von Karman Institute for Fluid Dynamics: OpenFOAM seminar, January 24, 2025

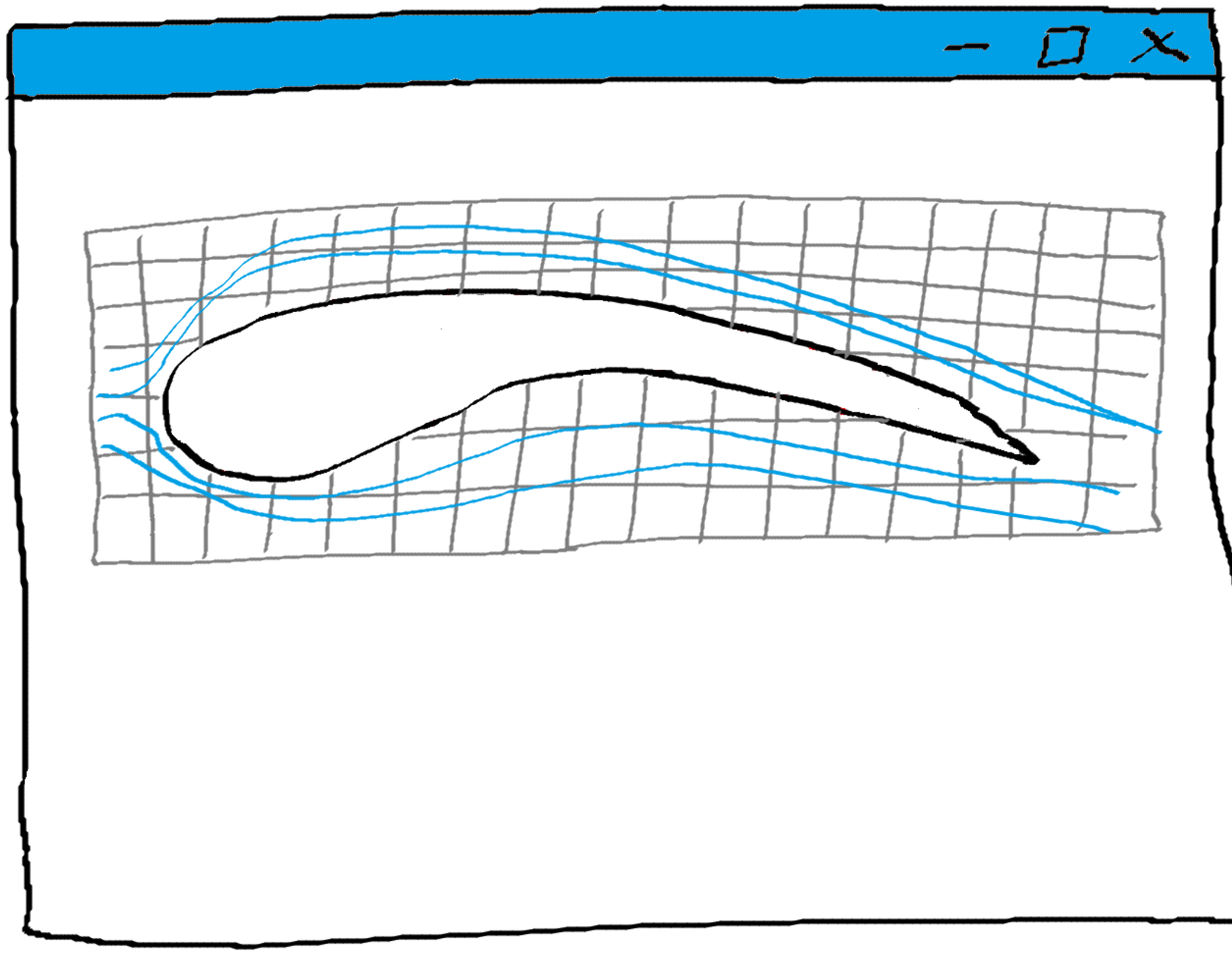
Coupling OpenFOAM with external codes via preCICE

Gerasimos Chourdakis, University of Stuttgart
gerasimos.chourdakis@ipvs.uni-stuttgart.de

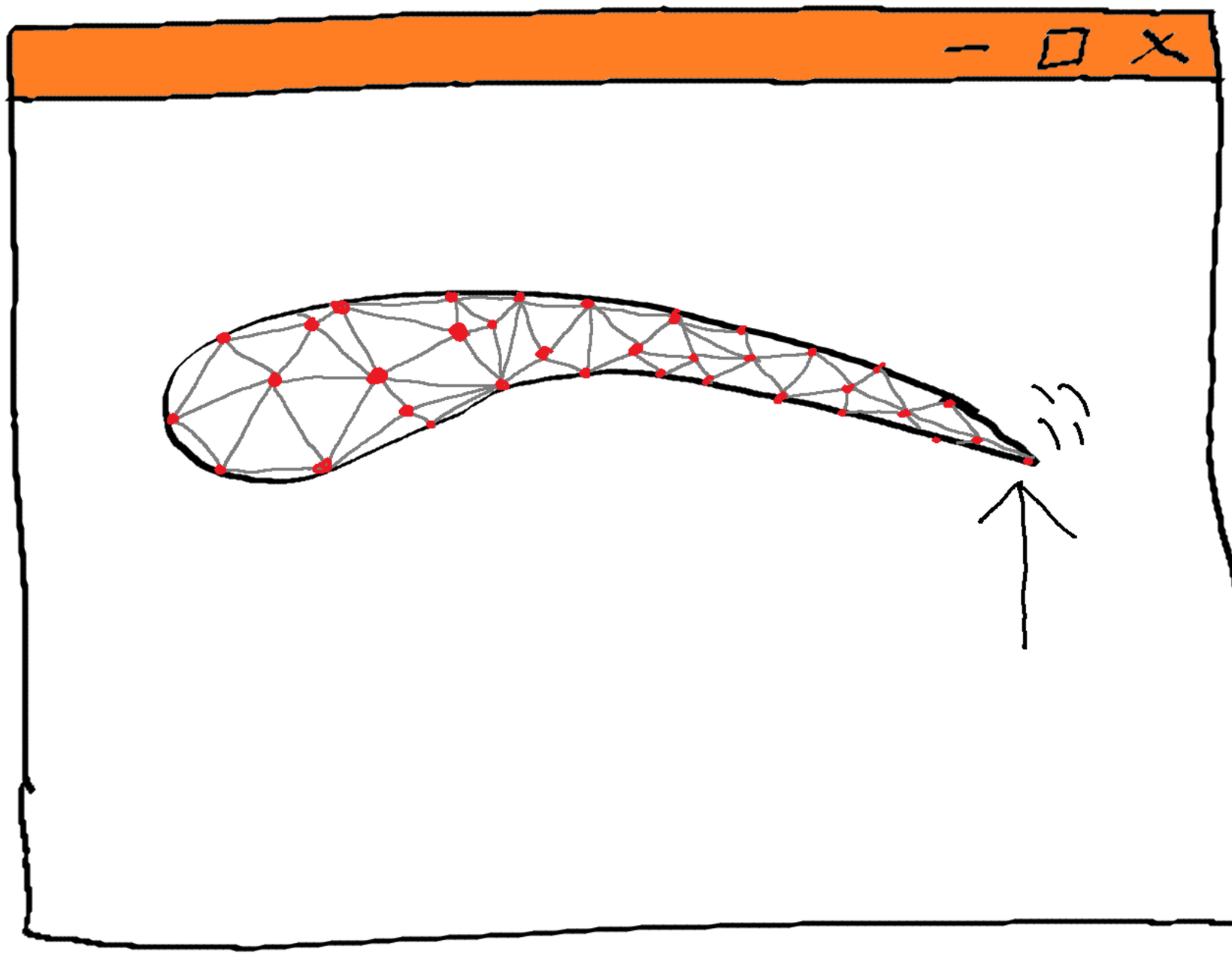
Jun Chen, University of Stuttgart

University of Stuttgart
Germany

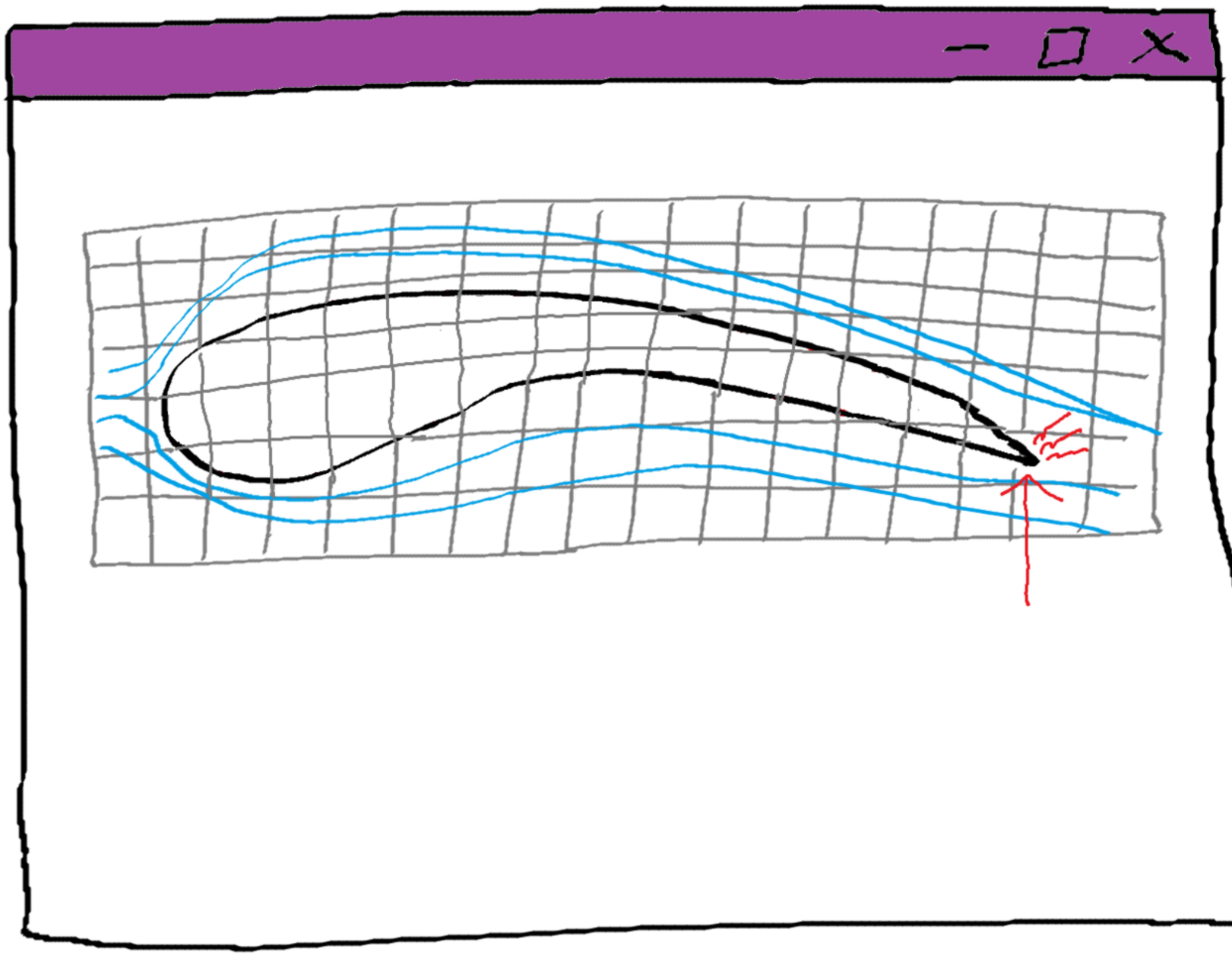
CFD: Simulating flow around a wing



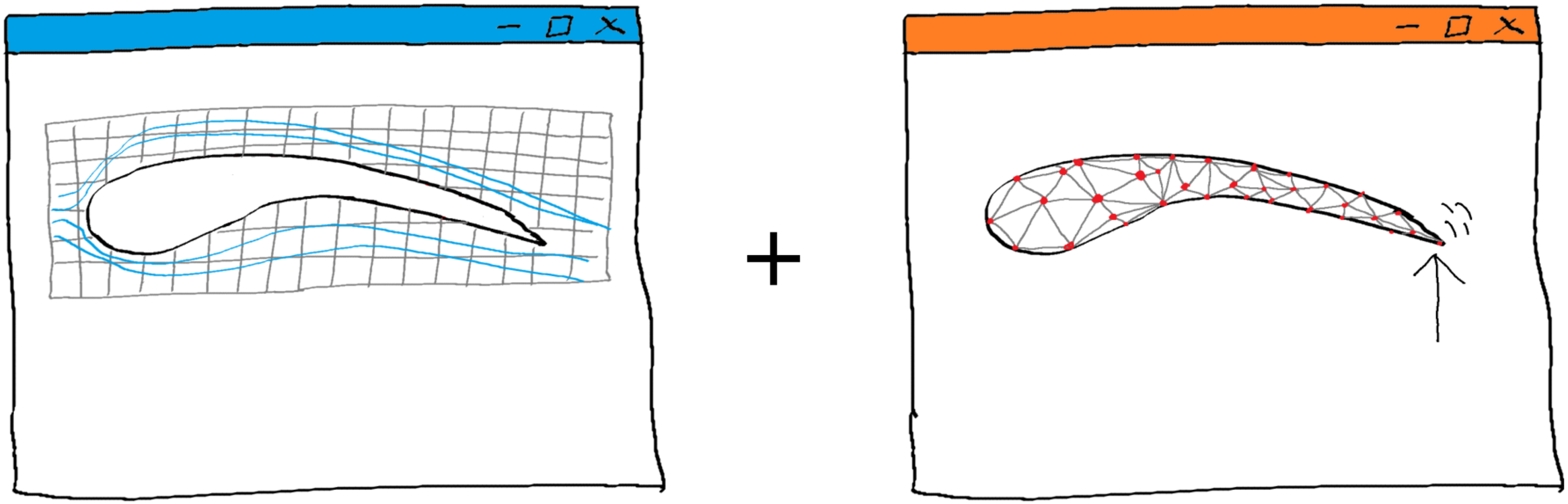
FEM: Simulating stresses on a wing



Can we simulate both at the same time?

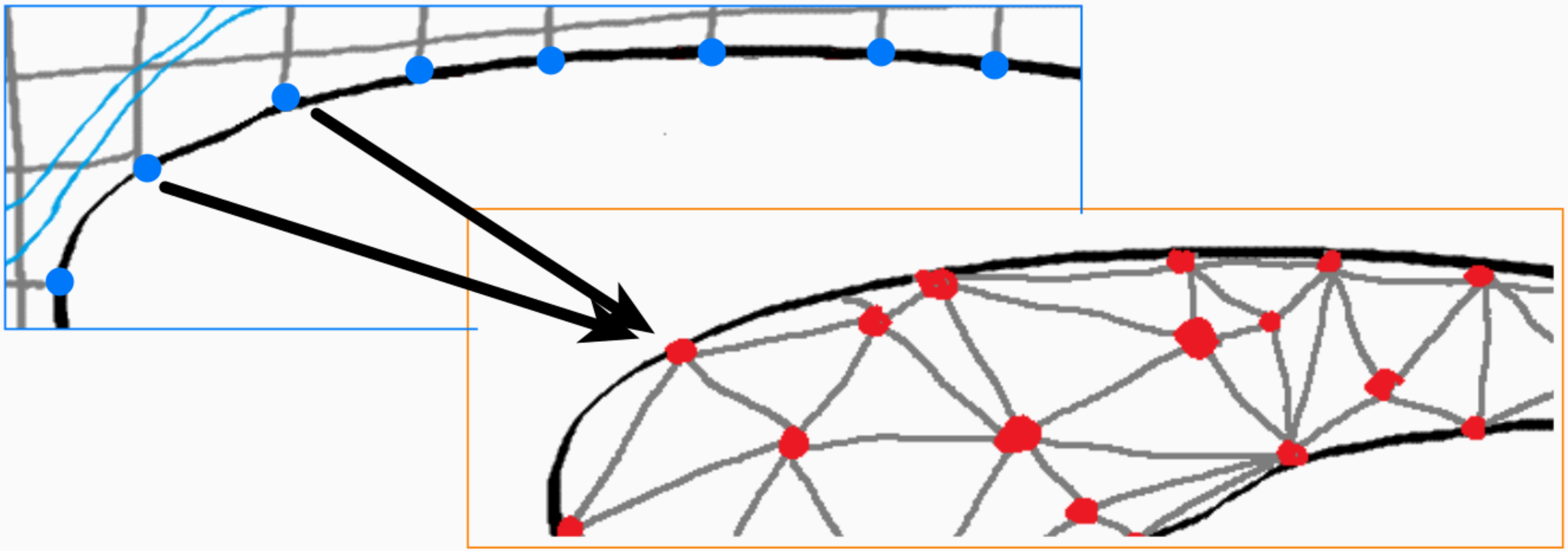


...but reusing the CFD and FEM codes?

