

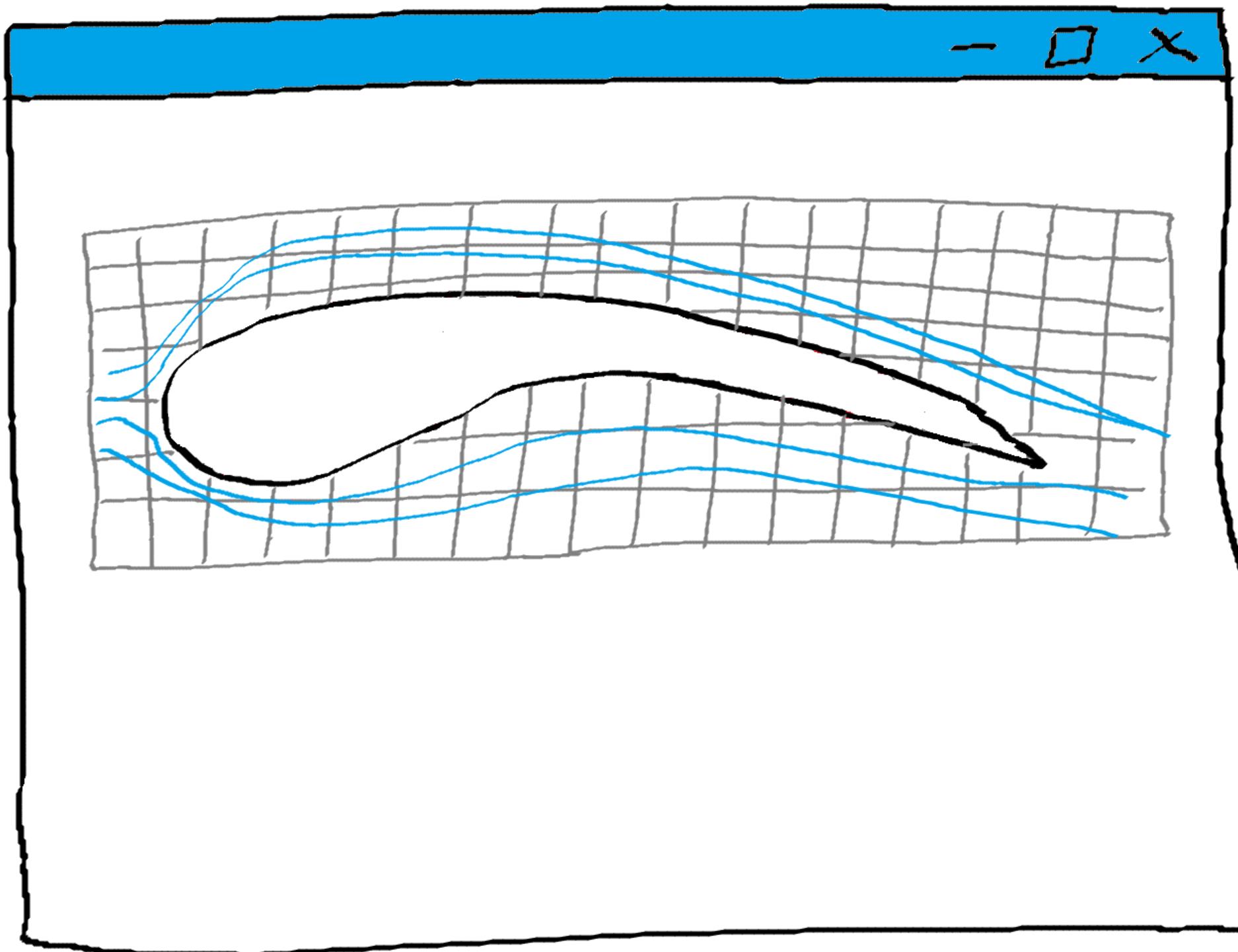
18th OpenFOAM Workshop - Genoa, July 11-14, 2023

