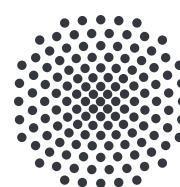


19th OpenFOAM Workshop - Beijing, June 25-28, 2024

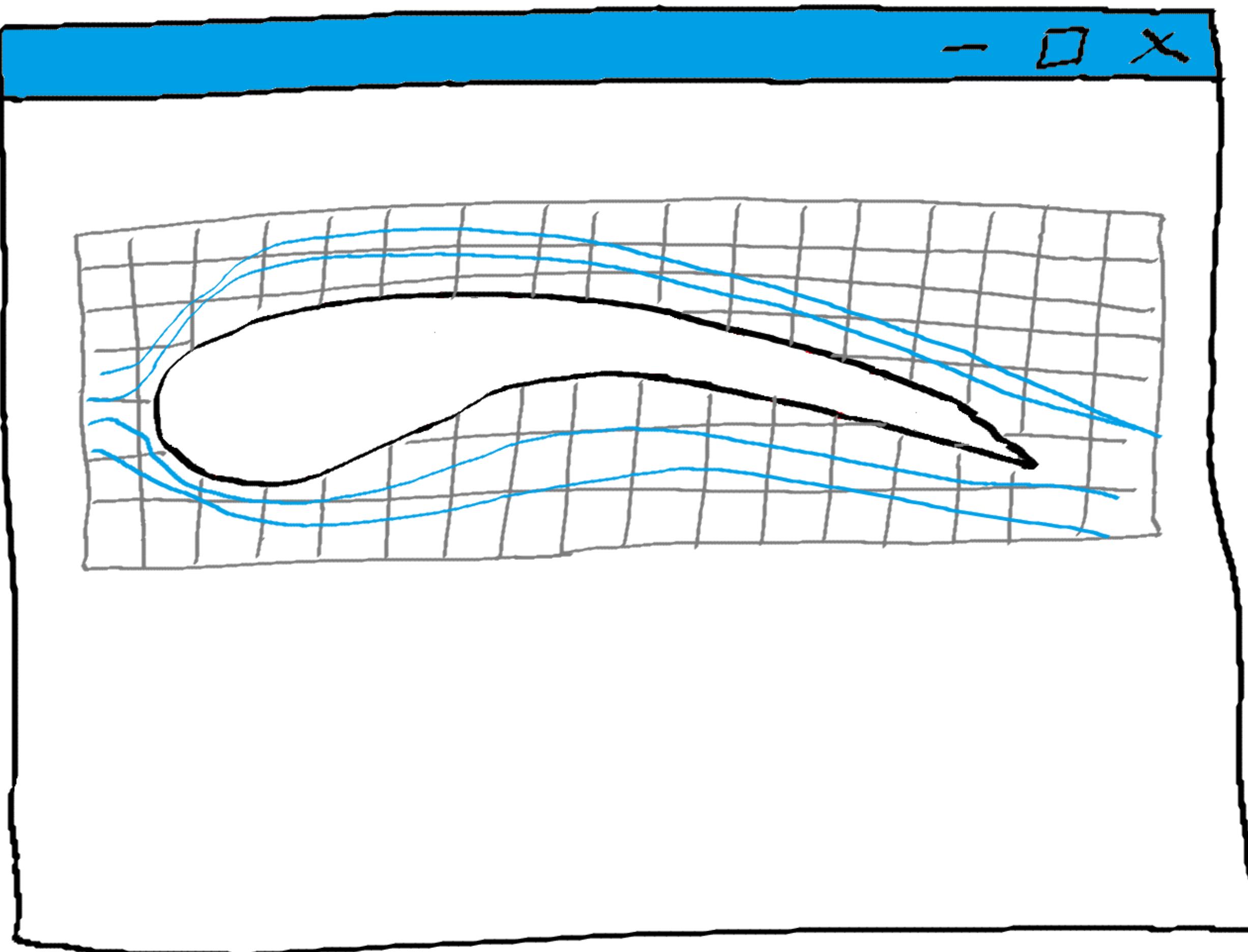
# Getting started with OpenFOAM-preCICE for FSI simulations

Jun Chen & Gerasimos Chourdakis, University of Stuttgart  
[jun.chen@ipvs.uni-stuttgart.de](mailto:jun.chen@ipvs.uni-stuttgart.de)

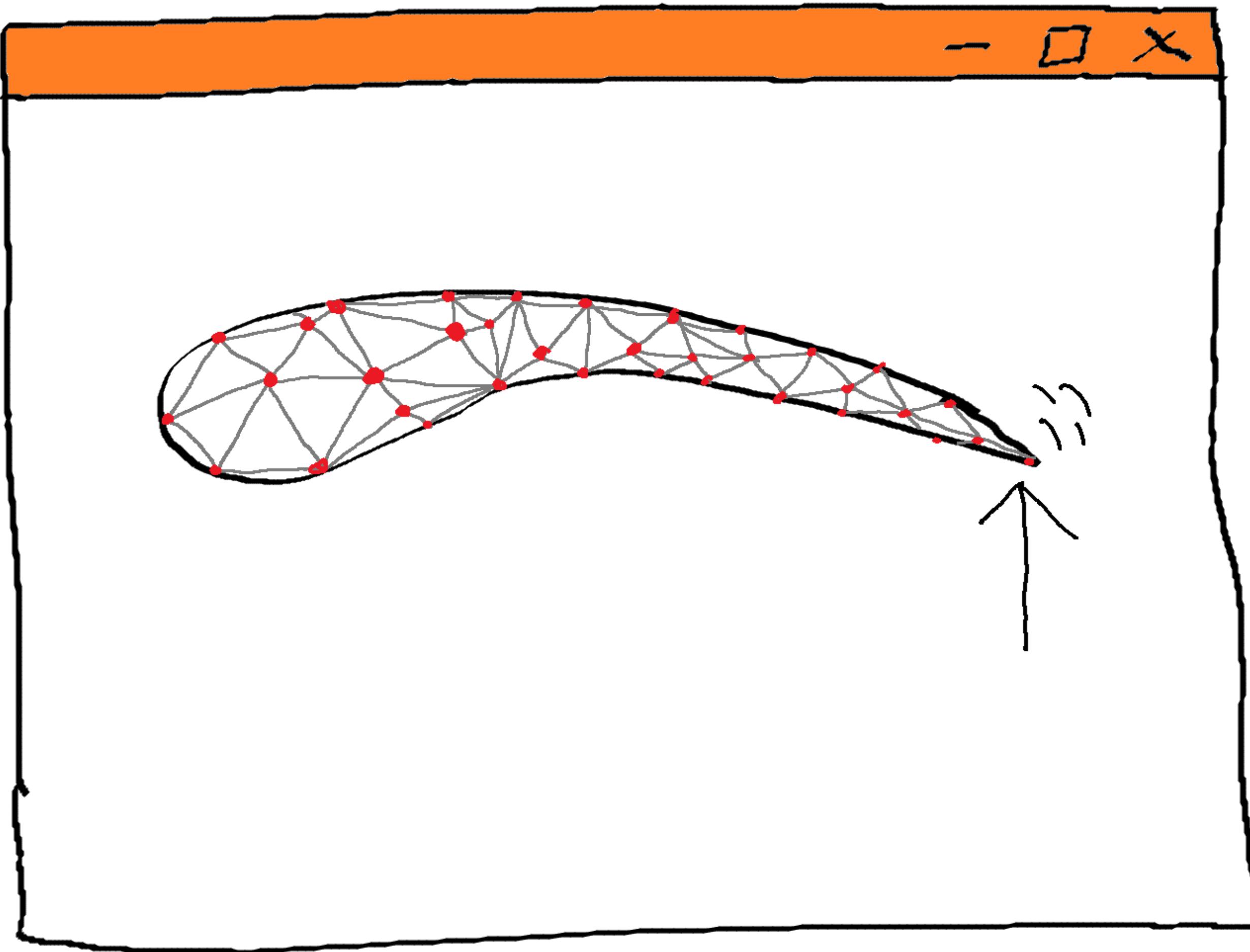


University of Stuttgart  
Germany

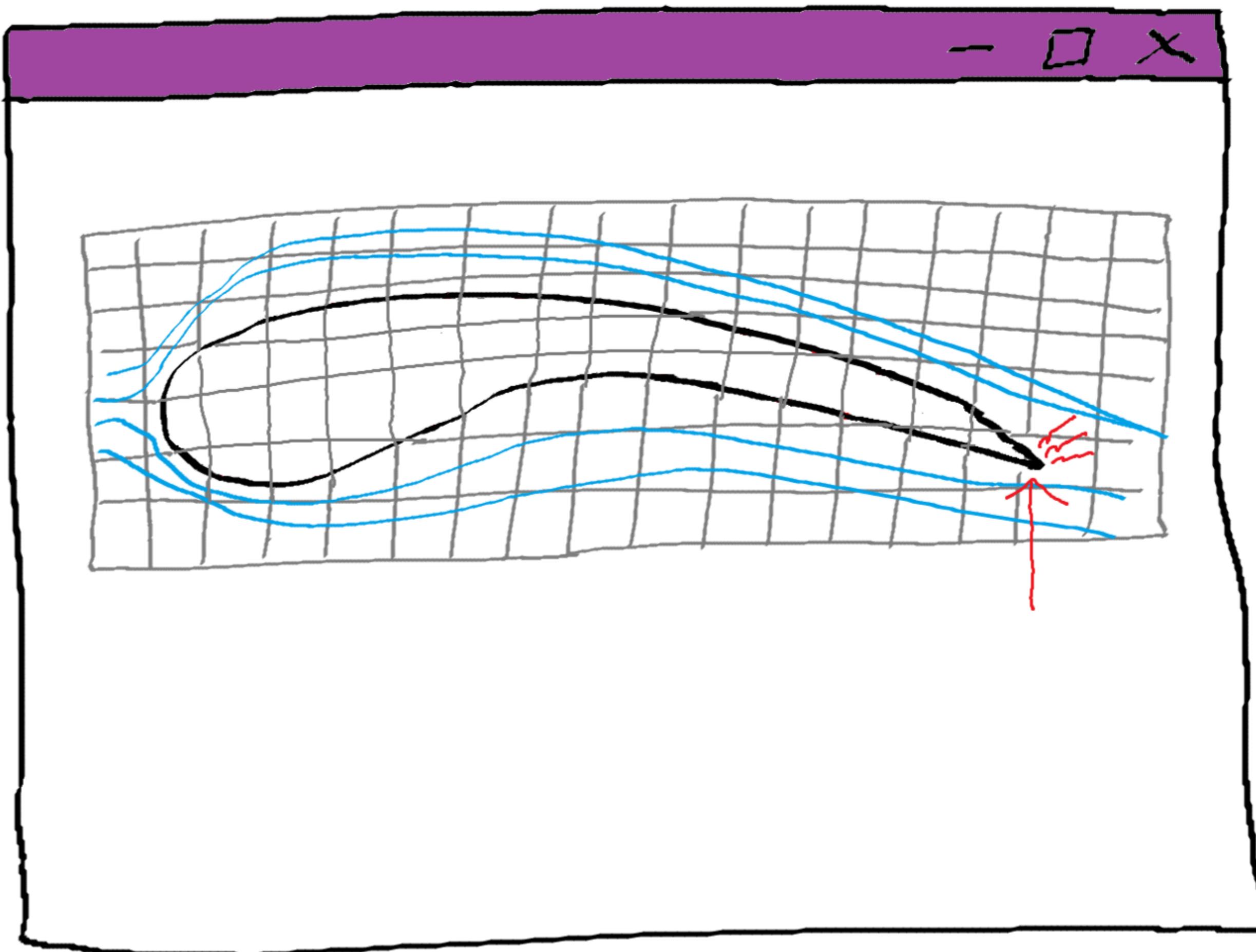
# CFD: Simulating flow around a wing



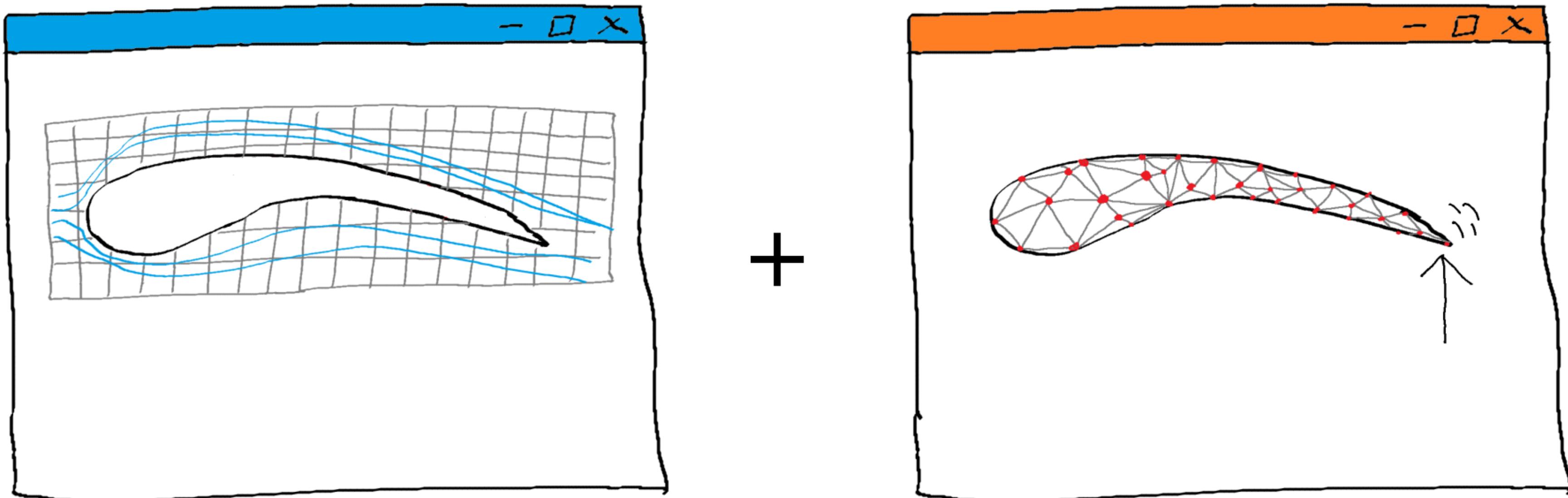
# FEM: Simulating stresses on a wing



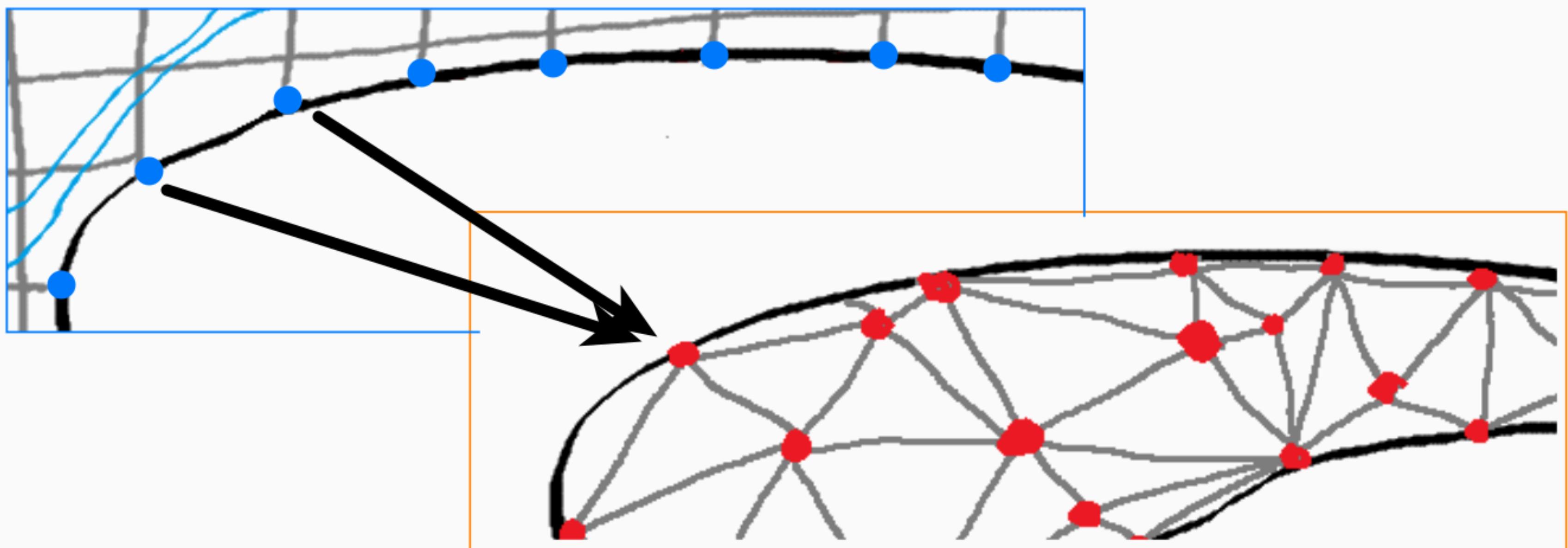
# Can we simulate both at the same time?



...but reusing the CFD and FEM codes?