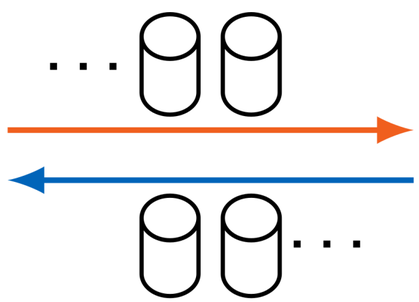


preCICE

From the CFD world to the FEM world



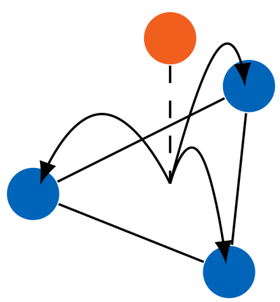
Where preCICE helps



Communication

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

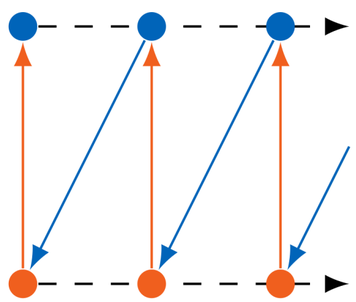
Fully-parallel, peer-to-peer



Data mapping

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

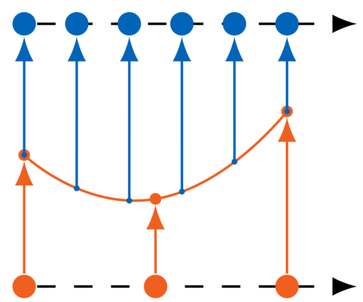
Compute on any side



Coupling schemes

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

Same high-level API
Configurable at runtime

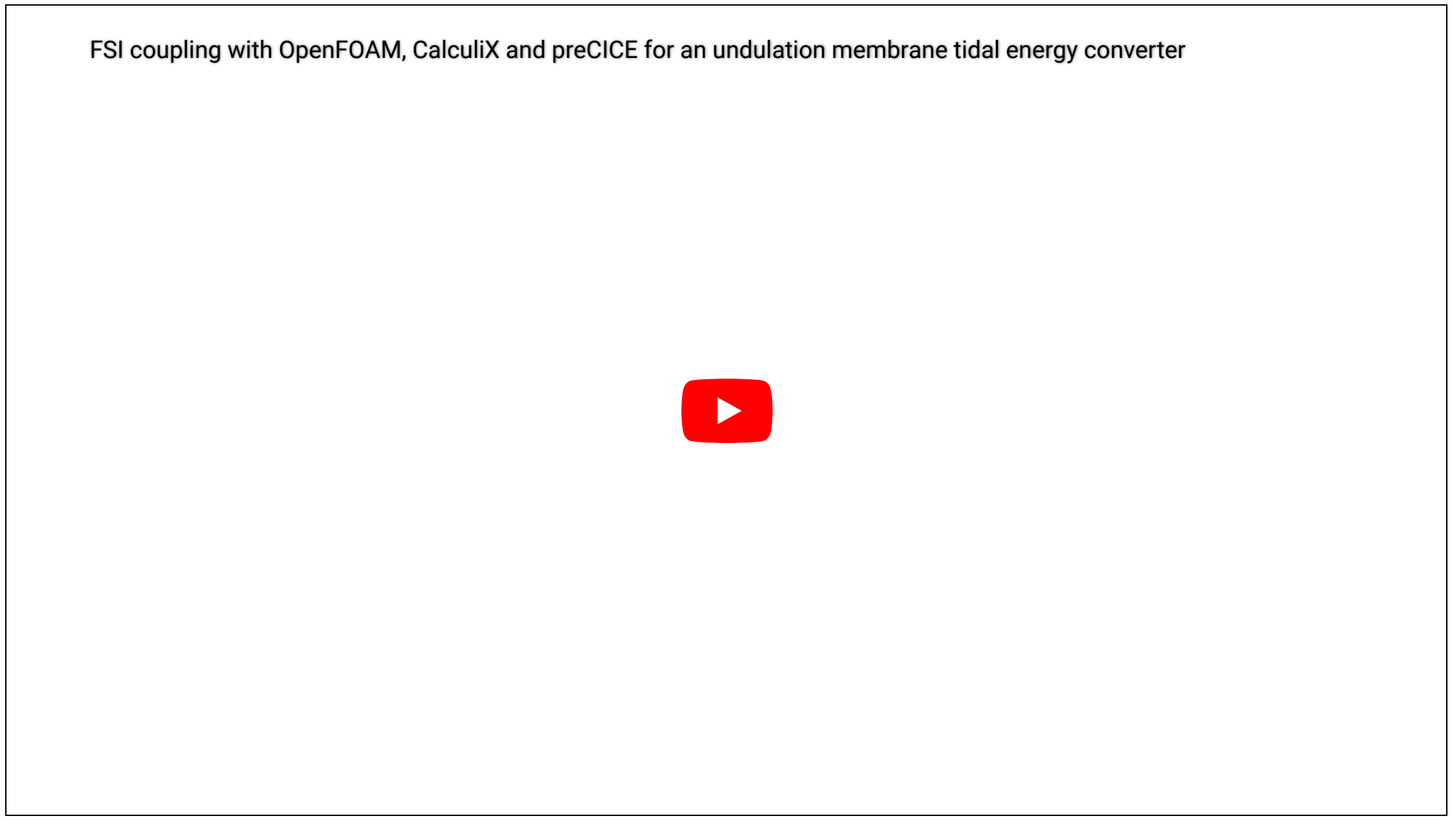


Time interpolation

Options:
- waveform iteration

What people do with preCICE

What people do with preCICE (1)



Credit: Ulrich Heck, DHCAE Tools GmbH (Germany)

What people do with preCICE (2)

Credit: Prasad Adhav, Univ. of Luxembourg (Luxembourg)

What people do with preCICE (3)

Credit: ESLAB Simulations. SNU (South Korea)

What people do with preCICE (4)

Home › Software for Exascale Computing - SPPEXA 2016-2019 › Conference paper

ExaFSA: Parallel Fluid-Structure-Acoustic Simulation

Florian Lindner, Amin Totounferoush, Miriam Mehl ✉, Benjamin Uekermann, Neda Ebrahimi Pour, Verena Krupp, Sabine Roller, Thorsten Reimann, Dörte C. Sternel, Ryusuke Egawa, Hiroyuki Takizawa & Frédéric Simonis

Conference paper | Open Access | First Online: 31 July 2020

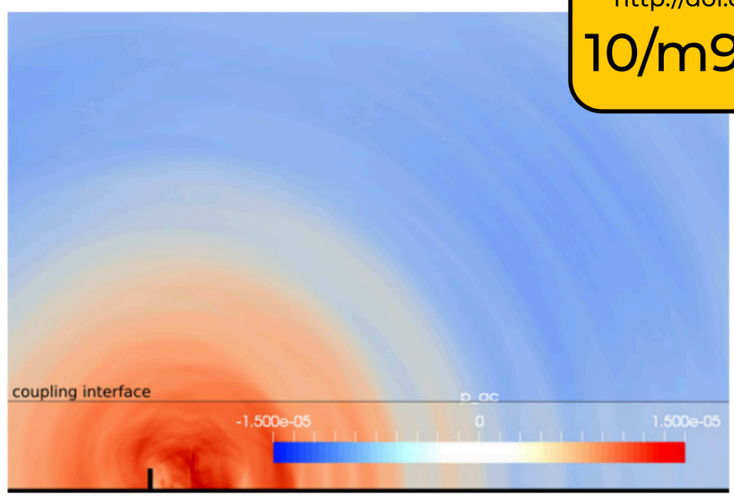
http://doi.org/
10/m9d9

fnb

SC SCIENTIFIC COMPUTING

Kolb, Elena (2022):
Aeroacoustic Simulation of Turbulent Fluid-Structure Interactions at Low Mach Numbers.
In: Ingenieurwissenschaften, Berlin, Verlag Dr. Hut, TU Darmstadt, ISBN 978-3-8439-5061-9, [Dissertation]

Mariño Salguero, Jessica Marcela (2021):
Numerical simulation of free surface flows interacting with flexible structures. (Verlagsversion)
Darmstadt, Technische Universität, → DOI: 10.26083/tuprints-00019193, → Offizielle URL, [Dissertation]



ExaFSA SPPEXA program, TU Darmstadt (Germany)

What people do with preCICE (5)

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

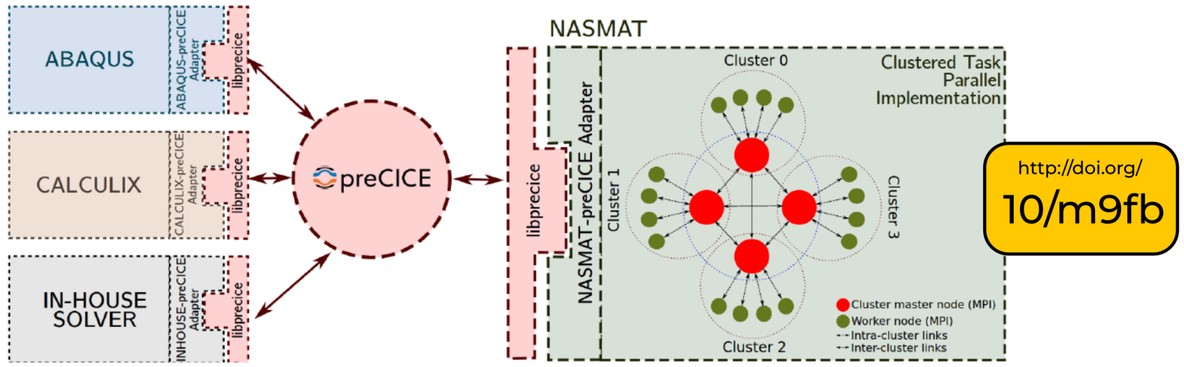
23-27 January 2023

National Harbor, MD & Online

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

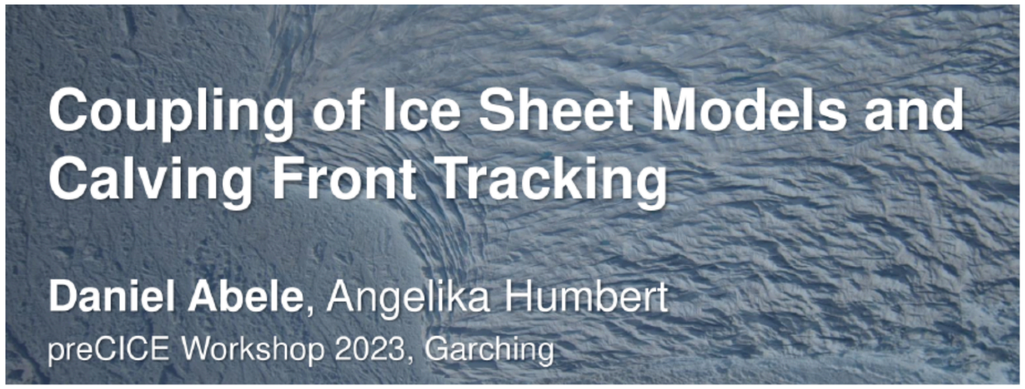
Crossmark

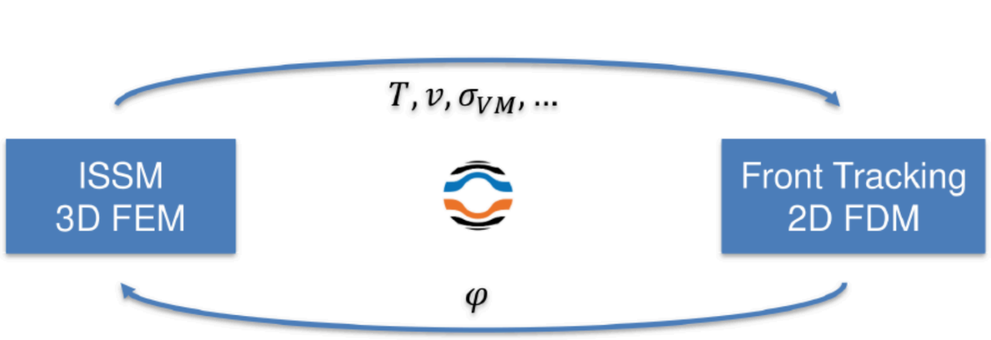
PDF

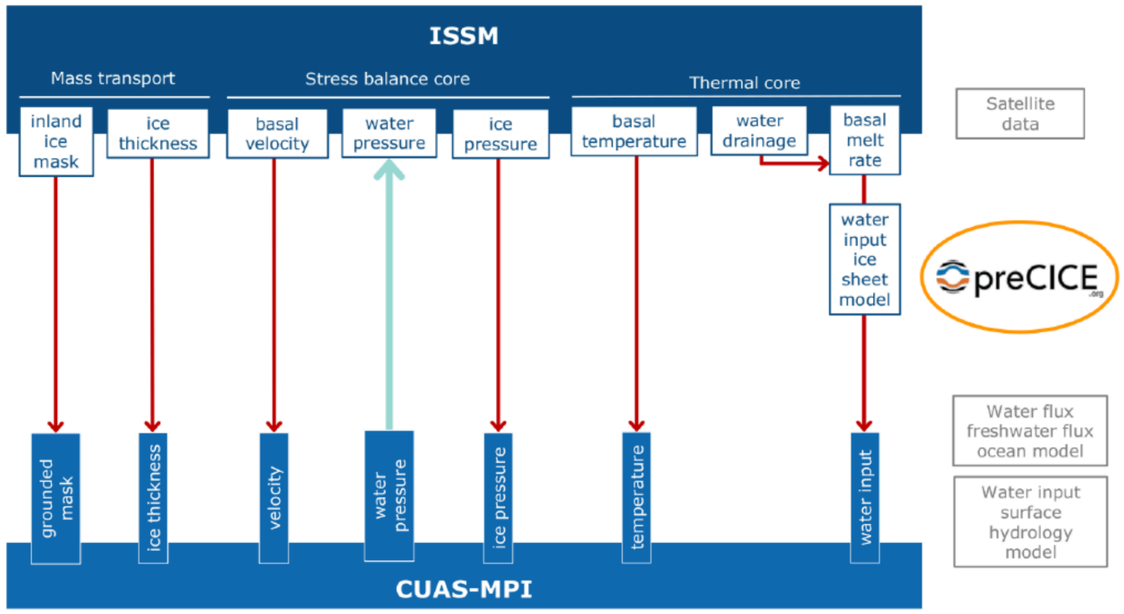


A publication by NASA Glenn Research Center (USA)

What people do with preCICE (6)







Ice-sheet simulations by AWI (Germany)

What people do with preCICE (6)

Contents lists available at ScienceDirect

SoftwareX

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

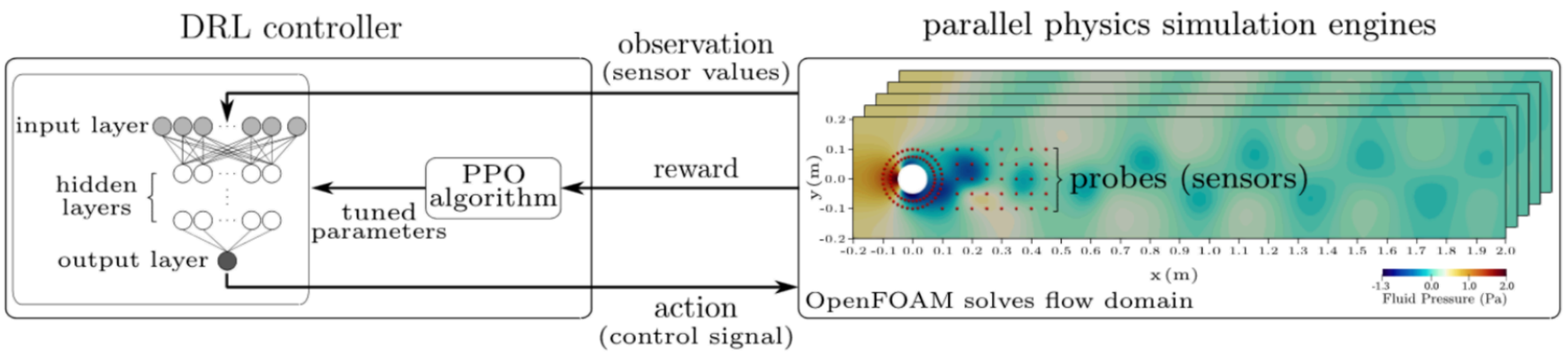
Original software publication

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)

In this talk

- The preCICE library: Couple two toy Python solvers
- The preCICE ecosystem: Couple OpenFOAM + deal.II

Get the slides

github.com/MakisH/vki-training

Organizational notes

1. You are not expected to try things live.
2. Ask questions live, feel free to interrupt me.
3. Find all software installed in a demo virtual machine: precice.org/installation-vm.html
4. Everything presented here is free software. preCICE and all the adapters are developed publicly on GitHub: github.com/precice

Using the preCICE library

Coupling two toy Python solvers

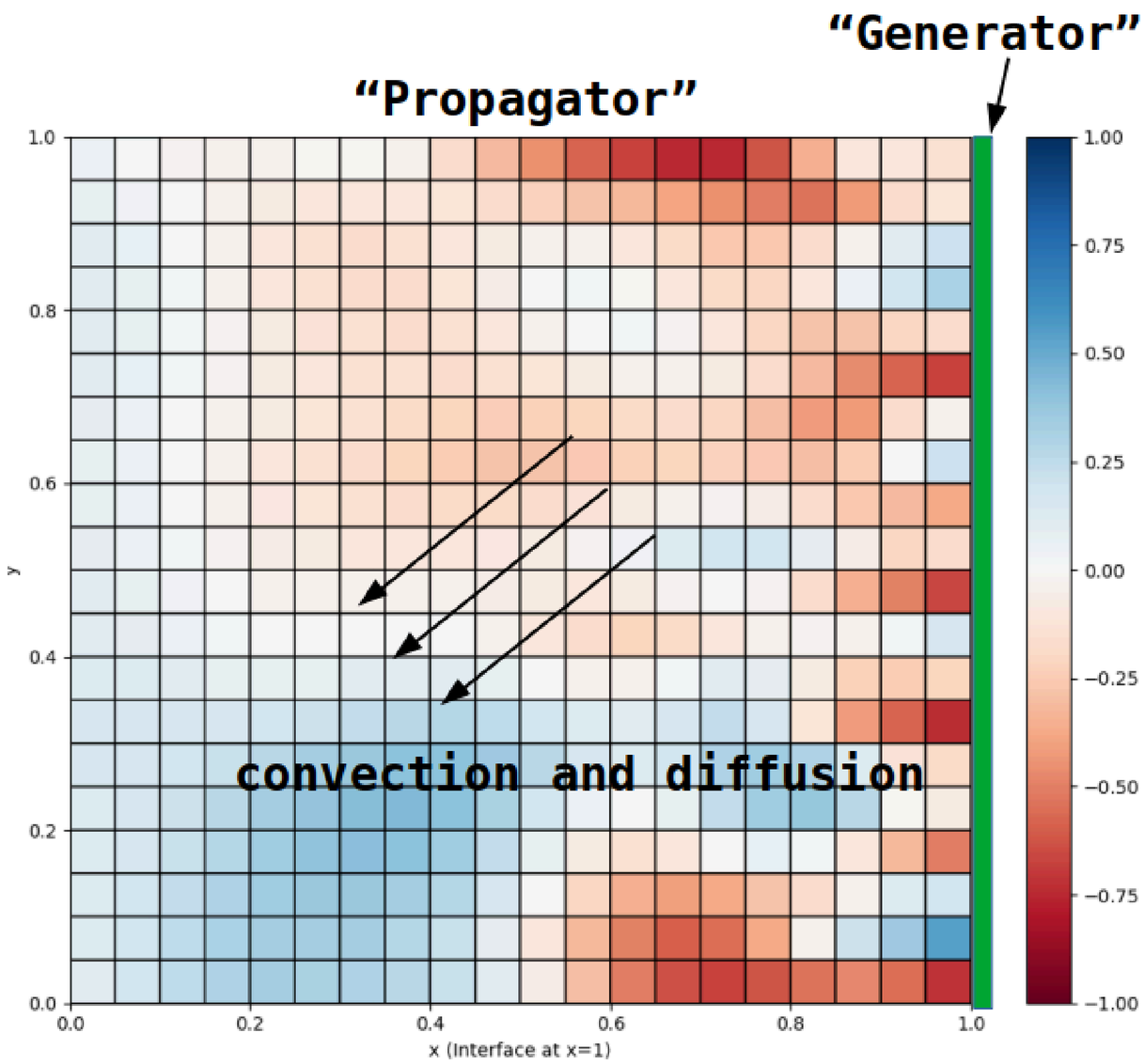
Dependencies

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

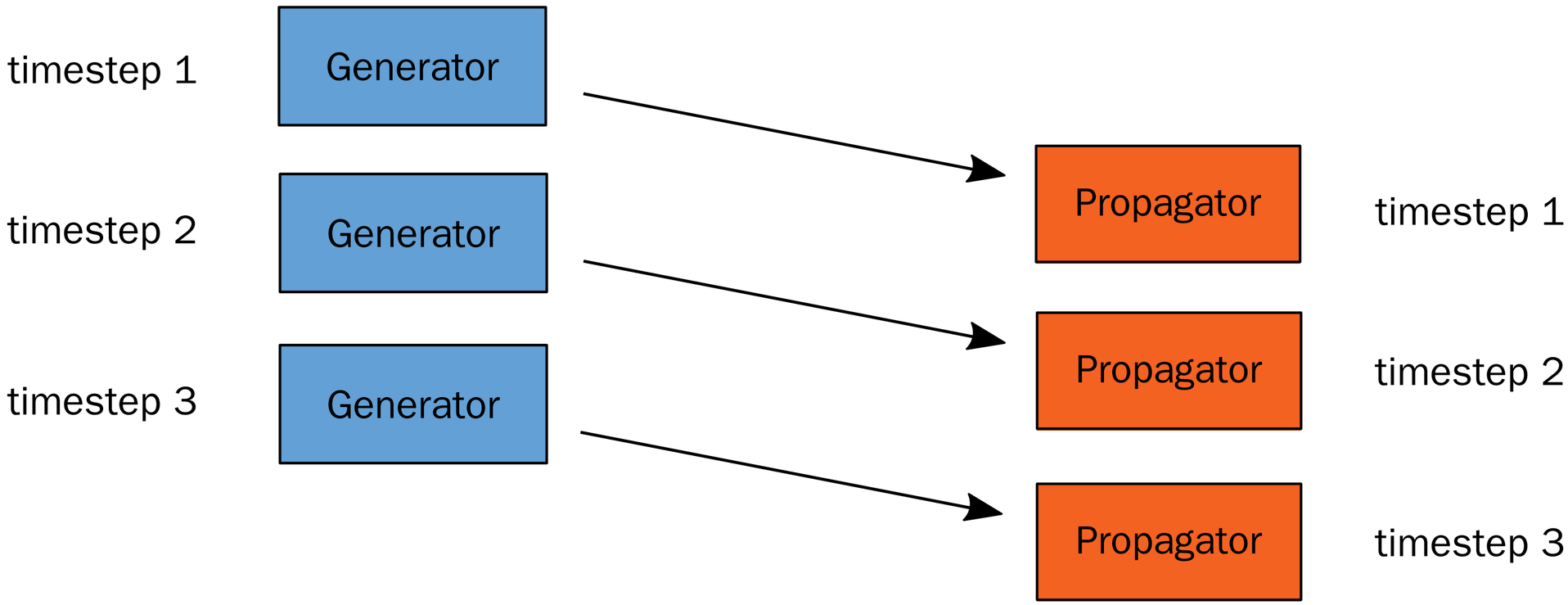
```
pip3 install --upgrade pip
python3 -m venv .venv && . .venv/bin/activate
pip3 install pyprecice
```