Getting started with OpenFOAM-preCICE for FSI simulations

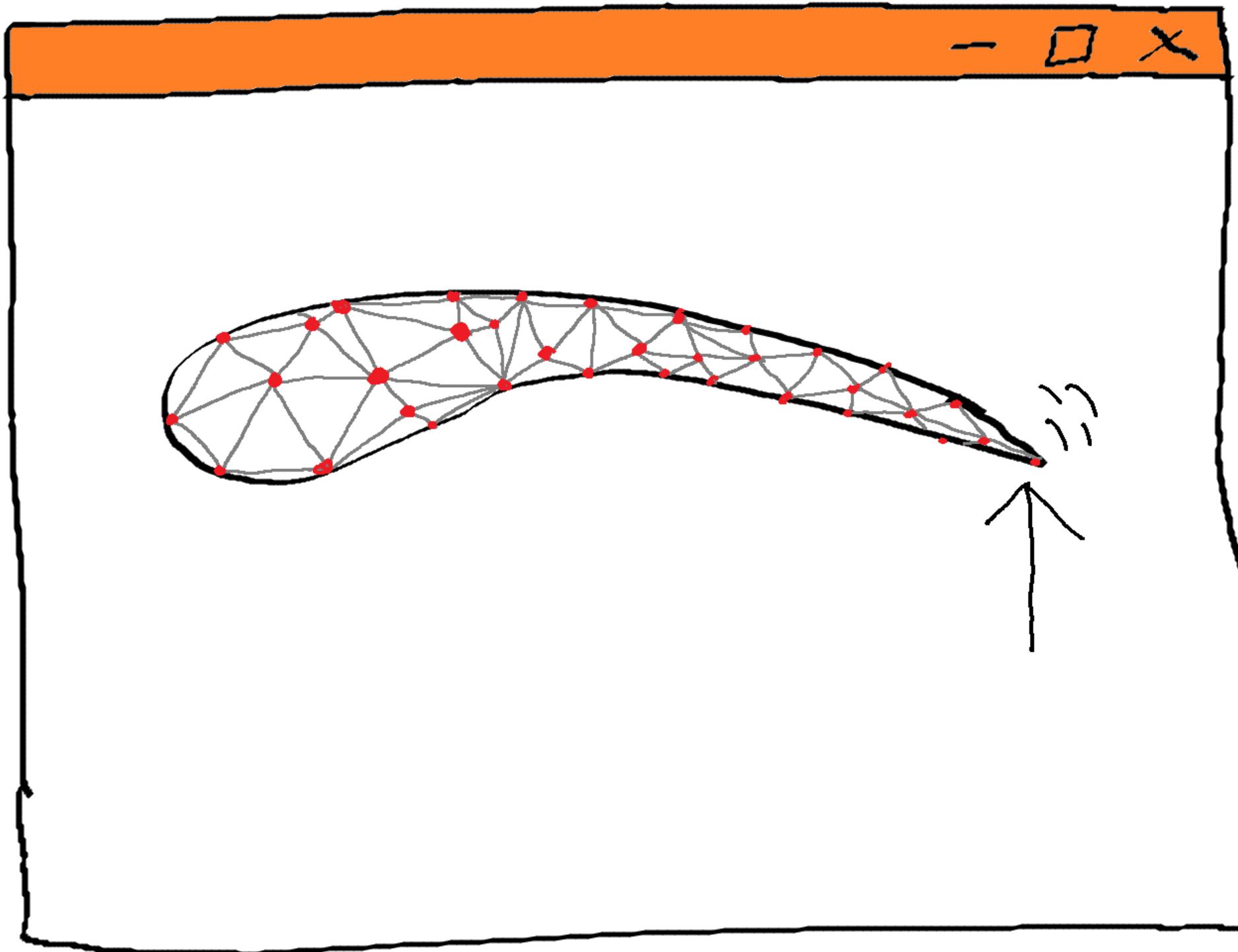
Gerasimos Chourdakis, Technical University of Munich
chourdak@in.tum.de