# From the CFD world to the FEM world



# 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

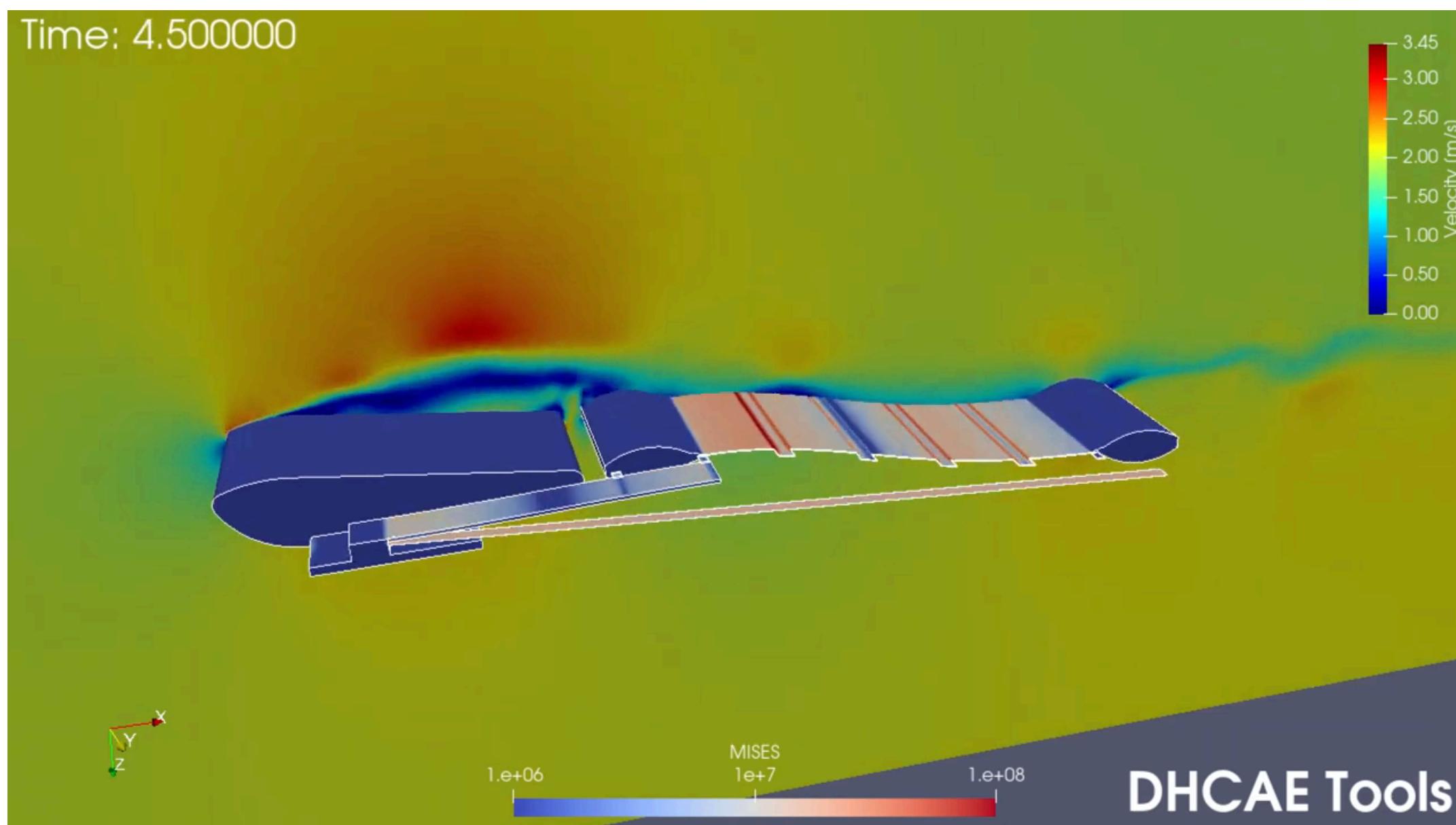
## Time interpolation

Options:

- waveform iteration

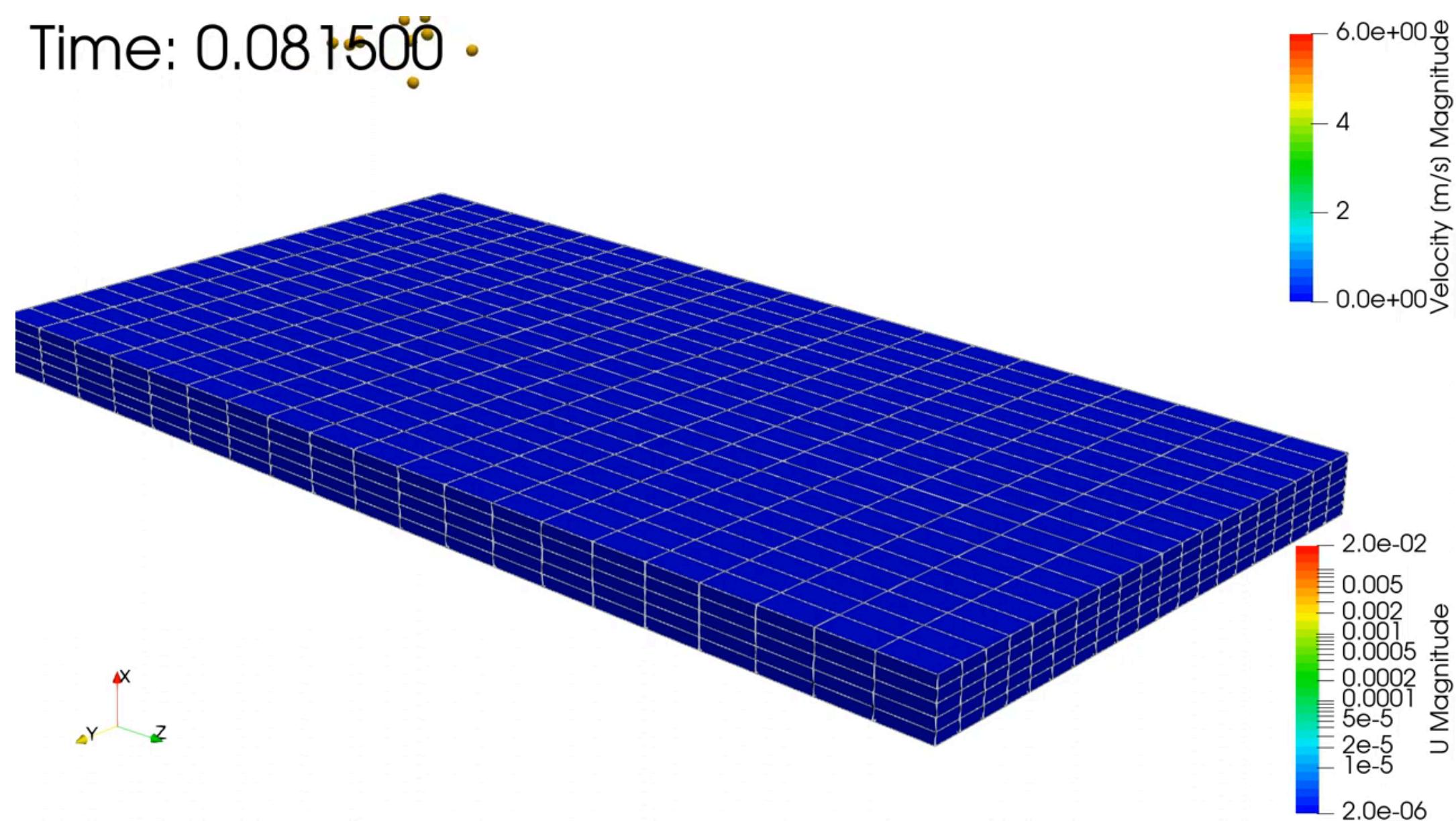


# What people do with preCICE (1)



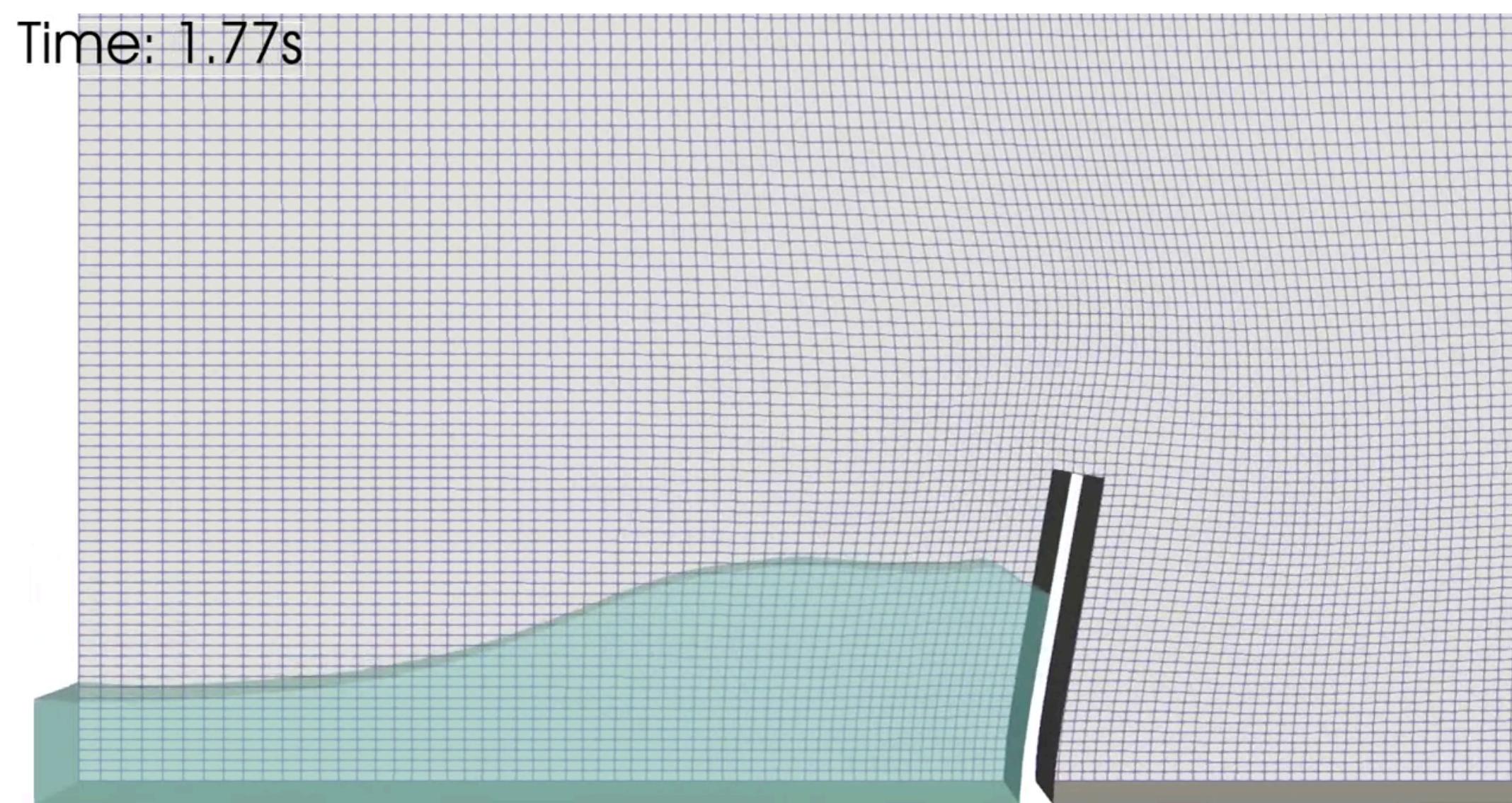
Credit: Ulrich Heck, DHCAE Tools GmbH

# What people do with preCICE (2)



Credit: Prasad Adhav, Univ. of Luxembourg

# What people do with preCICE (3)



Credit: Utkan Caliskan

# In this talk

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



# Get the slides



[github.com/MakisH/ofw19-training](https://github.com/MakisH/ofw19-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](http://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](https://github.com/precice)



# Using the preCICE library

Coupling two toy Python solvers

# Dependencies

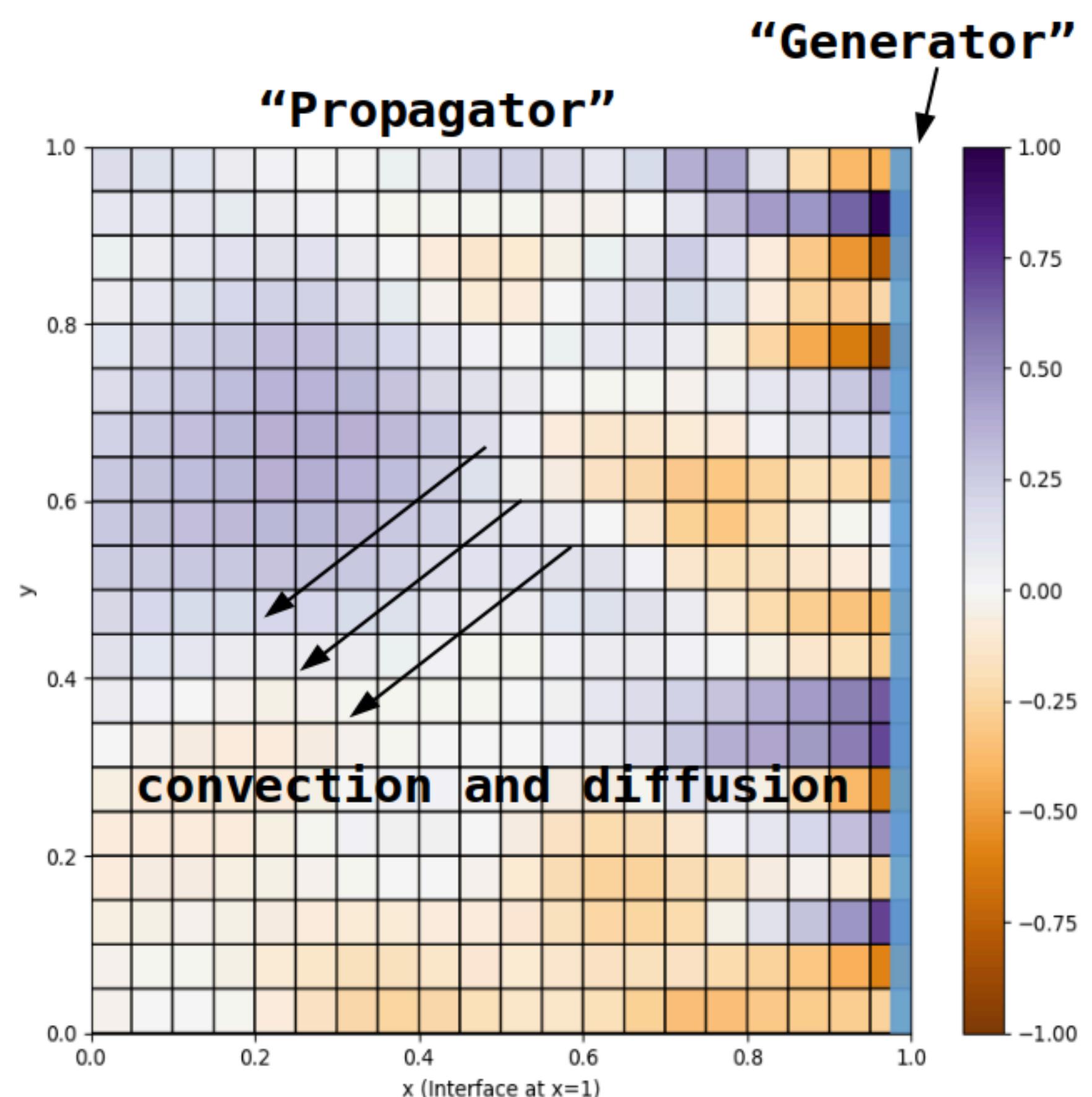
- preCICE v3 (e.g. packages for Ubuntu)
- Python 3
- Python packages numpy, matplotlib
- preCICE Python bindings:

```
pip3 install --upgrade pip  
pip3 install --user pyprecice
```

