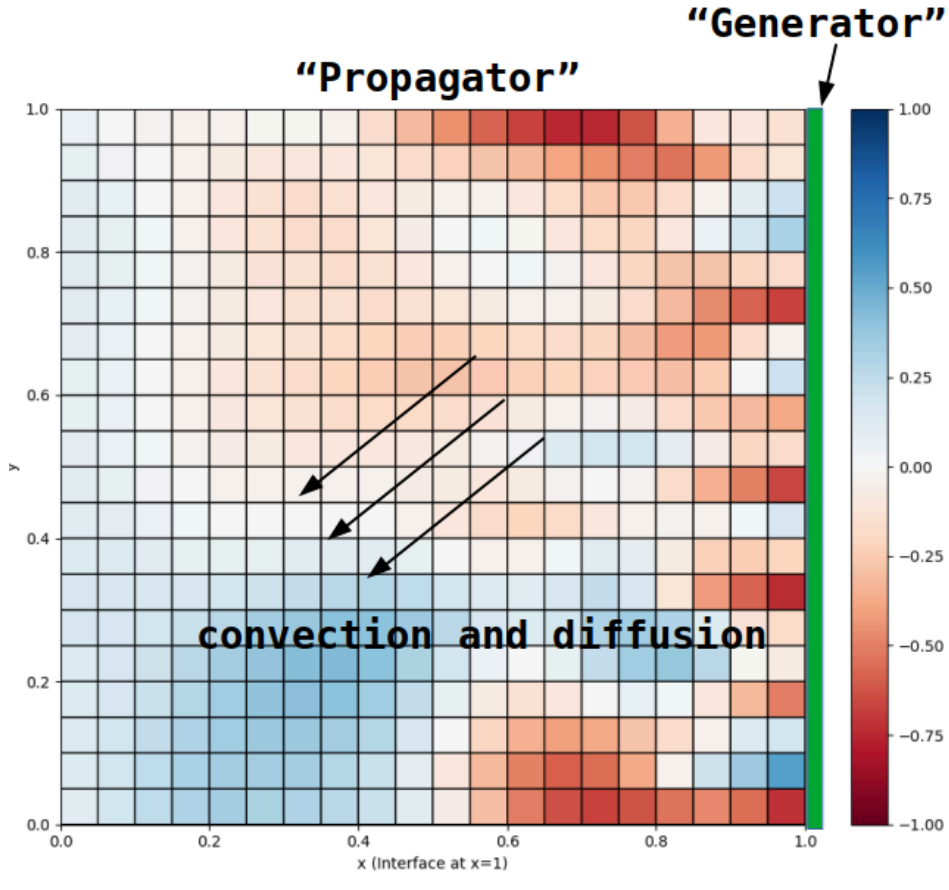
Dependencies

- preCICE v3 (e.g. packages for Ubuntu)
- Python 3
- Python packages `numpy`, `matplotlib`
- preCICE Python bindings:

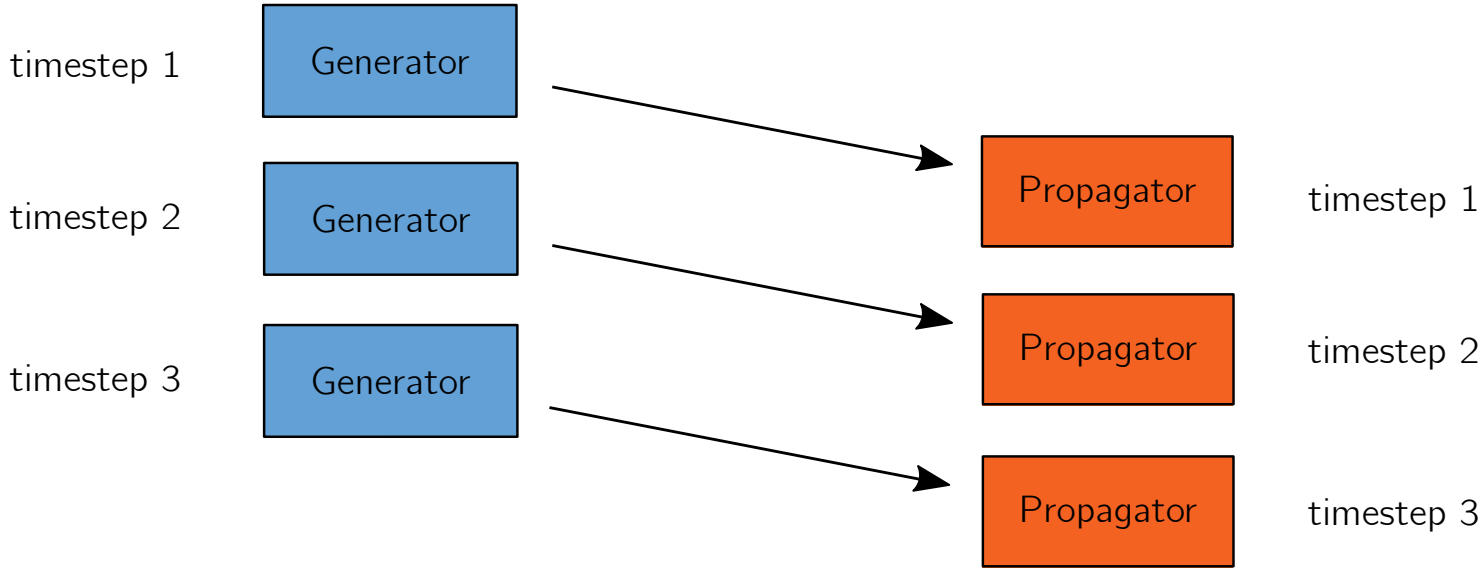
```
pip3 install --upgrade pip
python3 -m venv .venv && . .venv/bin/activate
pip3 install pyprecice
```

"Generator" and "Propagator"

- `generator.py`: generates random data in a 1D domain
- `propagator.py`: propagates boundary data over a 2D domain