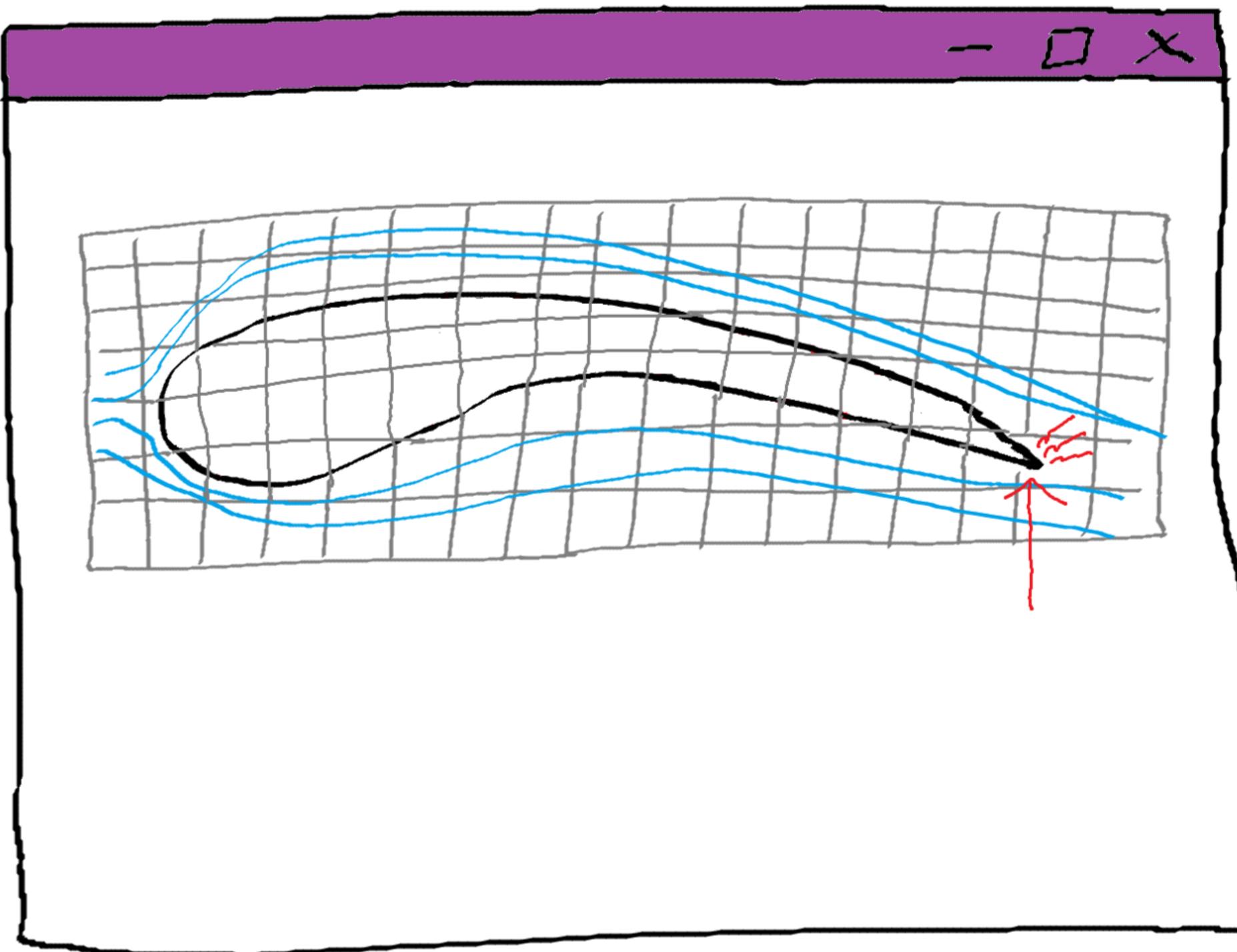
CFD: Simulating flow around a wing



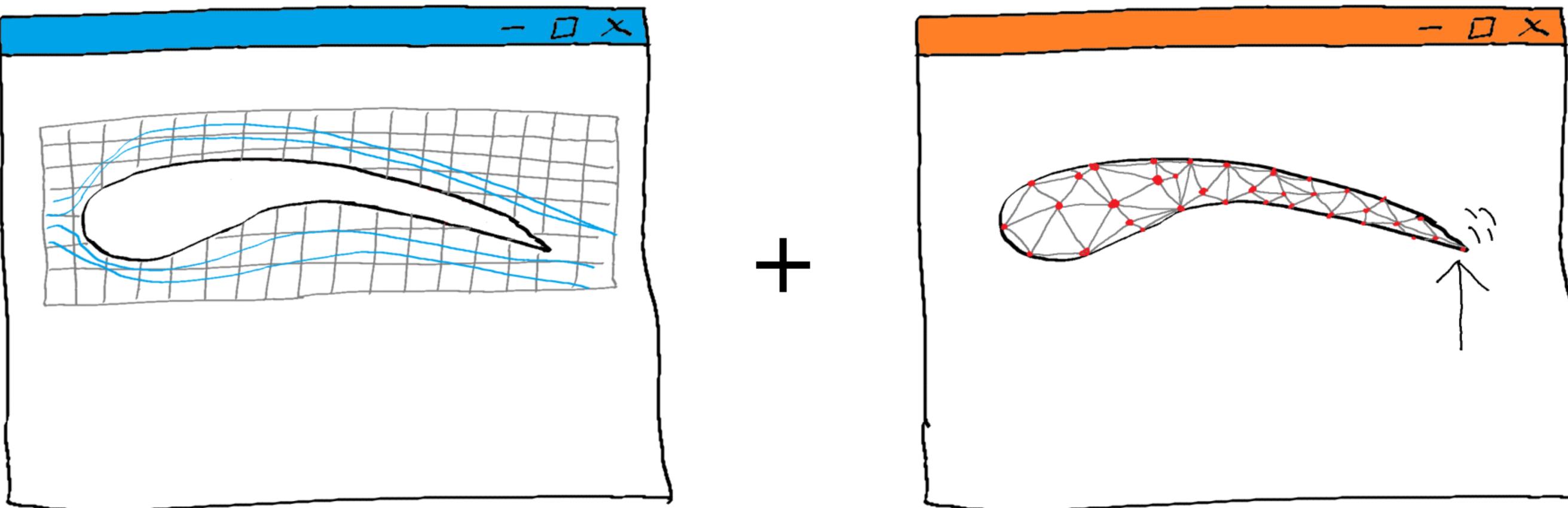