"Generator" and "Propagator"

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



Unidirectional coupling

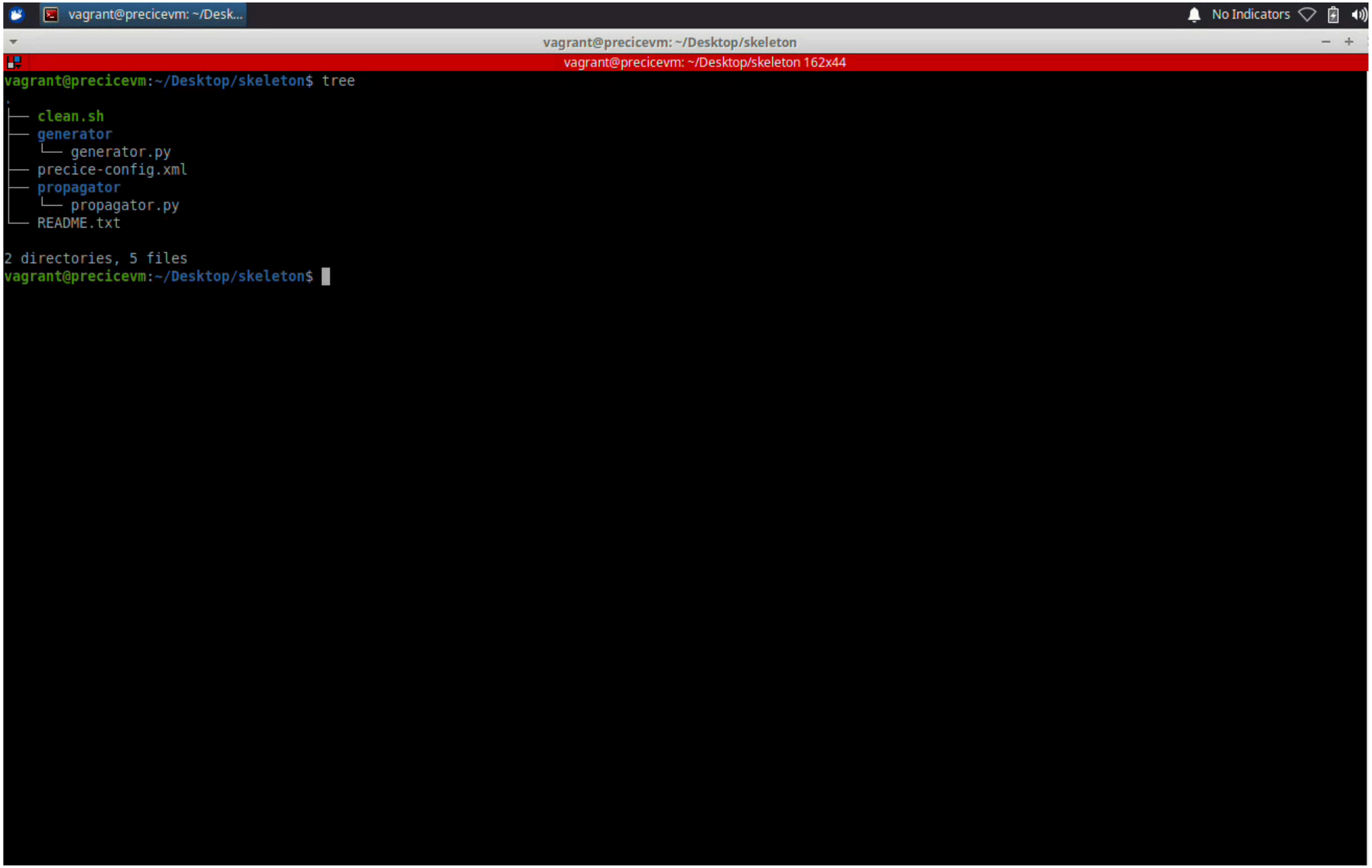


generator.py

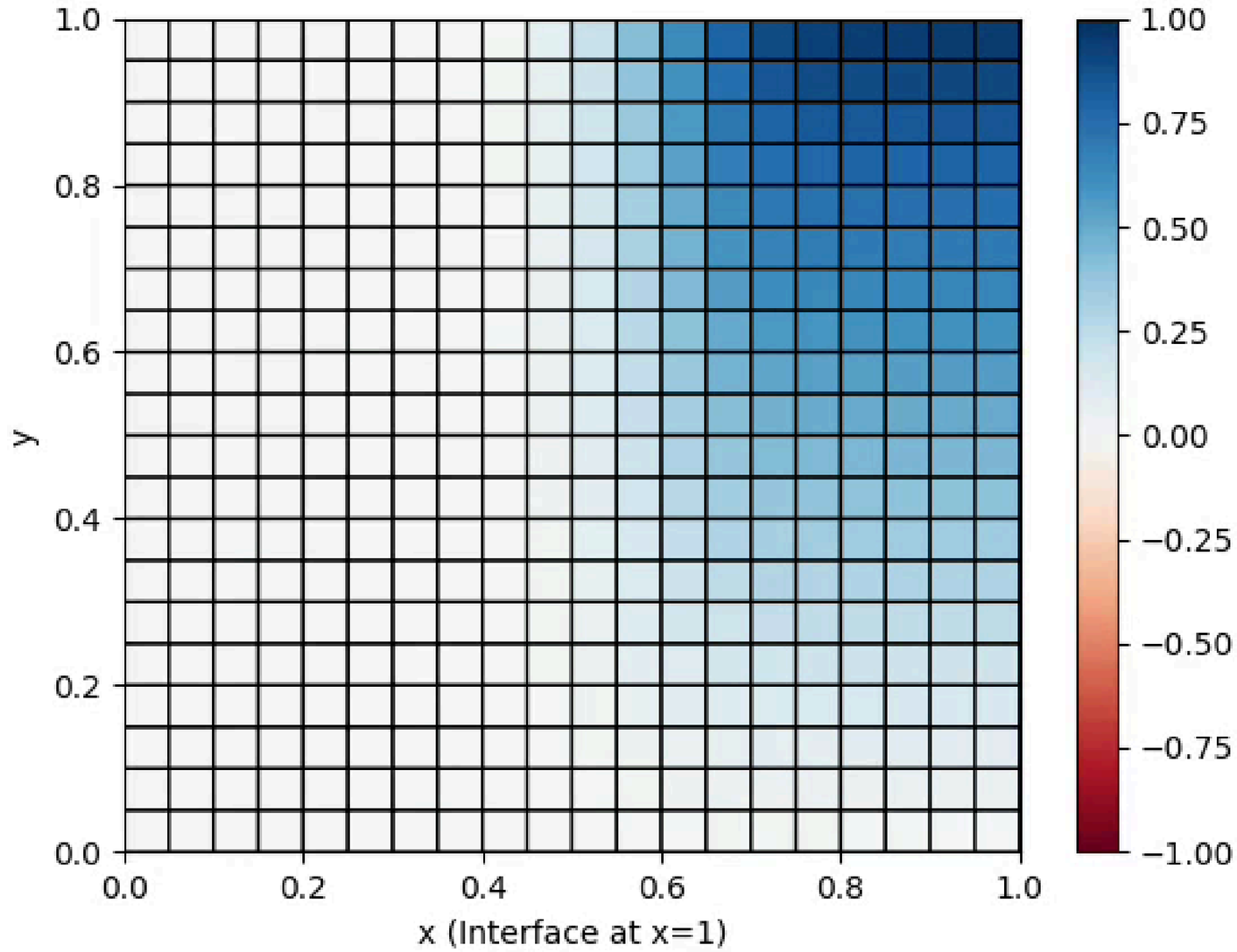
```
import numpy

# 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$ tree
.
├── clean.sh
├── generator
│   └── generator.py
├── precice-config.xml
├── propagator
│   └── propagator.py
└── README.txt
2 directories, 5 files
vagrant@precicevm:~/Desktop/skeleton$

Uncoupled simulation



Import preCICE

```
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

```
 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 # Define the coupling mesh name
22 mesh_name   = "Generator-Mesh"
```

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 vertices   = [[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

```
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 vertices = [[1, y0] for y0 in y[:-1]]
26 vertex_ids = participant.set_mesh_vertices(mesh_name, vertices)
27
28 participant.initialize()
29
30 dt = 0.01
31 t = 0
32
```

Initialize and finalize preCICE

```
20
21 # Define the coupling mesh name
22 mesh_name = "Generator-Mesh"
23
24 # Define the coupling mesh
25 vertices = [[1, y0] for y0 in y[:-1]]
26 vertex_ids = participant.set_mesh_vertices(mesh_name, vertices)
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

```
20
21 # Define the coupling mesh name
22 mesh_name = "Generator-Mesh"
23
24 # Define the coupling mesh
25 vertices = [[1, y0] for y0 in y[:-1]]
26 vertex_ids = participant.set_mesh_vertices(mesh_name, vertices)
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

```
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

```
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   participant.advance(dt)
38
39   t = t + dt
40   if(t > 0.1):
41     break
42
43 participant.finalize()
```

Advance the coupling

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

Advance the coupling

```
28 participant.initialize()
29
30 dt = 0.01
31 t = 0
32
33 while participant.is_coupling_ongoing():
34   print("Generating data")
35   u = 1 - 2 * numpy.random.rand(n)
36
37   participant.advance(dt)
38   precice_dt = participant.get_max_time_step_size()
39
40   t = t + dt
41
42
43
44 interface.finalize()
```

Advance the coupling

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

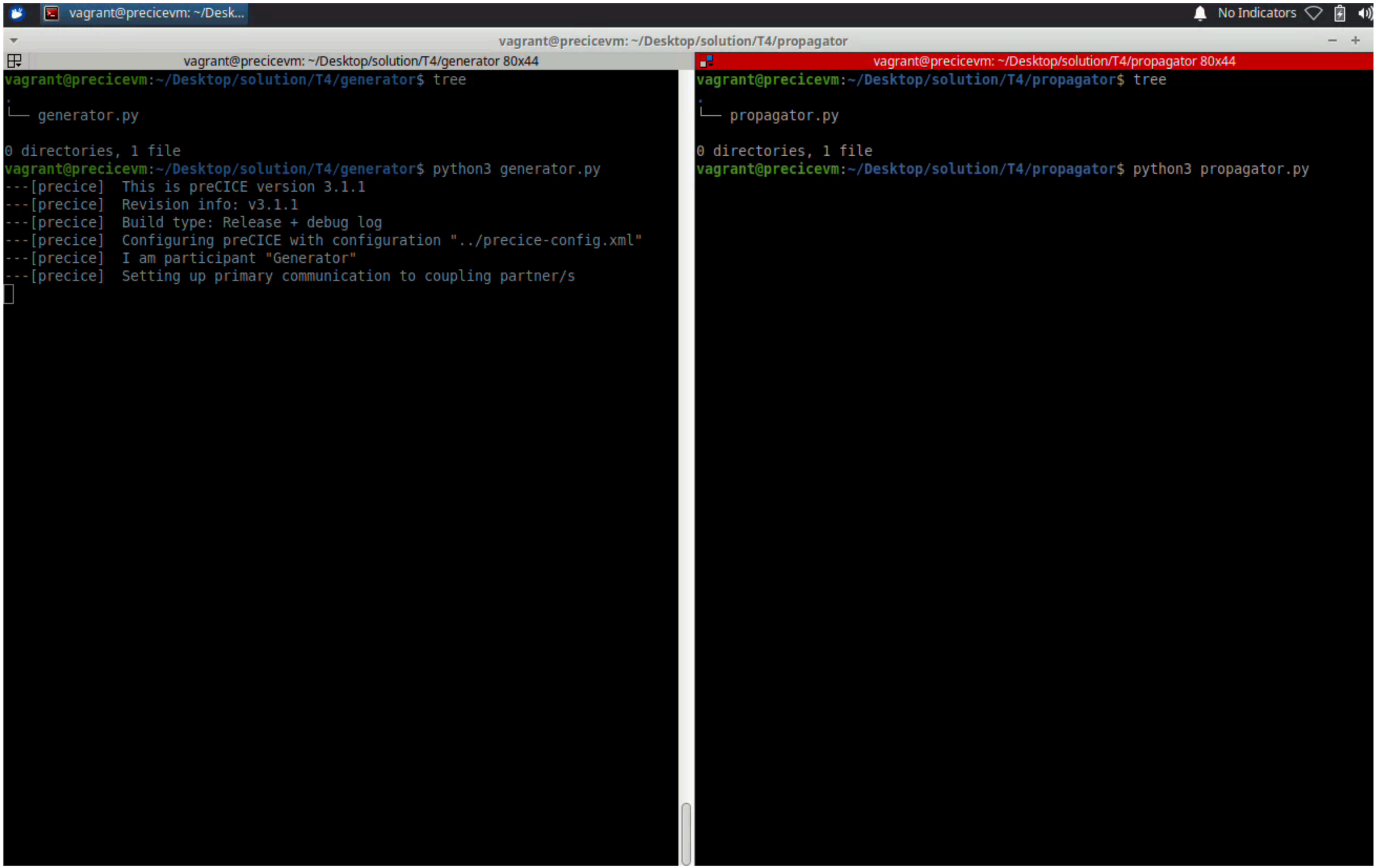
Advance the coupling

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

Not there yet, but let's run it

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

Nothing happening?



Steering works. But did we forget something?

Write & read data

```
20
21 # Get the preCICE mesh name
22 mesh_name  = "Generator-Mesh"
23
24 # Define the coupling mesh
25 vertices   = [[1, y0] for y0 in y[:-1]]
26 vertex_ids = participant.set_mesh_vertices(mesh_name, vertices)
27
28
29
30
31
32 participant.initialize()
33 precice_dt = participant.get_max_time_step_size()
34
35 dt = 0.01
36 t = 0
```

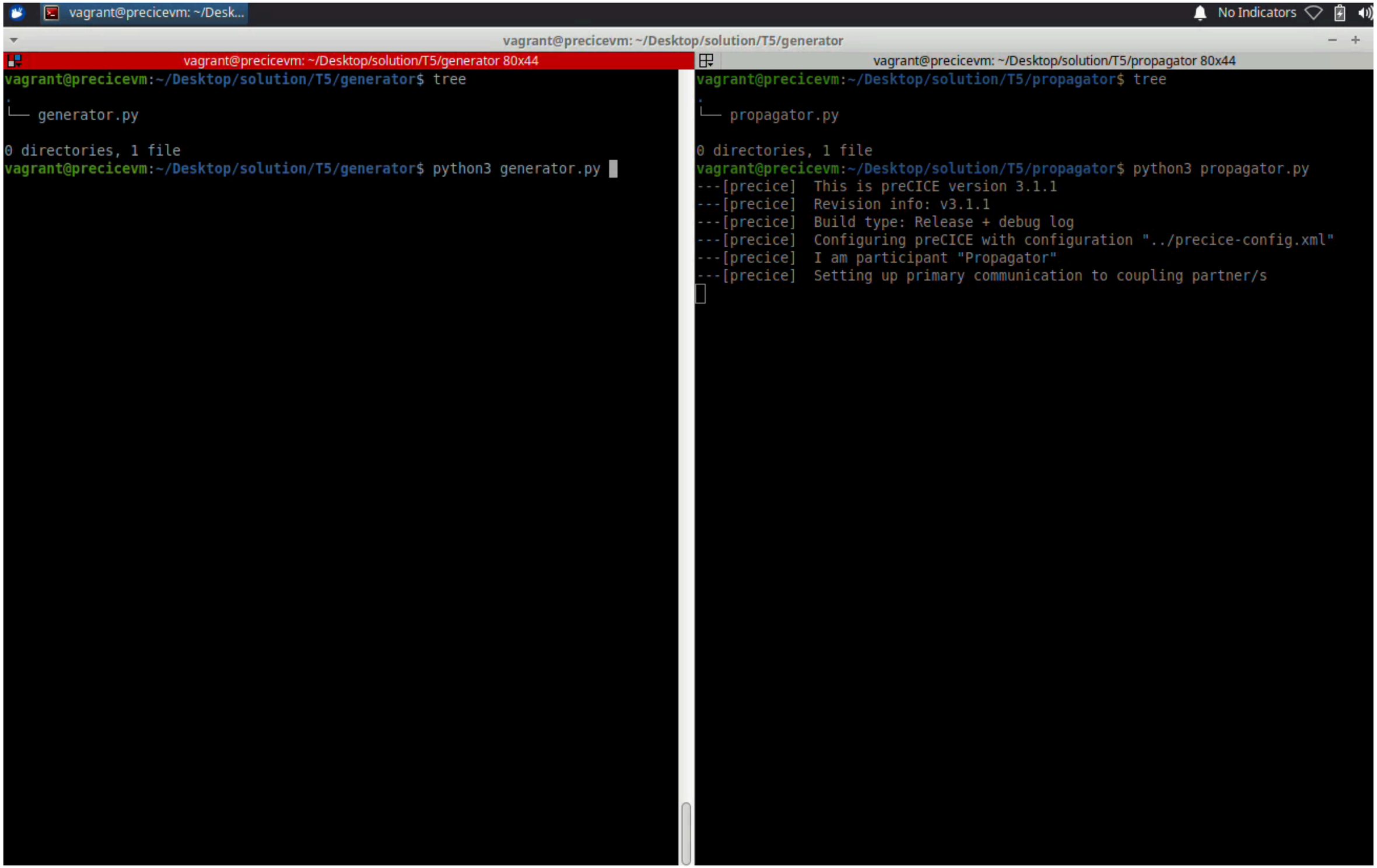
Write & read data