Unidirectional coupling

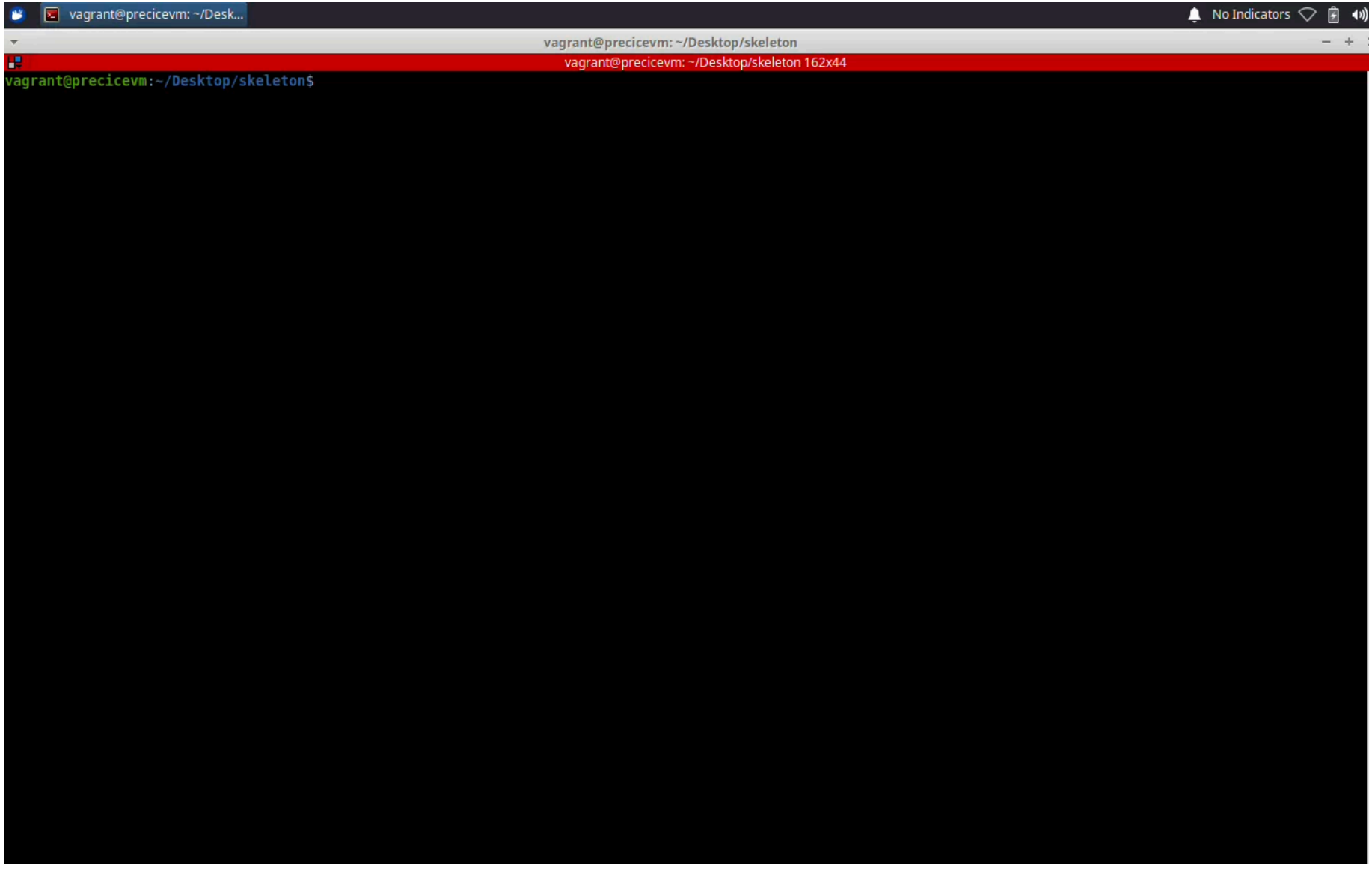


generator.py

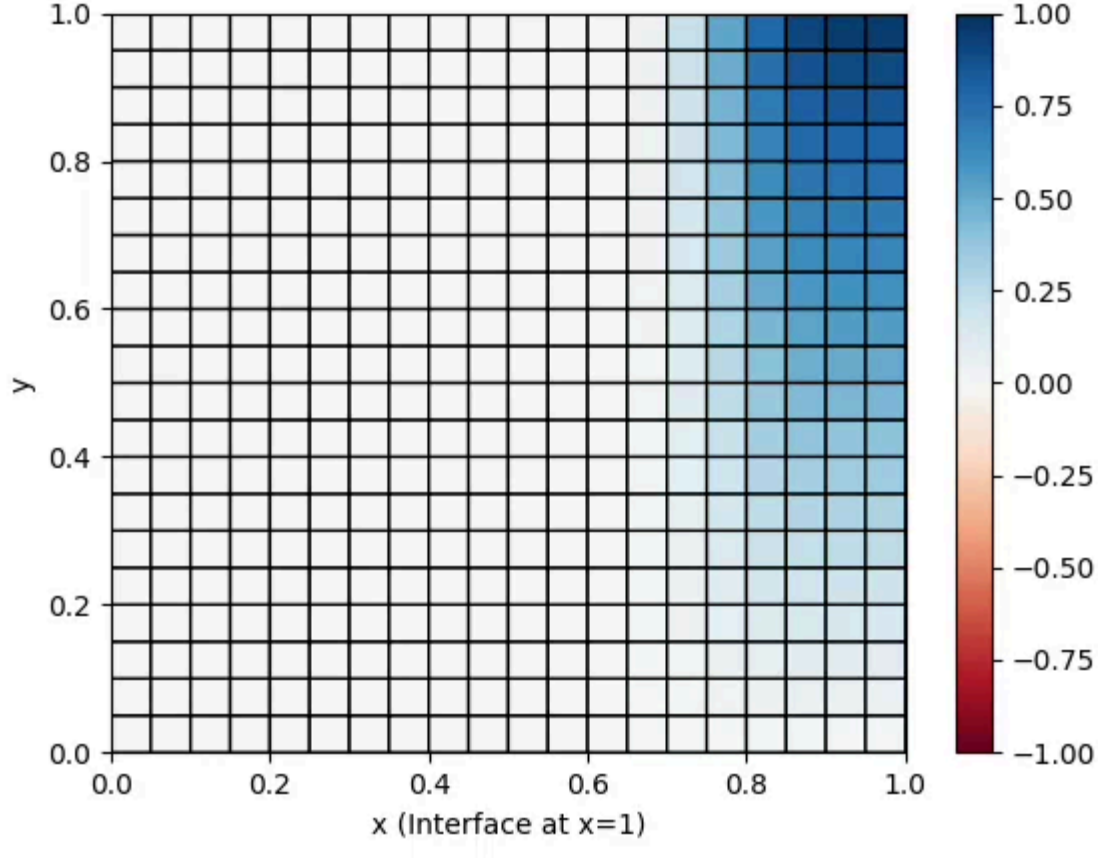
```
import numpy

# generate mesh
n = 20
y = numpy.linspace(0, 1, n + 1)

dt = 0.01
t = 0

while True:
  print("Generating data")
  u = 1 - 2 * numpy.random.rand(n)

  t = t + dt
  if(t > 0.1):
    break
```

propagator.py

```
# generate mesh
n = 20
x = numpy.linspace(0, 1, n+1)
y = numpy.linspace(0, 1, n+1)

# initial data, associated to cell centers
u = numpy.zeros([n, n])

dt = 0.01
t = 0

# boundary condition for u (arbitrary)
u[:,-1] = y[:-1]

while True:

  print("Propagating data")
```


vagrant@precicevm: ~/Desk...
No Indicators
vagrant@precicevm: ~/Desktop/skeleton
vagrant@precicevm: ~/Desktop/skeleton 162x44
vagrant@precicevm:~/Desktop/skeleton$

