

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

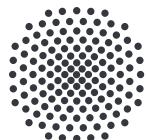
# Black-box coupling of simulation codes using preCICE

Gerasimos Chourdakis

Institute for Parallel and Distributed Systems

University of Stuttgart

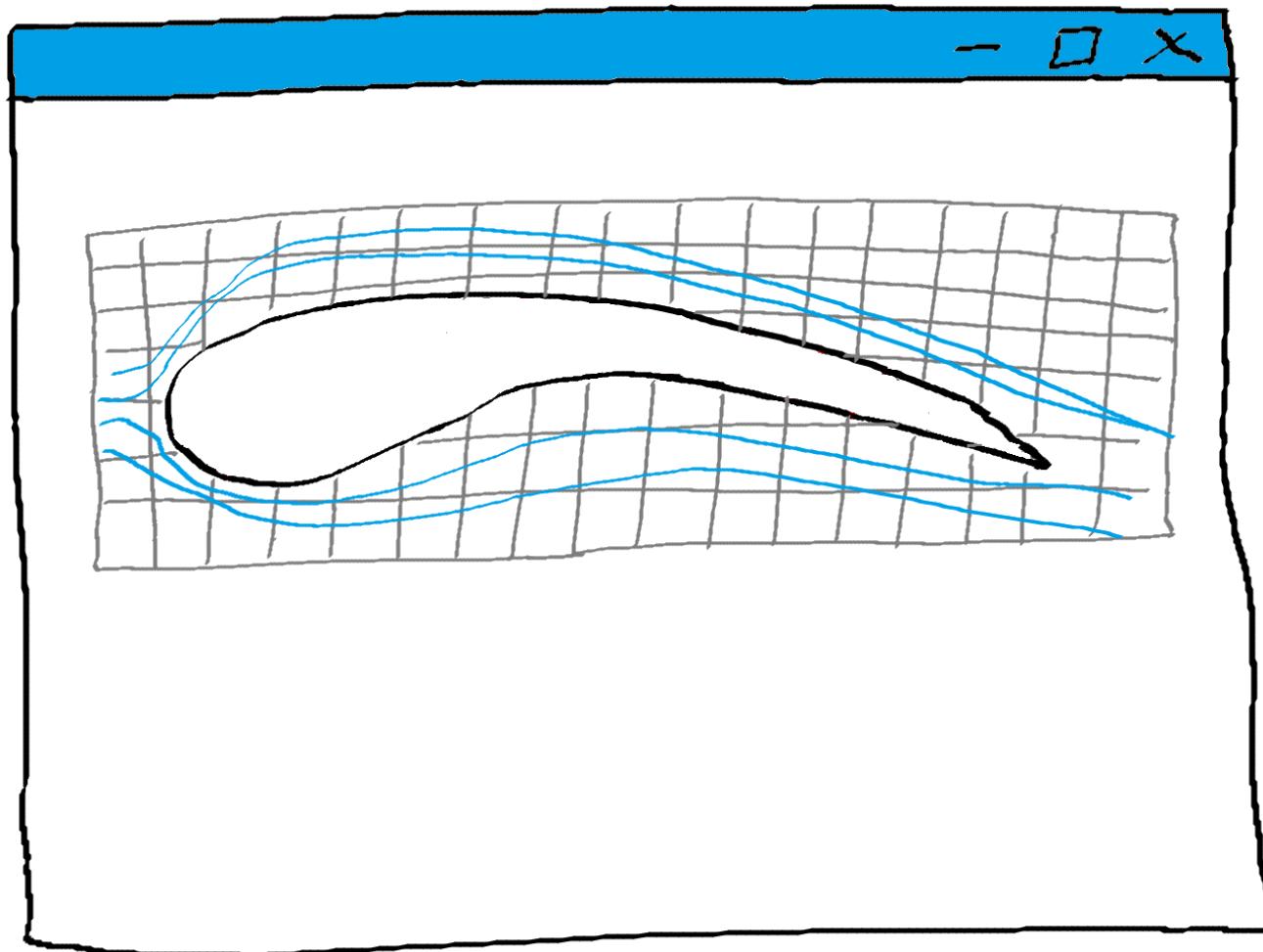
[gerasimos.chourdakis@ipvs.uni-stuttgart.de](mailto:gerasimos.chourdakis@ipvs.uni-stuttgart.de)