FEM: Simulating stresses on a wing



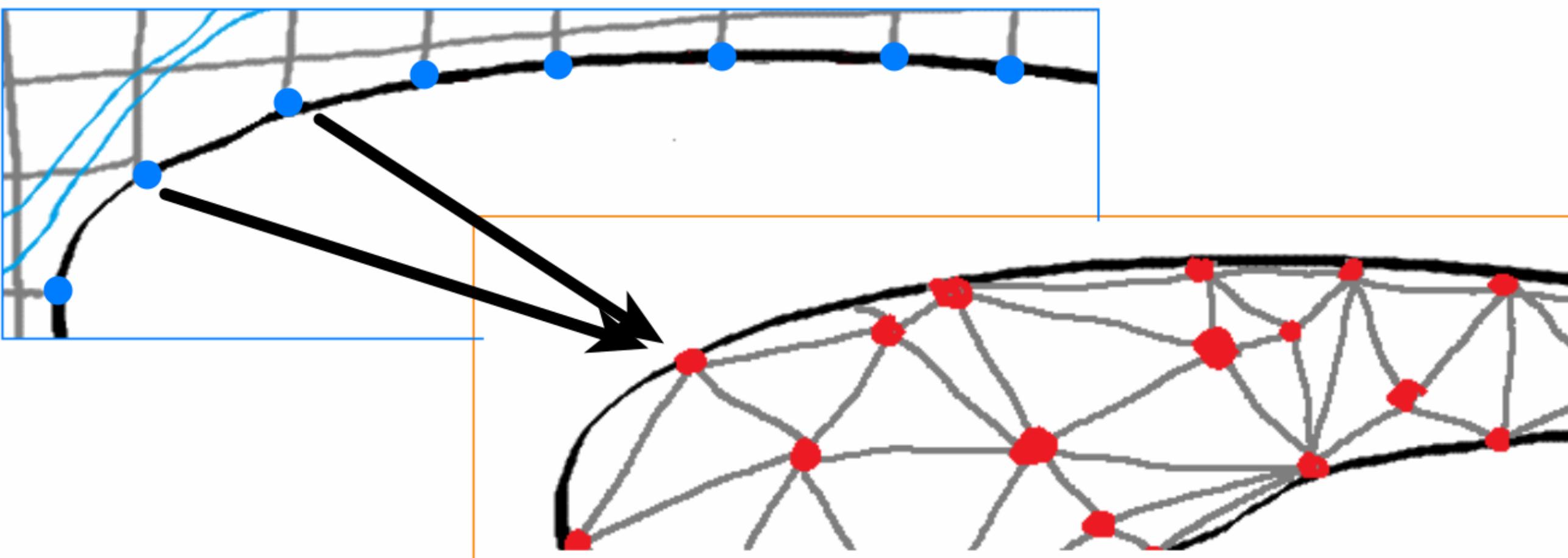
Can we simulate both at the same time?