Uncoupled simulation



Import preCICE

```python
import numpy


# generate mesh
n = 20
y = numpy.linspace(0, 1, n + 1)

dt = 0.01
t = 0

while True:
  print("Generating data")
  u = 1 - 2 * numpy.random.rand(n)

  t = t + dt
  if(t > 0.1):
    break
```

Import preCICE

```python
import numpy
import precice

# generate mesh
n = 20
y = numpy.linspace(0, 1, n + 1)

dt = 0.01
t = 0

while True:
  print("Generating data")
  u = 1 - 2 * numpy.random.rand(n)

  t = t + dt
  if(t > 0.1):
    break
```

Configure preCICE

```python
import numpy
import precice

# generate mesh
n = 20
y = numpy.linspace(0, 1, n + 1)

# preCICE setup
participant_name     = "Generator"
config_file_name     = "../precice-config.xml"
solver_process_index = 0
solver_process_size  = 1
participant =
    precice.Participant(
                        participant_name,
                        config_file_name,
                        solver_process_index,
```

Configure preCICE

```
 5 n = 20
 6 y = numpy.linspace(0, 1, n + 1)
 7
 8 # preCICE setup
 9 participant_name     = "Generator"
10 config_file_name     = "../precice-config.xml"
11 solver_process_index = 0
12 solver_process_size  = 1
13 participant =
14     precice.Participant(
15                         participant_name,
16                         config_file_name,
17                         solver_process_index,
18                         solver_process_size
19                        )
20
21 dt = 0.01
22 t = 0
```

Define the coupling mesh

```python
import numpy
import precice

# generate mesh
n = 20
y = numpy.linspace(0, 1, n + 1)

# preCICE setup
participant_name     = "Generator"
config_file_name     = "../precice-config.xml"
solver_process_index = 0
solver_process_size  = 1
participant =
    precice.Participant(
                        participant_name,
                        config_file_name,
                        solver_process_index,
```

Define the coupling mesh

```
16                          config_file_name,
17                          solver_process_index,
18                          solver_process_size
19                        )
20
21 # Define the coupling mesh name
22 mesh_name  = "Generator-Mesh"
23
24 # Define the coupling mesh
25 positions = [[1, y0] for y0 in y[:-1]]
26 vertex_ids = participant.set_mesh_vertices(mesh_name, positions)
27
28 dt = 0.01
29 t = 0
30
31 while True:
32   print("Generating data")
```

Initialize and finalize preCICE

```python
import numpy
import precice

# generate mesh
n = 20
y = numpy.linspace(0, 1, n + 1)

# preCICE setup
participant_name     = "Generator"
config_file_name     = "../precice-config.xml"
solver_process_index = 0
solver_process_size  = 1
participant =
    precice.Participant(
                        participant_name,
                        config_file_name,
                        solver_process_index,
```

Initialize and finalize preCICE

```
20
21 # Define the coupling mesh name
22 mesh_name = "Generator-Mesh"
23
24 # Define the coupling mesh
25 positions  = [[1, y0] for y0 in y[:-1]]
26 vertex_ids = participant.set_mesh_vertices(mesh_name, positions)
27
28 participant.initialize()
29
30 dt = 0.01
31 t = 0
32
33 while True:
34   print("Generating data")
35   u = 1 - 2 * numpy.random.rand(n)
36
```

Advance the coupling

```python
import numpy
import precice

# generate mesh
n = 20
y = numpy.linspace(0, 1, n + 1)

# preCICE setup
participant_name     = "Generator"
config_file_name     = "../precice-config.xml"
solver_process_index = 0
solver_process_size  = 1
participant =
    precice.Participant(
                        participant_name,
                        config_file_name,
                        solver_process_index,
```

Advance the coupling

```
27
28 participant.initialize()
29
30 dt = 0.01
31 t = 0
32
33 while True:
34   print("Generating data")
35   u = 1 - 2 * numpy.random.rand(n)
36
37
38
39   t = t + dt
40   if(t > 0.1):
41     break
42
43 participant.finalize()
```

Advance the coupling

```python
import numpy
import precice

# generate mesh
n = 20
y = numpy.linspace(0, 1, n + 1)

# preCICE setup
participant_name     = "Generator"
config_file_name     = "../precice-config.xml"
solver_process_index = 0
solver_process_size  = 1
participant =
    precice.Participant(
                        participant_name,
                        config_file_name,
                        solver_process_index,
```

Advance the coupling

```python
import numpy
import precice

# generate mesh
n = 20
y = numpy.linspace(0, 1, n + 1)

# preCICE setup
participant_name     = "Generator"
config_file_name     = "../precice-config.xml"
solver_process_index = 0
solver_process_size  = 1
participant =
    precice.Participant(
                        participant_name,
                        config_file_name,
                        solver_process_index,
```