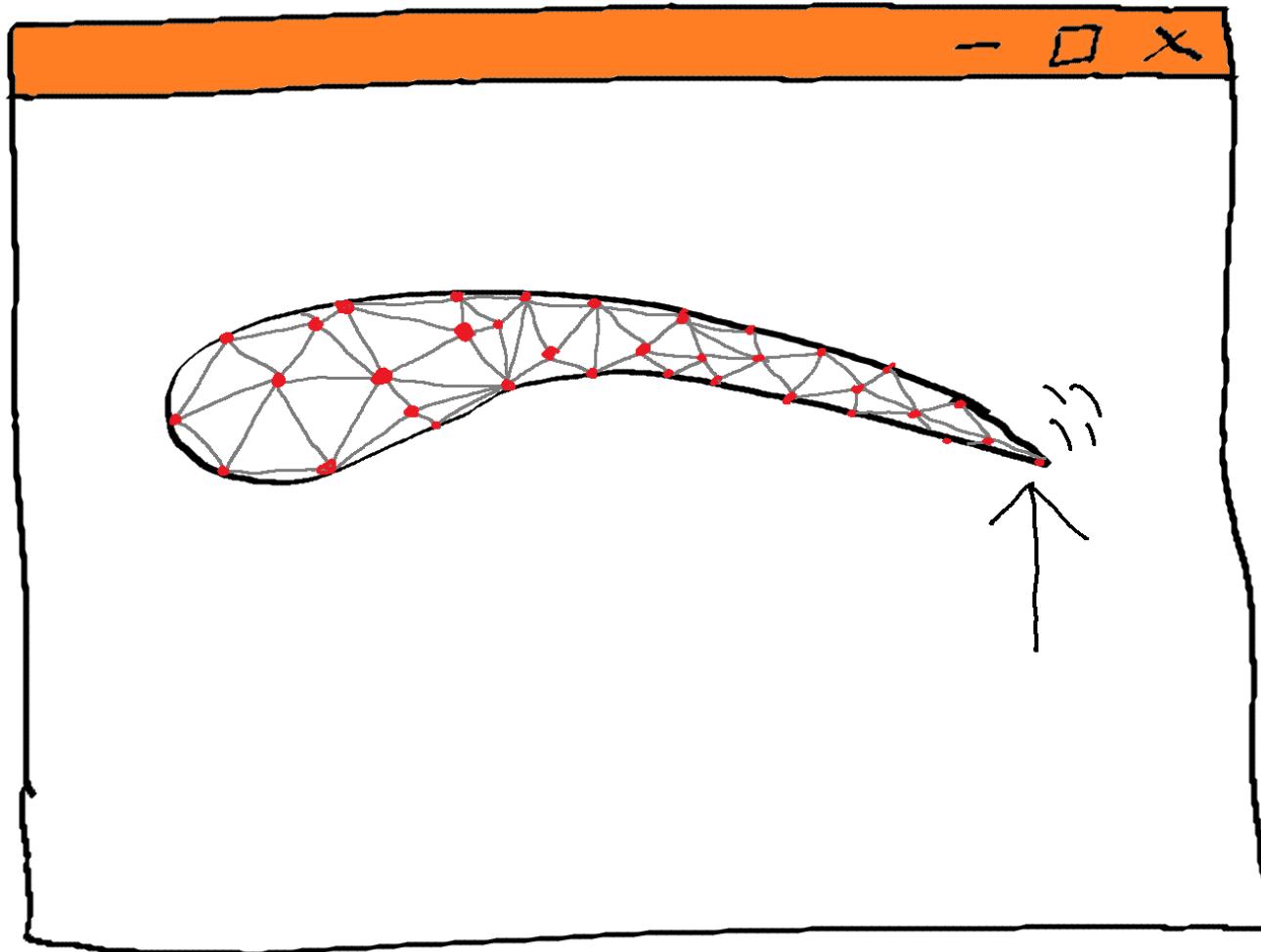
University of Stuttgart  
Germany



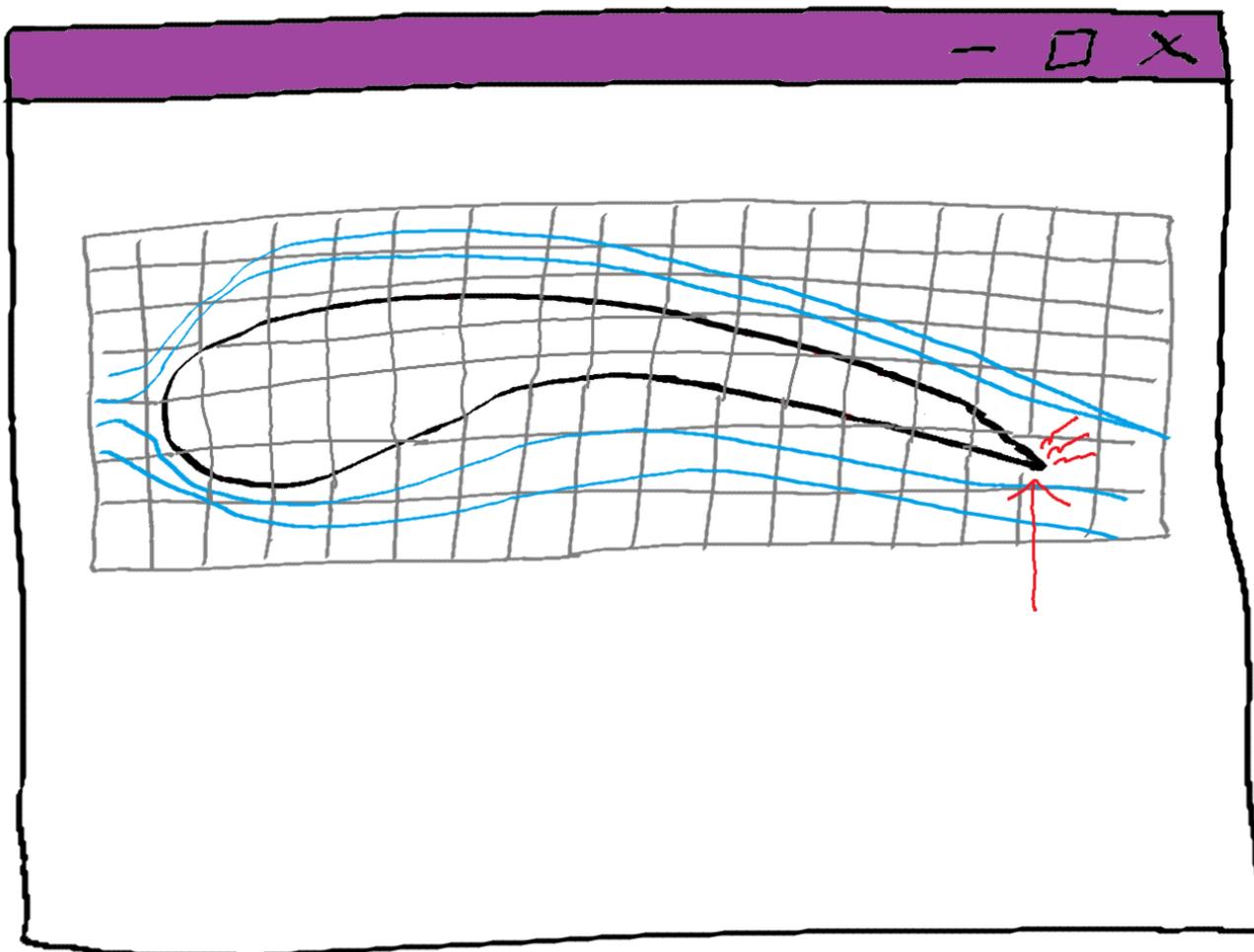
# CFD: Simulating flow around a wing



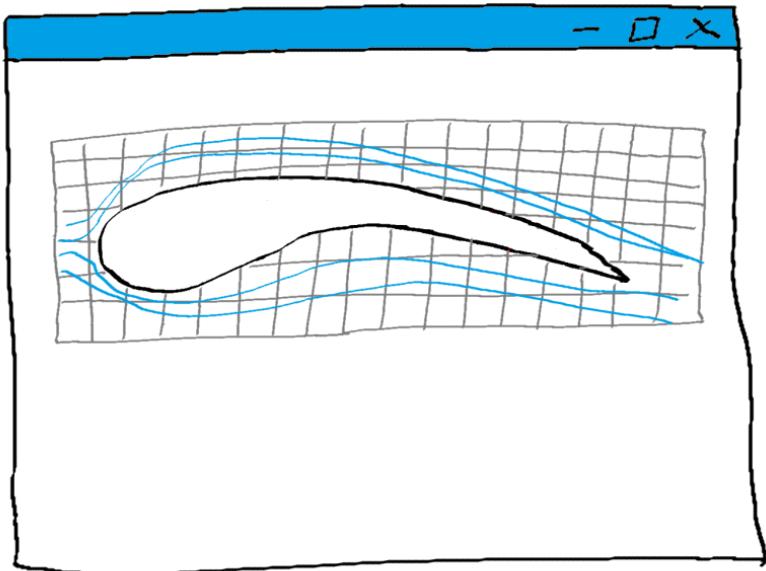
# FEM: Simulating stresses on a wing



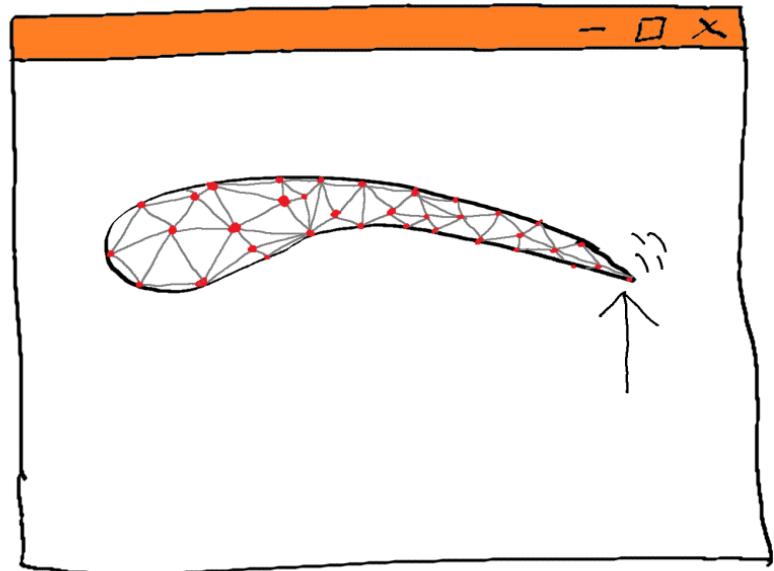
# Can we simulate both at the same time?



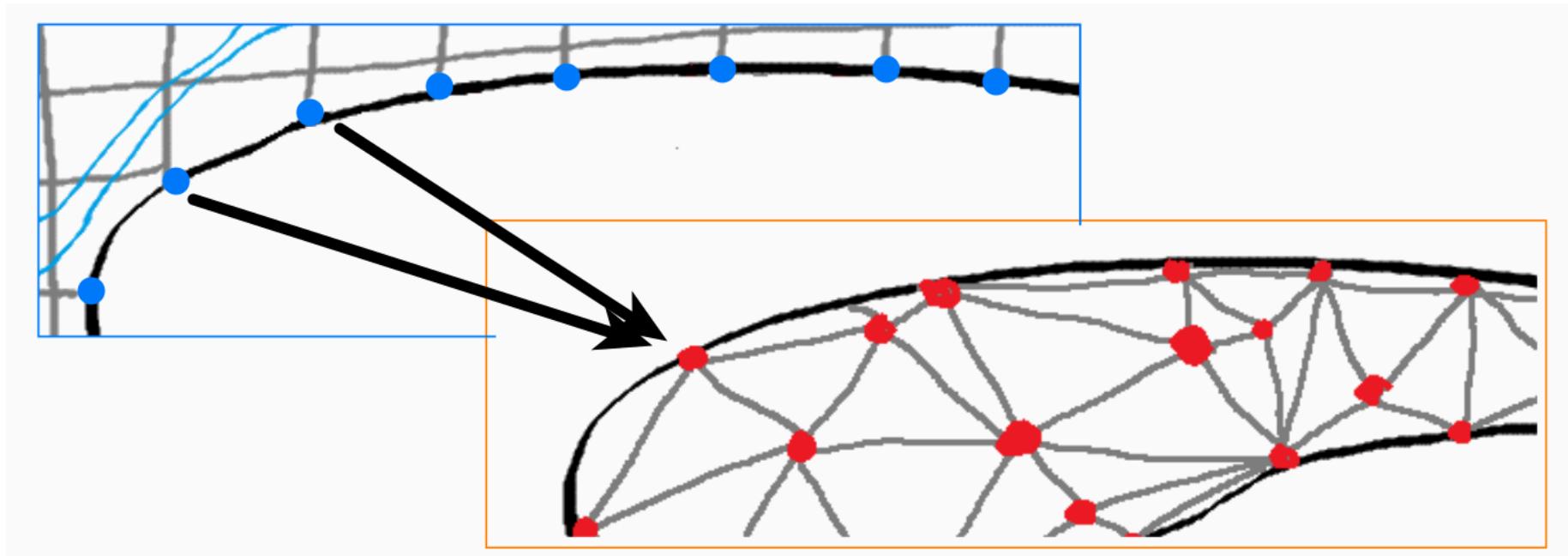
# ...but reusing the CFD and FEM codes?