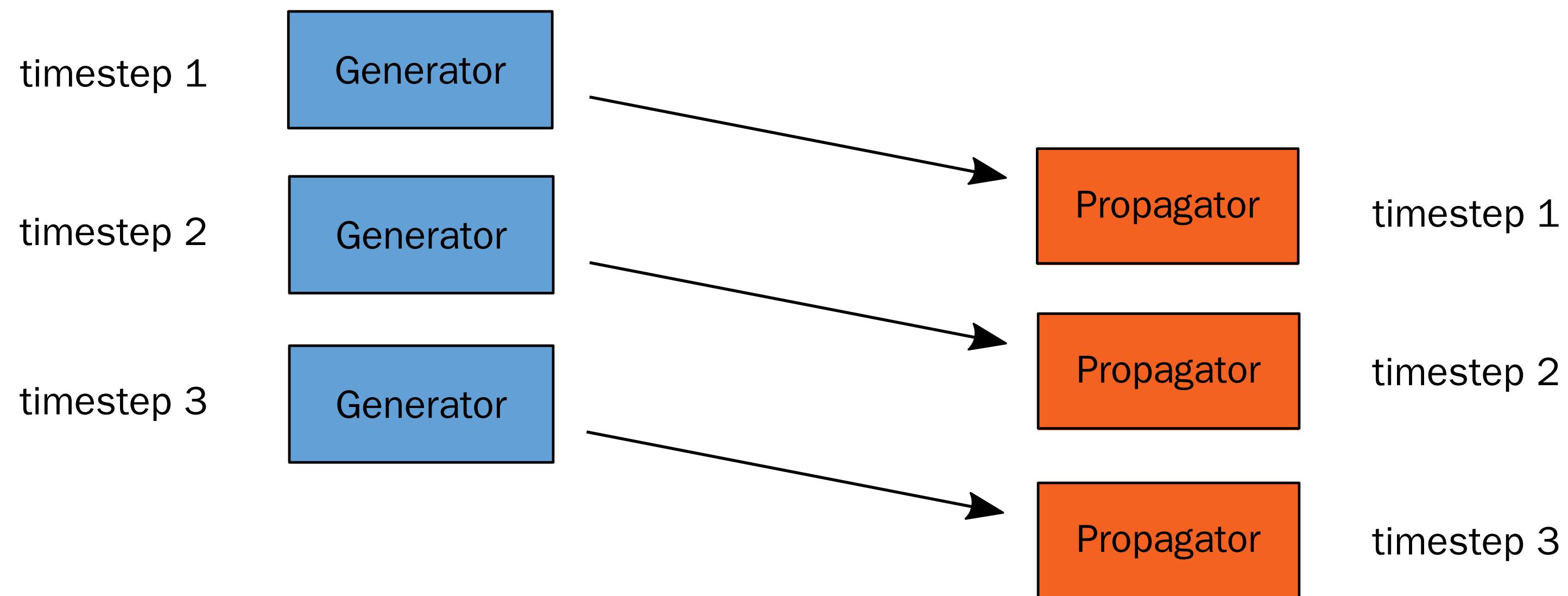


# "Generator" and "Propagator"

- `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:

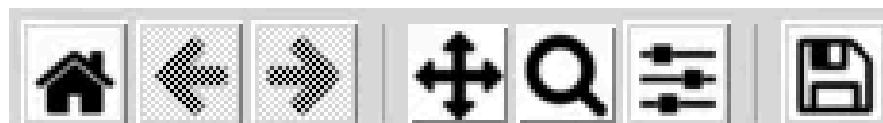
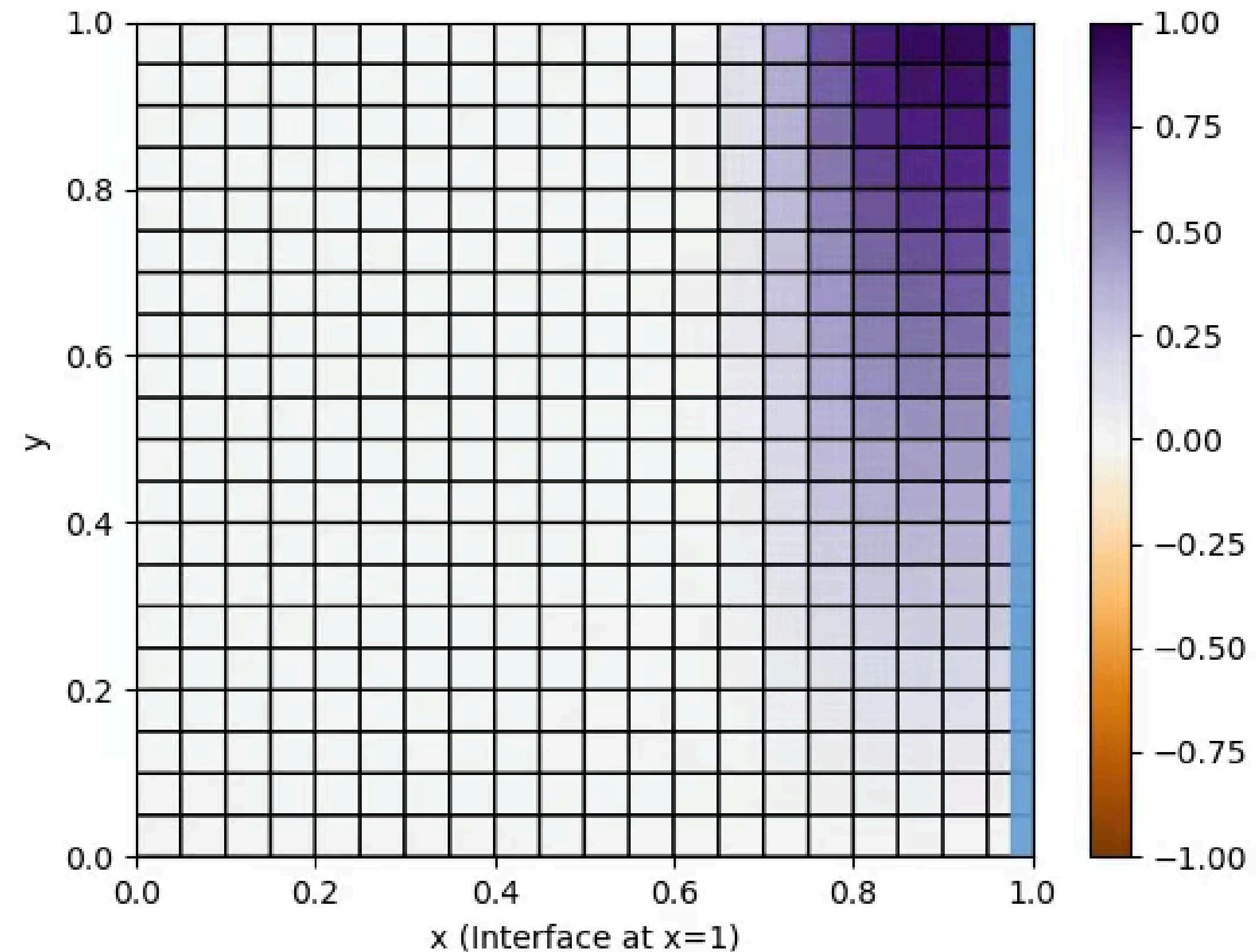
    print("Propagating data")
```



```
vagrant@precicevm:~/Desktop/skele... vagrant@precicevm: ~/Desktop/skeleton/generator
vagrant@precicevm: ~/Desktop/skeleton/generator 187x52
vagrant@precicevm:~/Desktop/skeleton$ tree
.
├── Allclean
├── generator
│   └── generator.py
├── precice-config.xml
└── propagator
    └── propagator.py
 README.txt

2 directories, 5 files
vagrant@precicevm:~/Desktop/skeleton$ cd generator/
vagrant@precicevm:~/Desktop/skeleton/generator$ python3 generator.py
Generating data
Generating data
Generating data
```

# 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

```
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

```
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
37     t = t + dt
```



# Advance the coupling

```
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
37
```



# 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?

```
chenju@lapsgs05:~/Desktop/T5/generator 94x49
chenju@lapsgs05: ~/Desktop/T5/generator
$ python3 generator.py
[17:03:48]

chenju@lapsgs05:~/Desktop/T5/propagator 94x49
chenju@lapsgs05: ~/Desktop/T5/propagator
$ [REDACTED]
[17:03:55]
```

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
```

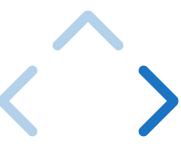
# Write & read data

```
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()
```

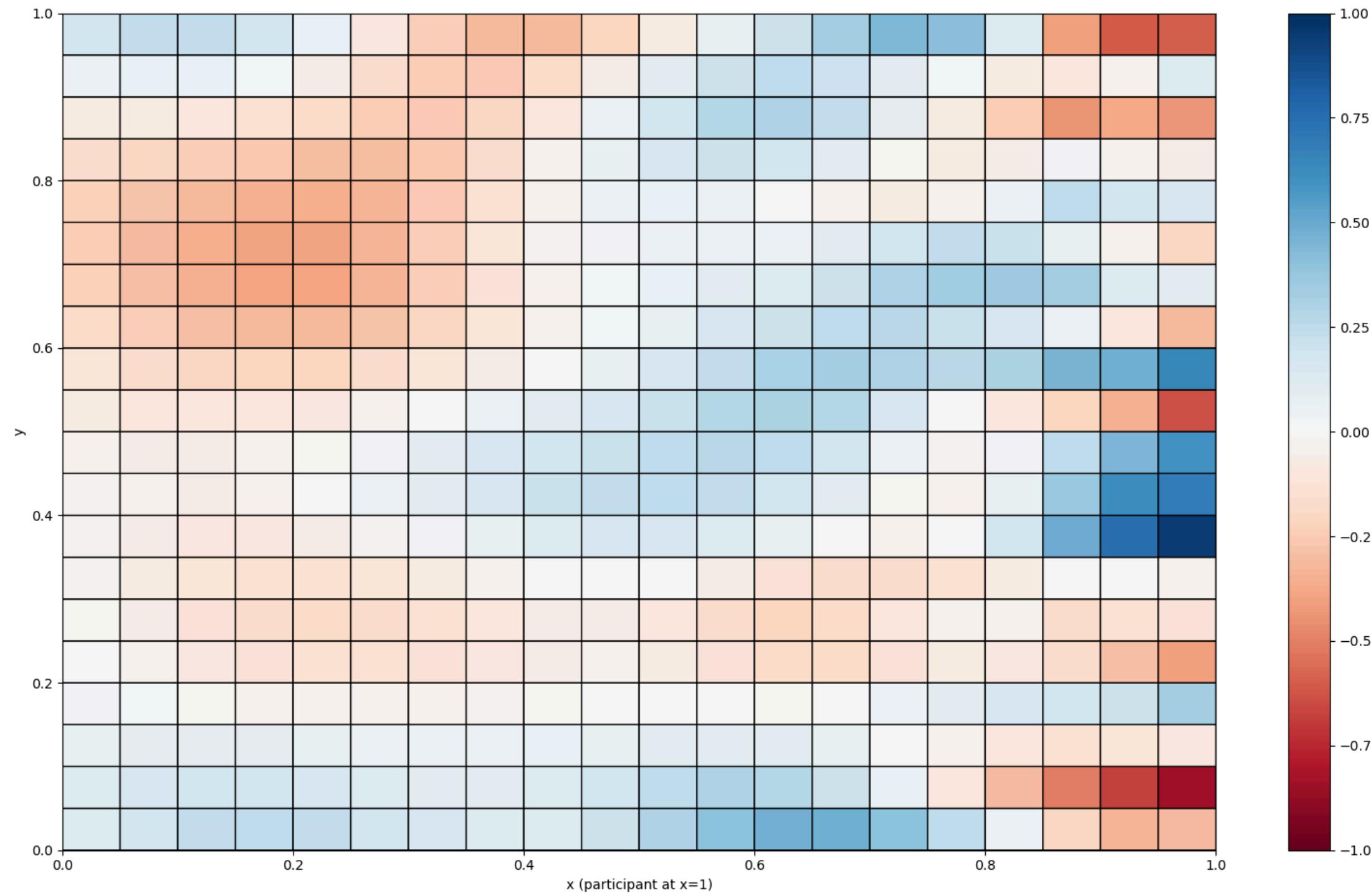


**It should work now!**

```
chenju@lapsgs05: ~/Desktop/T5/generator 94x49
chenju@lapsgs05: ~/Desktop/T5/generator 94x49
[17:12:36] $ python3 generator.py