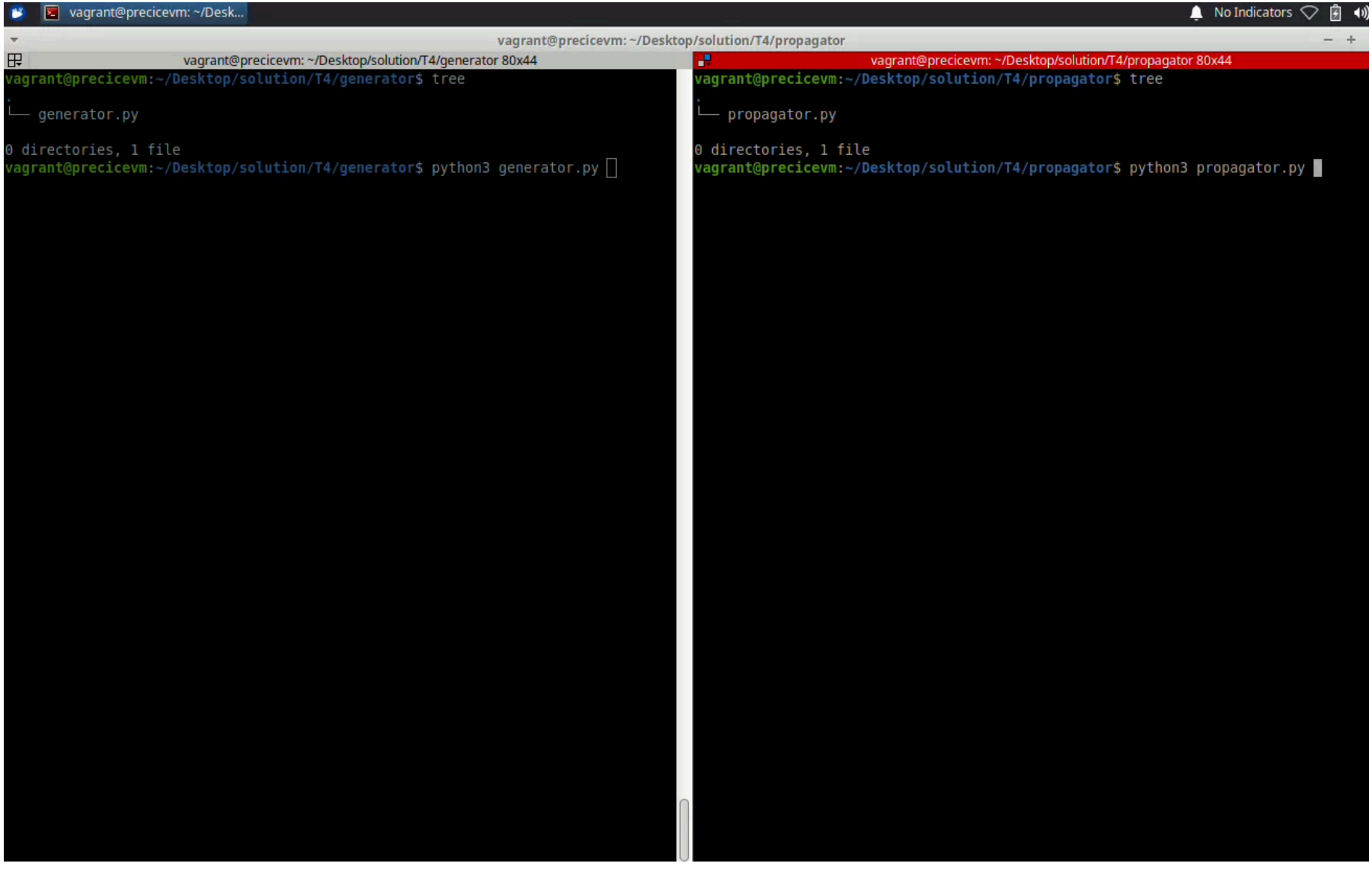
Advance the coupling

```python
import numpy
import precice

# generate mesh
n = 20
y = numpy.linspace(0, 1, n + 1)

# preCICE setup
participant_name     = "Generator"
config_file_name     = "../precice-config.xml"
solver_process_index = 0
solver_process_size  = 1
participant =
    precice.Participant(
                        participant_name,
                        config_file_name,
                        solver_process_index,
```

Not there yet, but let's run it

(similar changes in propagator.py - try it at home)

Nothing happening?



Write & read data

```python
import numpy
import precice

# generate mesh
n = 20
y = numpy.linspace(0, 1, n + 1)

# preCICE setup
participant_name     = "Generator"
config_file_name     = "../precice-config.xml"
solver_process_index = 0
solver_process_size  = 1
participant =
    precice.Participant(
                        participant_name,
                        config_file_name,
                        solver_process_index,
```

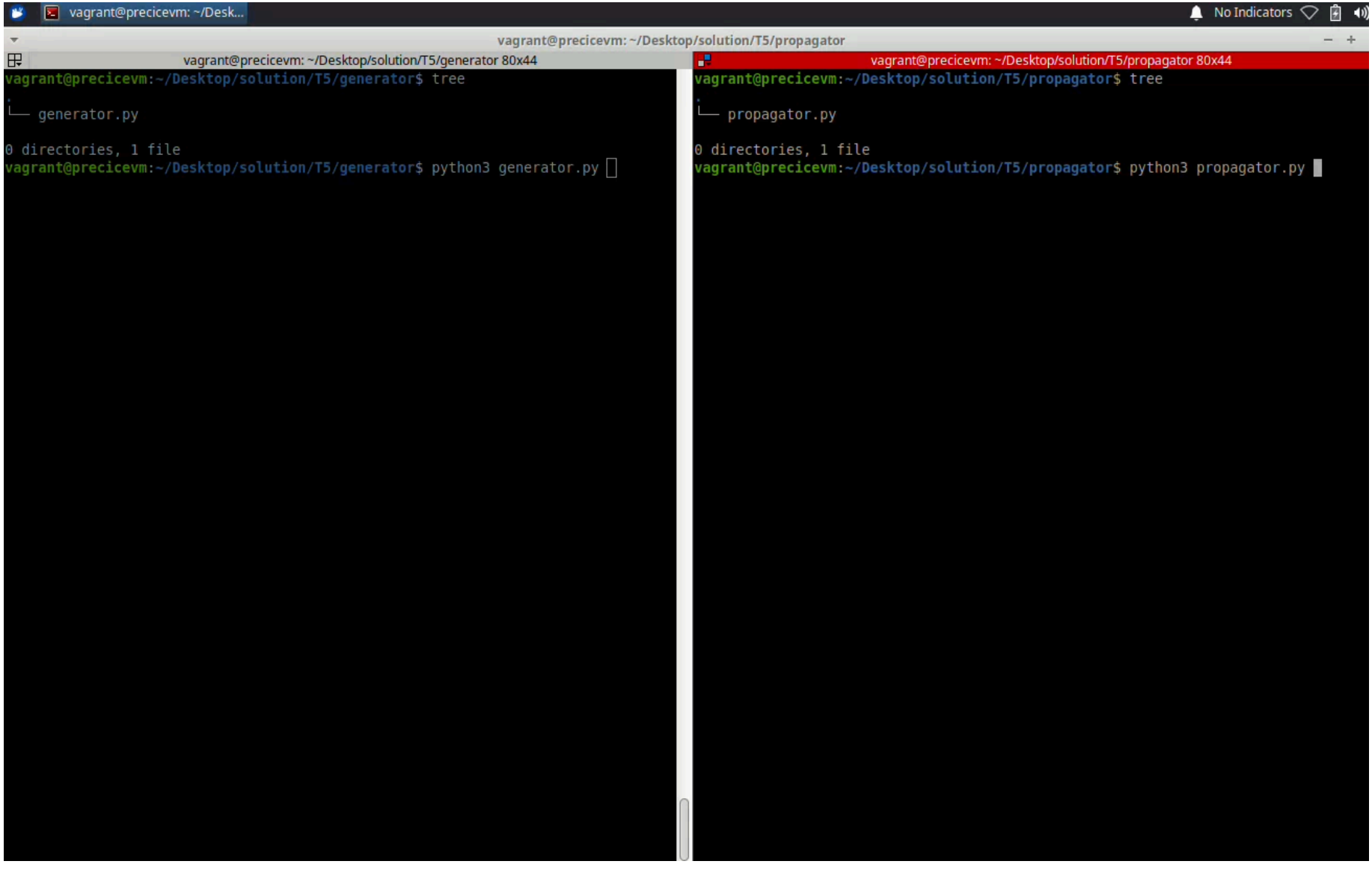
Write & read data

```python
import numpy
import precice

# generate mesh
n = 20
y = numpy.linspace(0, 1, n + 1)

# preCICE setup
participant_name     = "Generator"
config_file_name     = "../precice-config.xml"
solver_process_index = 0
solver_process_size  = 1
participant =
    precice.Participant(
                        participant_name,
                        config_file_name,
                        solver_process_index,
```