+

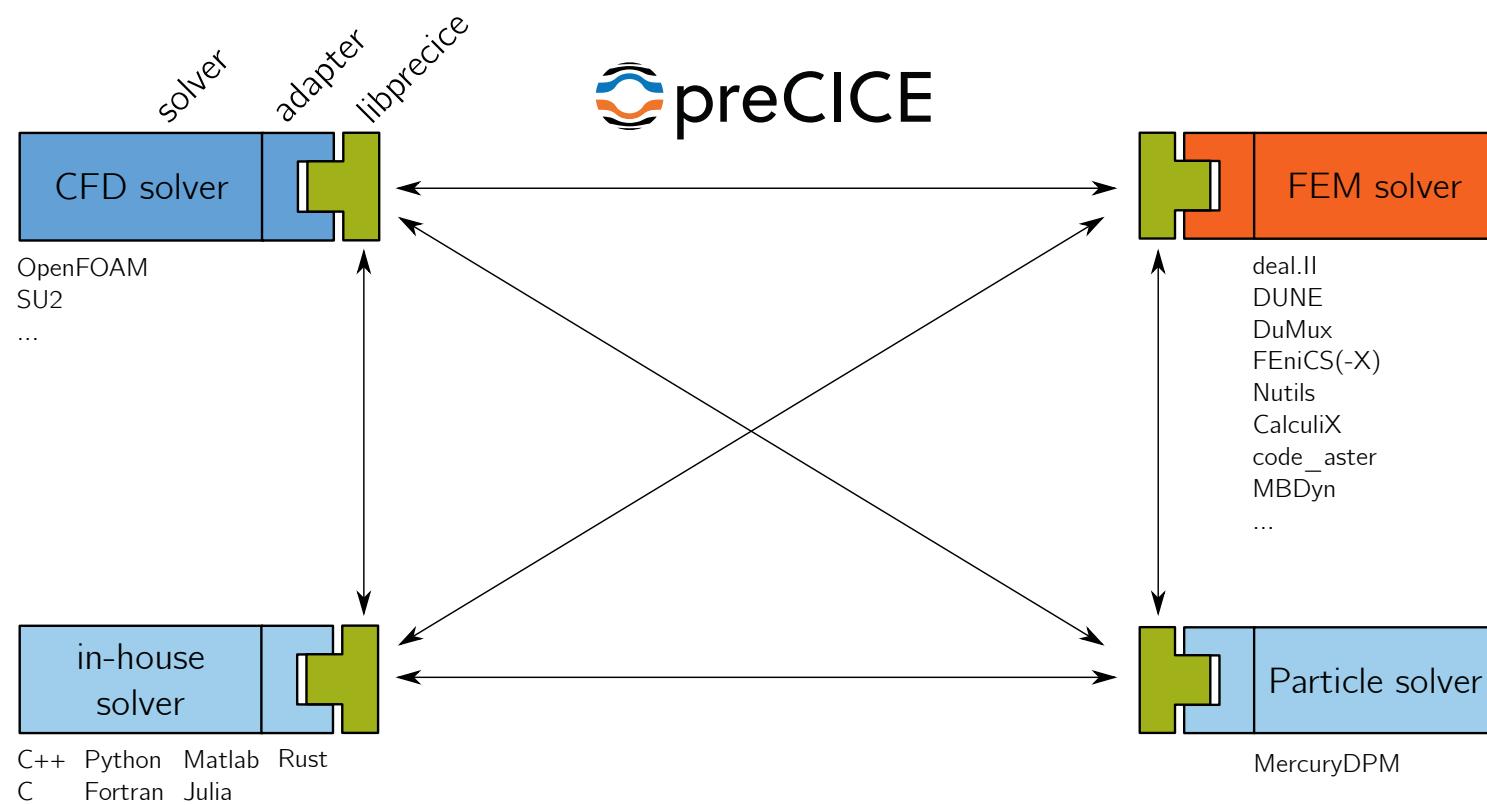


# From the CFD world to the FEM world

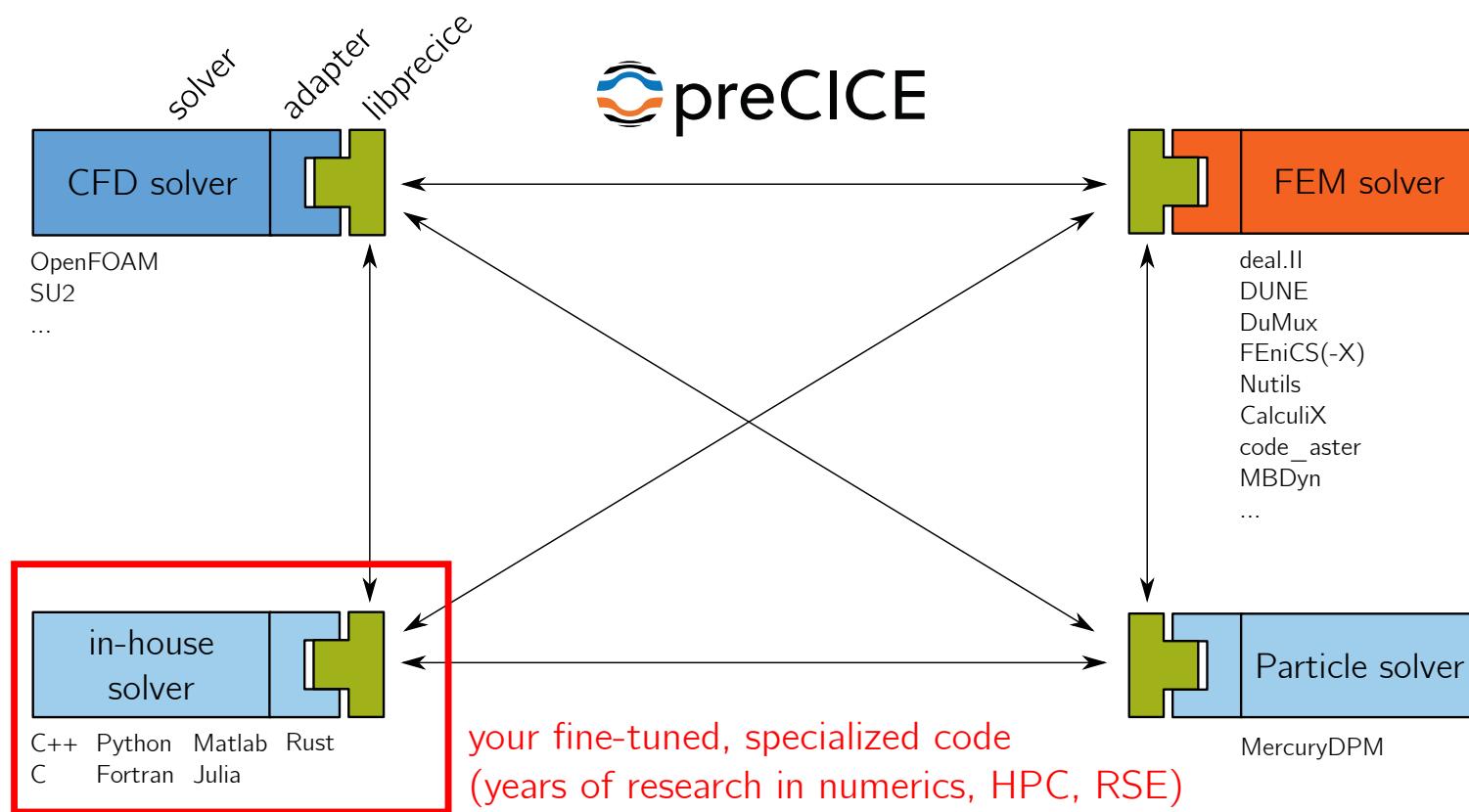




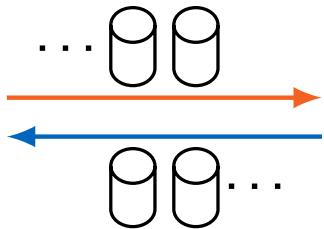
# preCICE



# preCICE



# Where preCICE helps



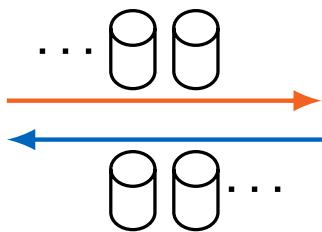
Communication

Options:

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

Fully-parallel, peer-to-peer

# Where preCICE helps

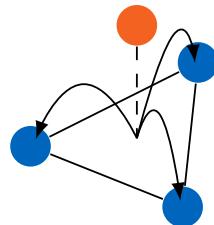


Communication

Options:

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

Fully-parallel, peer-to-peer



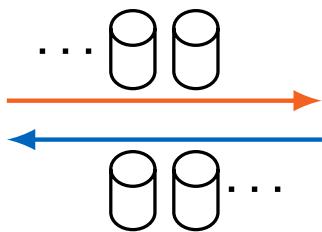
Data mapping

Options:

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

Compute on any side

# Where preCICE helps

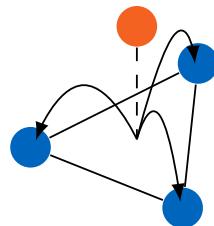


Communication

Options:

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

Fully-parallel, peer-to-peer

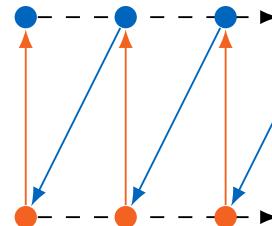


Data mapping

Options:

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

Compute on any side



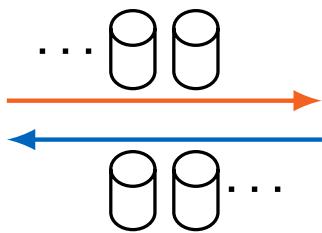
Coupling schemes

Options:

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

Same high-level API  
Configurable at runtime

# Where preCICE helps

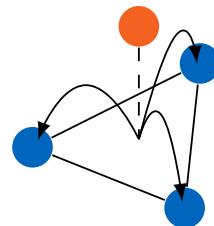


Communication

Options:

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

Fully-parallel, peer-to-peer

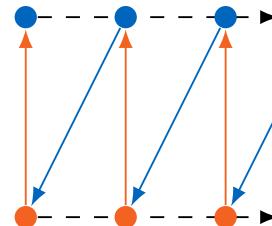


Data mapping

Options:

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

Compute on any side

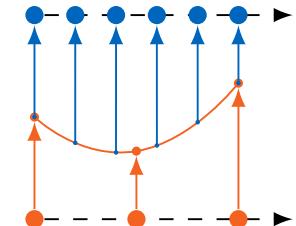


Coupling schemes

Options:

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

Same high-level API  
Configurable at runtime

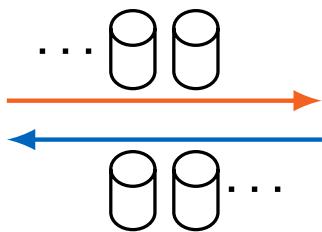


Time interpolation

Options:

- waveform iteration

# Where preCICE helps

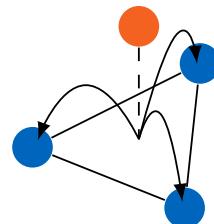


## Communication

### Options:

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

Fully-parallel, peer-to-peer

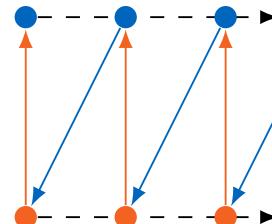


## Data mapping

### Options:

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

Compute on any side

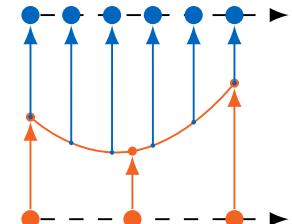


## Coupling schemes

### Options:

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

Same high-level API  
Configurable at runtime



## Time interpolation

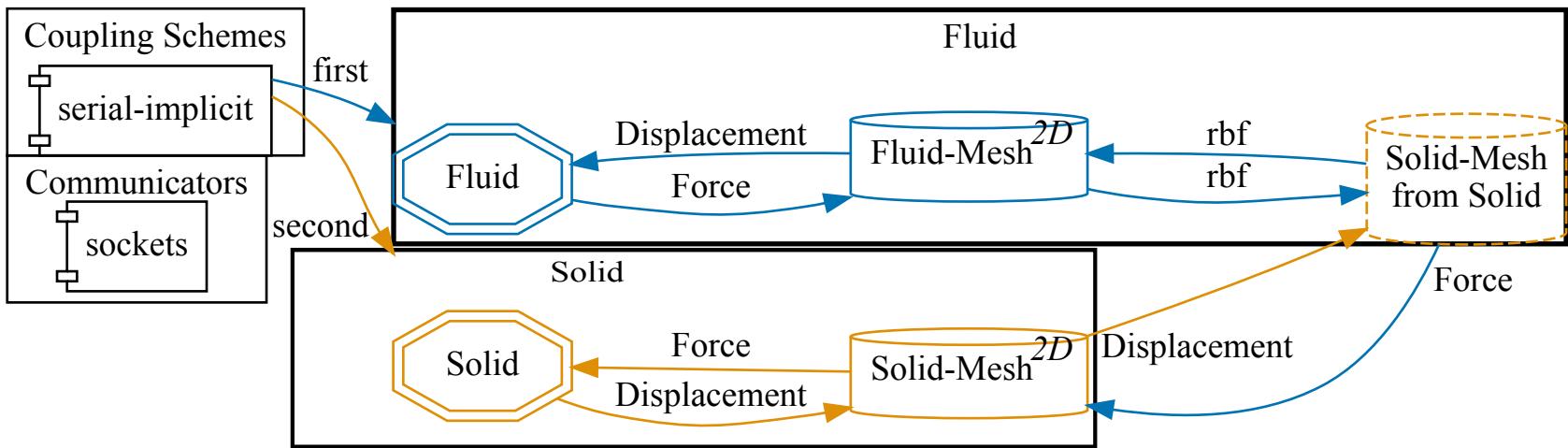
### Options:

- waveform iteration

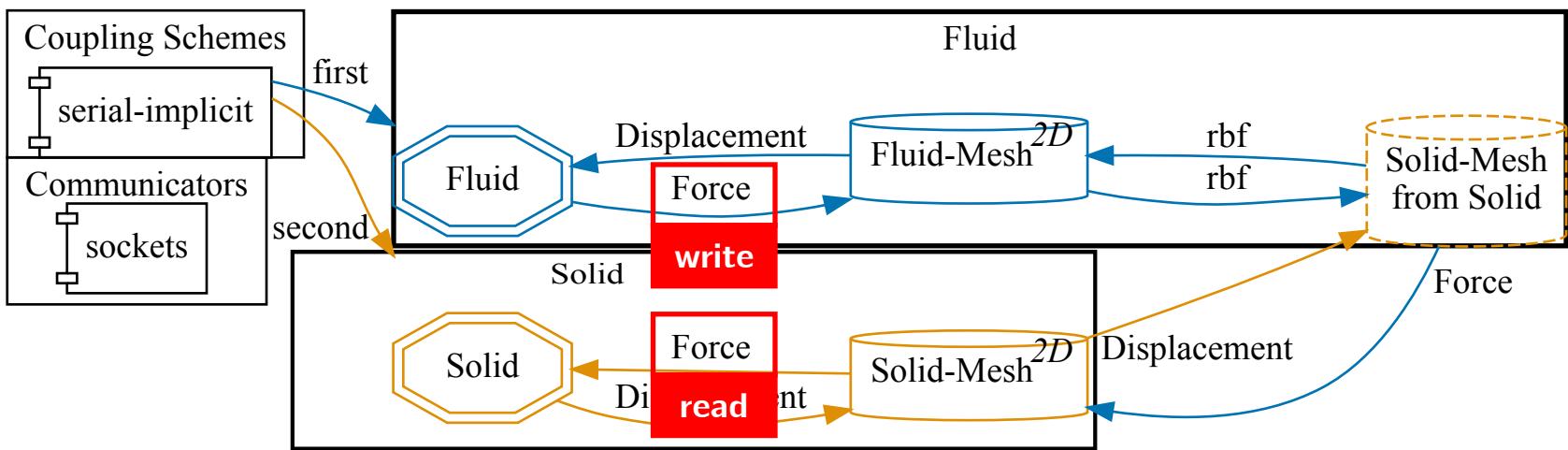
<http://doi.org/>

10/m9d8

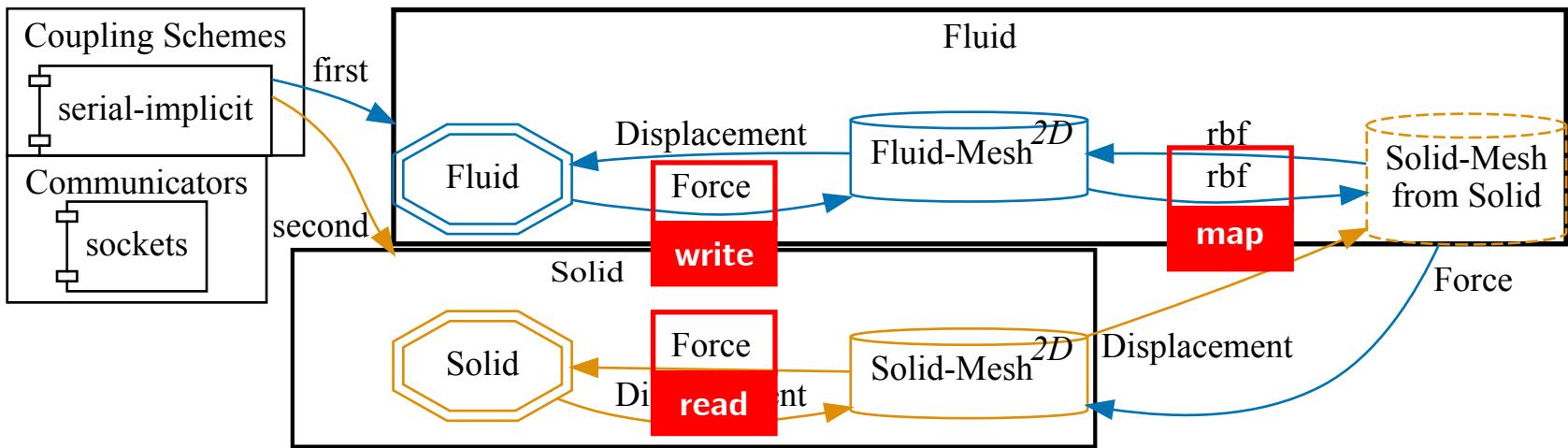
# An FS example setup



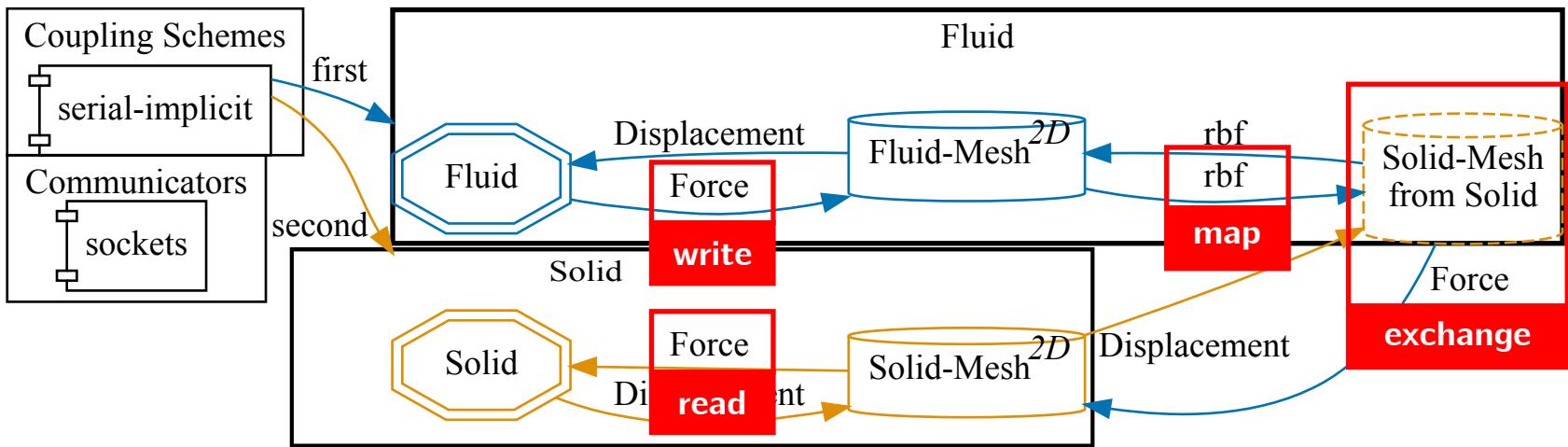
# An FS example setup



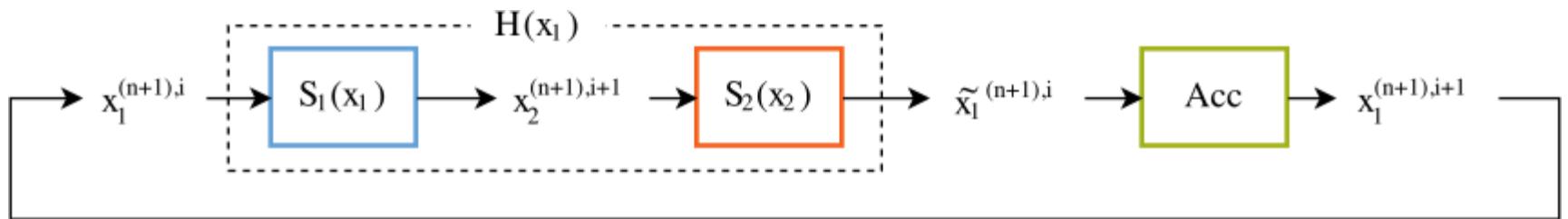
# An FS example setup



# An FS 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 allows 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 framework are evaluated by analyzing multiscale problems with varying degrees of complexity and computational loads.

Figures References Related Details

**SCI-TECH FORUM**

AIAA SCITECH 2023 Forum  
23-27 January 2023  
National Harbor, MD & Online  
<https://doi.org/10.2514/6.2023-2027>