```
20
21 # Get the preCICE mesh name
22 mesh_name  = "Generator-Mesh"
23
24 # Define the coupling mesh
25 vertices   = [[1, y0] for y0 in y[:-1]]
26 vertex_ids = participant.set_mesh_vertices(mesh_id, vertices)
27
28 # Get the exchanged data name
29 data_name = "Data"
30
31 participant.initialize()
32
33 dt = 0.01
34 t = 0
35
36 while participant.is_coupling_ongoing():
```

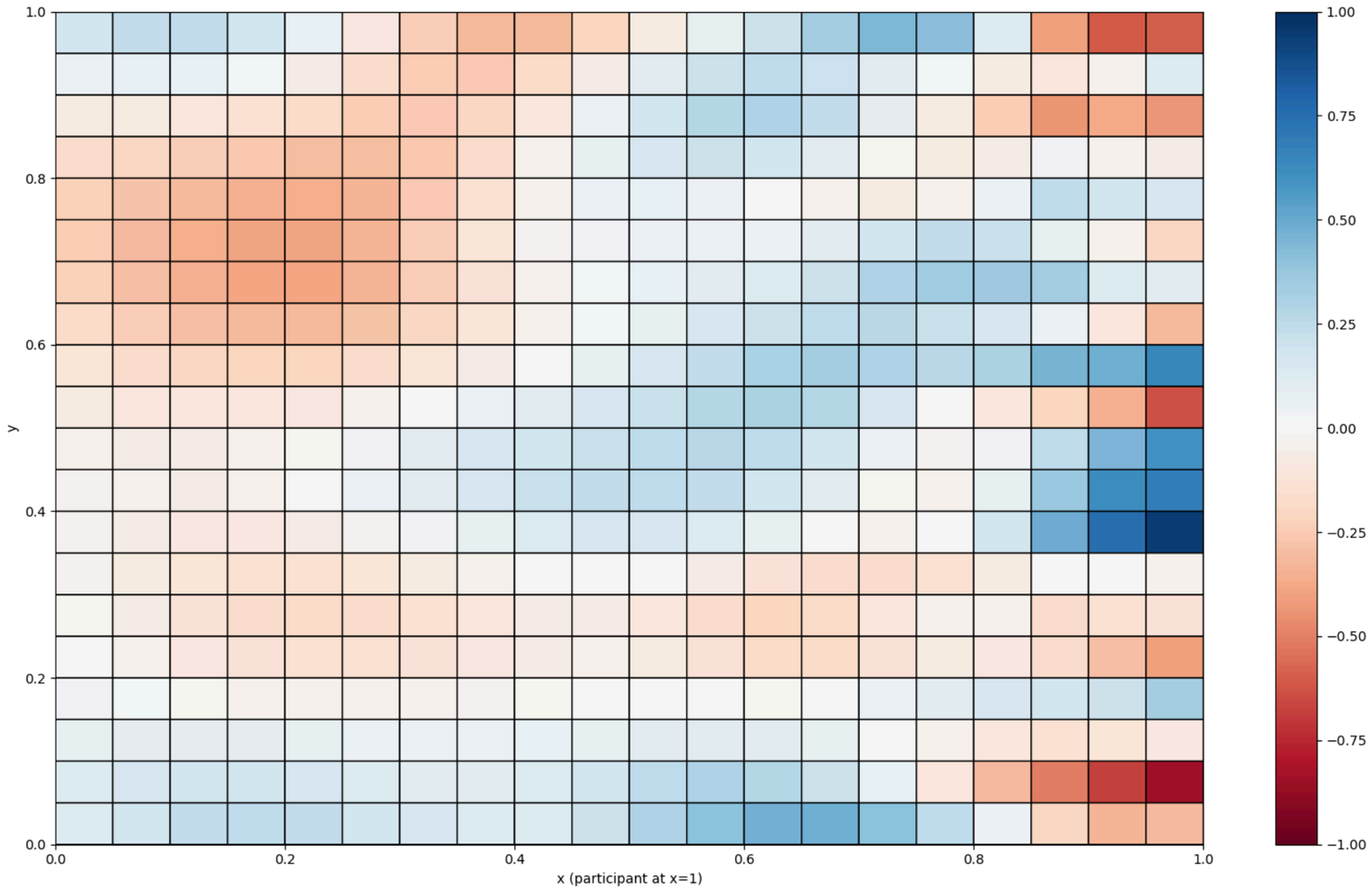
Write & read data

```
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!

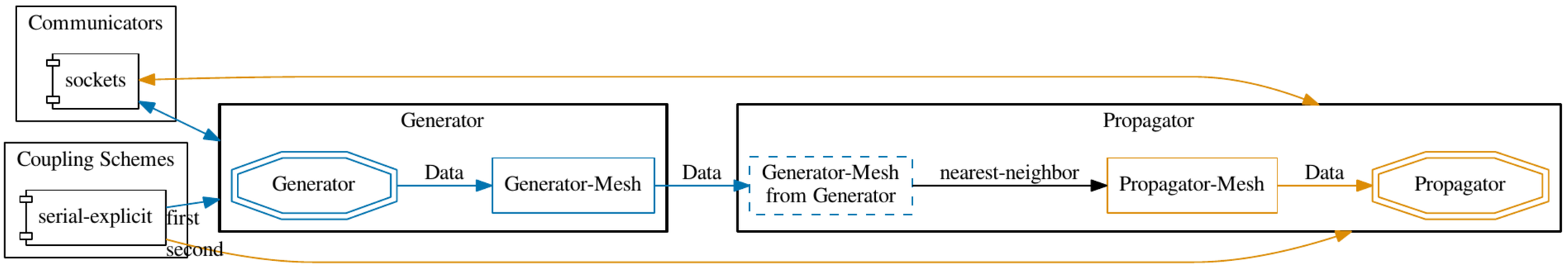

vagrant@precicevm: ~/Desk...
No Indicators
vagrant@precicevm: ~/Desktop/solution/T5/generator
vagrant@precicevm: ~/Desktop/solution/T5/generator 80x44
vagrant@precicevm:~/Desktop/solution/T5/generator$ tree
.
└── generator.py
0 directories, 1 file
vagrant@precicevm:~/Desktop/solution/T5/generator$ python3 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
---[precice] This is preCICE version 3.1.1
---[precice] Revision info: v3.1.1
---[precice] Build type: Release + debug log
---[precice] Configuring preCICE with configuration "../precice-config.xml"
---[precice] I am participant "Propagator"
---[precice] Setting up primary communication to coupling partner/s

Data is being transfered!



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

preCICE configuration



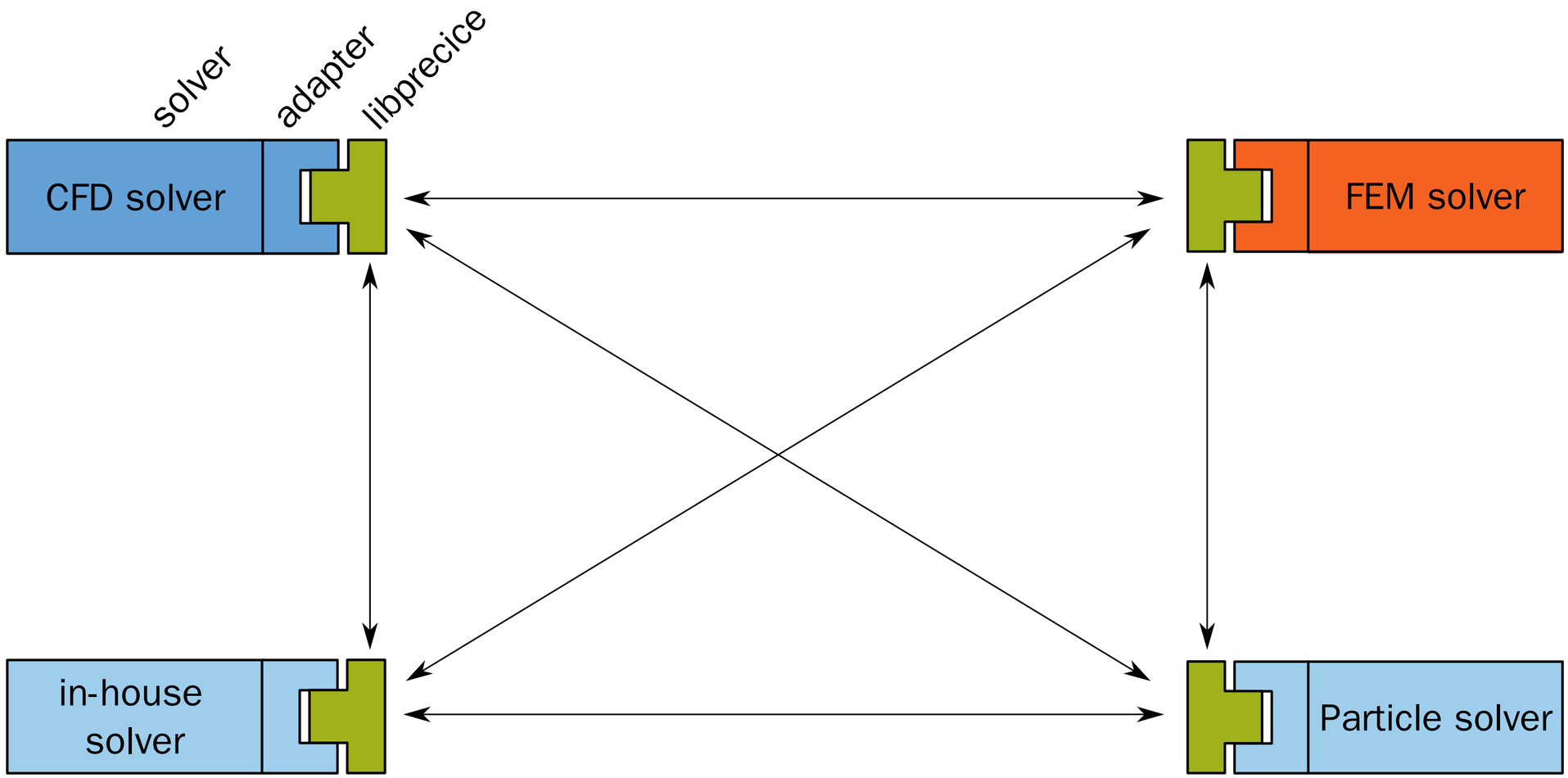
Visual representation of precice-config.xml using the config visualizer.

Using the preCICE ecosystem

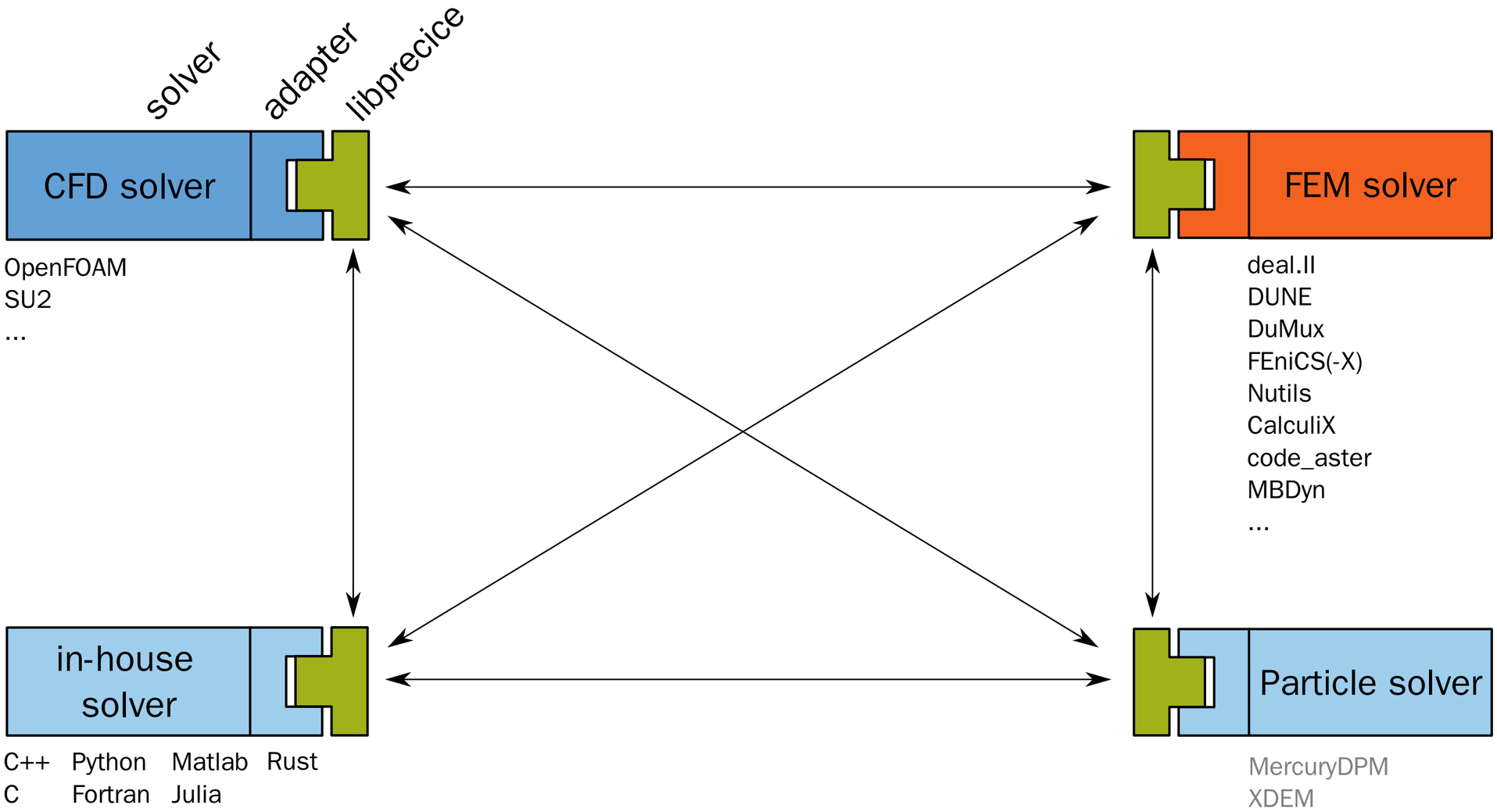
Example: Coupling OpenFOAM and deal.II

The big picture

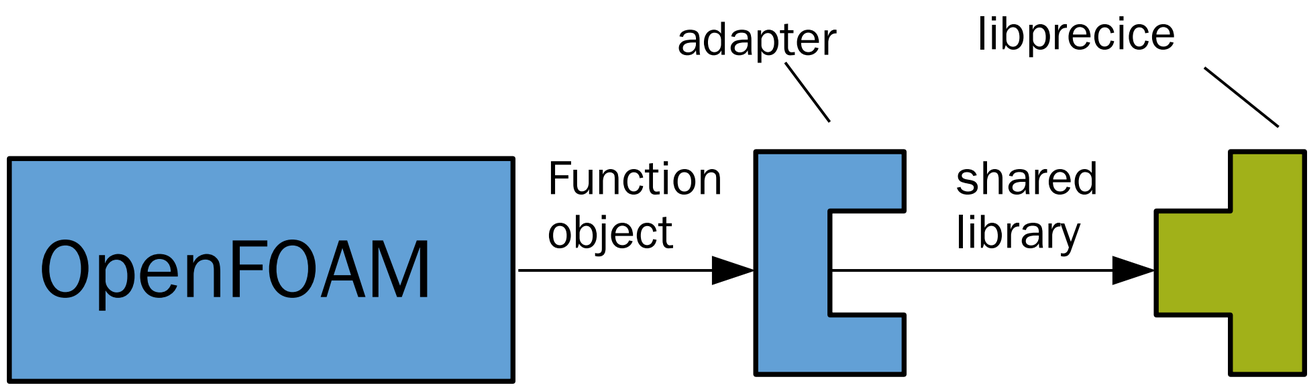
The big picture



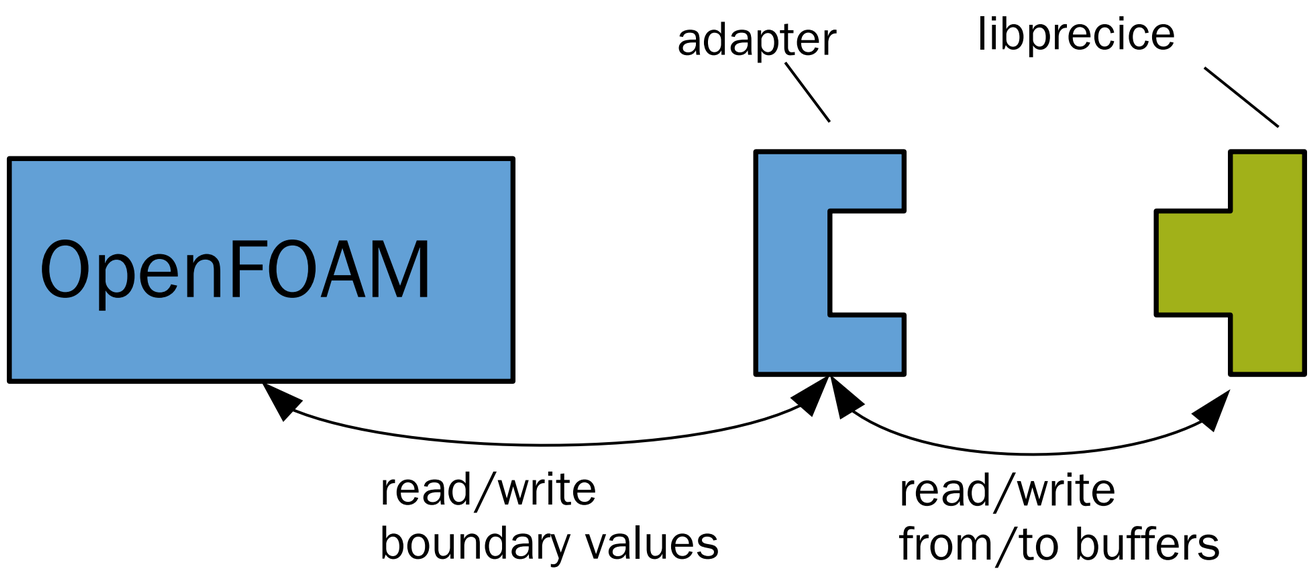
The big picture



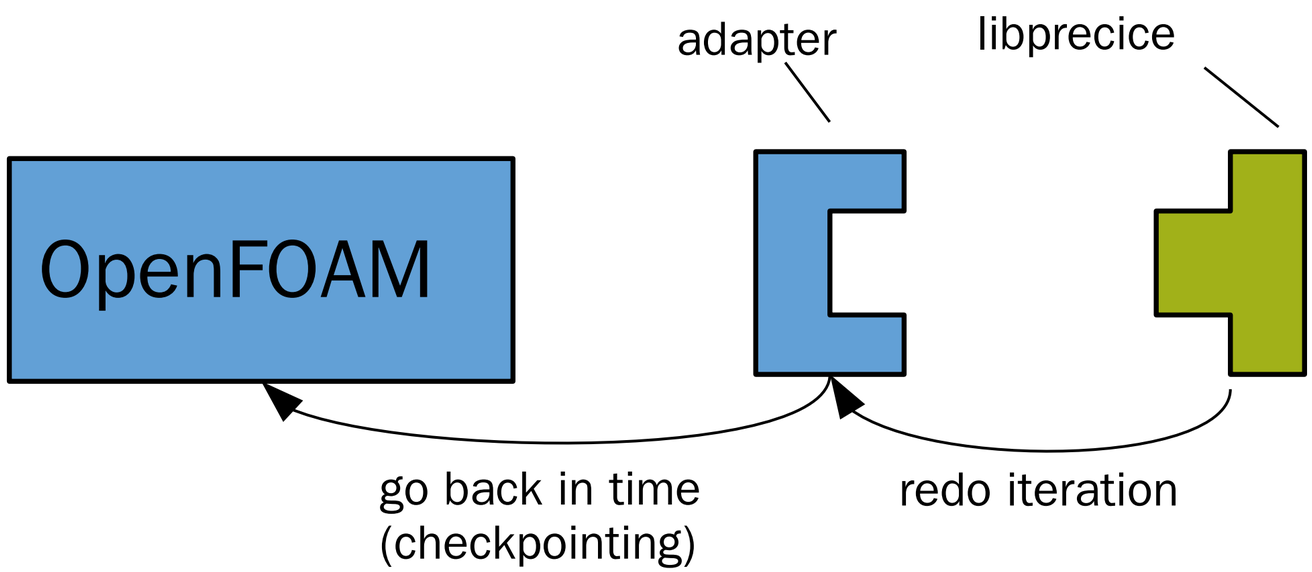
What does the adapter do?