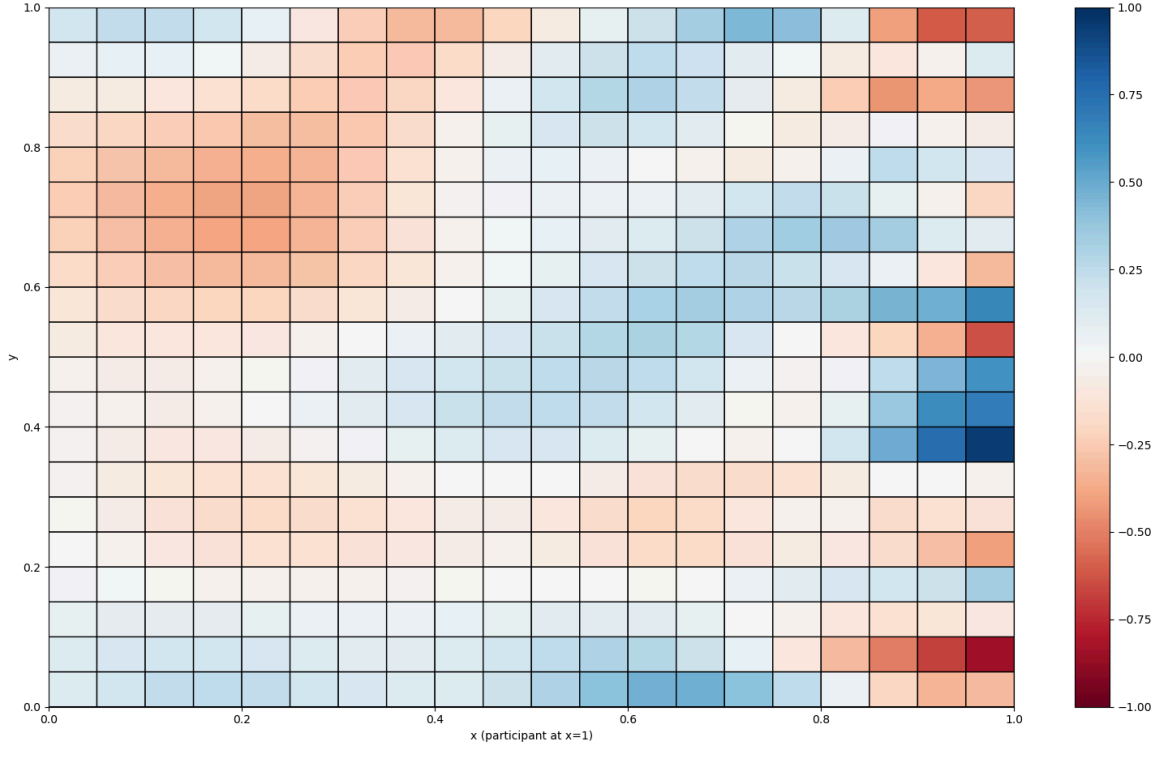
Write & read data

```
32
33 dt = 0.01
34 t = 0
35
36 while participant.is_coupling_ongoing():
37   dt = np.minimum(dt, precice_dt)
38
39   print("Generating data")
40   u = 1 - 2 * numpy.random.rand(n)
41
42   participant.write_data(mesh_name, data_name, vertex_ids, u)
43   precice_dt = participant.advance(dt)
44   # u[:, -1] = participant.read_data(mesh_name, data_name, vertex_ids
45
46   t = t + dt
47
48 participant.finalize()
```

It should work now!

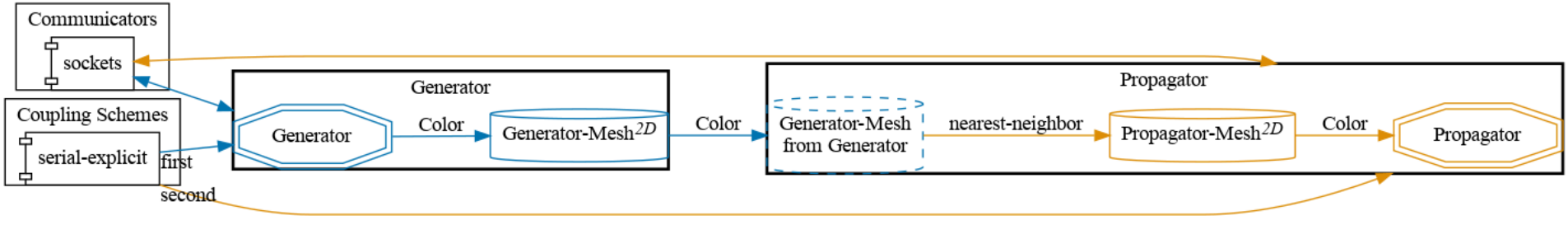

vagrant@precicevm: ~/Desk...
No Indicators
vagrant@precicevm: ~/Desktop/solution/T5/propagator
vagrant@precicevm: ~/Desktop/solution/T5/generator 80x44
vagrant@precicevm:~/Desktop/solution/T5/generator$ tree
.
└── generator.py
0 directories, 1 file
vagrant@precicevm:~/Desktop/solution/T5/generator$ python3 generator.py
vagrant@precicevm: ~/Desktop/solution/T5/propagator 80x44
vagrant@precicevm:~/Desktop/solution/T5/propagator$ tree
.
└── propagator.py
0 directories, 1 file
vagrant@precicevm:~/Desktop/solution/T5/propagator$ python3 propagator.py

Data is being transfered!