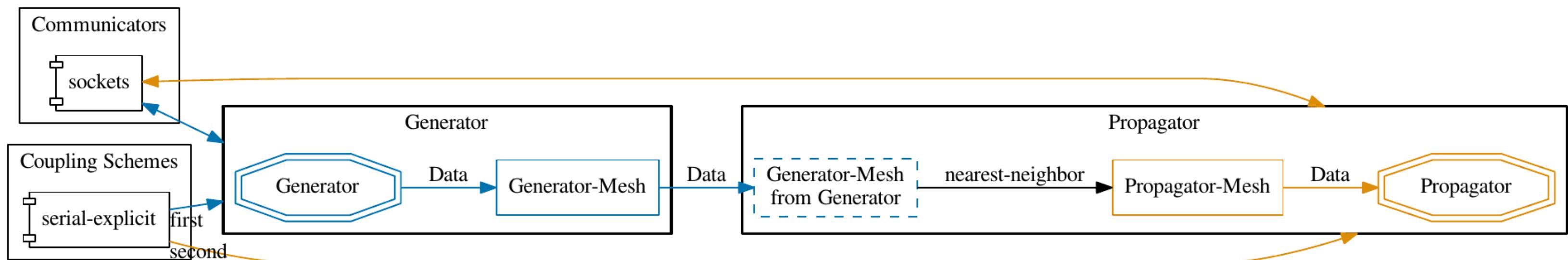
chenju@lapsgs05: ~/Desktop/T5/propagator 94x49
chenju@lapsgs05: ~/Desktop/T5/propagator 94x49
[17:12:34] $ python3 propagator.py
```

# 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](#).

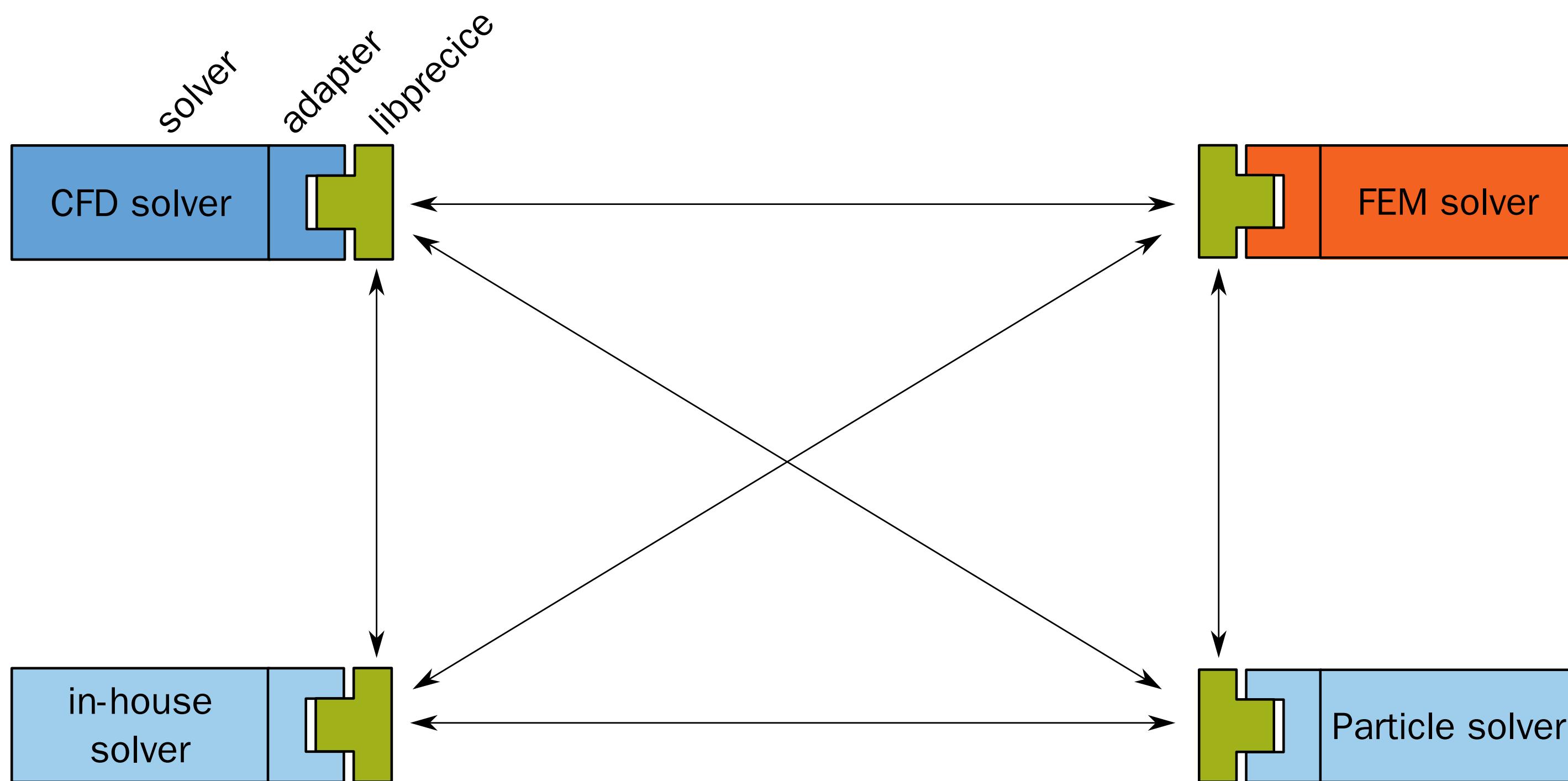
# Using the preCICE ecosystem

Example: Coupling OpenFOAM and deal.II

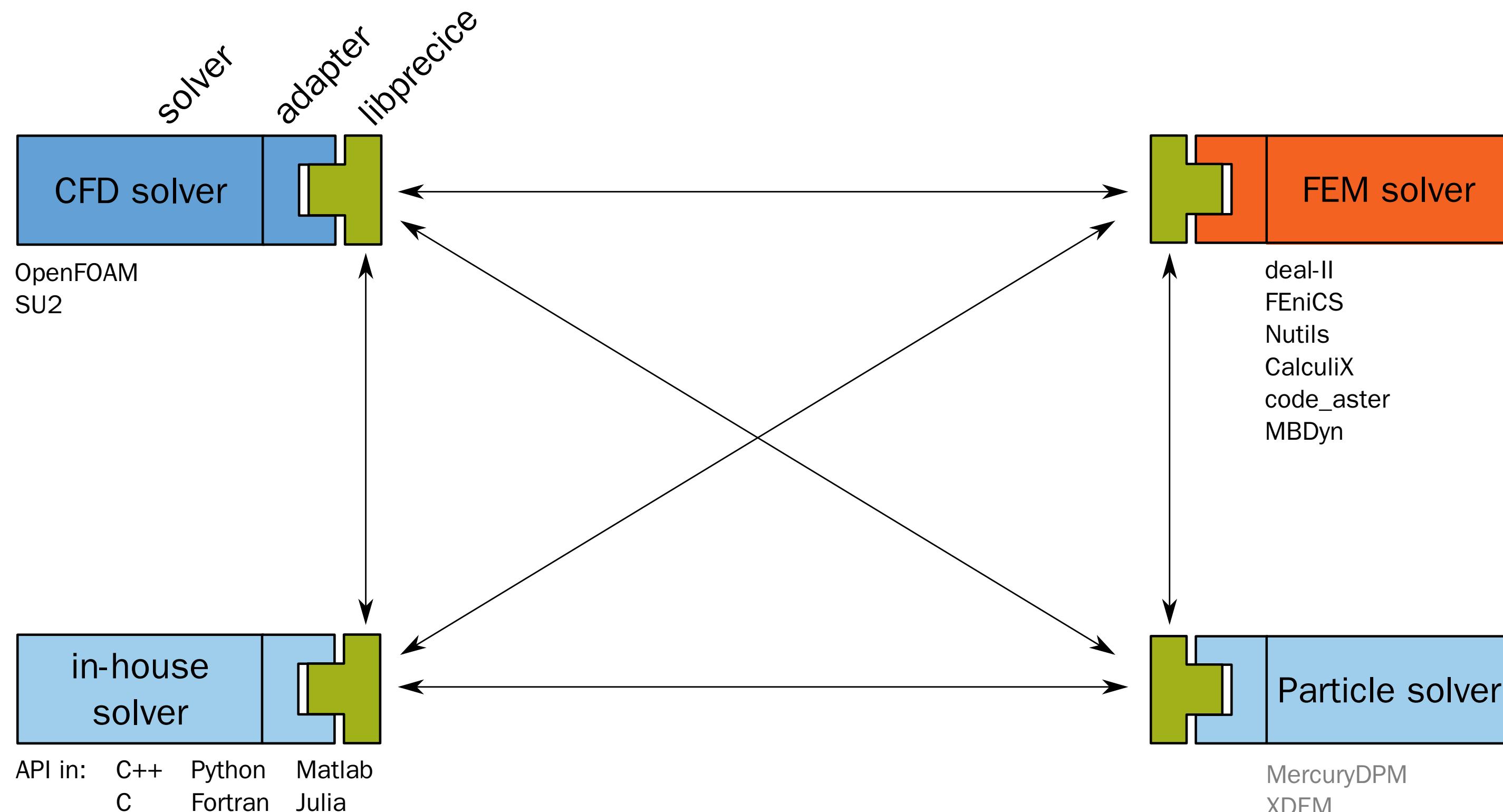
# The big picture



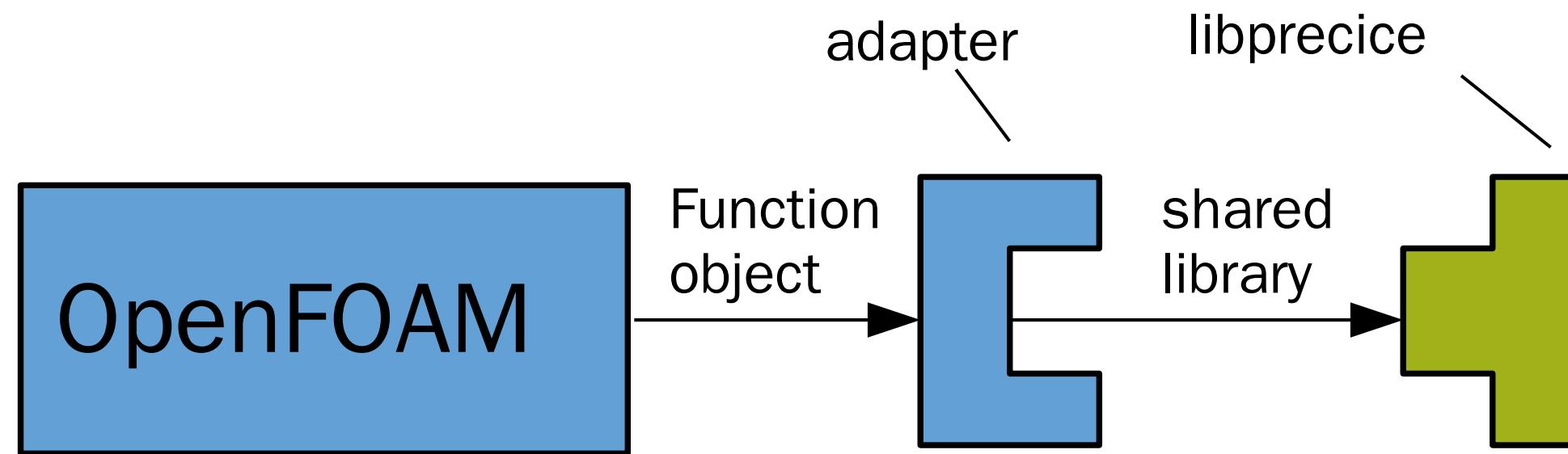
# The big picture



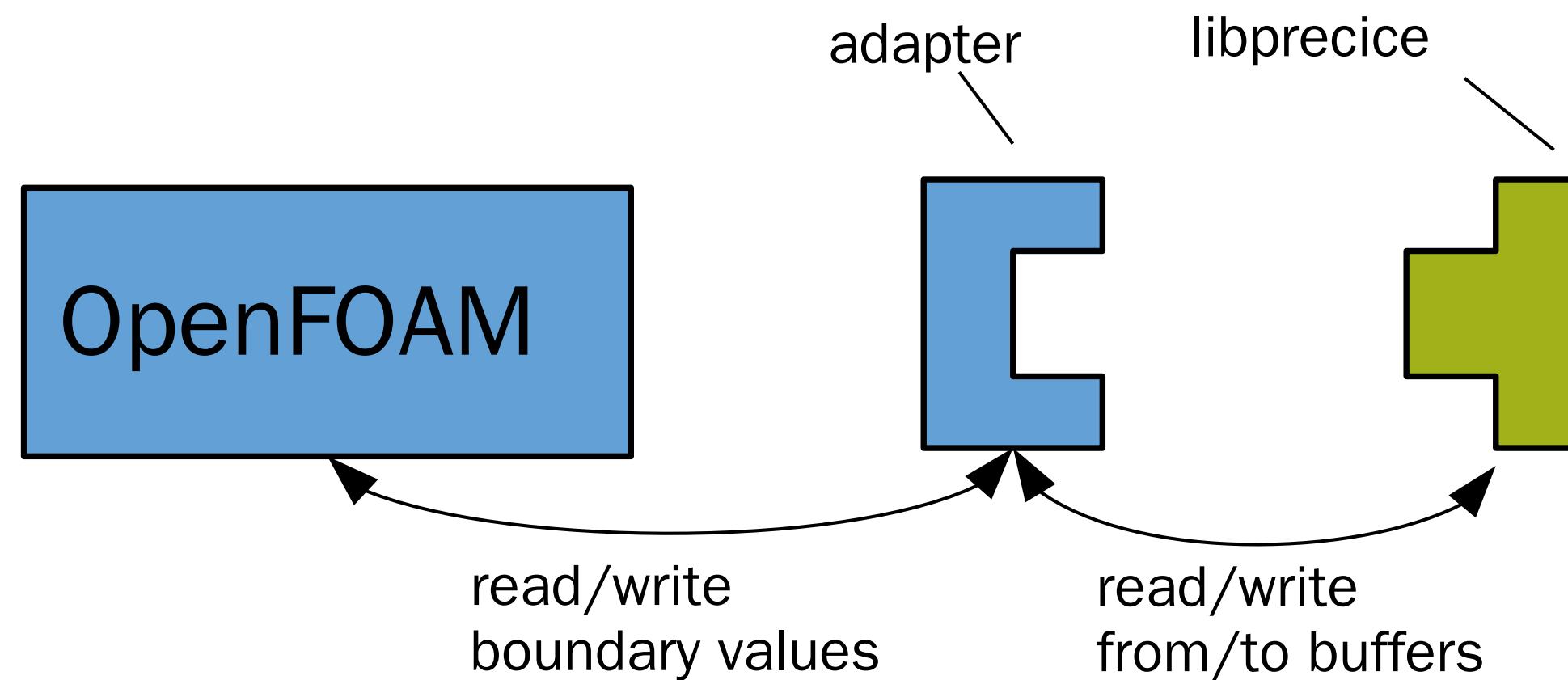
# The big picture



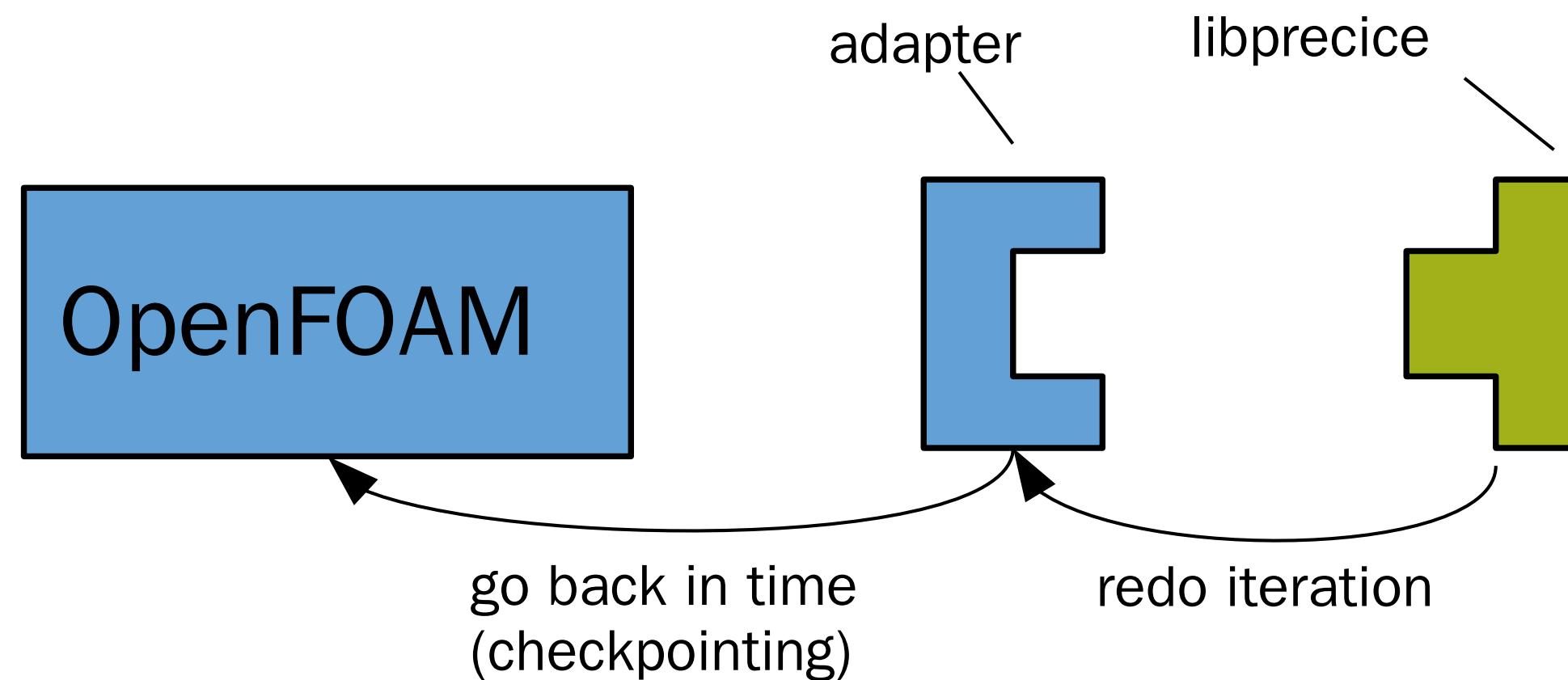
# What does the adapter do?



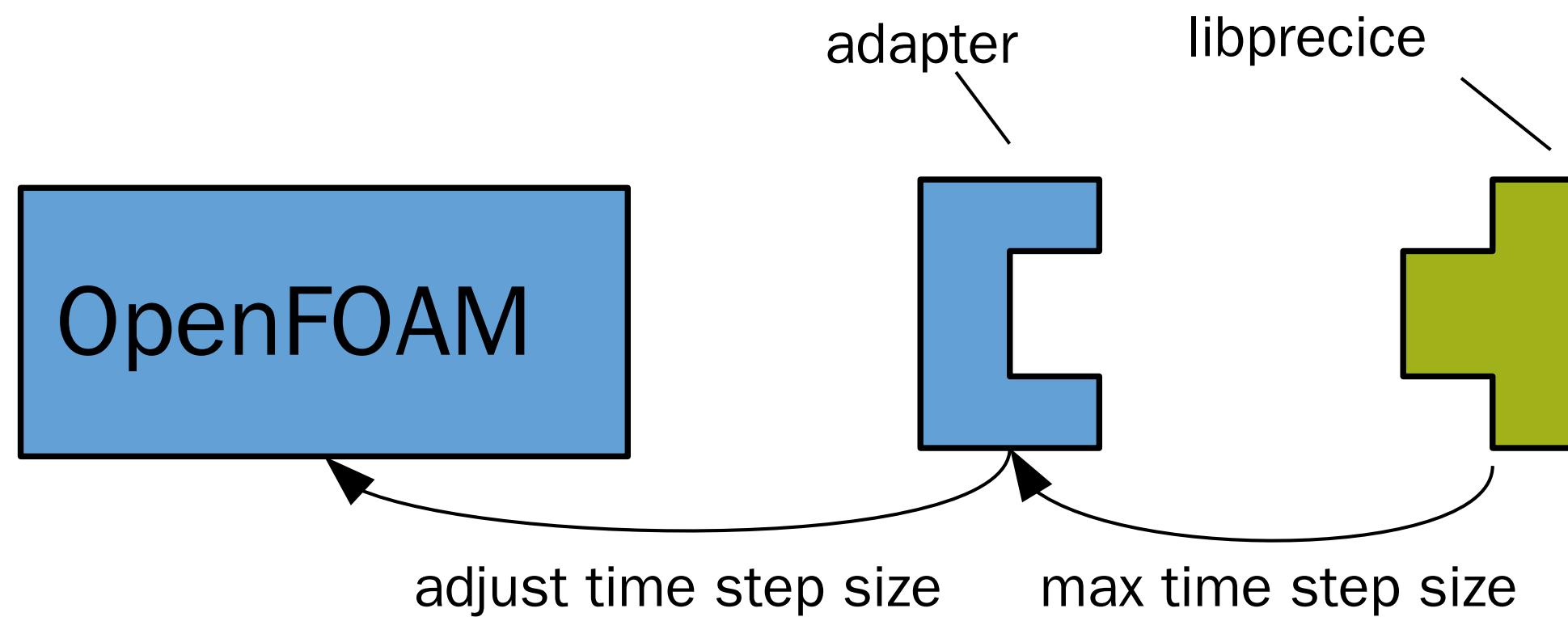
# What does the adapter do?



# 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 05 Jun 24

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 22.04 (Jammy Jellyfish), this is pretty easy: [download](#) and install our binary package by clicking on it or using the following commands:

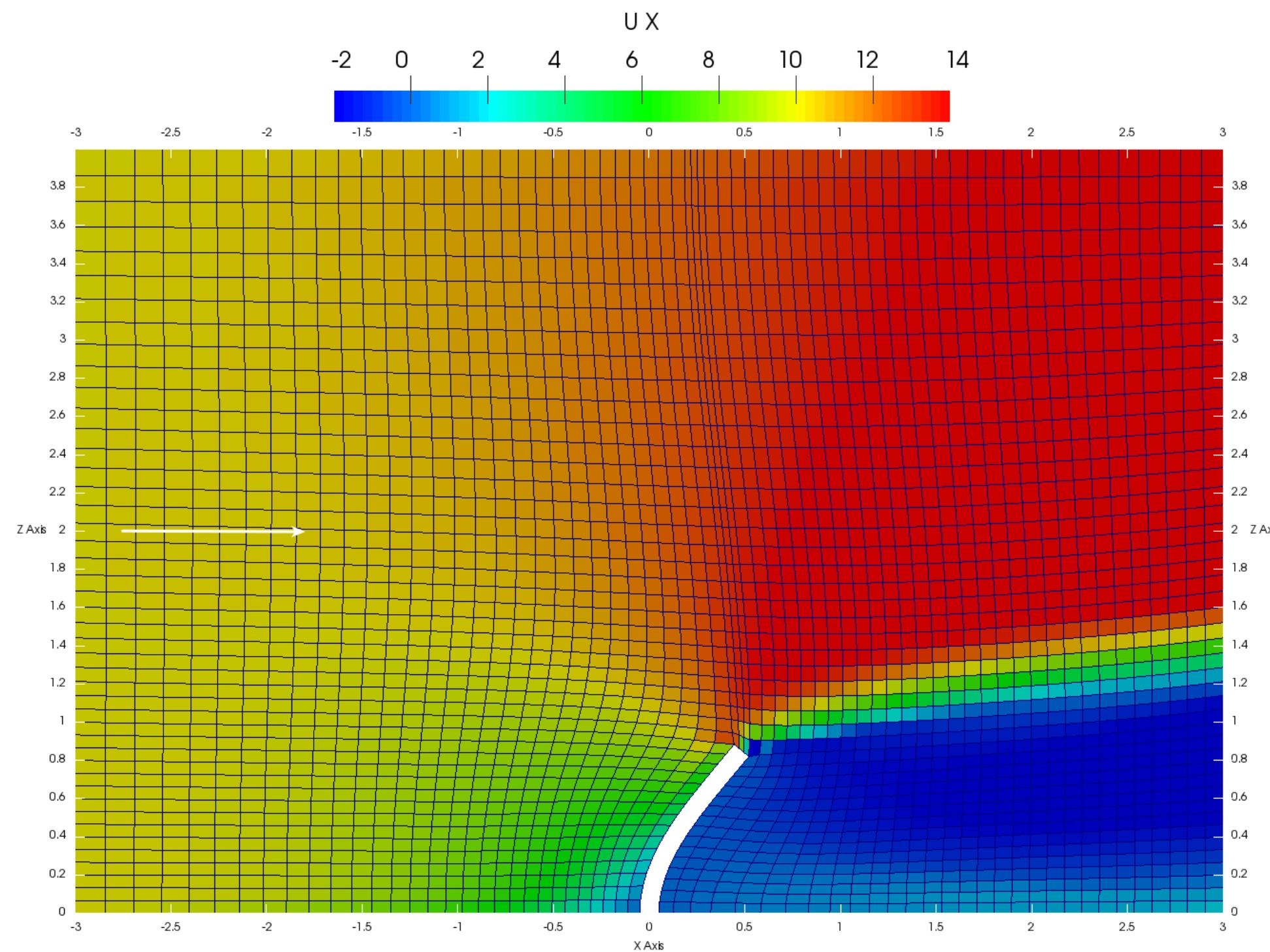
```
wget https://github.com/precice/precice/releases/download/v3.1.1/libprecice3_3.1.1_jammy.deb  
sudo apt install ./libprecice3_3.1.1_jammy.deb
```

OS	Package
Ubuntu 20.04 Focal Fossa	<a href="#">libprecice3_3.1.1_focal.deb</a>
Ubuntu 22.04 Jammy Jellyfish	<a href="#">libprecice3_3.1.1_jammy.deb</a>
Debian 11 "bullseye"	<a href="#">libprecice3_3.1.1_bullseye.deb</a>
Debian 12 "bookworm"	<a href="#">libprecice3_3.1.1_bookworm.deb</a>
Something else	See an <a href="#">overview of options</a>

- Facing any problems? [Ask for help](#).

2. We will use OpenFOAM here and in many of our tutorial cases, so [install OpenFOAM](#):

# Tutorial: Channel with a perpendicular flap



Find this tutorial on [precice.org/tutorials-perpendicular-flap.html](http://precice.org/tutorials-perpendicular-flap.html).

# Arbitrary solver combinations

FLUID

SOLID

---

pimpleFoam

CalculiX

---

SU2

deal.II

---

Nutils

FEniCS

---

DUNE

---

solidDisplacementFoam

---

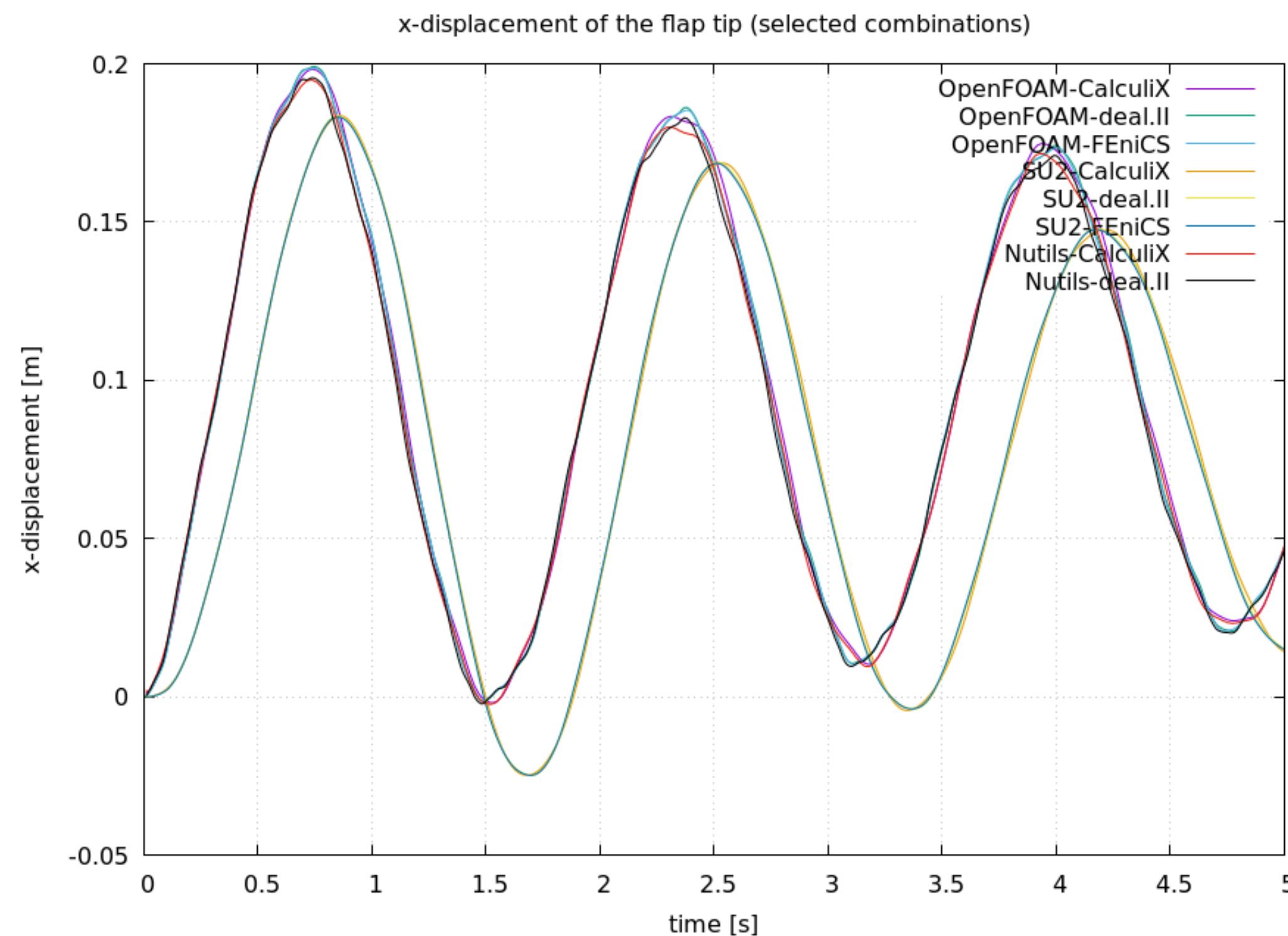
solids4Foam

---

Nutils

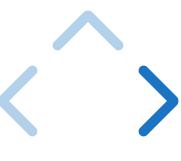


# Arbitrary solver combinations



# 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