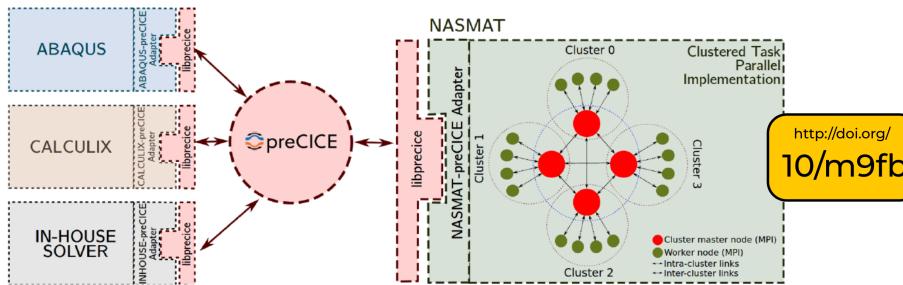
Crossmark

Check for updates

Information

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

PDF



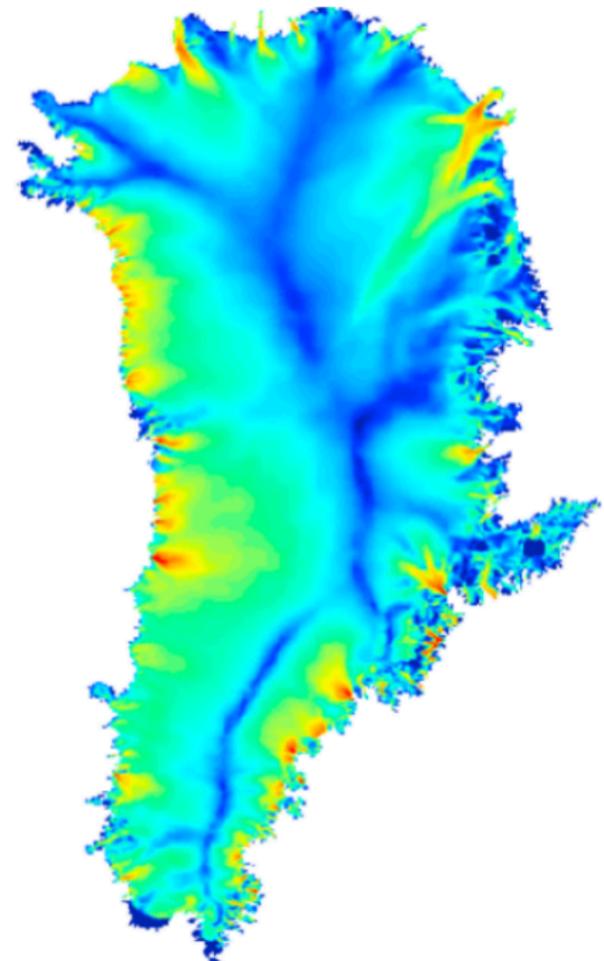
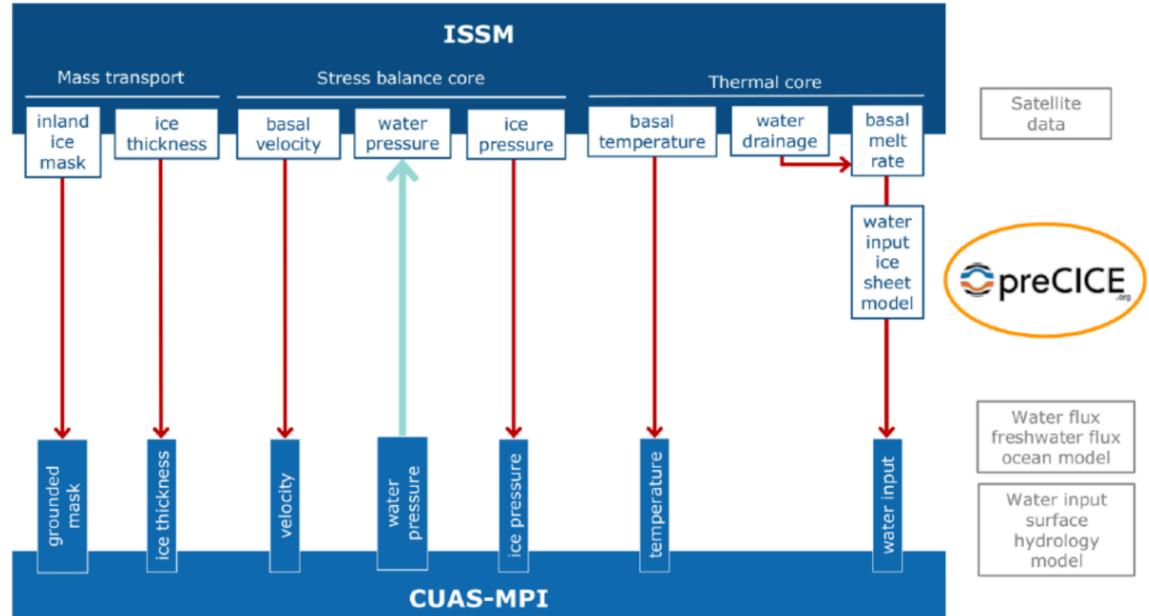
## Hierarchical materials, NASA Glenn Research Center (USA)

# Application example: Glaciology

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

Yannic Fischler  
[yannic.fischler@tu-darmstadt.de](mailto:yannic.fischler@tu-darmstadt.de)  
preCICE Workshop 2023

[http://doi.org/  
10/m9fc](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](http://www.elsevier.com/locate/softx)



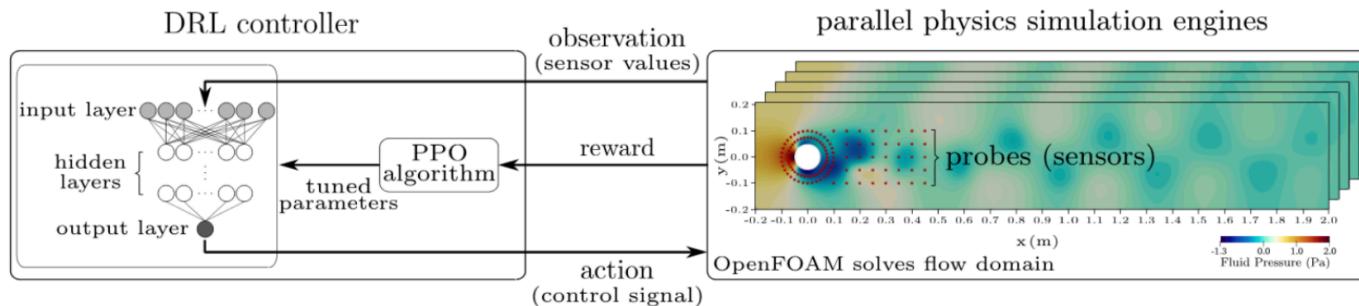
Original software publication

Gym-preCICE: Reinforcement learning environments for active flow control

Mosayeb Shams\*, Ahmed H. Elsheikh

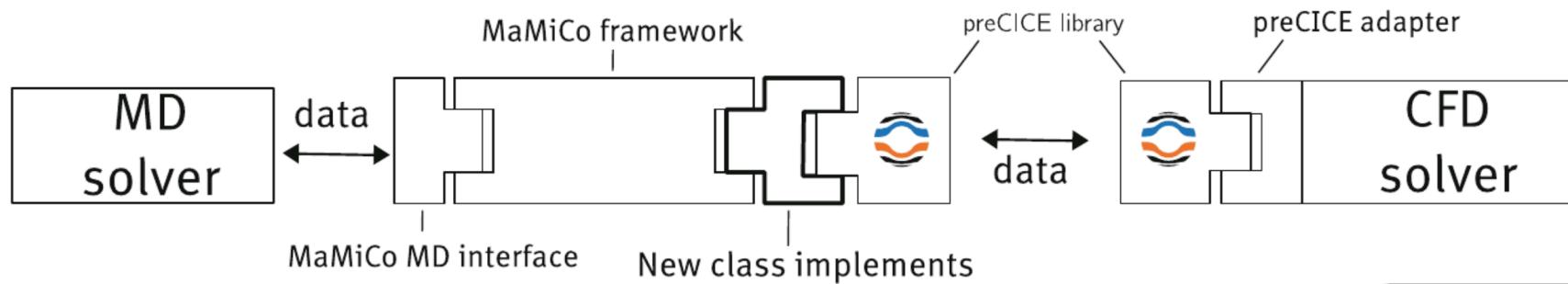
*Heriot-Watt University, Edinburgh, United Kingdom*

[http://doi.org/  
10/n3rd](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](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