Most basic case: uni-directional, serial-explicit, nearest-neighbor mapping, ...

preCICE configuration



Visual representation of precice-config.xml using the config visualizer.

Research and development

University of Stuttgart
Germany

Technical University of Munich


Prof. Miriam Schulte


Jun.-Prof. Benjamin Uekermann


Benjamin Rodenberg


Felix Neubauer


Gerasimos Chourdakis

preCICE contributors + vibrant and rapidly growing community of


Jun Chen

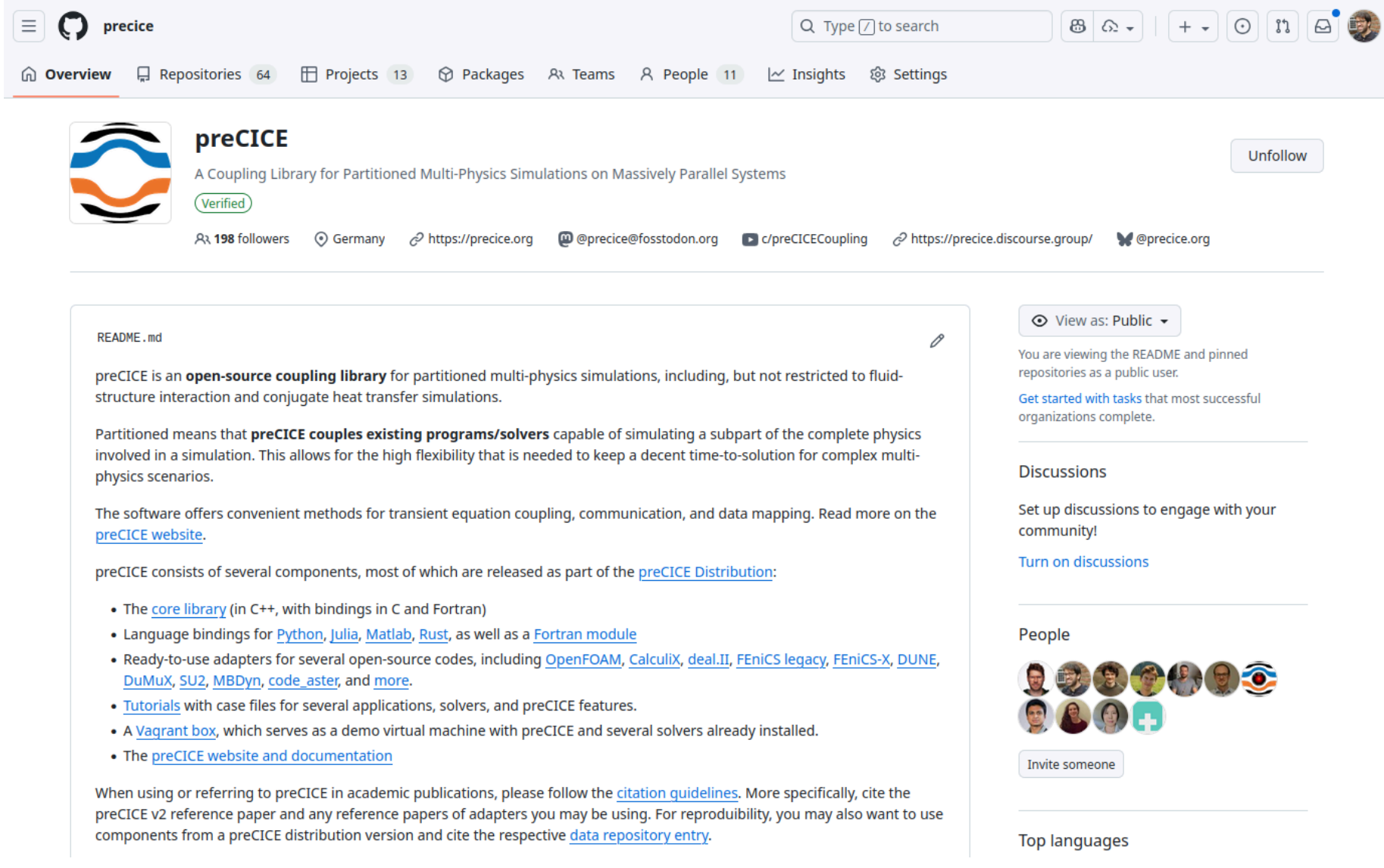

Frédéric Simonis


Carme Homs Pons

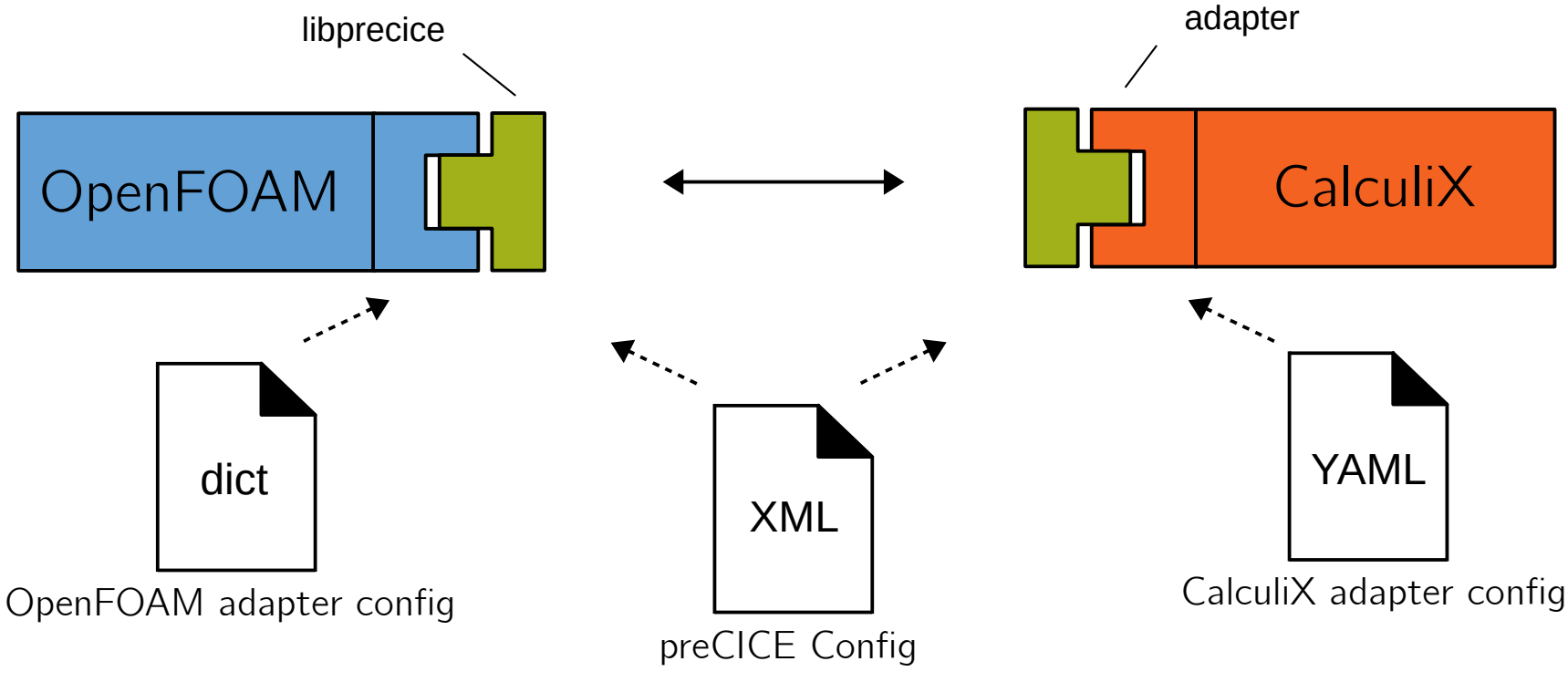

Ishaan Desai


David Schneider

Open-source development on GitHub



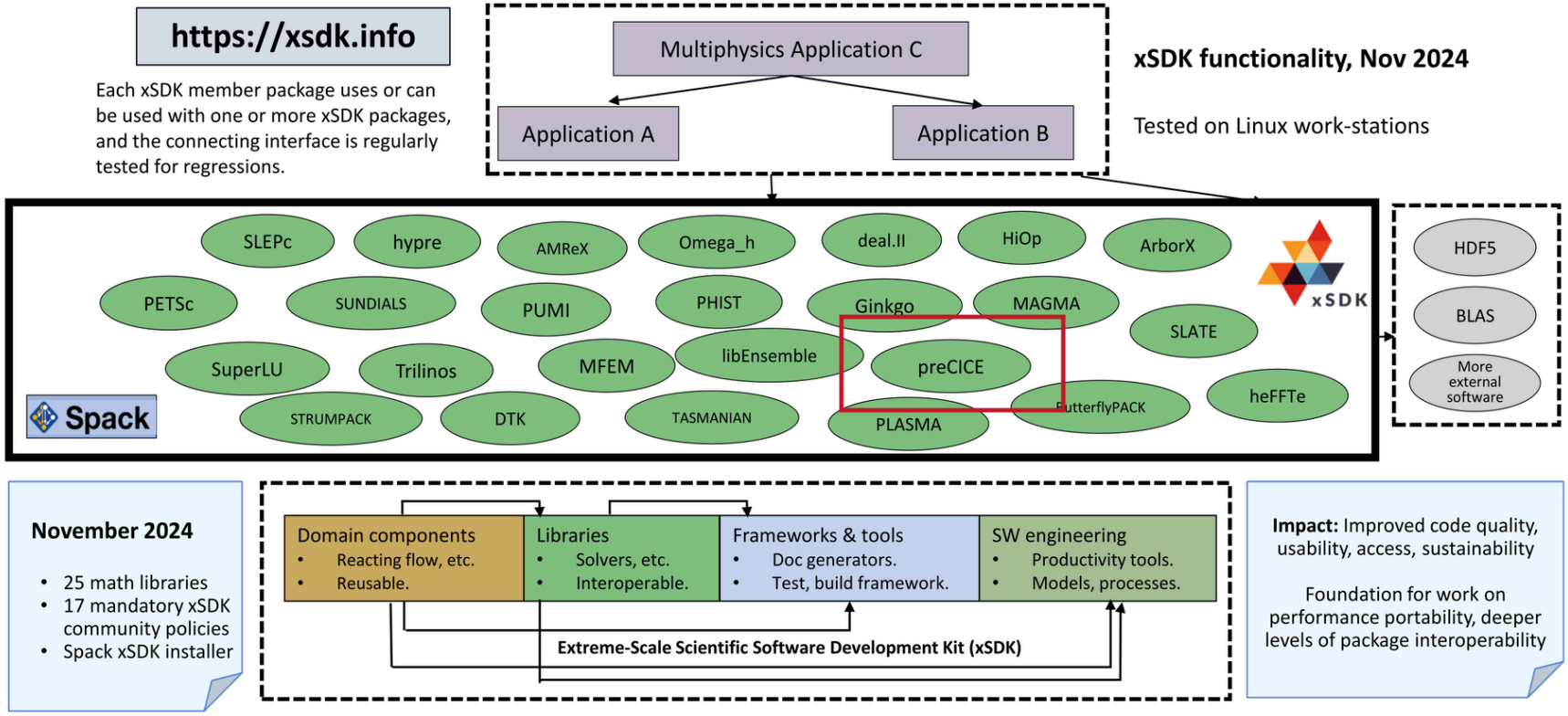
Defining community standards



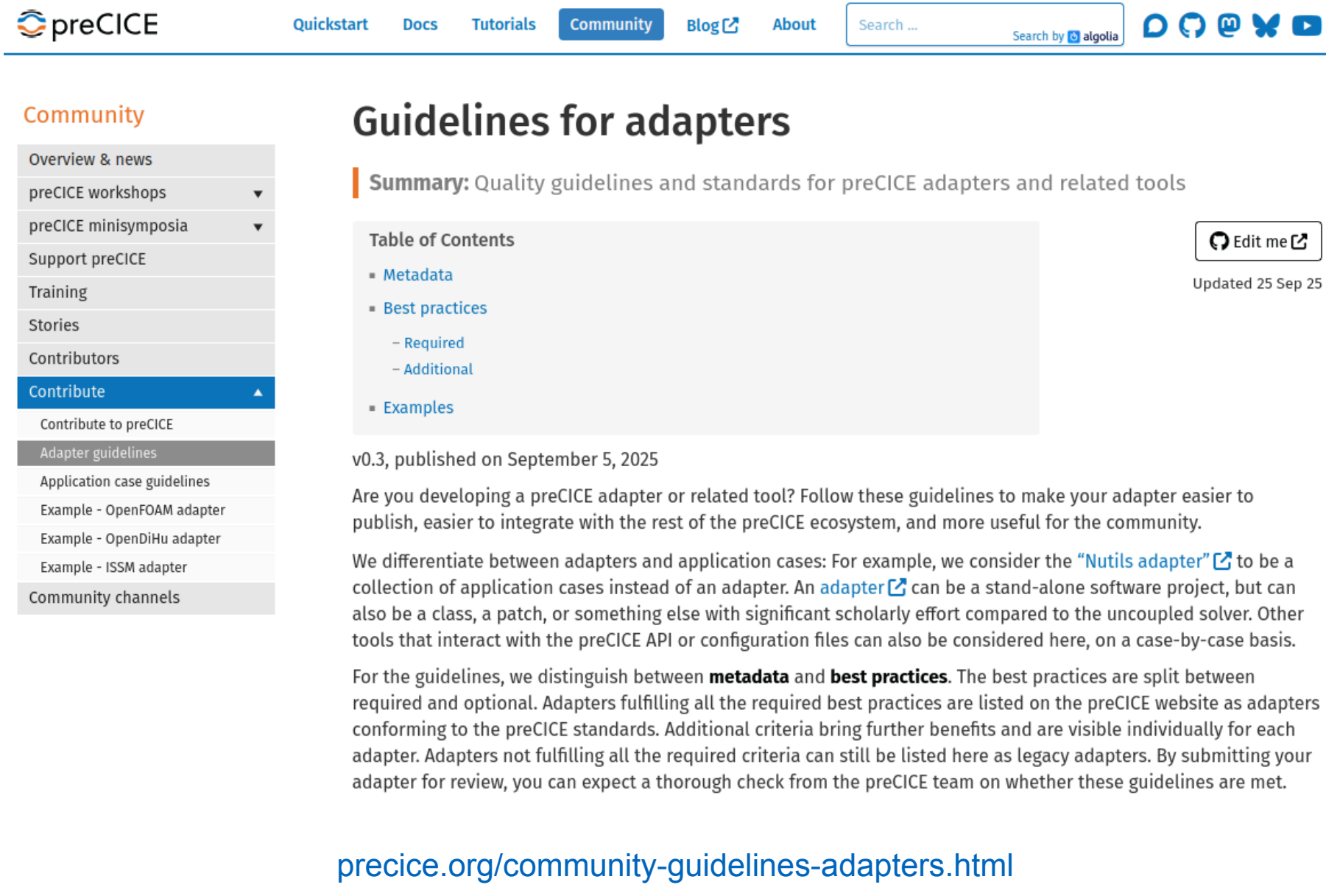
Overview of an FSI setup: How easy is it to replace CalculiX by FEniCS?

xSDK Version 1.1.0: November 2024

https://xsdk.info

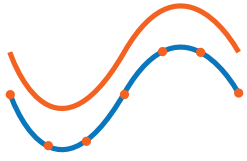


We can learn from the Extreme-Scale Development Toolkit: xsdk.info/packages/



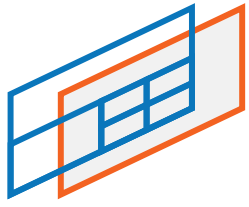
precice.org/community-guidelines-adapters.html

Current research



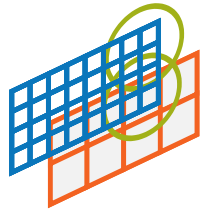
time interpolation