```
1 // fluid-openfoam/0/U
2
3 flap
4 {
5     type          movingWallVelocity;
6     value         uniform (0 0 0);
7 }
```

```
1 // fluid-openfoam/0/pointDisplacement
2
3 flap
4 {
5     type          fixedValue;
6     value         $internalField;
7 }
```

```
1 // fluid-openfoam/constant/dynamicMeshDict
2
3 solver      displacementLaplacian;
```

# Load the OpenFOAM adapter

```
1 // fluid-openfoam/system/controlDict
2 functions
3 {
4     precICE_Adapter
5     {
6         type preciceAdapterFunctionObject;
7         libs ("libpreciceAdapterFunctionObject.so");
8     }
9 }
10
11 // Don't forget to build the adapter with Allwmake
```



# Configure the OpenFOAM adapter

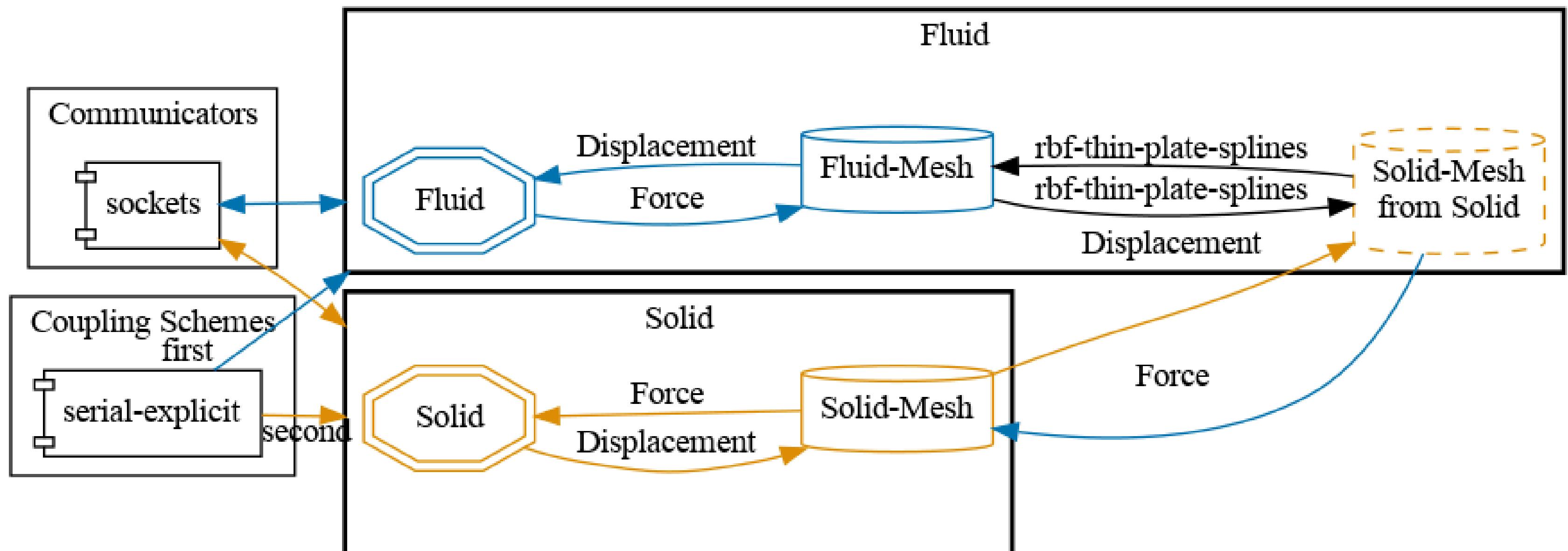
```
1 // fluid-openfoam/system/preciceDict
2 preciceConfig "../../precice-config.xml";
3 participant Fluid;
4
5 modules (FSI);
6
7 interfaces {
8     Interface1 {
9         mesh Fluid-Mesh;
10        patches (flap);
11        locations faceCenters;
12
13        readData (Displacement);
14
15        writeData (Force);
16    };
17}
```



# Configure the dealii adapter

```
1 // solid-dealii/parameters.prm
2
3 subsection precice configuration
4 # Cases: FSI3 or PF for perpendicular flap
5 set Scenario = PF
6
7 # Name of the precice configuration file
8 set precice config-file = ./precice-config.xml
9
10 # Name of the participant in the precice-config.xml file
11 set Participant name = Solid
12
13 # Name of the coupling mesh in the precice-config.xml file
14 set Mesh name = Solid-Mesh
15
16 # Name of the read data in the precice-config.xml file
17 set Read data name = Force
```

# Configure preCICE



# Configure preCICE

```
1 <data:vector name="Force" />
2 <data:vector name="Displacement" />
3
4 <mesh name="Fluid-Mesh" dimensions="2">
5   <use-data name="Force" />
6   <use-data name="Displacement" />
7 </mesh>
8
9 <mesh name="Solid-Mesh" dimensions="2">
10  <use-data name="Displacement" />
11  <use-data name="Force" />
12 </mesh>
13
14 <participant name="Fluid">
15   <export:vtk directory="results" />
16   <provide-mesh name="Fluid-Mesh" />
17   <receive-mesh name="Solid-Mesh" from="Solid" />
```

Notice the serial-explicit coupling scheme



# Configure preCICE

```
1 <data:vector name="Force" />
2 <data:vector name="Displacement" />
3
4 <mesh name="Fluid-Mesh" dimensions="2">
5   <use-data name="Force" />
6   <use-data name="Displacement" />
7 </mesh>
8
9 <mesh name="Solid-Mesh" dimensions="2">
10  <use-data name="Displacement" />
11  <use-data name="Force" />
12 </mesh>
13
14 <participant name="Fluid">
15   <export:vtk directory="results" />
16   <provide-mesh name="Fluid-Mesh" />
17   <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-thin-plate-splines
21     direction="write"
22     from="Fluid-Mesh"
23     to="Solid-Mesh"
24     constraint="conservative" />
25   <mapping:rbf-thin-plate-splines
26     direction="read"
27     from="Solid-Mesh"
28     to="Fluid-Mesh"
29     constraint="consistent" />
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-thin-plate-splines
21     direction="write"
22     from="Fluid-Mesh"
23     to="Solid-Mesh"
24     constraint="conservative" />
25   <mapping:rbf-thin-plate-splines
26     direction="read"
27     from="Solid-Mesh"
28     to="Fluid-Mesh"
29     constraint="consistent" />
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-thin-plate-splines
21     direction="write"
22     from="Fluid-Mesh"
23     to="Solid-Mesh"
24     constraint="conservative" />
25   <mapping:rbf-thin-plate-splines
26     direction="read"
27     from="Solid-Mesh"
28     to="Fluid-Mesh"
29     constraint="consistent" />
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" exchange-directory=".">
40
41 <coupling-scheme:serial-explicit>
42   <participants first="Fluid" second="Solid" />
43   <max-time value="1.5" />
44   <time-window-size value="0.01" />
45   <exchange data="Force" mesh="Solid-Mesh" from="Fluid" to="Solid" />
46   <exchange data="Displacement" mesh="Solid-Mesh" from="Solid" to="Fl
47 </coupling-scheme:serial-explicit>
```

Notice the serial-explicit coupling scheme



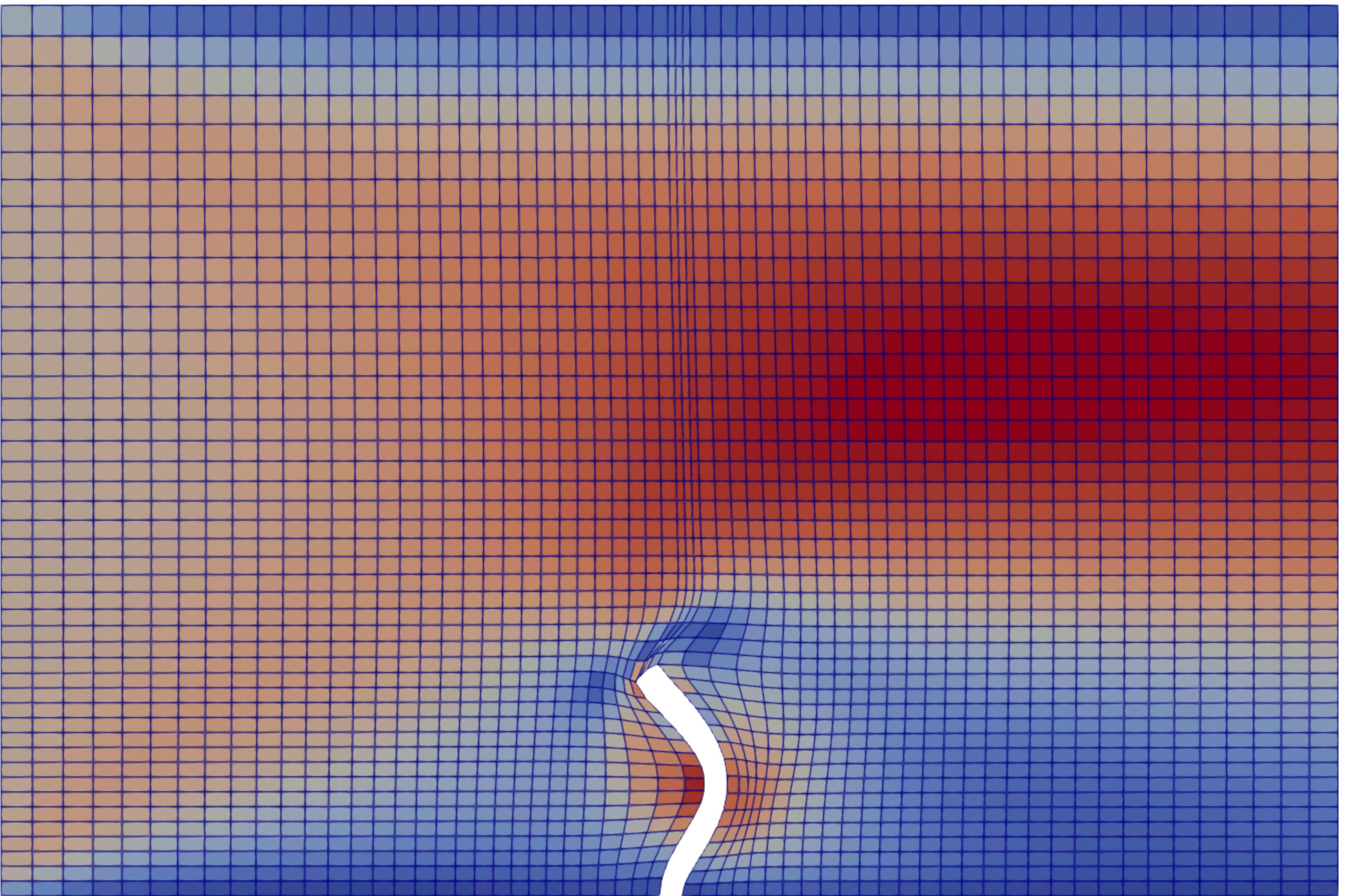
# Configure preCICE