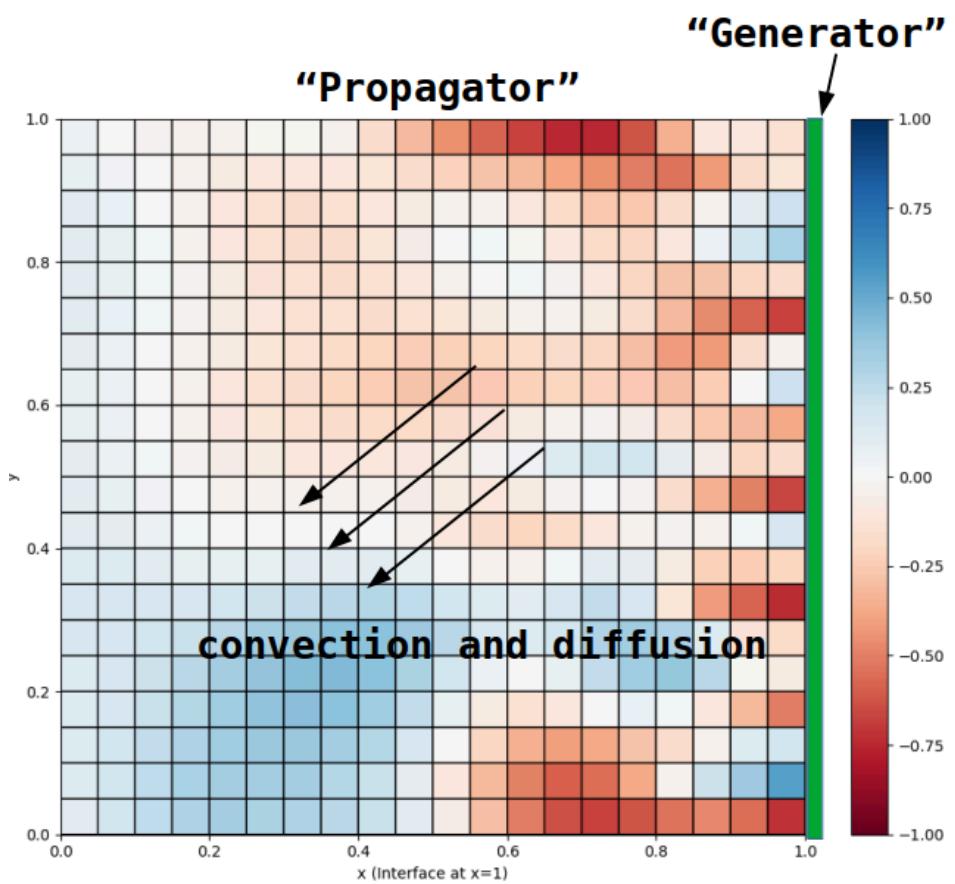
# Dependencies

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

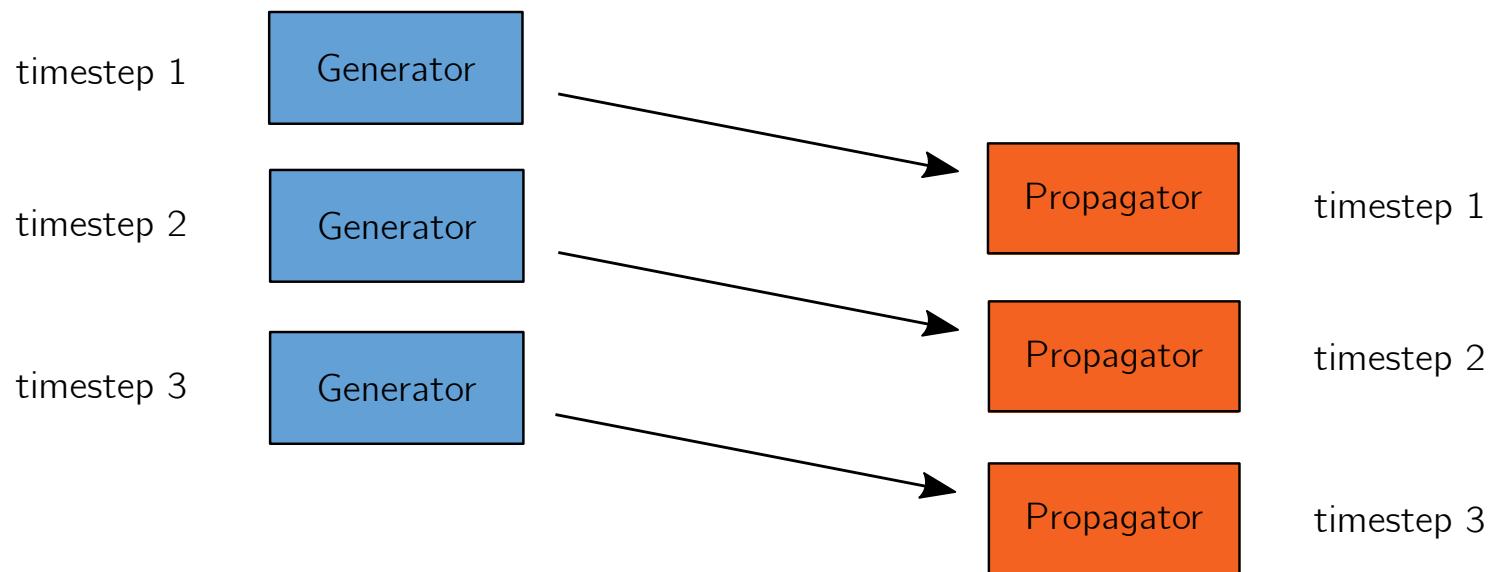
```
pip3 install --upgrade pip
python3 -m venv .venv && . .venv/bin/activate
pip3 install pyprecice
```

# "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)

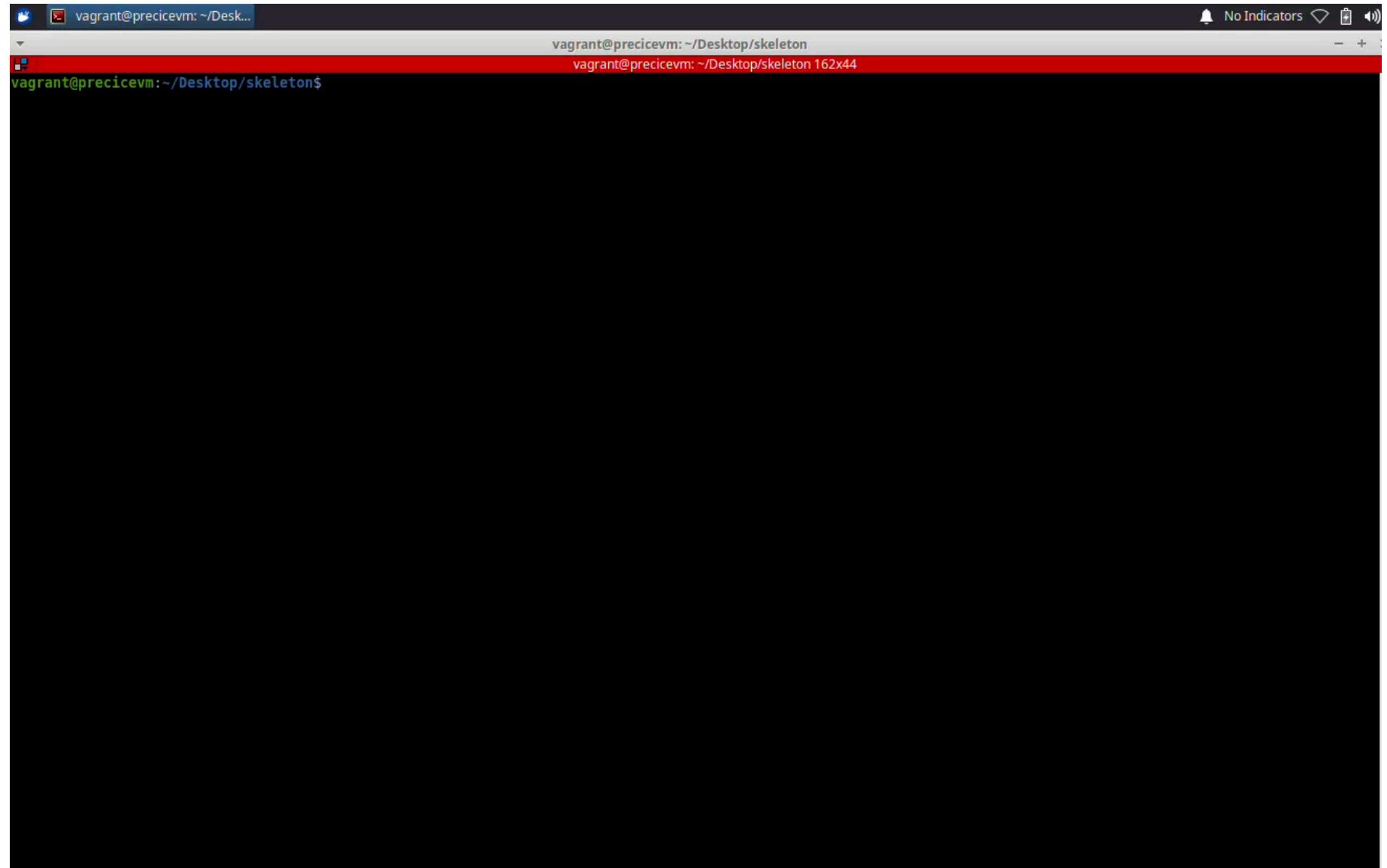
# initial data, associated to cell centers
u = numpy.zeros([n, n])

dt = 0.01
t = 0

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

while True:

    print("Propagating data")
```



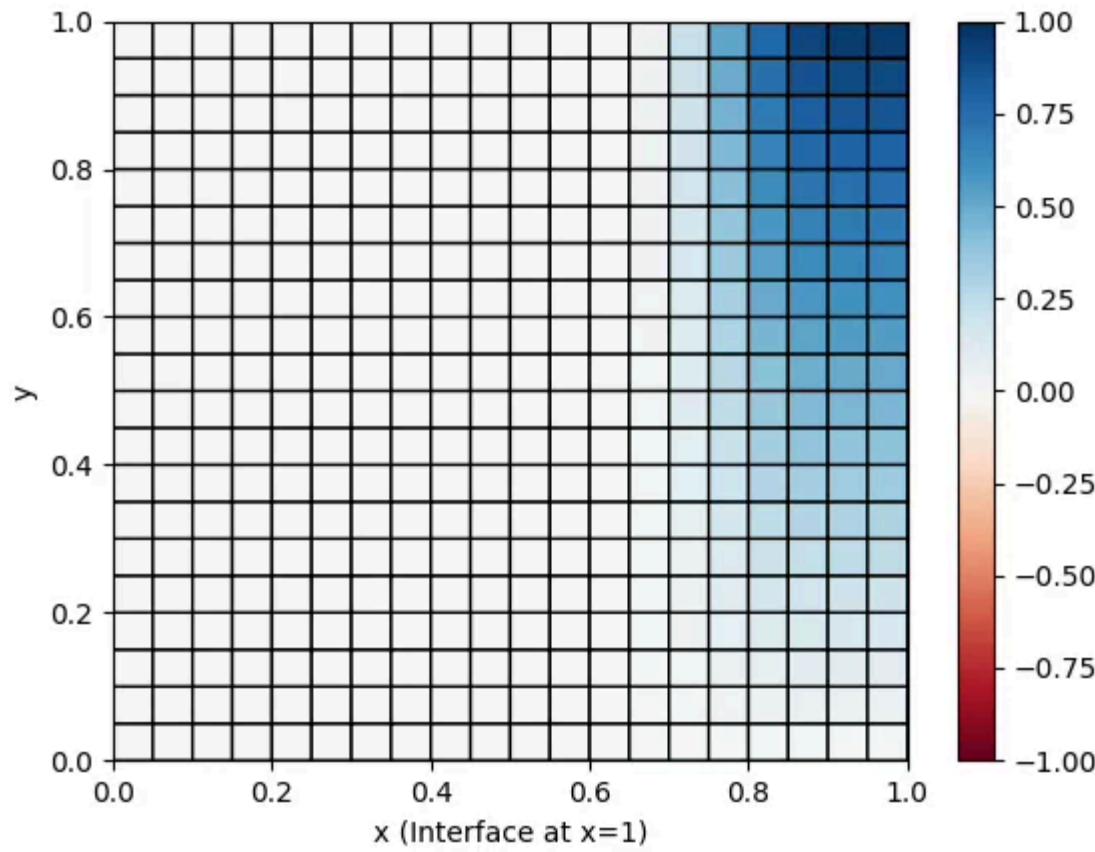
vagrant@precicevm: ~/Des...

vagrant@precicevm: ~/Desktop/skeleton

vagrant@precicevm: ~/Desktop/skeleton 162x44

vagrant@precicevm:~/Desktop/skeleton\$

# Uncoupled simulation



# Import preCICE

```
1 import numpy
2
3
4 # generate mesh
5 n = 20
6 y = numpy.linspace(0, 1, n + 1)
7
8 dt = 0.01
9 t = 0
10
11 while True:
12     print("Generating data")
13     u = 1 - 2 * numpy.random.rand(n)
14
15     t = t + dt
16     if(t > 0.1):
17         break
```

# Import preCICE

```
1 import numpy
2 import precice
3
4 # generate mesh
5 n = 20
6 y = numpy.linspace(0, 1, n + 1)
7
8 dt = 0.01
9 t = 0
10
11 while True:
12     print("Generating data")
13     u = 1 - 2 * numpy.random.rand(n)
14
15     t = t + dt
16     if(t > 0.1):
17         break
```

# Configure preCICE

```
1 import numpy
2 import precice
3
4 # generate mesh
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,
```

# 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

```
1 import numpy
2 import precice
3
4 # generate mesh
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,
```

# 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

```
1 import numpy
2 import precice
3
4 # generate mesh
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,
```

# 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

```
1 import numpy
2 import precice
3
4 # generate mesh
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,
```

# 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

```
1 import numpy
2 import precice
3
4 # generate mesh
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,
```

# Advance the coupling

```
1 import numpy
2 import precice
3
4 # generate mesh
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,
```

# Advance the coupling

```
1 import numpy
2 import precice
3
4 # generate mesh
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,
```

**Not there yet, but let's run it**  
(similar changes in propagator.py - try it at home)

# Nothing happening?