Benjamin Rodenberg



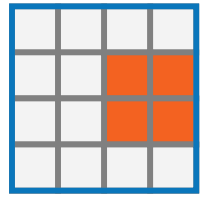
dynamic meshes

Frédéric Simonis



data mapping efficiency

David Schneider

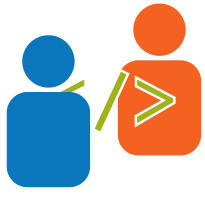


adaptive multi-scale

Ishaan Desai

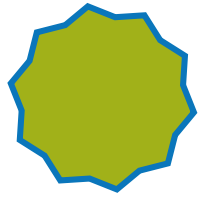
ecosystem

Gerasimos Chourdakis



configuration automation

Felix Neubauer



application: muscles

Carme Homs Pons

application: porous media

Jun Chen

Further research and development

- Coupling with Functional Mock-Up units
- Non-mesh-related global data exchange
- Volume coupling (overlapping domains)
- Coupling with particle codes
- Automatic acceleration configuration

©HSU MZ Christian W. Gelhausen

Resources

Start here: `precice.org`

preCICE Quickstart Docs Tutorials Community Blog About Search ... Search by algolia

Welcome to preCICE

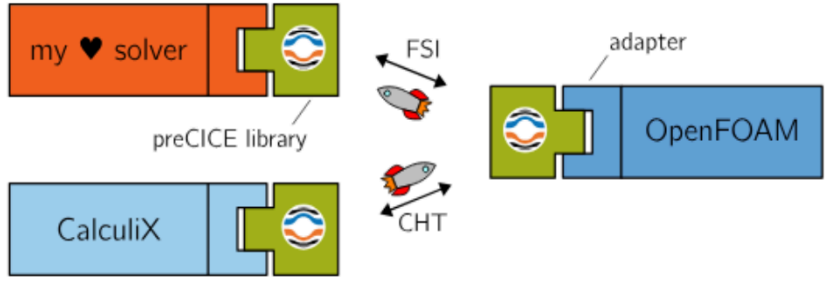
The coupling library for partitioned multi-physics simulations.

Star on Github 831 | Latest (Apr 1, 2025) | Get started

preCICE is an **open-source coupling library** for partitioned multi-physics simulations, including, but not restricted to fluid-structure interaction and conjugate heat transfer simulations.

Partitioned means that **preCICE couples existing programs/solvers** capable of simulating a subpart of the complete physics involved in a simulation. This allows for the high flexibility that is needed to keep a decent time-to-solution for complex multi-physics scenarios.

The software offers convenient methods for transient equation coupling, communication, and data mapping.



Everything is open and user-centered

- Developed in the open: github.com/precice
- Extensive documentation
- (Diamond) open-access publications
- Public recordings of talks

Extra effort to maintain and update

Further channels

precice.discourse.group

preCICECoupling

groups/9073912/

project/preCICE

fosstodon.org/@precicerecice

@precice.org

preCICE is free because of

Research Software
Sustainability

EXC 2075
SimTech

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 754462

and the code/issues/testing/documentation contributions of people like you (thank you!).

Summary

1. preCICE offers efficient algorithms for black-box coupling
2. New directions beyond FSI: looking for collaborations
3. Open source, open science, community

Gerasimos Chourdakis (US/TUM) + many more (see precice.org/about)