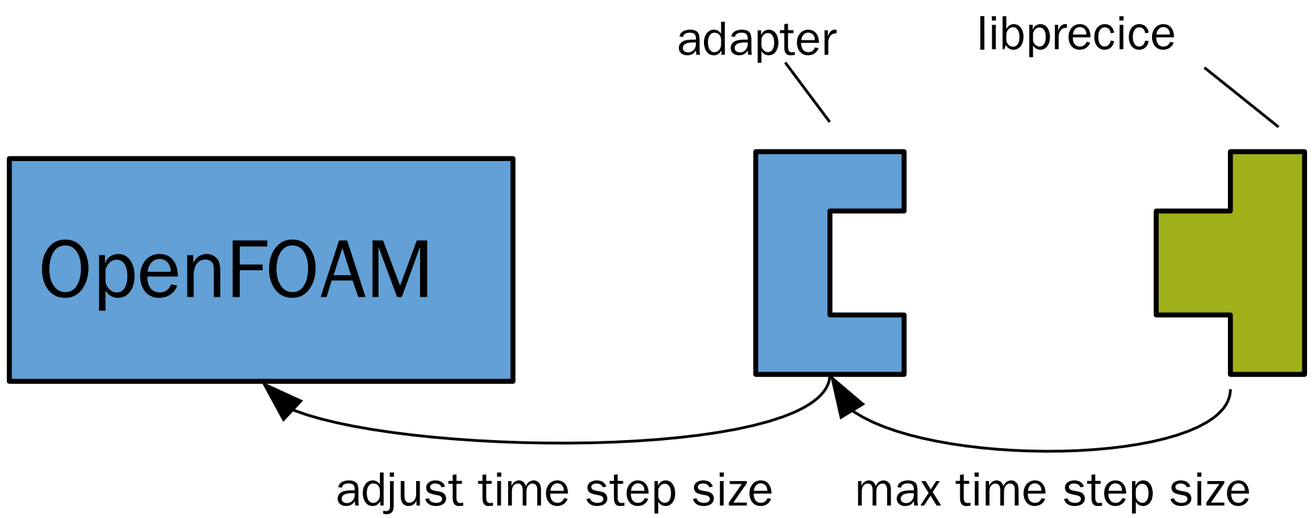
What does the adapter do?



What does the adapter do?



What does the adapter do?



Quickstart

Summary: Install preCICE on Linux (e.g. via a Debian package) and couple an OpenFOAM fluid solver (using the OpenFOAM-preCICE adapter) with an example rigid body solver in C++.

Edit me

Updated 24 Jan 25

This is the first step you may want to try if you are new to preCICE: install preCICE and some solvers, and run a simple coupled case.

To get a feeling what preCICE does, watch a short presentation or a longer talk on the fundamentals.

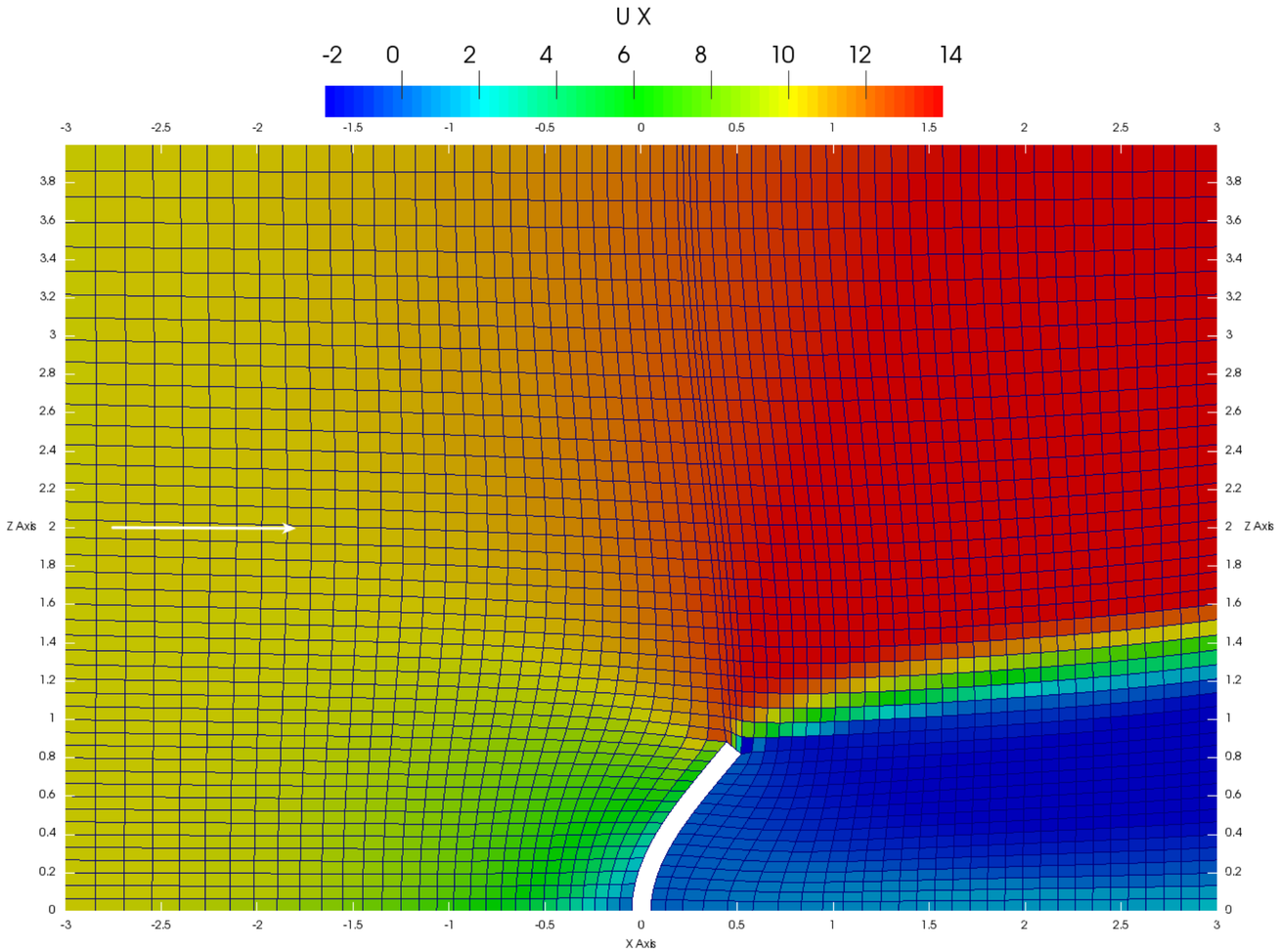
Installation

1. Get and install preCICE. For Ubuntu 24.04 (Noble Numbat), this is pretty easy: download and install our binary package by clicking on it or using the following commands:

```
wget https://github.com/precice/precice/releases/download/v3.1.2/libprecice3_3.1.2_noble.deb
sudo apt install ./libprecice3_3.1.2_noble.deb
```

OS	Package
Ubuntu 20.04 Focal Fossa	libprecice3_3.1.2_focal.deb
Ubuntu 22.04 Jammy Jellyfish	libprecice3_3.1.2_jammy.deb
Ubuntu 24.04 Noble Numbat	libprecice3_3.1.2_noble.deb

Tutorial: Channel with a perpendicular flap

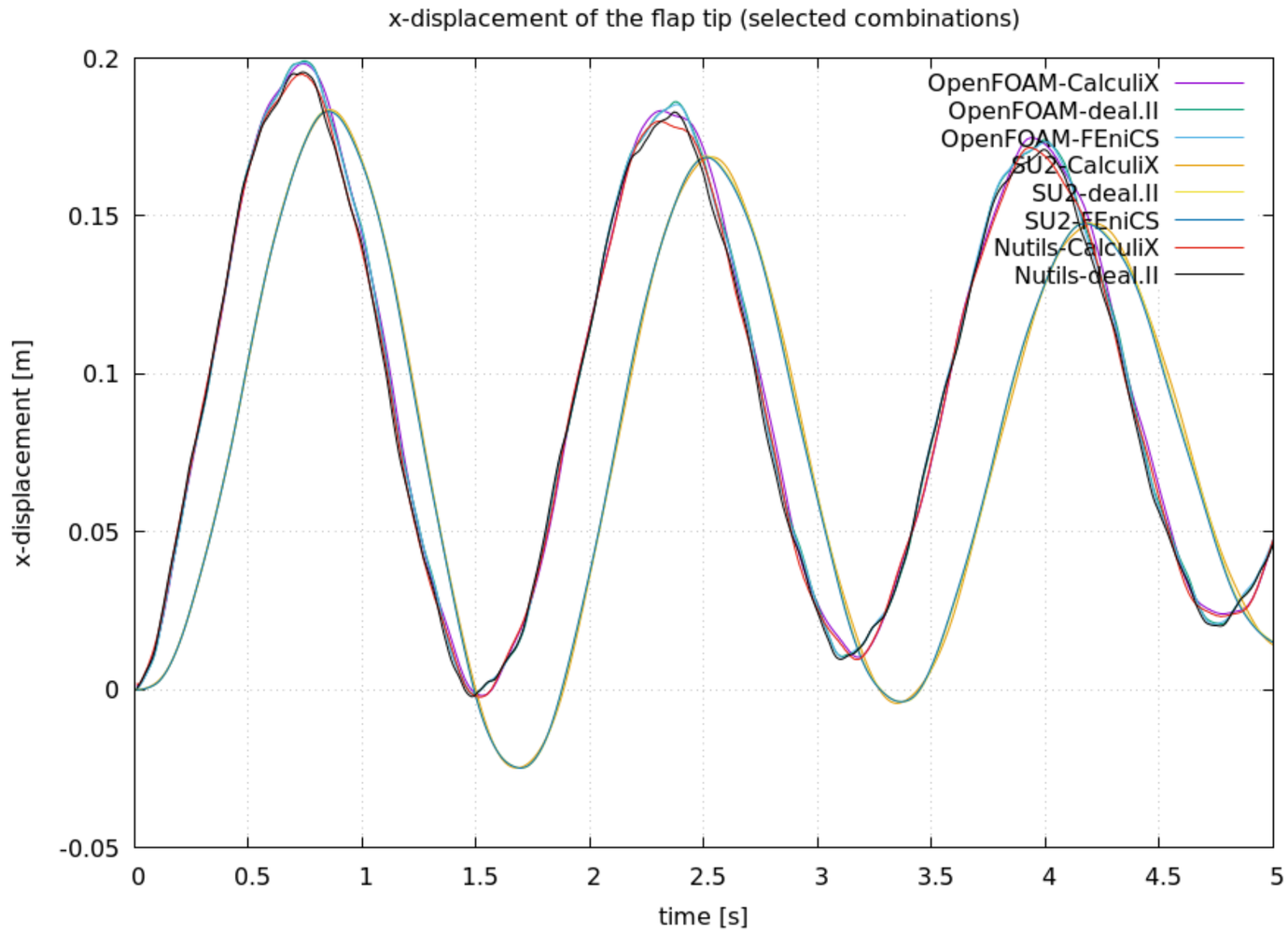


Find this tutorial on precice.org/tutorials-perpendicular-flap.html.

Arbitrary solver combinations

FLUID	SOLID
pimpleFoam	CalculiX
SU2	deal.II
Nutils	FEniCS
	DUNE
	solidDisplacementFoam
	solids4Foam
	Nutils

Arbitrary solver combinations



Dependencies

- preCICE v3 (e.g. packages for Ubuntu)
- Recent OpenFOAM (e.g., v2312)
- OpenFOAM-preCICE adapter v1.3.0
- deal.II 9.2 or greater
- deal.II-preCICE adapter 9.5.0

File structure

```
- precice-config.xml
- fluid-openfoam/
    - 0/U ...
    - constant/dynamicMeshDict ...
    - system/
      - controlDict
      - preciceDict
    - ...
- solid-dealii/
    - parameters.prm
    - ...
```

For demonstration purposes

- In `precice-config.xml`:
  - Reduce the `max-time` from 5s to 1.5s
  - Switch to a serial-explicit coupling scheme
- In `solid-dealii/parameters.prm`:
  - Lower the solid density from 3000 to 42

Configure OpenFOAM

```
// fluid-openfoam/0/U

flap
{
    type            movingWallVelocity;
    value           uniform (0 0 0);
}
```

```
// fluid-openfoam/0/pointDisplacement

flap
{
    type            fixedValue;
    value           $internalField;
}
```

```
// fluid-openfoam/constant/dynamicMeshDict

solver      displacementLaplacian;
```

Load the OpenFOAM adapter

```
// fluid-openfoam/system/controlDict
functions
{
    preCICE_Adapter
    {
        type preciceAdapterFunctionObject;
        libs ("libpreciceAdapterFunctionObject.so");
    }
}

// Don't forget to build the adapter with Allwmake
```

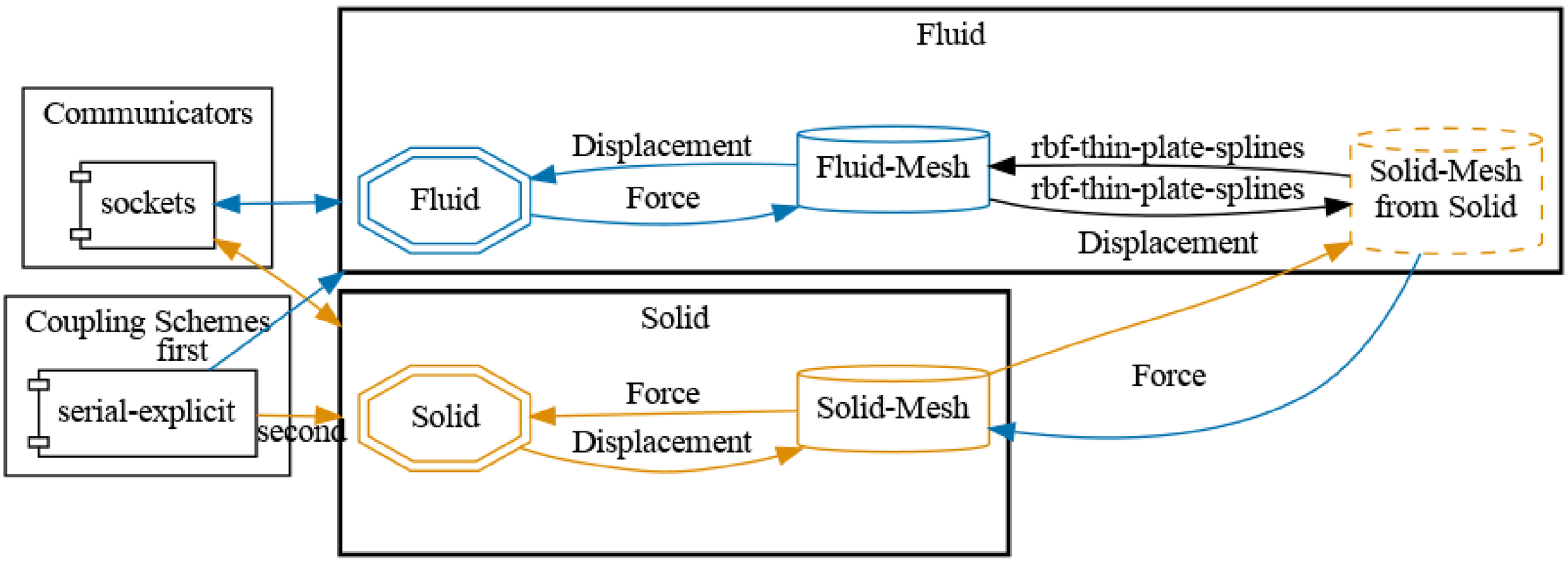
Configure the OpenFOAM adapter

```
// fluid-openfoam/system/preciceDict
preciceConfig "../precice-config.xml";
participant Fluid;

modules (FSI);

interfaces {
  Interface1 {
    mesh Fluid-Mesh;
    patches (flap);
    locations faceCenters;

    readData (Displacement);

    writeData (Force);
  };
};
```

Configure the dealii adapter

```
// solid-dealii/parameters.prm

subsection precice configuration
# Cases: FSI3 or PF for perpendicular flap
set Scenario = PF

# Name of the precice configuration file
set precice config-file = ../precice-config.xml

# Name of the participant in the precice-config.xml file
set Participant name = Solid

# Name of the coupling mesh in the precice-config.xml file
set Mesh name = Solid-Mesh

# Name of the read data in the precice-config.xml file
set Read data name = Force
```

Configure preCICE



Configure preCICE

```
<data:vector name="Force" />
<data:vector name="Displacement" />

<mesh name="Fluid-Mesh" dimensions="2">
  <use-data name="Force" />
  <use-data name="Displacement" />
</mesh>

<mesh name="Solid-Mesh" dimensions="2">
  <use-data name="Displacement" />
  <use-data name="Force" />
</mesh>

<participant name="Fluid">
  <export:vtk directory="results" />
  <provide-mesh name="Fluid-Mesh" />
  <receive-mesh name="Solid-Mesh" from="Solid" />
```

Notice the serial-explicit coupling scheme

Configure preCICE

```
<data:vector name="Force" />
<data:vector name="Displacement" />

<mesh name="Fluid-Mesh" dimensions="2">
  <use-data name="Force" />
  <use-data name="Displacement" />
</mesh>

<mesh name="Solid-Mesh" dimensions="2">
  <use-data name="Displacement" />
  <use-data name="Force" />
</mesh>

<participant name="Fluid">
  <export:vtk directory="results" />
  <provide-mesh name="Fluid-Mesh" />
  <receive-mesh name="Solid-Mesh" from="Solid" />
```