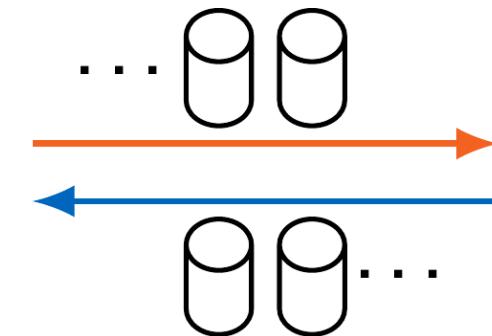
...but reusing the CFD and FEM codes?



From the CFD world to the FEM world



Where preCICE helps

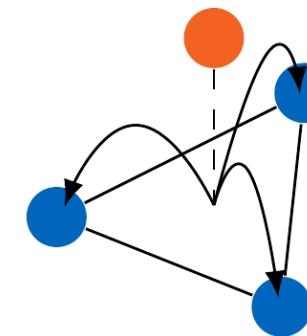


Communication

Options:

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

Fully-parallel, peer-to-peer

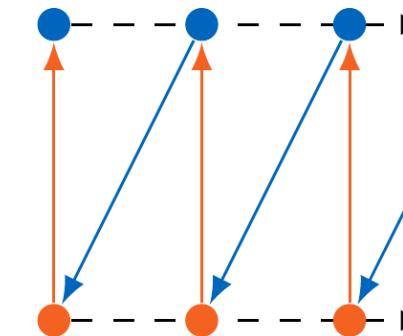


Data mapping

Options:

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

Compute on any side

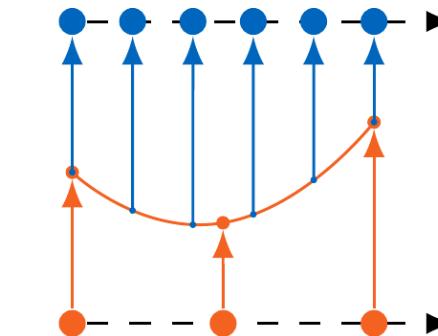


Coupling schemes

Options:

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

Same high-level API
Configurable at runtime



Time interpolation

Options:

- waveform iteration

Experimental since v2.4.0



What people do with preCICE (1)

FSI coupling with OpenFOAM, CalculiX and preCICE for an undulation membrane tidal energy...



Credit: Ulrich Heck, DHCAE Tools GmbH



What people do with preCICE (2)

Particle impacts on flat plate with deformation



Credit: Prasad Adhav, Univ. of Luxembourg

What people do with preCICE (3)

FSI coupling using OpenFOAM and preCICE for a free surface flow



Credit: Utkan Caliskan

In this talk

- The preCICE library: Couple two toy Python solvers
- The preCICE ecosystem: Couple OpenFOAM + CalculiX



Organizational notes

1. You are not expected to try things live.
2. Ask questions live, feel free to interrupt me.
3. Get these slides: go.tum.de/530822
4. Some Live USB sticks available during the workshop
5. Find all software installed in a demo virtual machine:
precice.org/installation-vm.html
6. 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