```
vagrant@precicevm: ~/Desktop/solution/T4/generator 80x44
vagrant@precicevm:~/Desktop/solution/T4/generator$ tree
.
└── generator.py

0 directories, 1 file
vagrant@precicevm:~/Desktop/solution/T4/generator$ python3 generator.py [REDACTED]

vagrant@precicevm: ~/Desktop/solution/T4/propagator 80x44
vagrant@precicevm:~/Desktop/solution/T4/propagator$ tree
.
└── propagator.py

0 directories, 1 file
vagrant@precicevm:~/Desktop/solution/T4/propagator$ python3 propagator.py [REDACTED]
```

# Write & read data

```
1 import numpy
2 import precice
3
4 # generate mesh
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,
```

# Write & read data

```
1 import numpy
2 import precice
3
4 # generate mesh
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,
```

# 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!**

The image shows a dual-terminal window on a Linux desktop environment. Both terminals are running under the user `vagrant`.

**Left Terminal:**

- Running command: `generator`
- Output:

```
vagrant@precicevm:~/Desktop/solution/T5/generator 80x44
vagrant@precicevm:~/Desktop/solution/T5/generator$ tree
.
└── generator.py

0 directories, 1 file
```
- Running command: `python3 generator.py`

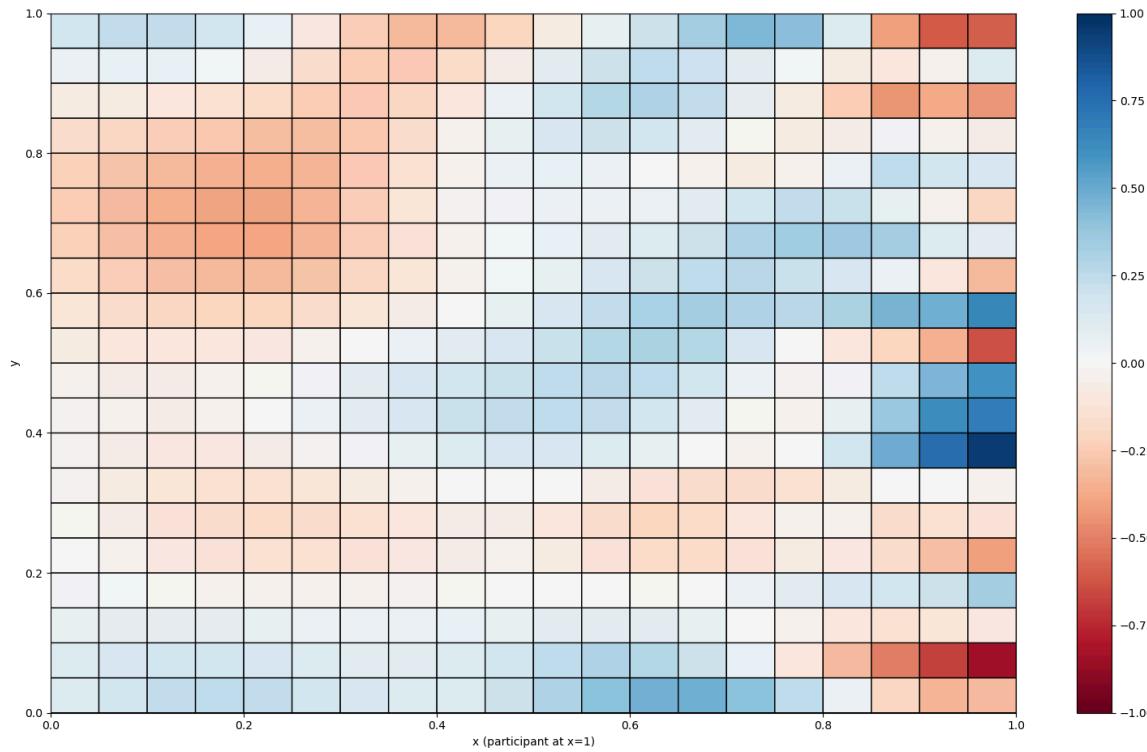
**Right Terminal:**

- Running command: `propagator`
- Output:

```
vagrant@precicevm:~/Desktop/solution/T5/propagator 80x44
vagrant@precicevm:~/Desktop/solution/T5/propagator$ tree
.
└── propagator.py