Notice the serial-explicit coupling scheme

Configure preCICE

```
14 <participant name="Fluid">
15   <export:vtk directory="results" />
16   <provide-mesh name="Fluid-Mesh" />
17   <receive-mesh name="Solid-Mesh" from="Solid" />
18   <write-data name="Force" mesh="Fluid-Mesh" />
19   <read-data name="Displacement" mesh="Fluid-Mesh" />
20   <mapping:rbf direction="write"
21     from="Fluid-Mesh" to="Solid-Mesh"
22     constraint="conservative">
23     <basis-function:compact-polynomial-c6 support-radius="1." />
24   </mapping:rbf>
25   <mapping:rbf direction="read"
26     from="Solid-Mesh" to="Fluid-Mesh"
27     constraint="consistent">
28     <basis-function:compact-polynomial-c6 support-radius="1." />
29   </mapping:rbf>
30 </participant>
```

Notice the serial-explicit coupling scheme

Configure preCICE

```
14 <participant name="Fluid">
15   <export:vtk directory="results" />
16   <provide-mesh name="Fluid-Mesh" />
17   <receive-mesh name="Solid-Mesh" from="Solid" />
18   <write-data name="Force" mesh="Fluid-Mesh" />
19   <read-data name="Displacement" mesh="Fluid-Mesh" />
20   <mapping:rbf direction="write"
21     from="Fluid-Mesh" to="Solid-Mesh"
22     constraint="conservative">
23     <basis-function:compact-polynomial-c6 support-radius="1." />
24   </mapping:rbf>
25   <mapping:rbf direction="read"
26     from="Solid-Mesh" to="Fluid-Mesh"
27     constraint="consistent">
28     <basis-function:compact-polynomial-c6 support-radius="1." />
29   </mapping:rbf>
30 </participant>
```

Notice the serial-explicit coupling scheme

Configure preCICE

```
14 <participant name="Fluid">
15   <export:vtk directory="results" />
16   <provide-mesh name="Fluid-Mesh" />
17   <receive-mesh name="Solid-Mesh" from="Solid" />
18   <write-data name="Force" mesh="Fluid-Mesh" />
19   <read-data name="Displacement" mesh="Fluid-Mesh" />
20   <mapping:rbf direction="write"
21     from="Fluid-Mesh" to="Solid-Mesh"
22     constraint="conservative">
23     <basis-function:compact-polynomial-c6 support-radius="1." />
24   </mapping:rbf>
25   <mapping:rbf direction="read"
26     from="Solid-Mesh" to="Fluid-Mesh"
27     constraint="consistent">
28     <basis-function:compact-polynomial-c6 support-radius="1." />
29   </mapping:rbf>
30 </participant>
```

Notice the serial-explicit coupling scheme

Configure preCICE

```
31
32 <participant name="Solid">
33   <provide-mesh name="Solid-Mesh" />
34   <receive-data name="Displacement" mesh="Solid-Mesh" />
35   <read-data name="Force" mesh="Solid-Mesh" />
36   <watch-point mesh="Solid-Mesh" name="Flap-Tip" coordinate="0.0;1" /
37 </participant>
38
39 <m2n:sockets acceptor="Fluid" connector="Solid"
40              exchange-directory=".." /> <!-- initial handshake -->
41
42 <coupling-scheme:serial-explicit>
43   <participants first="Fluid" second="Solid" />
44   <max-time value="1.5" />
45   <time-window-size value="0.01" />
46   <exchange data="Force" mesh="Solid-Mesh" from="Fluid" to="Solid" />
47   <exchange data="Displacement" mesh="Solid-Mesh" from="Solid" to="Fl
48 </coupling-scheme:serial-explicit>
```

Notice the serial-explicit coupling scheme

Configure preCICE

```
32 <participant name="Solid">
33   <provide-mesh name="Solid-Mesh" />
34   <receive-data name="Displacement" mesh="Solid-Mesh" />
35   <read-data name="Force" mesh="Solid-Mesh" />
36   <watch-point mesh="Solid-Mesh" name="Flap-Tip" coordinate="0.0;1" /
37 </participant>
38
39 <m2n:sockets acceptor="Fluid" connector="Solid"
40              exchange-directory=".." /> <!-- initial handshake -->
41
42 <coupling-scheme:serial-explicit>
43   <participants first="Fluid" second="Solid" />
44   <max-time value="1.5" />
45   <time-window-size value="0.01" />
46   <exchange data="Force" mesh="Solid-Mesh" from="Fluid" to="Solid" />
47   <exchange data="Displacement" mesh="Solid-Mesh" from="Solid" to="Fl
48 </coupling-scheme:serial-explicit>
```

Notice the serial-explicit coupling scheme

vagrant@precicevm: ~/tutor... fsi-course - File Manager No Indicators

vagrant@precicevm: ~/tutorials/fsi-course/fluid-openfoam

vagrant@precicevm: ~/tutorials/fsi-course 160x3

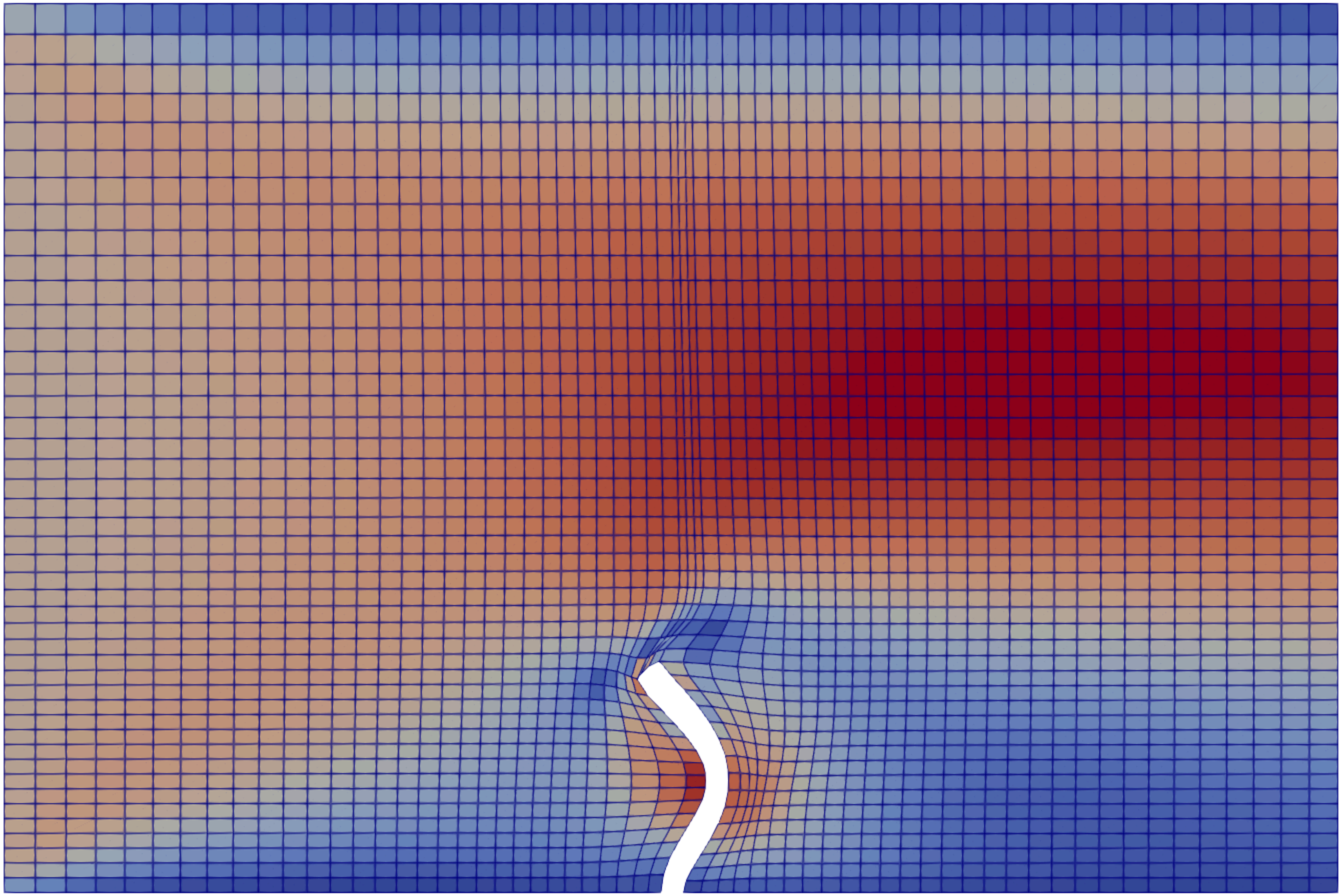
```
vagrant@precicevm:~/tutorials/fsi-course$ ls
clean-tutorial.sh  fluid-openfoam  plot-displacement.sh  precice-config.xml  solid-calculix
vagrant@precicevm:~/tutorials/fsi-course$
```

vagrant@precicevm: ~/tutorials/fsi-course/fluid-openfoam 79x44

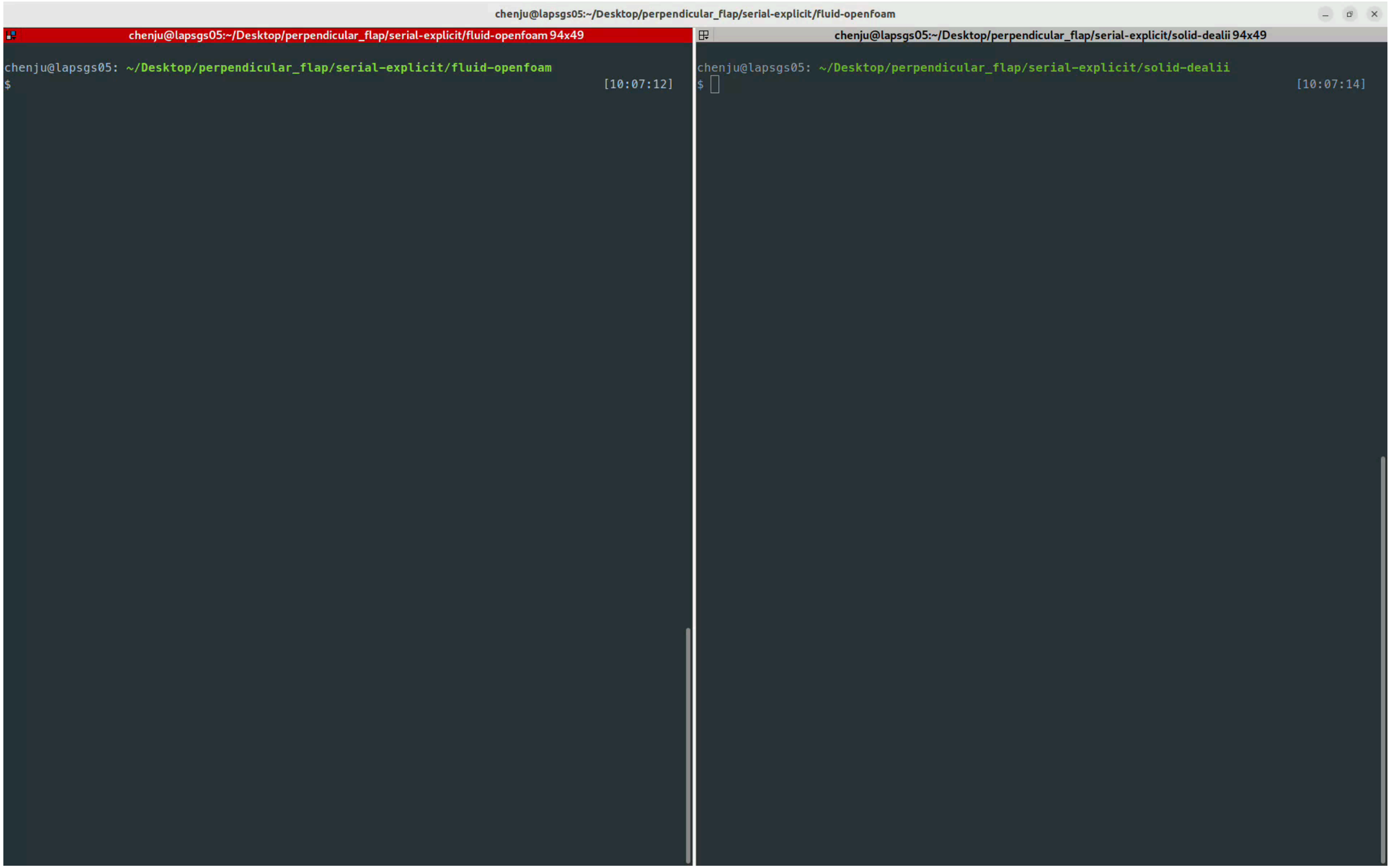
```
vagrant@precicevm:~/tutorials/fsi-course/fluid-openfoam$ tree
.
├── 0
│   ├── p
│   ├── phi
│   ├── pointDisplacement
│   └── U
├── clean.sh
├── constant
│   ├── dynamicMeshDict
│   ├── transportProperties
│   └── turbulenceProperties
├── run.sh
└── system
    ├── blockMeshDict
    ├── controlDict
    ├── decomposeParDict
    ├── fvSchemes
    ├── fvSolution
    └── preciceDict

3 directories, 15 files
vagrant@precicevm:~/tutorials/fsi-course/fluid-openfoam$ ./run.sh
```

vagrant@precicevm: ~/tutorials/fsi-course/solid-calculix 79x44

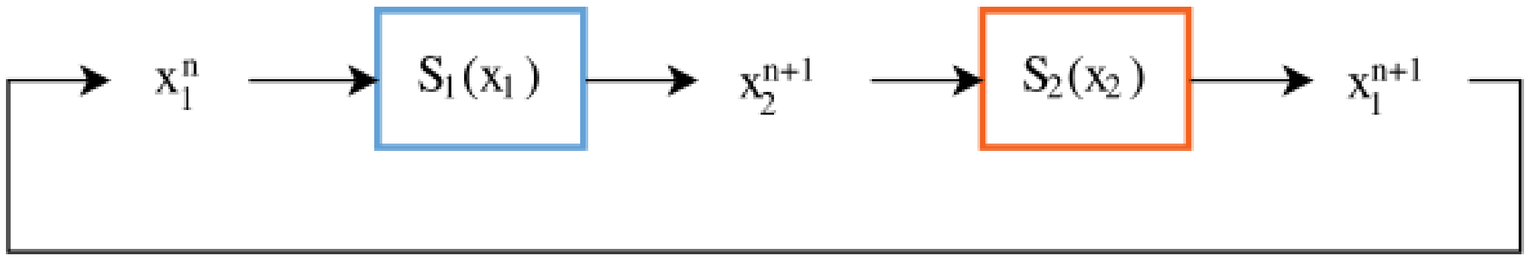
```
vagrant@precicevm:~/tutorials/fsi-course/solid-calculix$ ls
all.msh    config.yml     flap.inp        frequency.inp       run.sh
clean.sh   fix1_beam.nam  flap_modal.inp  interface_beam.nam
vagrant@precicevm:~/tutorials/fsi-course/solid-calculix$ ./run.sh
```



When we decrease the solid density further to 1, the participants have stronger coupling, then:

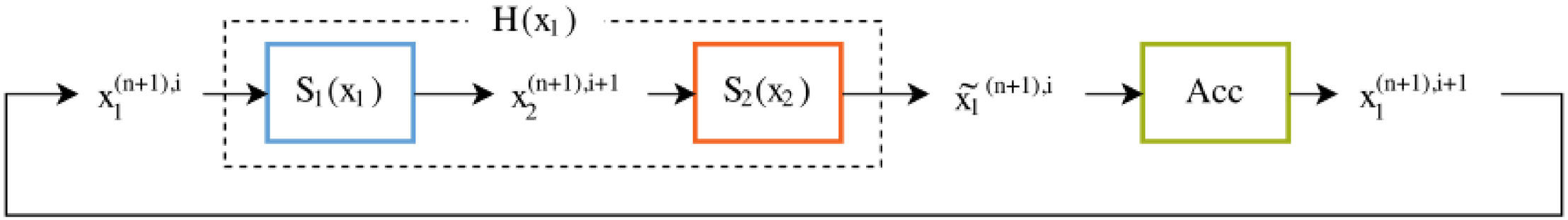


What we did so far: serial-explicit scheme



$n \leftarrow n + 1$

Let's try an implicit coupling scheme



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

Configure the coupling scheme

```
<coupling-scheme:serial-implicit>
  <participants first="Fluid" second="Solid" />
  <max-time value="1.5" />
  <time-window-size value="0.01" />
  <exchange data="Force" mesh="Solid-Mesh" from="Fluid" to="Solid" />
  <exchange data="Displacement" mesh="Solid-Mesh" from="Solid" to="Fl
  <relative-convergence-measure limit="5e-3" data="Displacement" mesh
  <relative-convergence-measure limit="5e-3" data="Force" mesh="Solid
  <max-iterations value="50" />
</coupling-scheme:serial-implicit>
```

Only change from explicit to implicit coupling

Configure the coupling scheme

```
<coupling-scheme:serial-implicit>
  <participants first="Fluid" second="Solid" />
  <max-time value="1.5" />
  <time-window-size value="0.01" />
  <exchange data="Force" mesh="Solid-Mesh" from="Fluid" to="Solid" />
  <exchange data="Displacement" mesh="Solid-Mesh" from="Solid" to="Fl
  <relative-convergence-measure limit="5e-3" data="Displacement" mesh
  <relative-convergence-measure limit="5e-3" data="Force" mesh="Solid
  <max-iterations value="50" />
</coupling-scheme:serial-implicit>
```

Only change from explicit to implicit coupling

Configure the coupling scheme

```
<coupling-scheme:serial-implicit>
  <participants first="Fluid" second="Solid" />
  <max-time value="1.5" />
  <time-window-size value="0.01" />
  <exchange data="Force" mesh="Solid-Mesh" from="Fluid" to="Solid" />
  <exchange data="Displacement" mesh="Solid-Mesh" from="Solid" to="Fl
  <relative-convergence-measure limit="5e-3" data="Displacement" mesh
  <relative-convergence-measure limit="5e-3" data="Force" mesh="Solid
  <max-iterations value="50" />
</coupling-scheme:serial-implicit>
```

Only change from explicit to implicit coupling

Configure the coupling scheme

```
<coupling-scheme:serial-implicit>
  <participants first="Fluid" second="Solid" />
  <max-time value="1.5" />
  <time-window-size value="0.01" />
  <exchange data="Force" mesh="Solid-Mesh" from="Fluid" to="Solid" />
  <exchange data="Displacement" mesh="Solid-Mesh" from="Solid" to="Fl
  <relative-convergence-measure limit="5e-3" data="Displacement" mesh
  <relative-convergence-measure limit="5e-3" data="Force" mesh="Solid
  <max-iterations value="50" />
</coupling-scheme:serial-implicit>
```