```
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" exchange-directory=".">
40
41 <coupling-scheme:serial-explicit>
42   <participants first="Fluid" second="Solid" />
43   <max-time value="1.5" />
44   <time-window-size value="0.01" />
45   <exchange data="Force" mesh="Solid-Mesh" from="Fluid" to="Solid" />
46   <exchange data="Displacement" mesh="Solid-Mesh" from="Solid" to="Fl
47 </coupling-scheme:serial-explicit>
```

Notice the serial-explicit coupling scheme



```
vagrant@precicevm:~/tutor... fsi-course - File Manager 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/fluid-openfoam  
vagrant@precicevm: ~/tutorials/fsi-course 160x3  
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
```

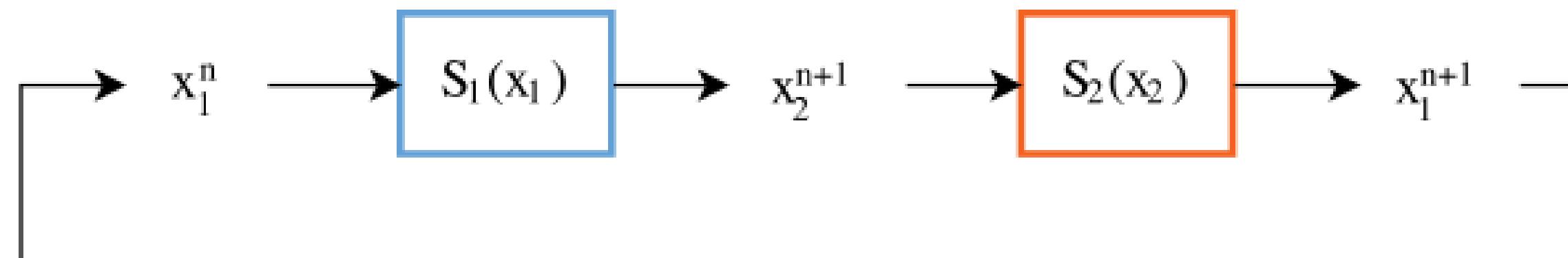


When we decrease the solid density further to 1, the participants have stronger coupling, then:

The image shows two terminal windows side-by-side. Both windows have a red header bar with the text "chenju@lapsgs05: ~/Desktop/perpendicular\_flap/serial-explicit/fluid-openfoam 94x49". The left window's header also includes "[10:07:12]". The right window's header also includes "[10:07:14]". The main body of both windows is dark gray and mostly empty, indicating they are running simulations.

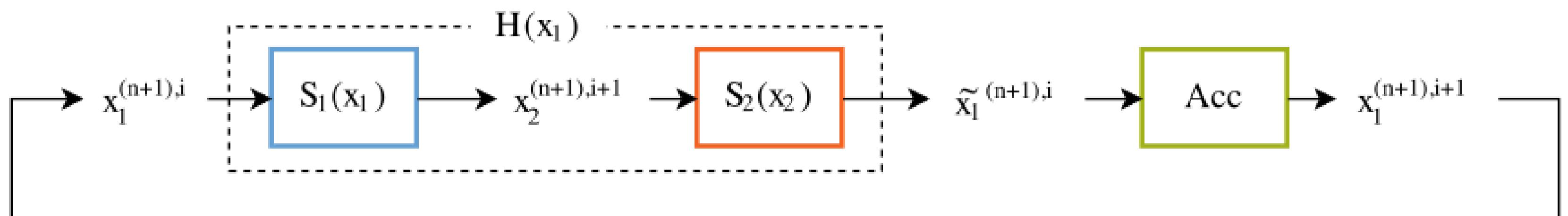
```
chenju@lapsgs05: ~/Desktop/perpendicular_flap/serial-explicit/fluid-openfoam 94x49
chenju@lapsgs05: ~/Desktop/perpendicular_flap/serial-explicit/fluid-openfoam [10:07:12]
chenju@lapsgs05: ~/Desktop/perpendicular_flap/serial-explicit/solid-dealii 94x49
chenju@lapsgs05: ~/Desktop/perpendicular_flap/serial-explicit/solid-dealii [10:07:14]
```

# 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

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

Only change from explicit to implicit coupling



# Configure the coupling scheme

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

Only change from explicit to implicit coupling



# Configure the coupling scheme

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

Only change from explicit to implicit coupling



# Configure the coupling scheme

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

Only change from explicit to implicit coupling



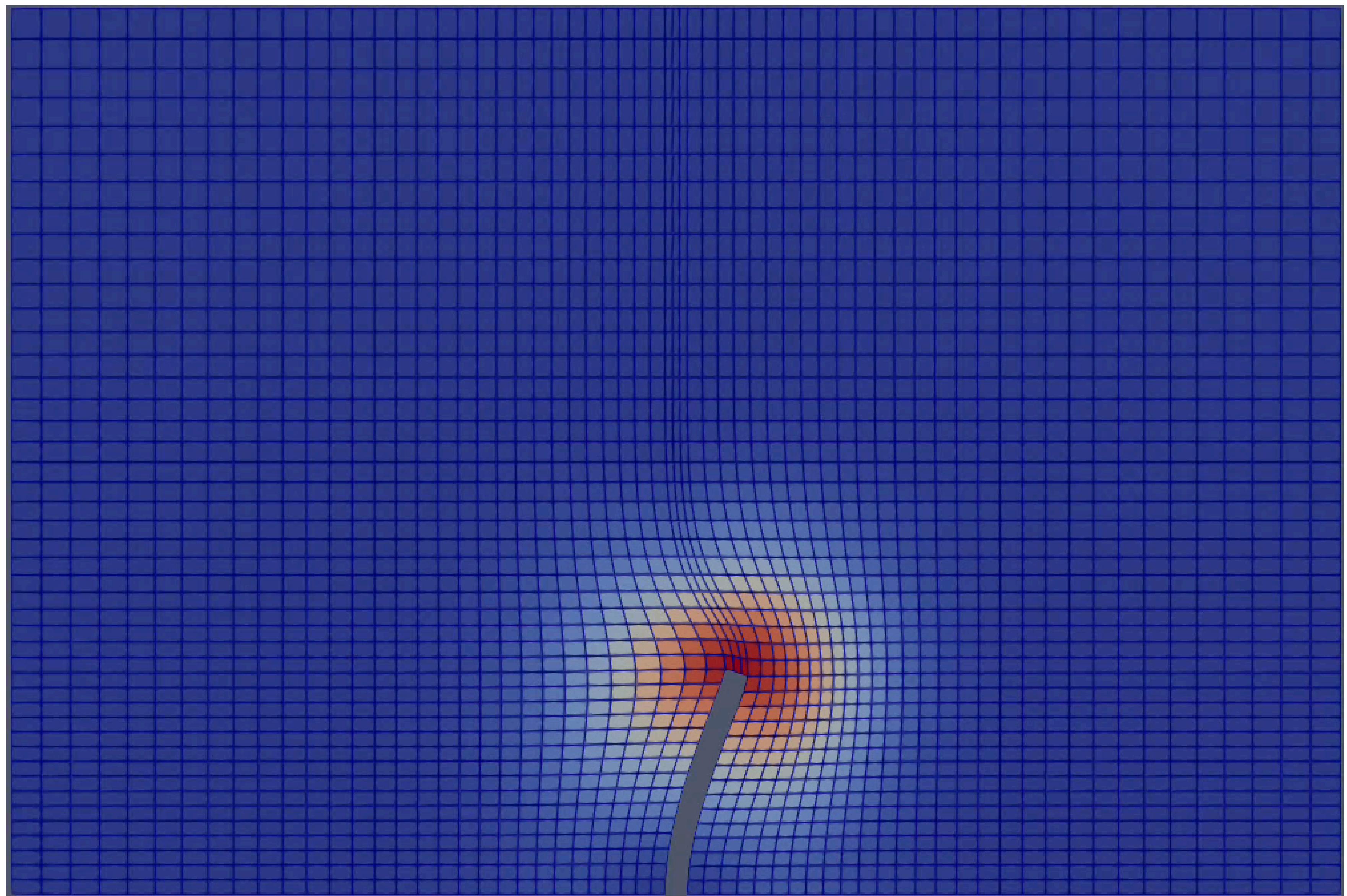
# Configure the coupling scheme

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

Only change from explicit to implicit coupling



With solid density equal to 42:



With solid density equal to 1: break again!

Still not enough stability for strong coupling!

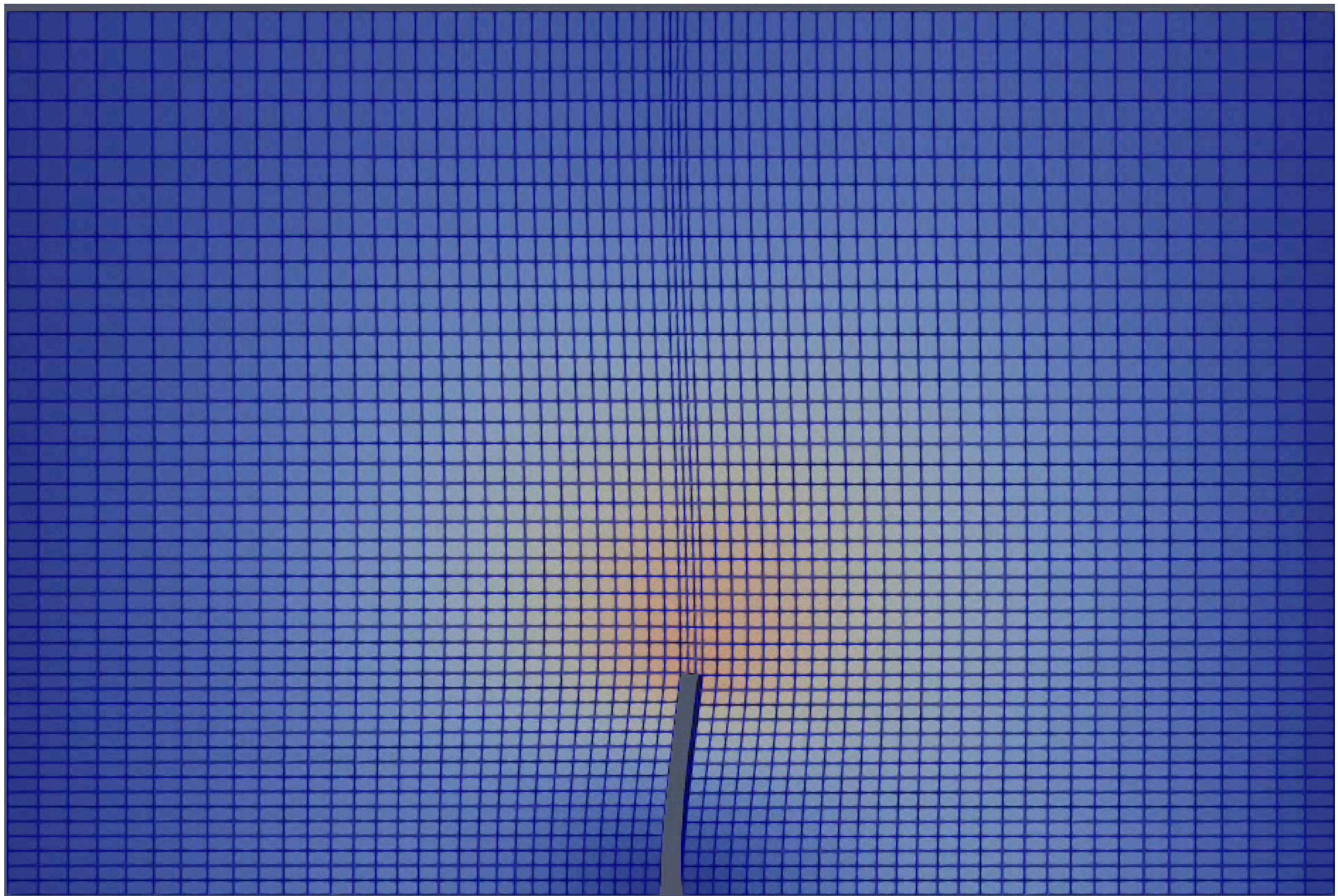
# Configure the coupling scheme

```
1 <coupling-scheme:serial-implicit>
2   <participants first="Fluid" second="Solid" />
3   <max-time value="1.5" />
4   <time-window-size value="0.01" />
5   <exchange data="Force" mesh="Solid-Mesh" from="Fluid" to="Solid" />
6   <exchange data="Displacement" mesh="Solid-Mesh" from="Solid" to="Fl
7   <relative-convergence-measure limit="5e-3" data="Displacement" mesh=
8   <relative-convergence-measure limit="5e-3" data="Force" mesh="Solid"
9   <max-iterations value="50" />
10  <acceleration:constant>
11    <relaxation value="0.5" />
12  </acceleration:constant>
13 </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>
```

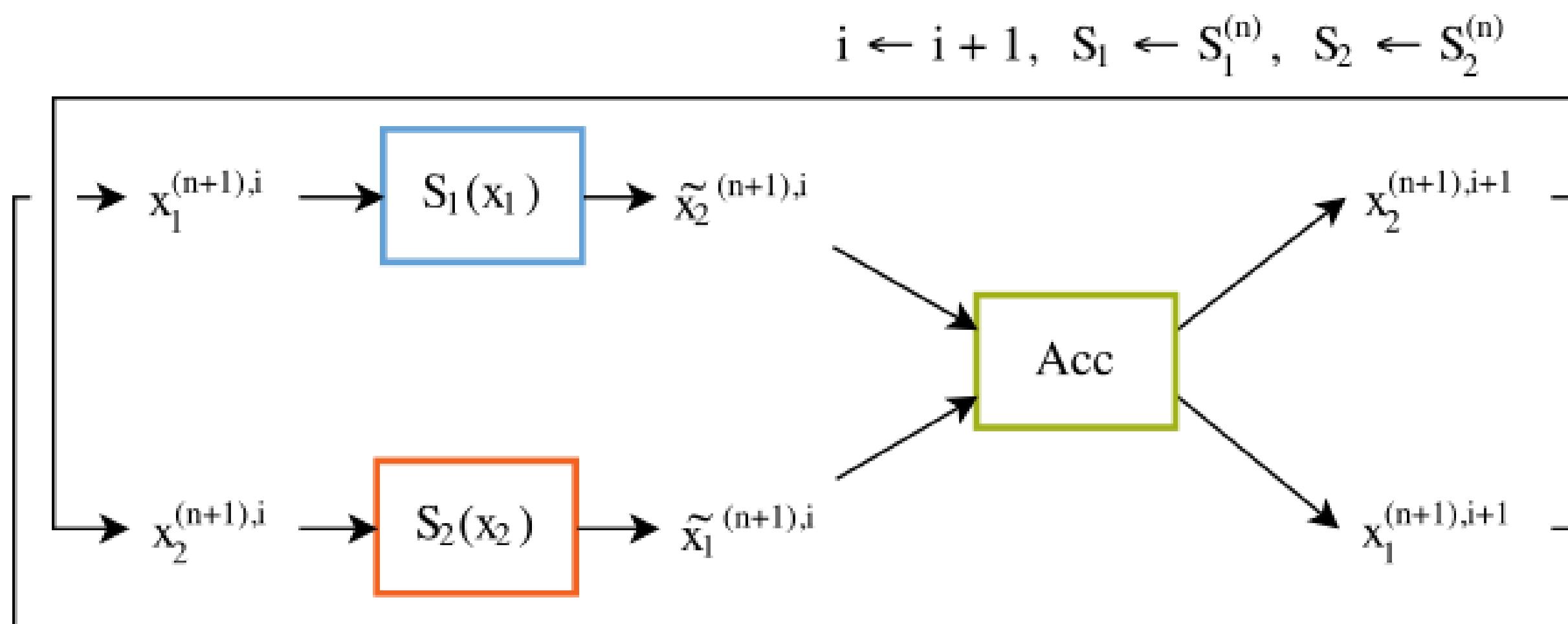
# Improvement 2: Anderson acceleration

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

# Improvement 2: Anderson acceleration

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

# There are also parallel schemes



Can use fields from both / all participants

# Improvement 3: Parallel coupling schemes