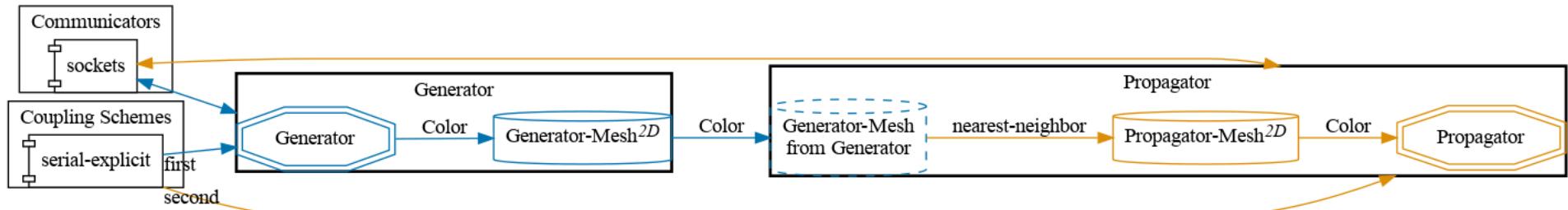
0 directories, 1 file
```
- Running command: `python3 propagator.py`

# Data is being transferred!



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**



# Open-source development on GitHub

Screenshot of the GitHub organization page for preCICE.

**Organization Overview:**

- Repositories: 64
- Projects: 13
- Packages: 0
- Teams: 0
- People: 11
- Insights
- Settings

**preCICE**  
A Coupling Library for Partitioned Multi-Physics Simulations on Massively Parallel Systems  
**Verified**

198 followers | Germany | <https://precice.org> | [@precice@fosstodon.org](mailto:@precice@fosstodon.org) | [c/precICEcoupling](#) | <https://precice.discourse.group/> | [@precice.org](#)

**README.md**

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. Read more on the [preCICE website](#).

preCICE consists of several components, most of which are released as part of the [preCICE Distribution](#):

- The [core library](#) (in C++, with bindings in C and Fortran)
- Language bindings for [Python](#), [Julia](#), [Matlab](#), [Rust](#), as well as a [Fortran module](#)
- Ready-to-use adapters for several open-source codes, including [OpenFOAM](#), [CalculiX](#), [deal.II](#), [FEniCS legacy](#), [FEniCS-X](#), [DUNE](#), [DuMuX](#), [SU2](#), [MBDyn](#), [code\\_aster](#), and [more](#).
- [Tutorials](#) with case files for several applications, solvers, and preCICE features.
- A [Vagrant box](#), which serves as a demo virtual machine with preCICE and several solvers already installed.
- The [preCICE website and documentation](#)

When using or referring to preCICE in academic publications, please follow the [citation guidelines](#). More specifically, cite the preCICE v2 reference paper and any reference papers of adapters you may be using. For reproducibility, you may also want to use components from a preCICE distribution version and cite the respective [data repository entry](#).

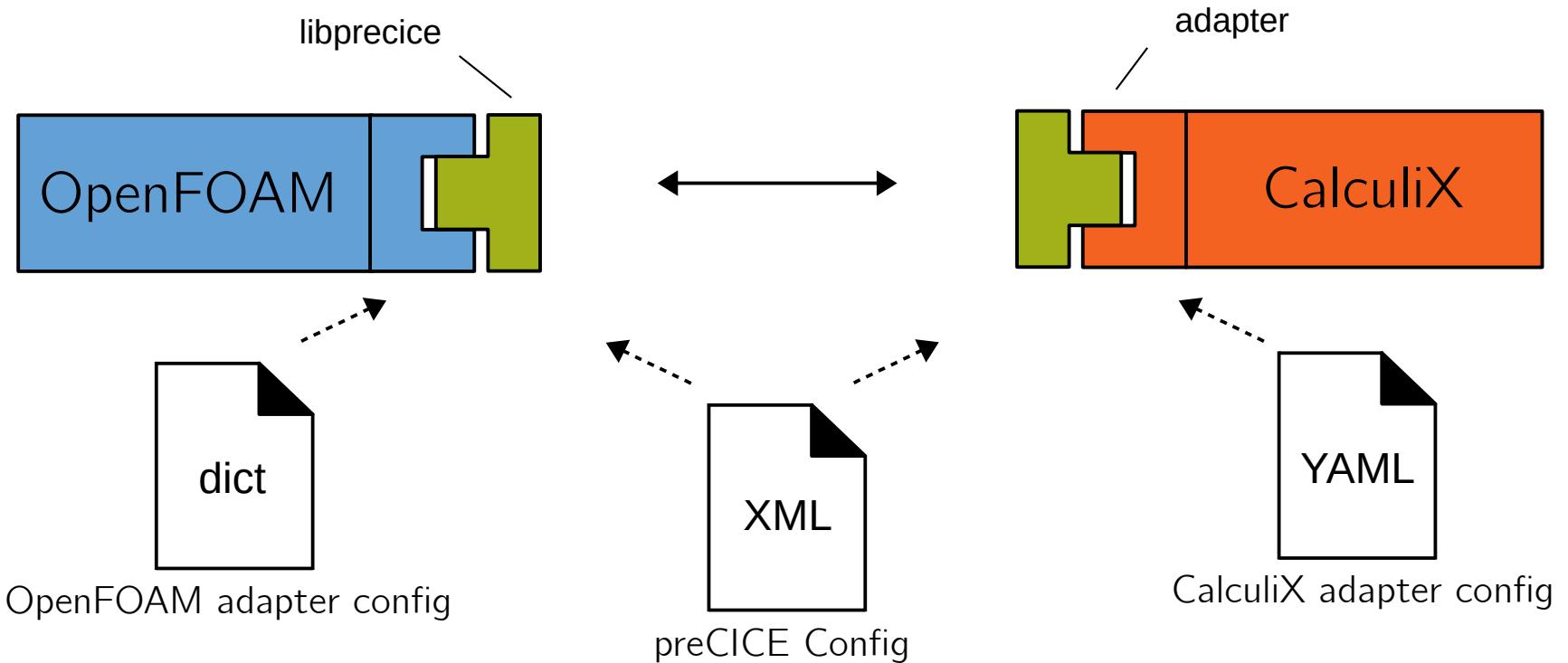
**View as: Public**  
You are viewing the README and pinned repositories as a public user.  
[Get started with tasks](#) that most successful organizations complete.

**Discussions**  
Set up discussions to engage with your community!  
[Turn on discussions](#)

**People**  
  
[Invite someone](#)

**Top languages**

# Defining community standards

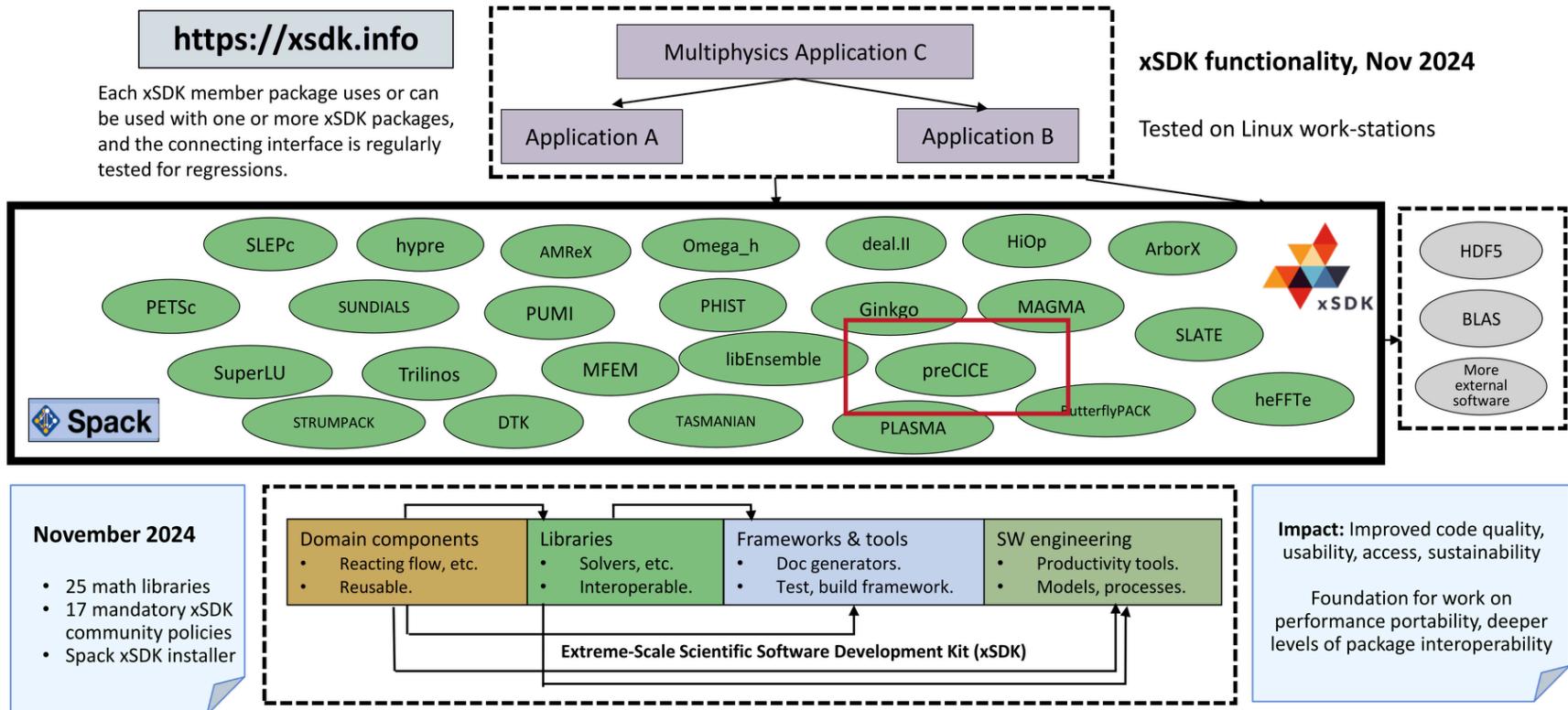


Overview of an FSI setup: How easy is it to replace CalculiX by FEniCS?



# xSDK Version 1.1.0: November 2024

<https://xsdk.info>



We can learn from the Extreme-Scale Development Toolkit: [xsdk.info/packages/](https://xsdk.info/packages/)

## Community

[Overview & news](#)[preCICE workshops](#) ▾[preCICE minisymposia](#) ▾[Support preCICE](#)[Training](#)[Stories](#)[Contributors](#)[Contribute](#) ▲[Contribute to preCICE](#)[Adapter guidelines](#)[Application case guidelines](#)[Example - OpenFOAM adapter](#)[Example - OpenDiHu adapter](#)[Example - ISSM adapter](#)[Community channels](#)

# Guidelines for adapters

**Summary:** Quality guidelines and standards for preCICE adapters and related tools

## Table of Contents

- [Metadata](#)
- [Best practices](#)
  - [Required](#)
  - [Additional](#)
- [Examples](#)

[Edit me !\[\]\(17c16d959c69905f647fdb0eb7c9ff8c\_img.jpg\)](#)

Updated 25 Sep 25

v0.3, published on September 5, 2025

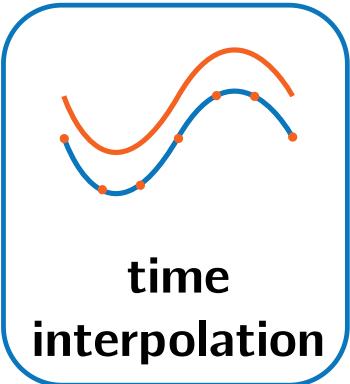
Are you developing a preCICE adapter or related tool? Follow these guidelines to make your adapter easier to publish, easier to integrate with the rest of the preCICE ecosystem, and more useful for the community.

We differentiate between adapters and application cases: For example, we consider the "[Nutils adapter](#)"  to be a collection of application cases instead of an adapter. An [adapter](#)  can be a stand-alone software project, but can also be a class, a patch, or something else with significant scholarly effort compared to the uncoupled solver. Other tools that interact with the preCICE API or configuration files can also be considered here, on a case-by-case basis.

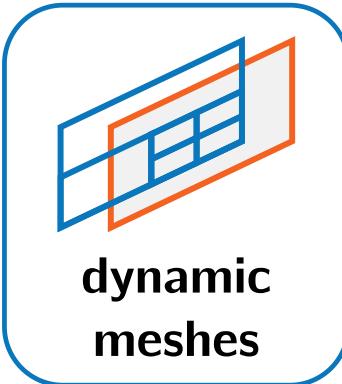
For the guidelines, we distinguish between **metadata** and **best practices**. The best practices are split between required and optional. Adapters fulfilling all the required best practices are listed on the preCICE website as adapters conforming to the preCICE standards. Additional criteria bring further benefits and are visible individually for each adapter. Adapters not fulfilling all the required criteria can still be listed here as legacy adapters. By submitting your adapter for review, you can expect a thorough check from the preCICE team on whether these guidelines are met.

[precice.org/community-guidelines-adapters.html](https://precice.org/community-guidelines-adapters.html)

# Current research



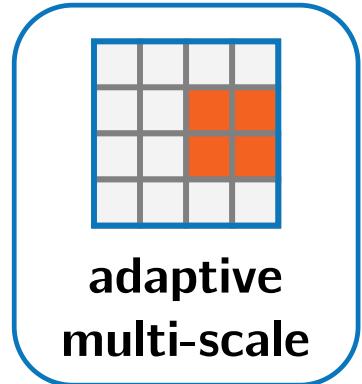
Benjamin Rodenberg



Frédéric Simonis



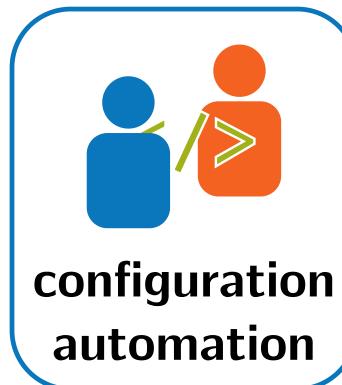
David Schneider



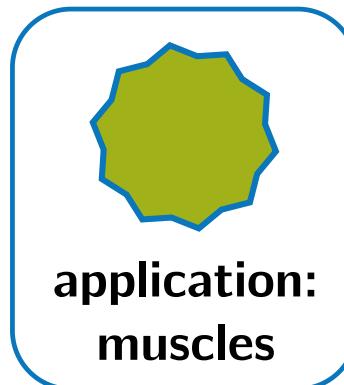
Ishaan Desai



Gerasimos Chourdakis



Felix Neubauer



Carme Homs Pons



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](https://preCICE.org)



## Welcome to preCICE

The coupling library for partitioned multi-physics simulations.

[Star on Github](#) ★ 831

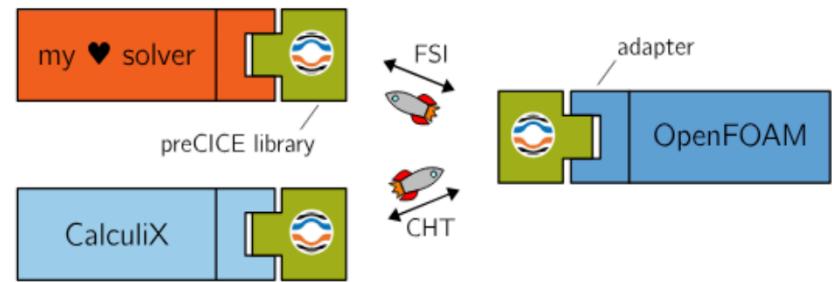
[Latest \(Apr 1, 2025\) !\[\]\(86468143c196eb0b0e5ef839d9b6c034\_img.jpg\)](#)

[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](https://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



Bundesministerium  
für Wirtschaft  
und Energie



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](http://precice.org/about))



This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).