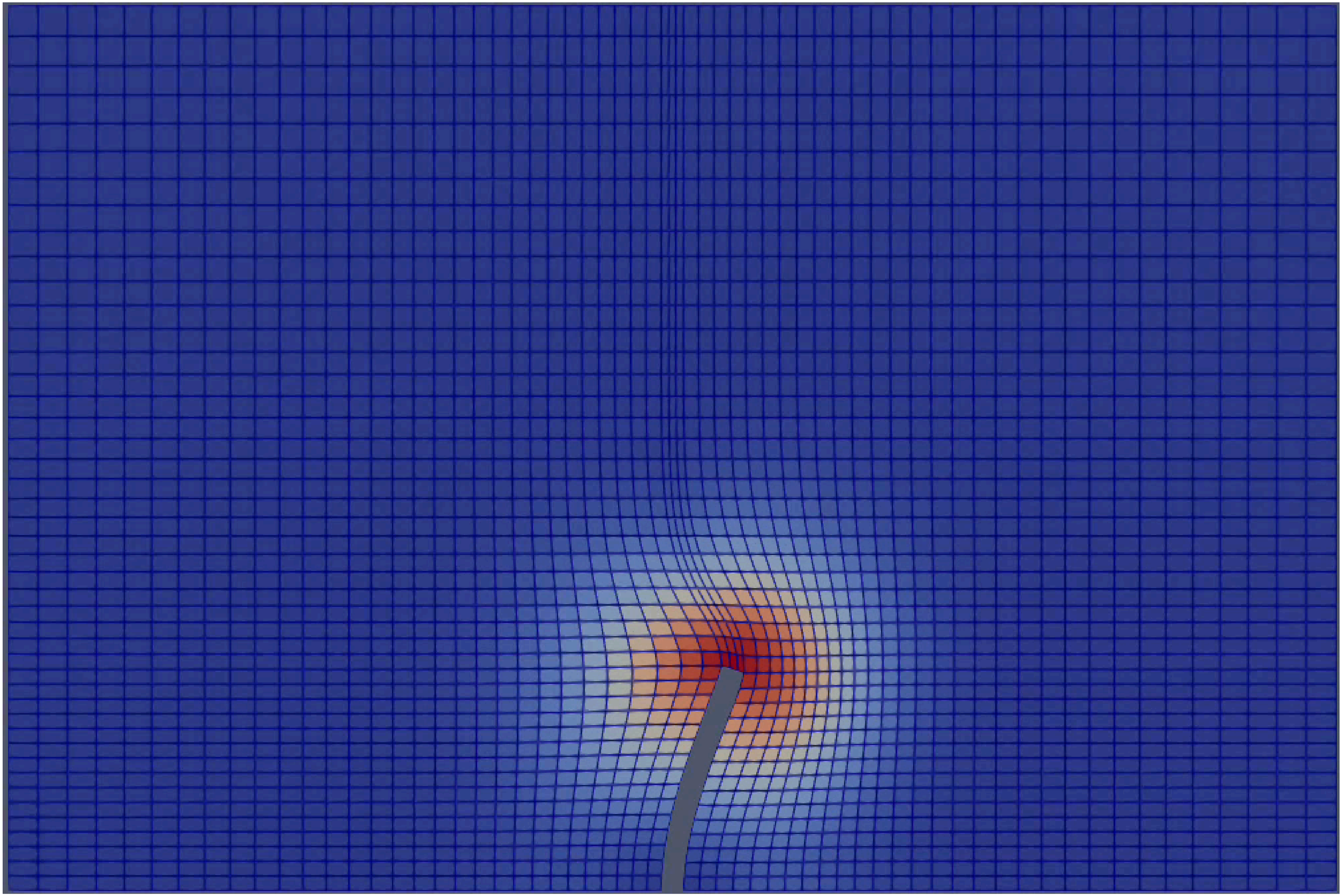
Only change from explicit to implicit coupling

Configure the coupling scheme

```
<coupling-scheme:serial-implicit>
  <participants first="Fluid" second="Solid" />
  <max-time value="1.5" />
  <time-window-size value="0.01" />
  <exchange data="Force" mesh="Solid-Mesh" from="Fluid" to="Solid" />
  <exchange data="Displacement" mesh="Solid-Mesh" from="Solid" to="Fl
  <relative-convergence-measure limit="5e-3" data="Displacement" mesh
  <relative-convergence-measure limit="5e-3" data="Force" mesh="Solid
  <max-iterations value="50" />
</coupling-scheme:serial-implicit>
```

Only change from explicit to implicit coupling

With solid density equal to 42:



Average iteration number per time step: 3.41

With solid density equal to 1: breaks again!

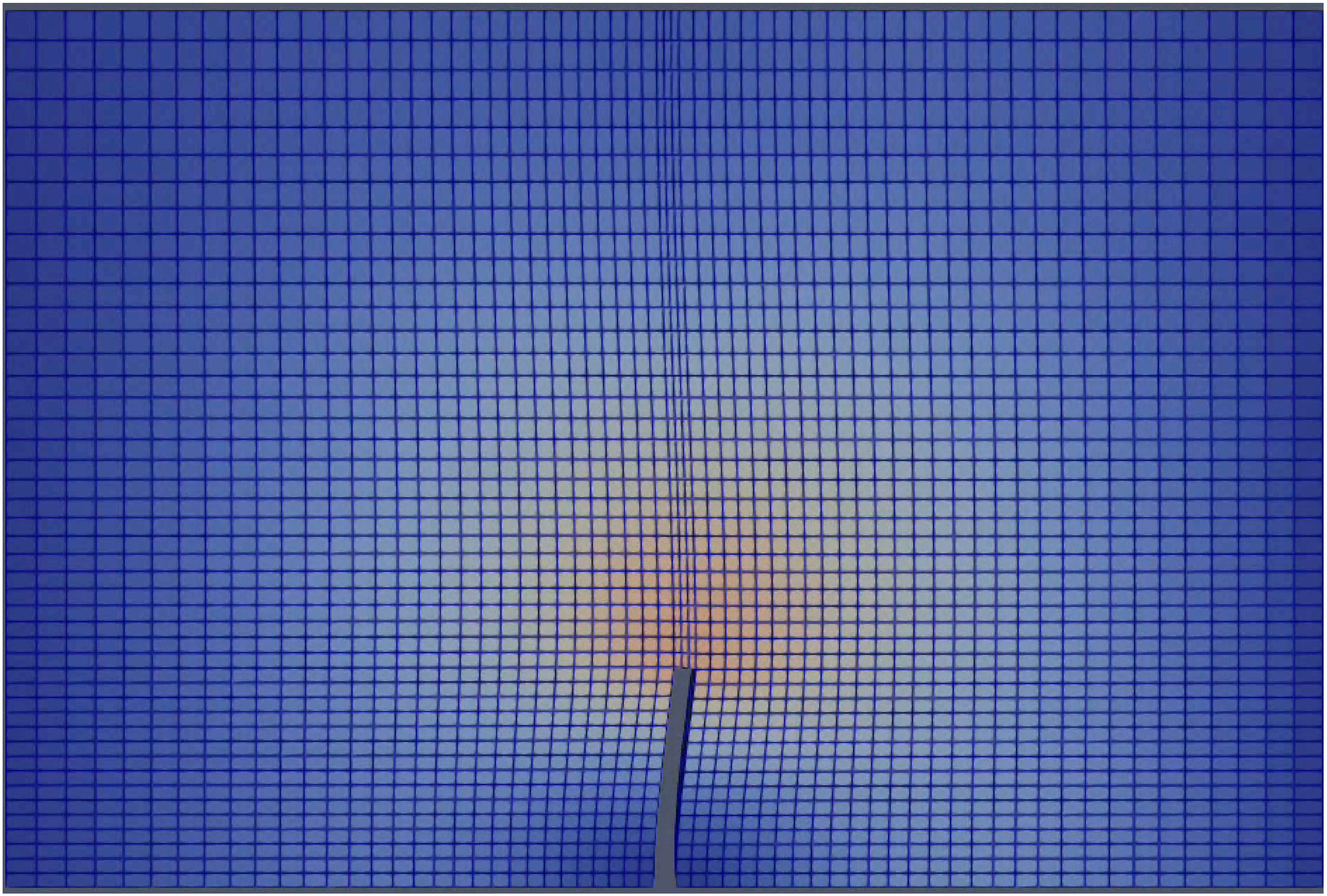
Still not enough stability for strong coupling!

Configure the coupling scheme

```
<coupling-scheme:serial-implicit>
  <participants first="Fluid" second="Solid" />
  <max-time value="1.5" />
  <time-window-size value="0.01" />
  <exchange data="Force" mesh="Solid-Mesh" from="Fluid" to="Solid" />
  <exchange data="Displacement" mesh="Solid-Mesh" from="Solid" to="Fl
  <relative-convergence-measure limit="5e-3" data="Displacement" mesh
  <relative-convergence-measure limit="5e-3" data="Force" mesh="Solid
  <max-iterations value="50" />
  <acceleration:constant>
    <relaxation value="0.5" />
  </acceleration:constant>
</coupling-scheme:serial-implicit>
```

Simplest acceleration: constant under-relaxation

With solid density equal to 1:



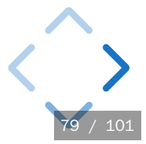
Too slow! Can we do better?

Improvement 1: Aitken under-relaxation

```
<acceleration:aitken>
  <data name="Displacement" mesh="Solid-Mesh" />
  <initial-relaxation value="0.5" />
</acceleration:aitken>
```

Improvement 2: Anderson acceleration

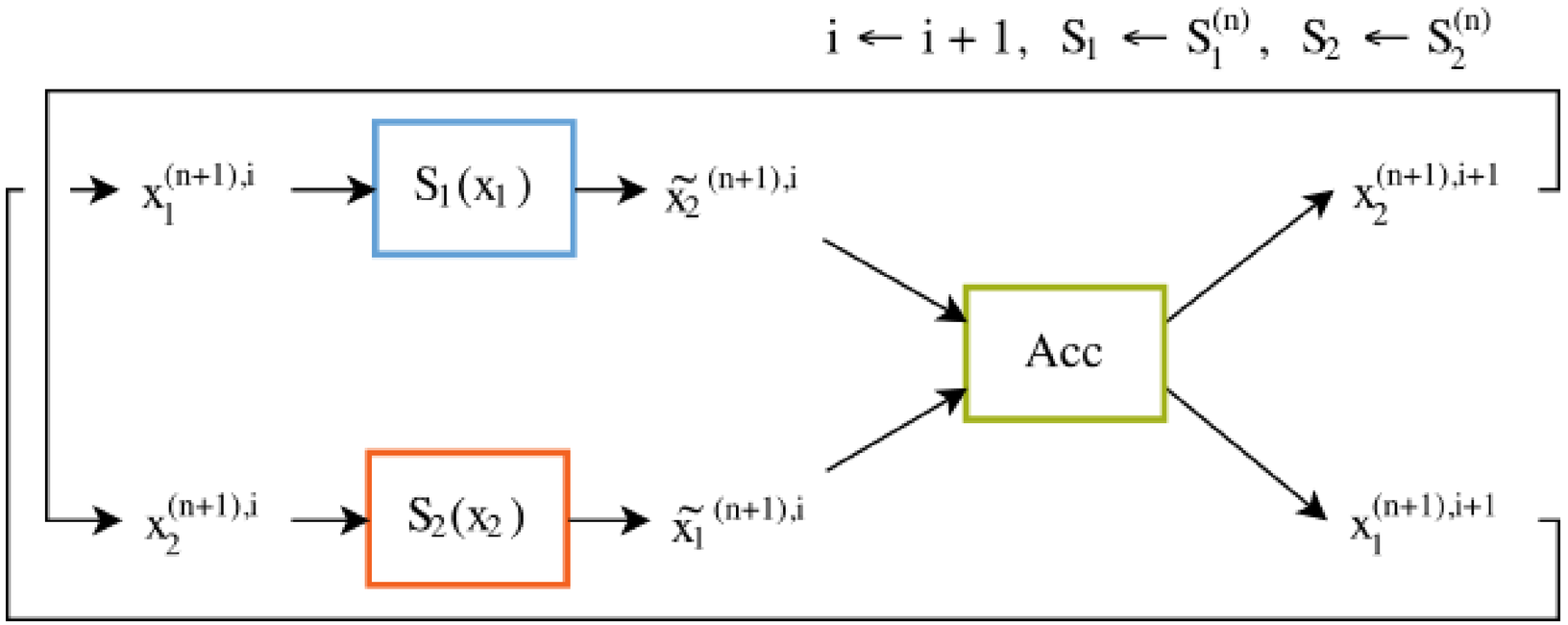
```
<acceleration:IQN-ILS>
  <data name="Displacement" mesh="Solid-Mesh" />
  <initial-relaxation value="0.5" />
  <preconditioner type="residual-sum" />
  <filter type="QR2" limit="1e-2" />
  <max-used-iterations value="100" />
  <time-windows-reused value="15" />
</acceleration:IQN-ILS>
```



Improvement 2: Anderson acceleration

```
<acceleration:IQN-ILS>
  <data name="Displacement" mesh="Solid-Mesh" />
  <initial-relaxation value="0.5" />
  <preconditioner type="residual-sum" />
  <filter type="QR2" limit="1e-2" />
  <max-used-iterations value="100" />
  <time-windows-reused value="15" />
</acceleration:IQN-ILS>
```

There are also parallel schemes



Can use fields from both / all participants

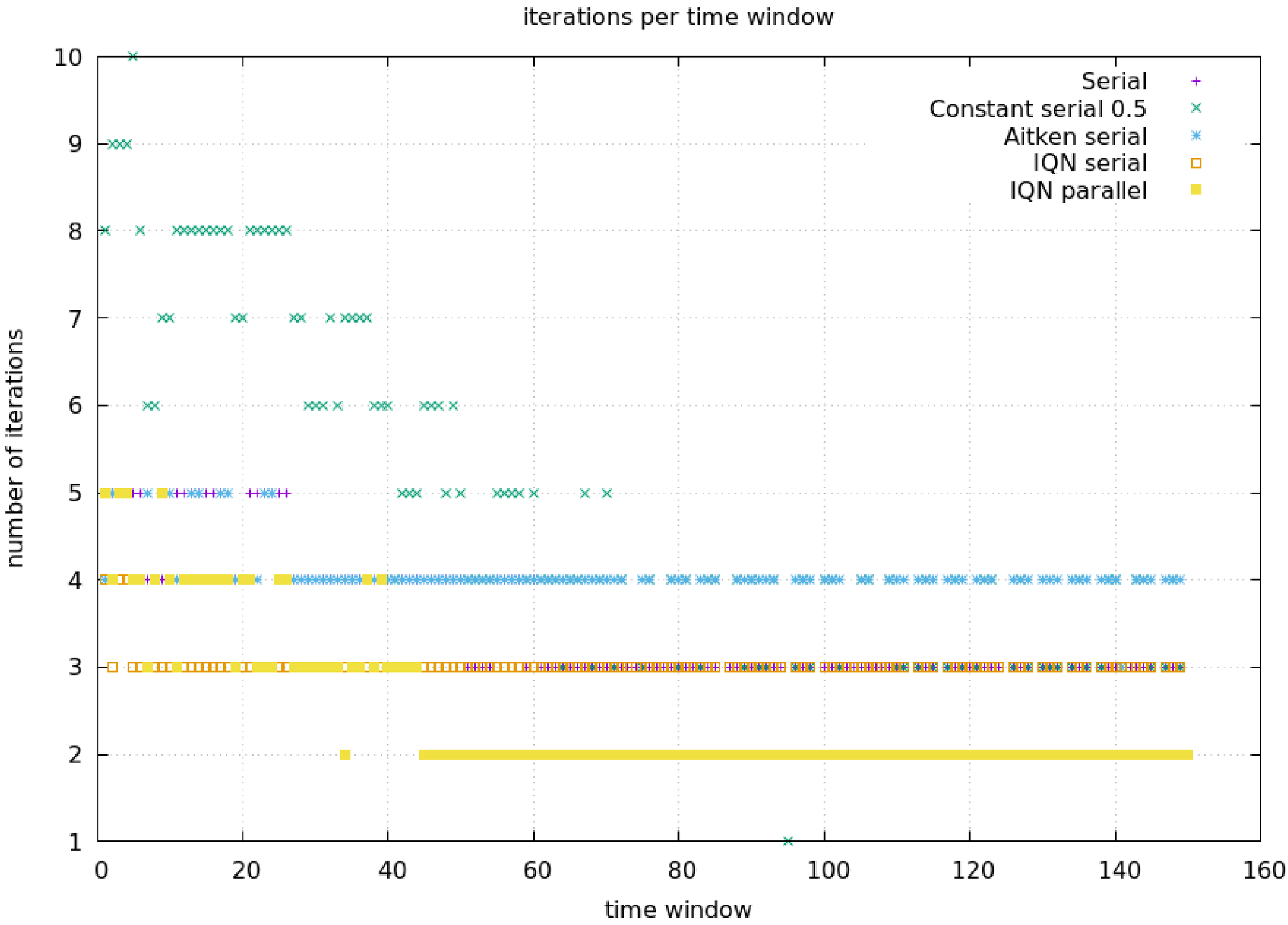
Improvement 3: Parallel coupling schemes

```
<coupling-scheme:parallel-implicit>
  <time-window-size value="0.01" />
  <max-time value="1.5" />
  <participants first="Fluid" second="Solid" />
  <exchange data="Force" mesh="Solid-Mesh" from="Fluid" to="Solid" />
  <exchange data="Displacement" mesh="Solid-Mesh" from="Solid" to="Fl
  <max-iterations value="50" />
  <relative-convergence-measure limit="5e-3" data="Displacement" mesh
  <relative-convergence-measure limit="5e-3" data="Force" mesh="Solid
  <acceleration:IQN-ILS>
    <data name="Displacement" mesh="Solid-Mesh" />
    <data name="Force" mesh="Solid-Mesh" />
    <initial-relaxation value="0.5" />
    <preconditioner type="residual-sum" />
    <filter type="QR2" limit="1e-2" />
    <max-used-iterations value="100" />
    <time-windows-reused value="15" />
```

Improvement 3: Parallel coupling schemes

```
3     <max-time value="1.5" />
4     <participants first="Fluid" second="Solid" />
5     <exchange data="Force" mesh="Solid-Mesh" from="Fluid" to="Solid" />
6     <exchange data="Displacement" mesh="Solid-Mesh" from="Solid" to="Fl
7     <max-iterations value="50" />
8     <relative-convergence-measure limit="5e-3" data="Displacement" mesh
9     <relative-convergence-measure limit="5e-3" data="Force" mesh="Solid
10    <acceleration:IQN-ILS>
11      <data name="Displacement" mesh="Solid-Mesh" />
12      <data name="Force" mesh="Solid-Mesh" />
13      <initial-relaxation value="0.5" />
14      <preconditioner type="residual-sum" />
15      <filter type="QR2" limit="1e-2" />
16      <max-used-iterations value="100" />
17      <time-windows-reused value="15" />
18    </acceleration:IQN-ILS>
19  </coupling-scheme:parallel-implicit>
```

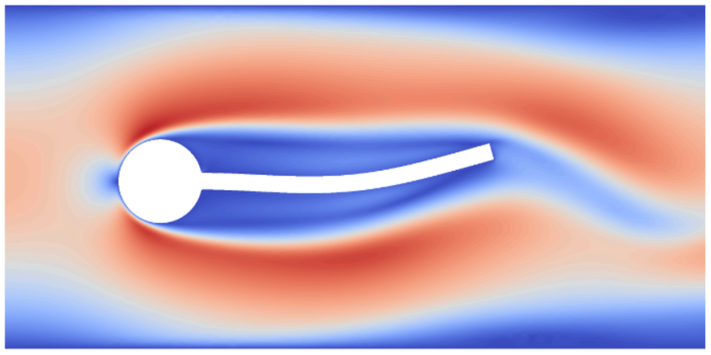
	d = 42	d = 1
serial-explicit	div	div
serial-implicit (const)	4.48	5.27
serial-implicit (Aitken)	3.74	5.30
serial-implicit (IQN)	2.95	3.07
parallel-implicit (IQN)	2.46	2.33



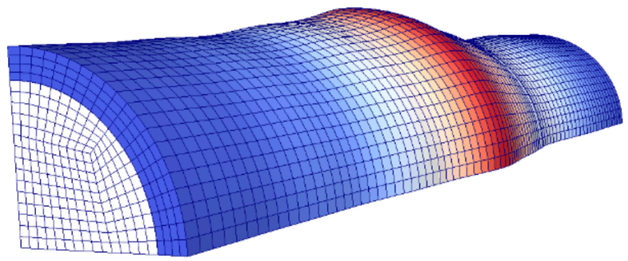
Beware: Configuration not fine-tuned, not rigorous comparison, application context.