```
1 <coupling-scheme:parallel-implicit>
2   <time-window-size value="0.01" />
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" />
```

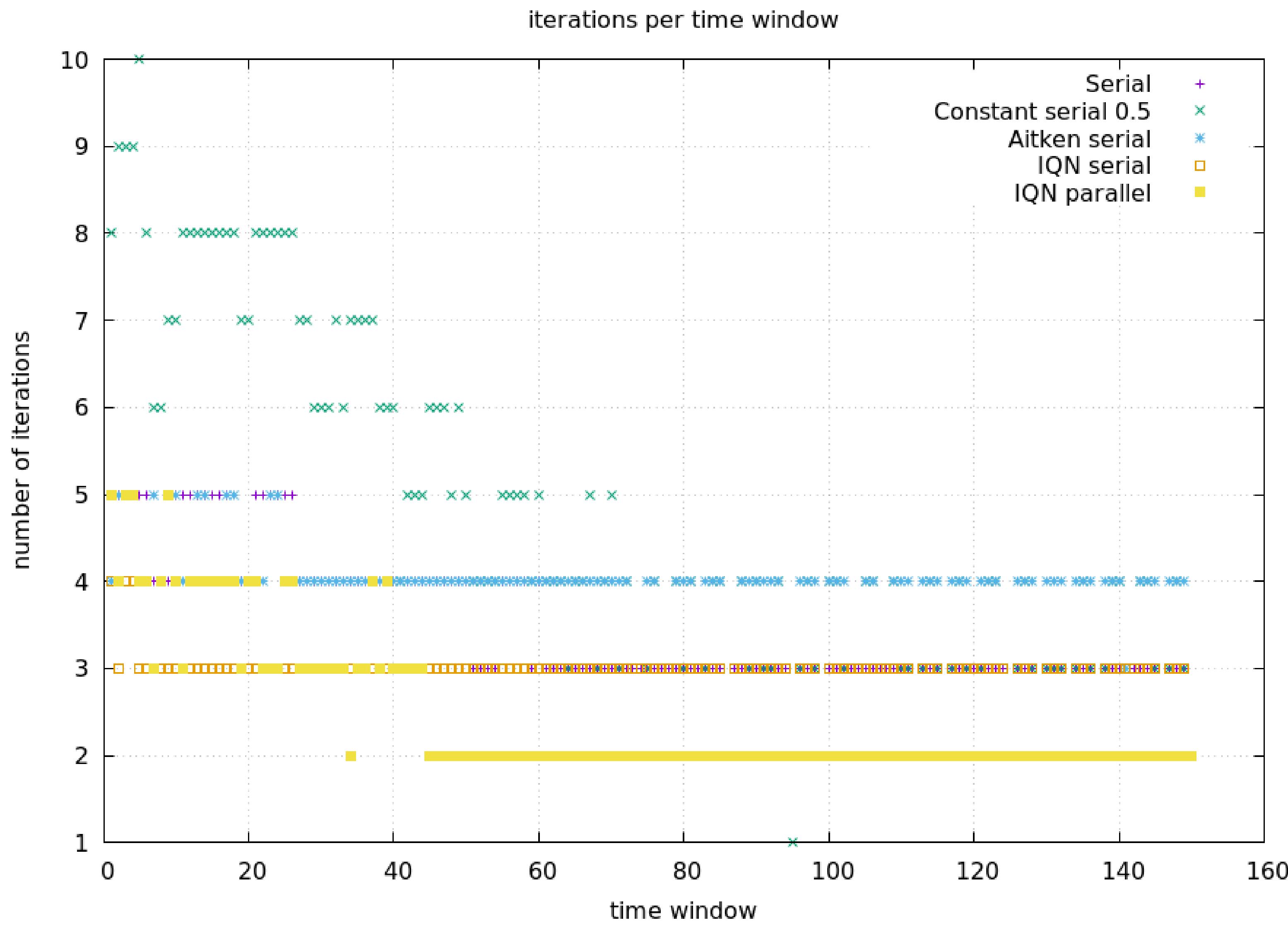
# Improvement 3: Parallel coupling schemes