Rigorous comparison from the literature

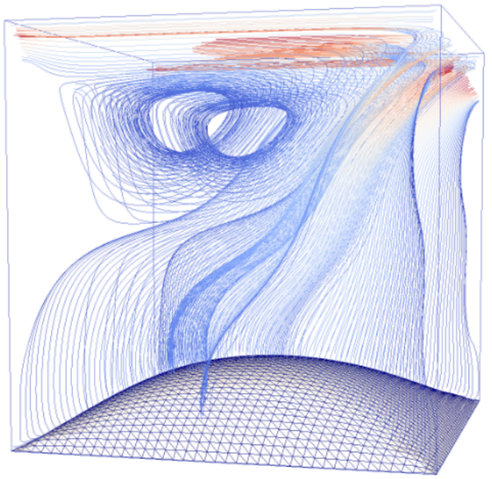
FSI3



3D-Tube



Driven Cavity



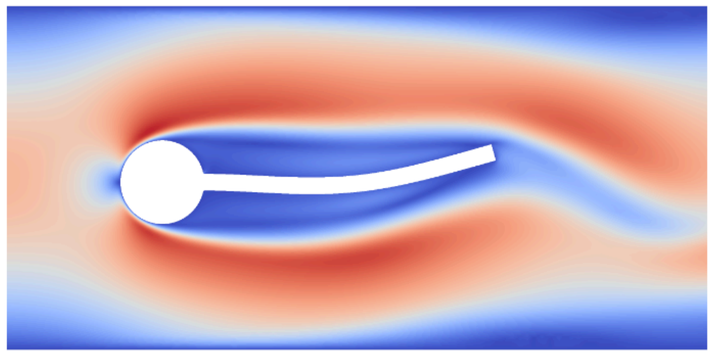
Mean Iterations	Aitken	Quasi-Newton
FSI3	17.0	3.7
3D-Tube	Div.	7.5
Driven Cavity	7.4	3.0

http://doi.org/
10/m9ff

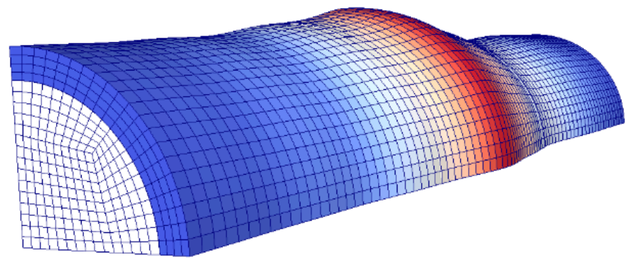
Source: Dissertation of Benjamin Uekermann (2016)

Rigorous comparison from the literature

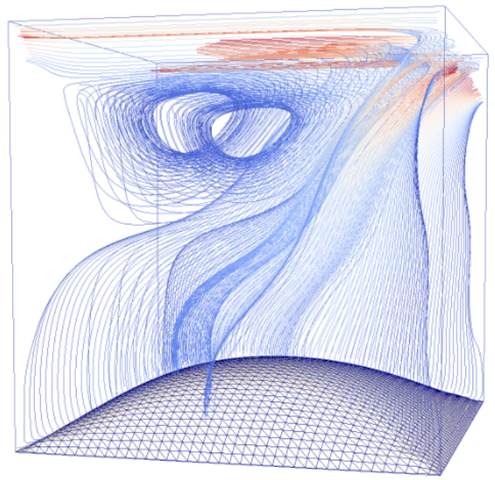
FSI3



3D-Tube



Driven Cavity



Mean Iterations	Serial Coupling	Parallel Coupling
FSI3	3.7	**3.3**
3D-Tube	**7.5**	10.4
Driven Cavity	3.0	**2.0**

Source: Dissertation of Benjamin Uekermann (2016)

Resources

Everything is open and user-centered

- Developed in the open: `github.com/precice`
- Documentation and many examples in one place
- (Diamond) open-access publications
- Public recordings of workshop talks
- Open collaboration, issues, projects, feedback, ...

Extra effort to maintain and update

Start here: precice.org

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

Save the date: preCICE Workshop 2025, Hamburg, Sep 8-12

Welcome to preCICE

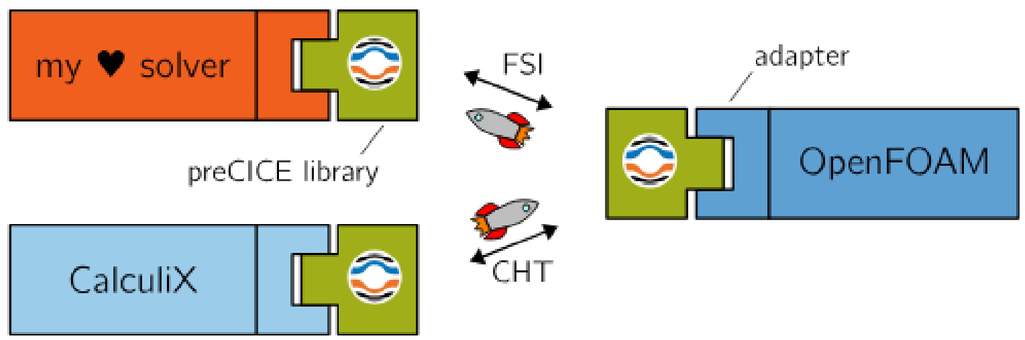
The coupling library for partitioned multi-physics simulations.

Star on Github 777 | Latest v3.1.2 (Jun 6, 2024) | 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.



Built & hosted on GitHub Pages.
Last 7 days: 700+ visitors, 4k pageviews

Documentation

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

Save the date: preCICE Workshop 2025, Hamburg, Sep 8-12

Documentation

- Fundamentals
  - Overview
  - Terminology
  - License information
  - Literature guide
  - Roadmap
  - Previous versions
- Installation
- Configuration
- Tooling
- Provided adapters
- Couple your code
- Running simulations
- Dev docs
- Documentation meta

The preCICE documentation

Summary: This page gives an overview of the complete preCICE documentation, including building, configuration, literature, the API, and much more.

Table of Contents

- The big picture
- Where to find what

Edit me

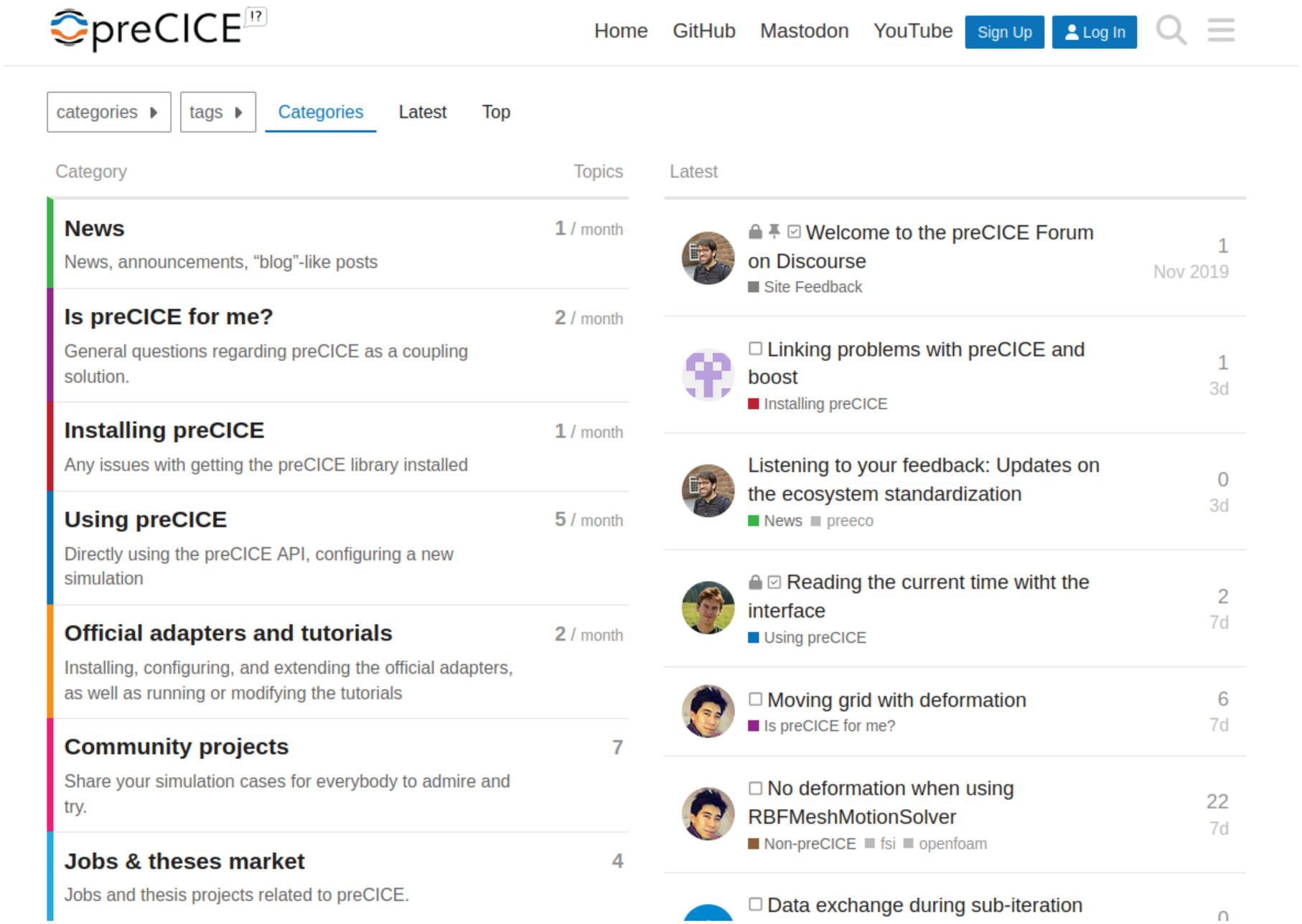
Updated 15 May 24

The big picture

preCICE stands for Precise Code Interaction Coupling Environment. Its main component is a library that can be used for partitioned multi-physics simulations, including, but not restricted to fluid-structure interaction and conjugate heat transfer simulations. Partitioned (as opposite to monolithic) means that preCICE couples existing programs (solvers) which simulate 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, reusing existing components. preCICE runs efficiently on a wide spectrum of systems, from low-end laptops up to complete compute clusters and has proven scalability on 10000s of MPI Ranks.

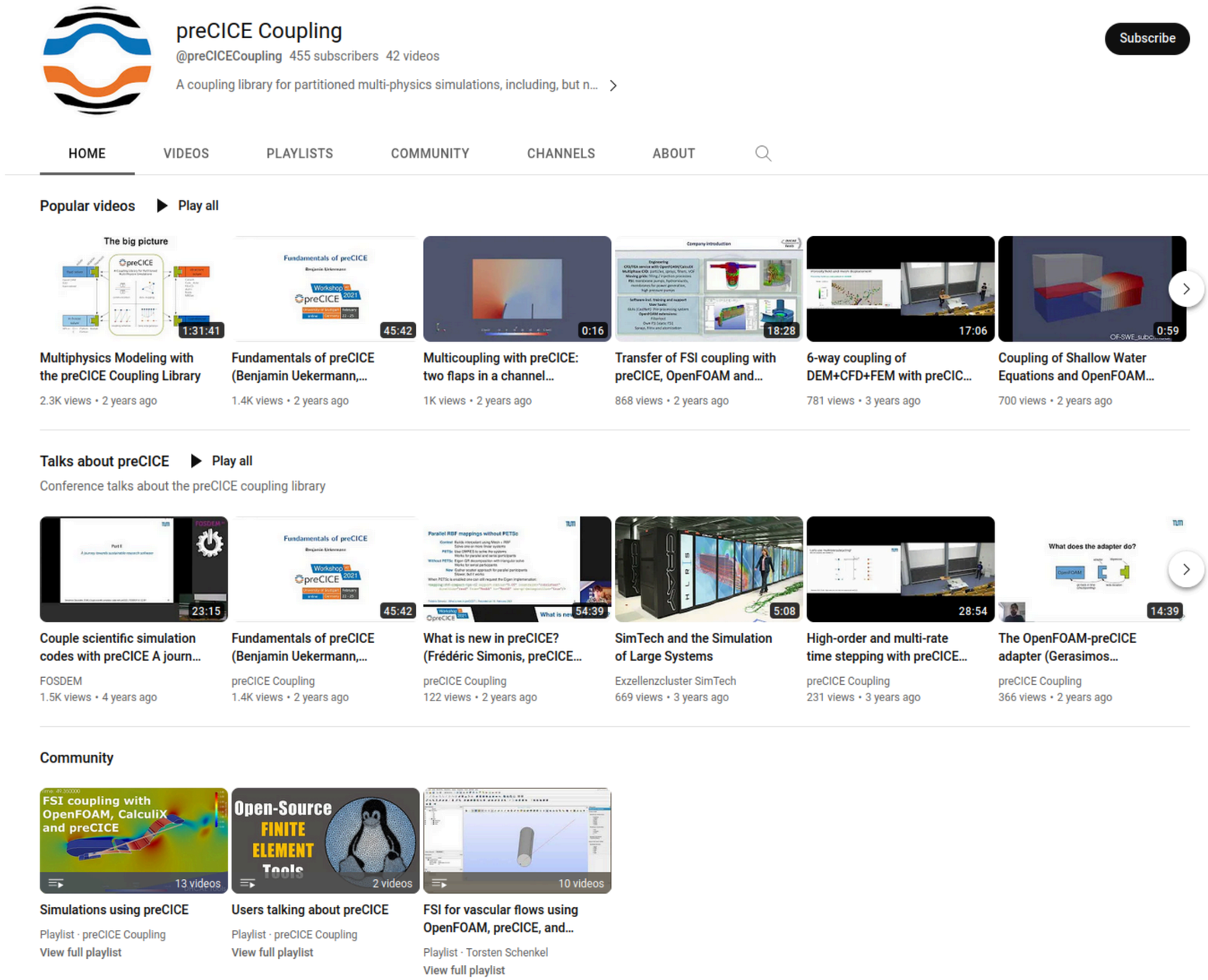
Everything in one place, user-editable on GitHub, 100+ pages

Discuss & get help



Active since October 2019, 500+ members

Learn: YouTube

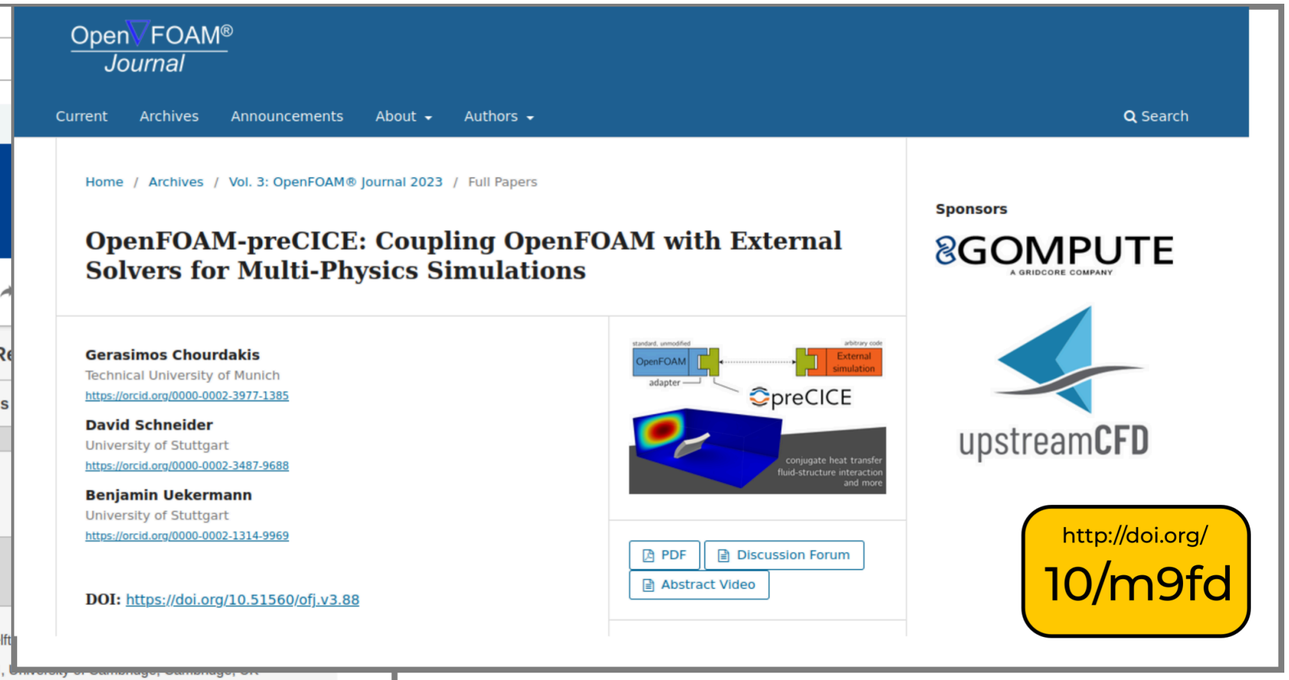


Active since 2020, 600+ subscribers, 50+ videos

Further channels

- Mastodon: https://fosstodon.org/@precice
- Bluesky: https://bsky.app/profile/precice.org
- LinkedIn: https://www.linkedin.com/groups/9073912/
- RG: https://www.researchgate.net/project/preCICE

Related publications



The people behind preCICE

University of Stuttgart
Germany

Technical University of Munich

Current PhD topics (TUM)

- Gerasimos Chourdakis: Ecosystem, geometric multi-scale
- Benjamin Rodenberg: Time interpolation
- Frédéric Simonis: Dynamic coupling meshes

Current PhD topics (Univ. Stuttgart)

- David Schneider: Data mapping (GPUs, PoU RBF)
- Ishaan Desai: Adaptive macro-micro simulations
- Jun Chen: Porous media macro-micro simulations
- Carme Homs Pons: Musculoskeletal multi-scale simulations (OpenDiHu)

Further research and development

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

Meet the people

Workshop #6 2025

preCICE

HSU Hamburg

Germany

September

8 - 12

preCICE is free because of

Research Software
Sustainability

Bundesministerium
für Wirtschaft
und Energie

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. Both for developers and for users

2. Many efficient algorithms are one setting away

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

Get these slides.