```
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>
```

density = 42    density = 1

serial-explicit	div	div
se-implicit	3.41	div
se-im-const	4.48	5.27
se-im-Aitken	3.74	5.3
se-im-ILS	2.95	3.07
parallel-im-ILS	2.46	2.33

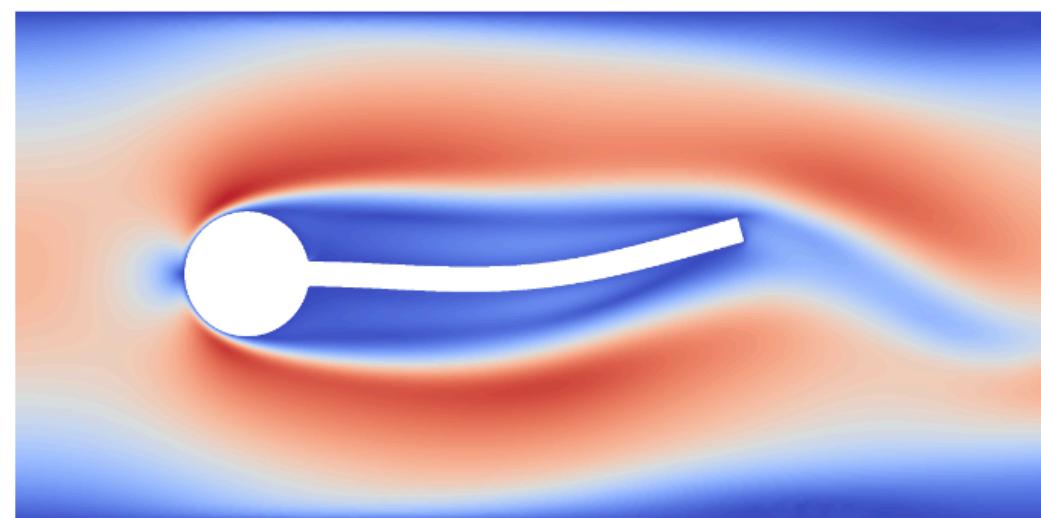




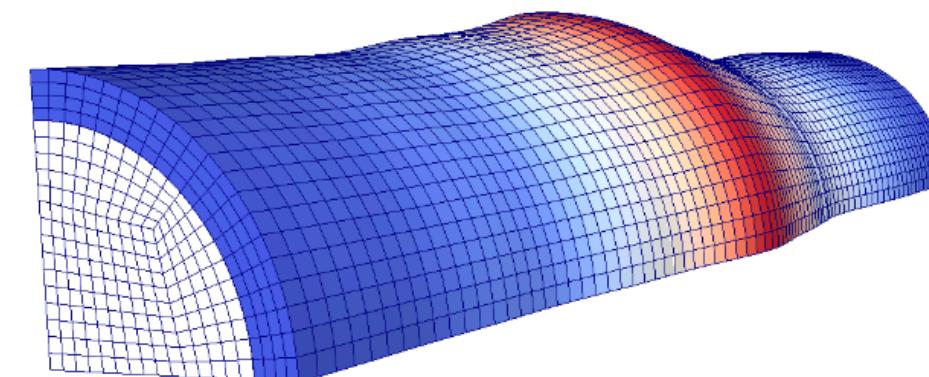
Beware: Configuration not fine-tuned, not rigorous comparison, application context.

# Rigorous comparison from the literature

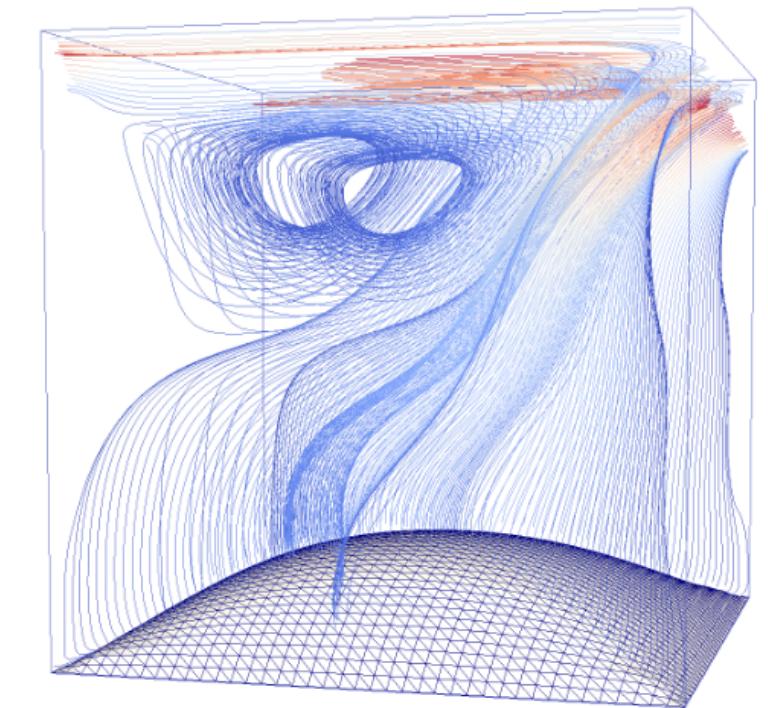
FSI3



3D-Tube



Driven Cavity

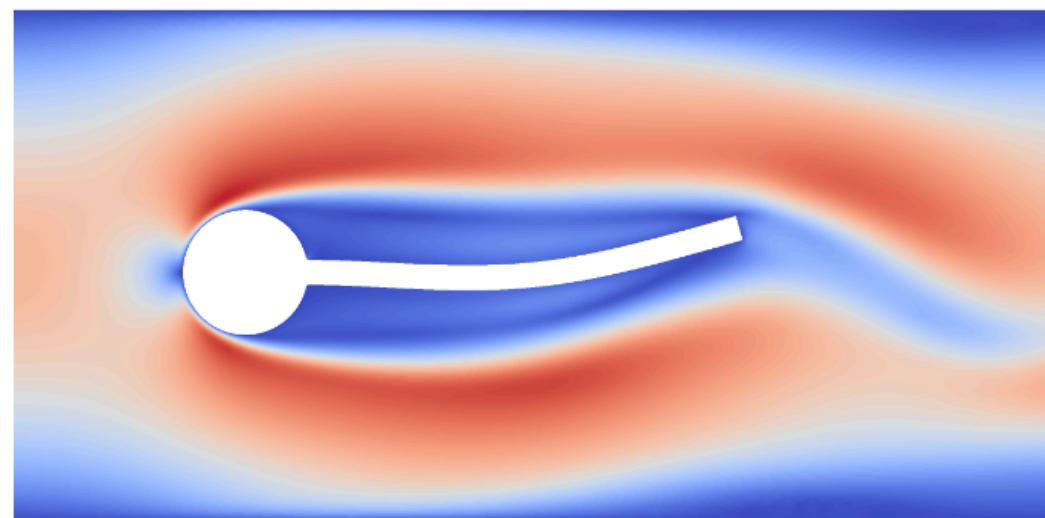


Mean Iterations	Aitken	Quasi-Newton
FSI3	17.0	3.7
3D-Tube	Div.	7.5
Driven Cavity	7.4	3.0

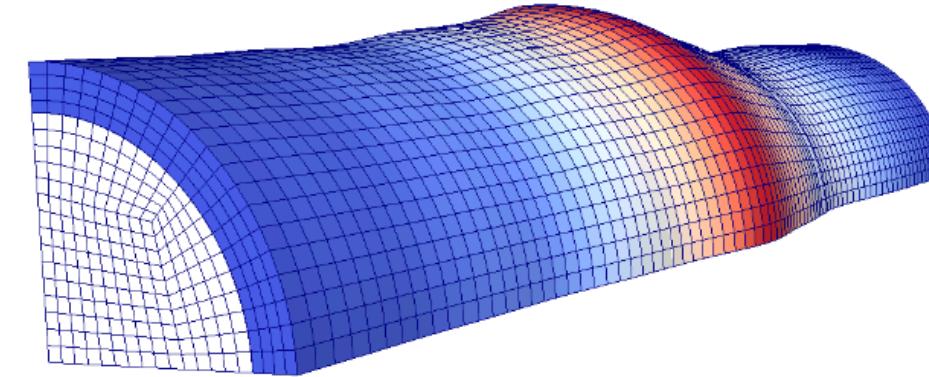
Source: Dissertation of Benjamin Uekermann (2016)

# Rigorous comparison from the literature

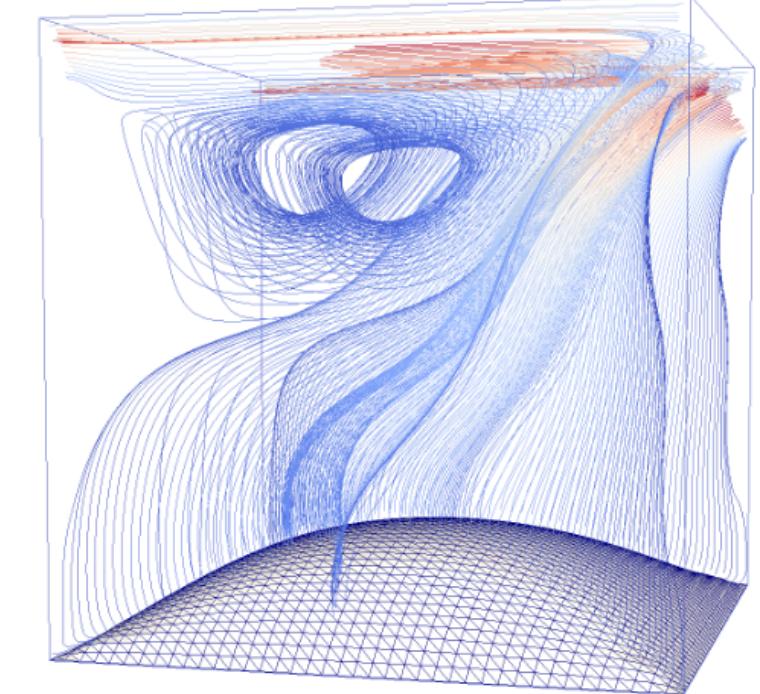
FSI3



3D-Tube



Driven Cavity



Mean Iterations	Serial Coupling	Parallel Coupling
FSI3	3.7	<b>3.3</b>
3D-Tube	<b>7.5</b>	10.4
Driven Cavity	3.0	<b>2.0</b>

Source: Dissertation of Benjamin Uekermann (2016)

# Resources

# Start here: [precice.org](https://precice.org)



Quickstart   Docs   Tutorials   Community   Blog ↗   About

Search ...

Search by algolia



Registration for the preCICE workshop 2024 is now open! Submit your abstract until June 30.

## Welcome to preCICE

The coupling library for partitioned multi-physics simulations.

Star on Github 686

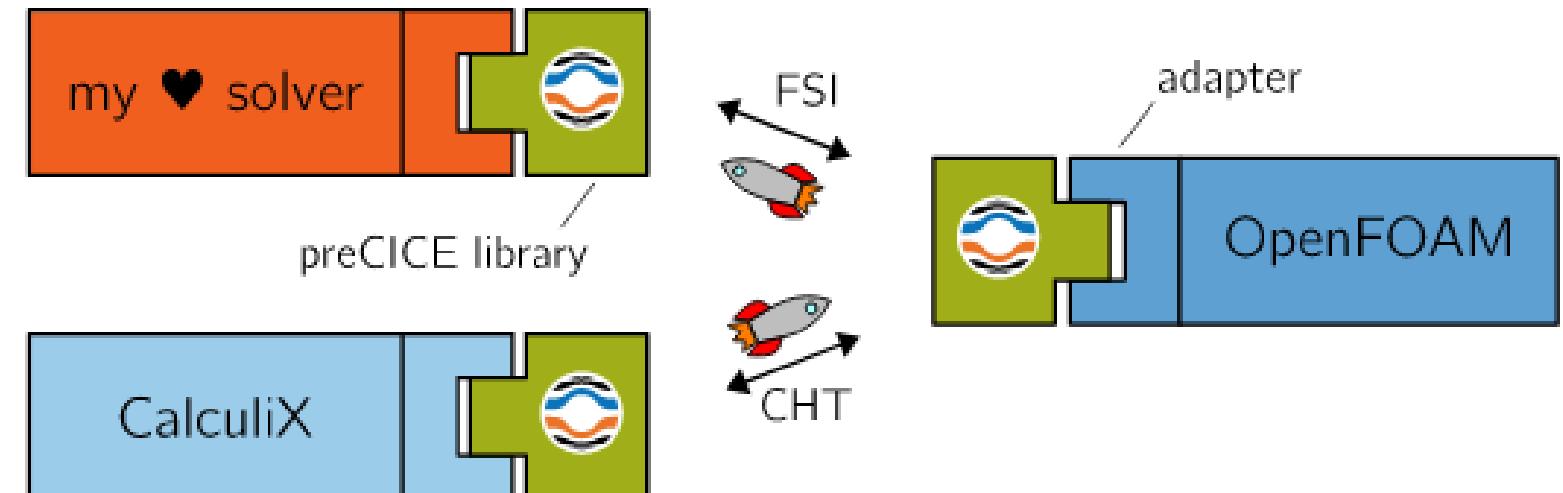
Latest v3.1.1 (Apr 12, 2024)

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: 600+ visitors, 4k pageviews



# Documentation

[Quickstart](#)[Docs](#)[Tutorials](#)[Community](#)[Blog](#)[About](#)

Search ...

Search by algolia



Registration for the preCICE workshop 2024 is now open! Submit your abstract until June 30.

**Docs v3.1.1**

Fundamentals	▲
<a href="#">Overview</a>	
<a href="#">Terminology</a>	
<a href="#">License information</a>	
<a href="#">Literature guide</a>	
<a href="#">Roadmap</a>	
<a href="#">Previous versions</a>	
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](#)

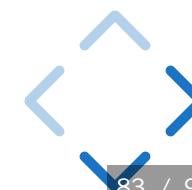
Updated 15 May 24

### 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.

The preCICE library offers parallel communication means, data mapping schemes, and methods for transient equation coupling. Additionally, we are actively developing methods for time interpolation and more features (see our [roadmap](#)). preCICE is written in C++ and offers additional bindings for C, Fortran, Python, and Matlab. Coupling your own solver is very easy, due to the minimally-invasive approach of preCICE. Once you add the (very few) calls to the preCICE library in your code, you can couple it with any other code at runtime. For well-known solvers such as OpenFOAM, deal.II, FEniCS, Nutils, CalculiX, or SU2, you can use one of our official adapters.

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



# Discuss & get help

 We look forward to meeting you in Stuttgart in September! Happy coupling!

categories ► tags ► Categories Latest New (1) Top Bookmarks + New Topic

Category	Topics
<b>News</b> News, announcements, "blog"-like posts	42
<b>Is preCICE for me?</b> General questions regarding preCICE as a coupling solution.	65
<b>Installing preCICE</b> Any issues with getting the preCICE library installed	84
<b>Using preCICE</b> Directly using the preCICE API, configuring a new simulation	324
<b>Official adapters and tutorials</b> Installing, configuring, and extending the official adapters, as well as running or modifying the tutorials	270
<b>Community projects</b> Share your simulation cases for everybody to admire and try.	5
<b>Jobs &amp; theses market</b> Jobs and thesis projects related to preCICE.	2

Latest

 Welcome to the preCICE Forum on Discourse Site Feedback	1 Nov 2019
 About driven cavity problem!	0 7h
 About the use of acceleration algorithms?	5 8h
 Floating point exception while using parallel-explicit coupling scheme for two different interfaces openfoam	2 1d
 A coupling problem between Openfoam and Calculix Using preCICE calculix fsi openfoam	0 2d
 Can preCICE be used with moving meshes? Using preCICE faq	27 4d

Active since October 2019, 480+ users, 7k posts in 800+ topics



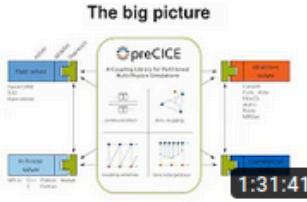
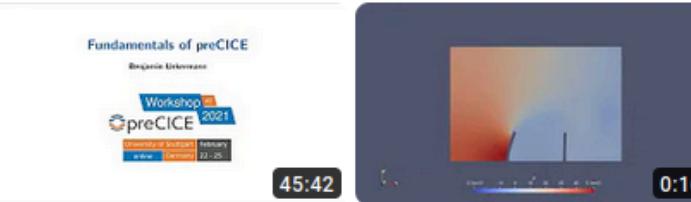
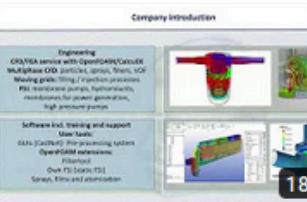
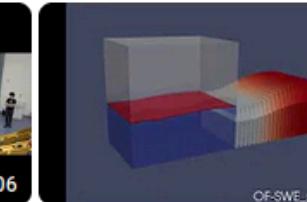
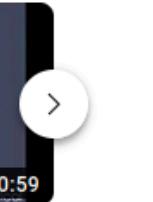
# Learn: YouTube

 **preCICE Coupling**  
@preCICECoupling 455 subscribers 42 videos  
A coupling library for partitioned multi-physics simulations, including, but n... >

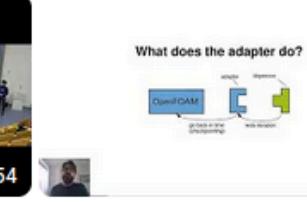
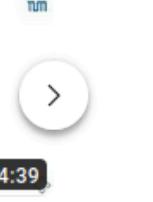
[Subscribe](#)

[HOME](#) [VIDEOS](#) [PLAYLISTS](#) [COMMUNITY](#) [CHANNELS](#) [ABOUT](#) [🔍](#)

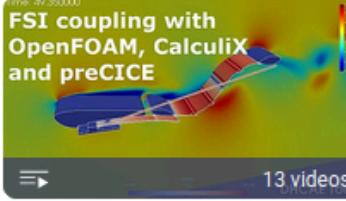
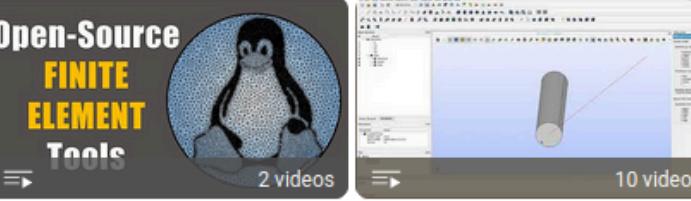
**Popular videos** ► [Play all](#)

 <b>Multiphysics Modeling with the preCICE Coupling Library</b> 2.3K views • 2 years ago	 <b>Fundamentals of preCICE (Benjamin Uekermann,...)</b> 1.4K views • 2 years ago	 <b>Multicoupling with preCICE: two flaps in a channel...</b> 1K views • 2 years ago	 <b>Transfer of FSI coupling with preCICE, OpenFOAM and...</b> 868 views • 2 years ago	 <b>6-way coupling of DEM+CFD+FEM with preCIC...</b> 781 views • 3 years ago	 <b>Coupling of Shallow Water Equations and OpenFOAM...</b> 700 views • 2 years ago
---	---	--	--	--	---

**Talks about preCICE** ► [Play all](#)  
Conference talks about the preCICE coupling library

 <b>Couple scientific simulation codes with preCICE A jour...</b> FOSDEM 1.5K views • 4 years ago	 <b>Fundamentals of preCICE (Benjamin Uekermann,...)</b> preCICE Coupling 1.4K views • 2 years ago	 <b>What is new in preCICE? (Frédéric Simonis, preCICE...)</b> preCICE Coupling 122 views • 2 years ago	 <b>SimTech and the Simulation of Large Systems</b> Exzellenzcluster SimTech 669 views • 3 years ago	 <b>High-order and multi-rate time stepping with preCICE...</b> preCICE Coupling 231 views • 3 years ago	 <b>The OpenFOAM-preCICE adapter (Gerasimos...)</b> preCICE Coupling 366 views • 2 years ago
--	--	---	--	--	--

**Community**

 <b>Simulations using preCICE</b> Playlist · preCICE Coupling View full playlist	 <b>Users talking about preCICE</b> Playlist · preCICE Coupling View full playlist	 <b>FSI for vascular flows using OpenFOAM, preCICE, and...</b> Playlist · Torsten Schenkel View full playlist
---	--	---

Active since 2020, 580+ subscribers, 50+ videos



# Get news: Twitter

← **preCICE (the coupling library)**  
295 Tweets



**preCICE (the coupling library)**  
@preCICE\_org

A free/open-source coupling library for partitioned multi-physics simulations, including fluid-structure interaction and more. Mastodon:  
[@precice@fosstodon.org](mailto:@precice@fosstodon.org)

⌚ Germany ⌂ [precice.org](https://precice.org) 📅 Joined April 2018

0 Following 462 Followers

Active since 2018, 400+ followers  
also on [@precice@fosstodon.org](mailto:@precice@fosstodon.org)



# Read more: OpenFOAM-preCICE paper

OpenFOAM®  
Journal

Current Archives Announcements About ▾ Authors ▾

Search

Home / Archives / Vol. 3: OpenFOAM® Journal 2023 / Full Papers

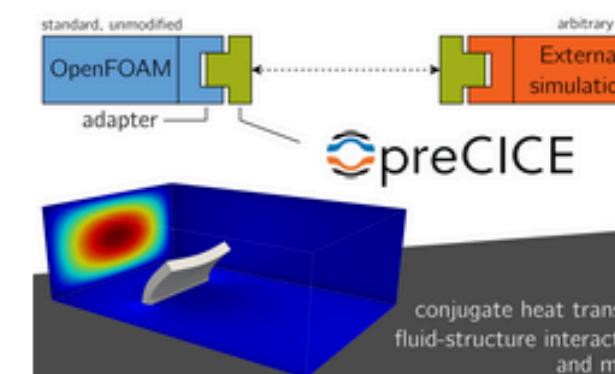
## OpenFOAM-preCICE: Coupling OpenFOAM with External Solvers for Multi-Physics Simulations

**Gerasimos Chourdakis**  
Technical University of Munich  
<https://orcid.org/0000-0002-3977-1385>

**David Schneider**  
University of Stuttgart  
<https://orcid.org/0000-0002-3487-9688>

**Benjamin Uekermann**  
University of Stuttgart  
<https://orcid.org/0000-0002-1314-9969>

**DOI:** <https://doi.org/10.51560/ofj.v3.88>



The diagram shows the preCICE coupling interface. It consists of two main components: 'OpenFOAM' (represented by a blue rectangle) and 'External simulation' (represented by an orange rectangle). They are connected via a green 'adapter' block. A dashed arrow points from the adapter to the external simulation, indicating the flow of data. The 'preCICE' logo is positioned below the adapter. Below the diagram, there is a 3D visualization of a fluid domain with a ship-like object, labeled 'conjugate heat transfer fluid-structure interaction and more'.

[PDF](#) [Discussion Forum](#)  
[Abstract Video](#)

Sponsors

**eGOMPUTE**  
A GRIDCORE COMPANY

**upstreamCFD**

# Read more: preCICE paper

The screenshot shows the Open Research Europe website interface. At the top, there is a European Commission logo and a search bar with a "Search" button. Below the header, there are navigation links for "Research and Innovation", "Open Research Europe", "Browse", "Gateways & Collections", "How to Publish", "About", "Resource Hub", "Blog", and "Sign in". The main content area displays a software tool article titled "preCICE v2: A sustainable and user-friendly coupling library [version 2; peer review: 2 approved]". The article has 441 views, 176 downloads, and 29 citations. It is categorized as a "SOFTWARE TOOL ARTICLE". The authors listed are Gerasimos Chourdakis, Kyle Davis, Benjamin Rodenberg, Miriam Schulte, Frédéric Simonis, Benjamin Uekermann, Georg Abrams, Hans-Joachim Bungartz, Lucia Cheung Yau, Ishaan Desai, Konrad Eder, Richard Hertrich, Florian Lindner, Alexander Rusch, Dmytro Sashko, David Schneider, Amin Totounferoush, Dominik Volland, Peter Vollmer, and Oguz Ziya Koseomur. There are sections for "Horizon 2020 gateway" and "Marie-Sklodowska-Curie Actions (MSCA) gateway". To the right, there is a "Open Peer Review" section showing approval status for two versions: Version 2 (Revision) from 30 Sep 22 and Version 1 from 29 Apr 22, both of which have been approved. The page also includes a "Comments on this article" section.

European Commission | Search | Search

Research and Innovation

Open Research Europe

Browse Gateways & Collections How to Publish About Resource Hub Blog Sign in

441 Views | 176 Downloads | 29 Citations

Cite | Download | Export | Share | Track

Home > Articles > preCICE v2: A sustainable and user-friendly coupling library

SOFTWARE TOOL ARTICLE

REVISED preCICE v2: A sustainable and user-friendly coupling library [version 2; peer review: 2 approved]

Gerasimos Chourdakis, Kyle Davis, Benjamin Rodenberg, Miriam Schulte, Frédéric Simonis, Benjamin Uekermann, Georg Abrams, Hans-Joachim Bungartz, Lucia Cheung Yau, Ishaan Desai, Konrad Eder, Richard Hertrich, Florian Lindner, Alexander Rusch, Dmytro Sashko, David Schneider, Amin Totounferoush, Dominik Volland, Peter Vollmer, Oguz Ziya Koseomur

This article is included in Horizon 2020 gateway

H2020

This article is included in Marie-Sklodowska-Curie Actions (MSCA) gateway

Open Peer Review

Approval Status ✓ ✓

	1	2
Version 2 (Revision) 30 Sep 22		
Version 1 29 Apr 22	<a href="#">view</a>	<a href="#">view</a>

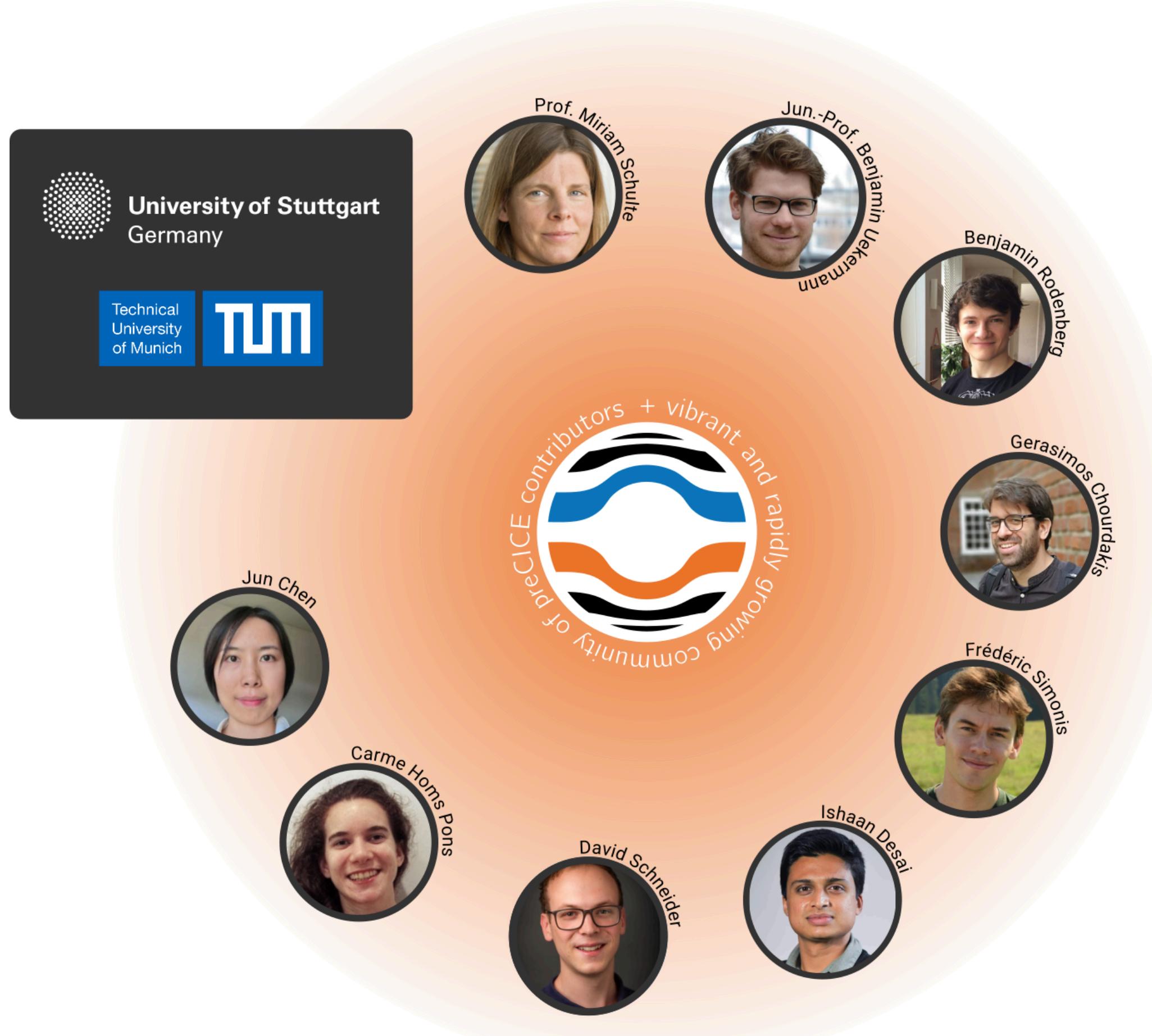
1. Axelle Viré, Delft University of Technology, Delft, The Netherlands

2. Garth Wells, University of Cambridge, Cambridge, UK

Comments on this article



# The people behind preCICE



# Meet the people



# Incoming Workshop



University of Stuttgart

Germany

September

24 - 27



# 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

Jun Chen(UST) + Gerasimos Chourdakis (UST/TUM) + many more (see [precice.org/about](http://precice.org/about))



This work is licensed under a [Creative Commons Attribution 4.0 International License](#).  
Get these slides.