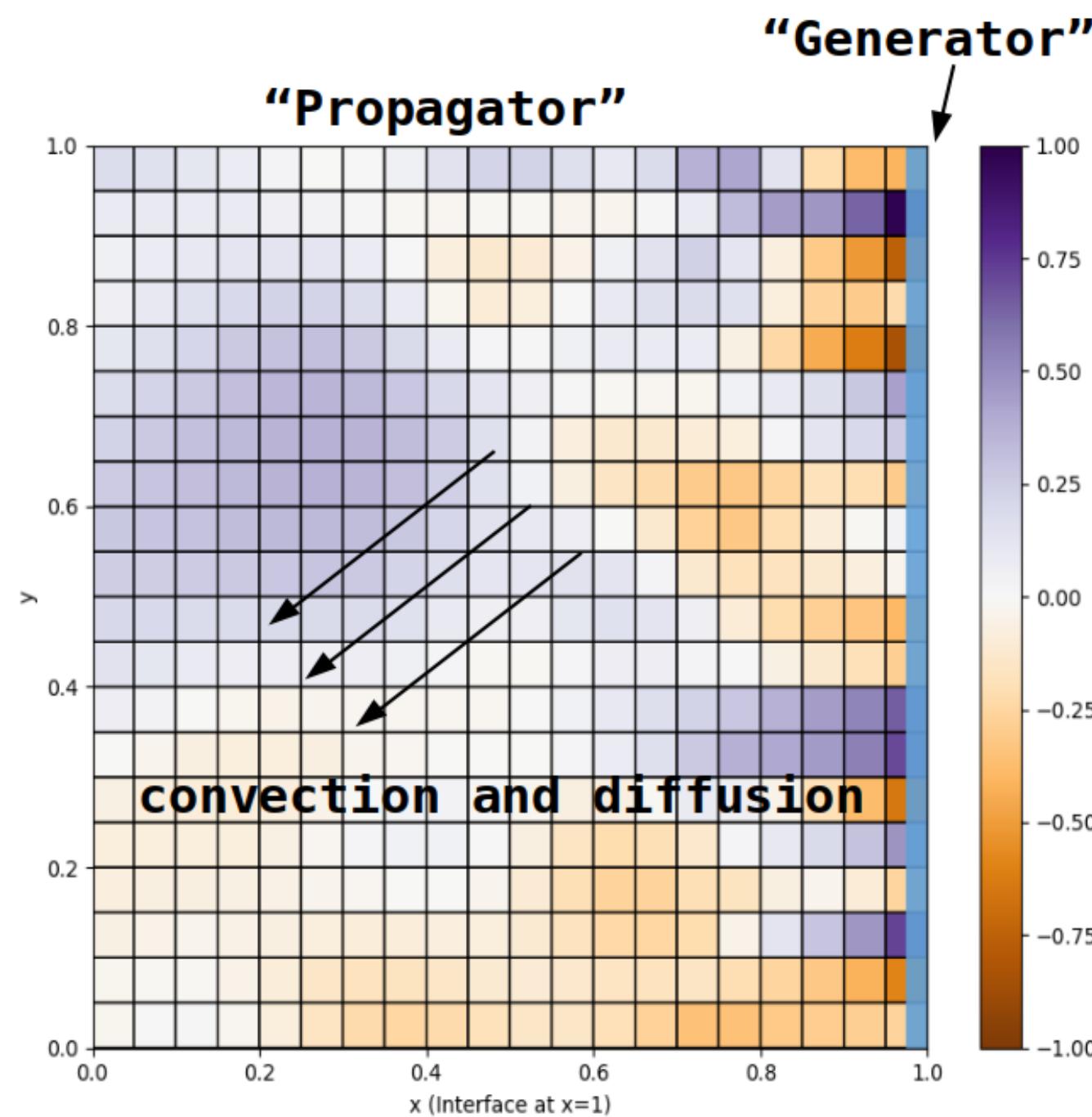
- preCICE v2 (e.g. packages for Ubuntu)
- Python 3
- Python packages numpy, matplotlib
- preCICE Python bindings:

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

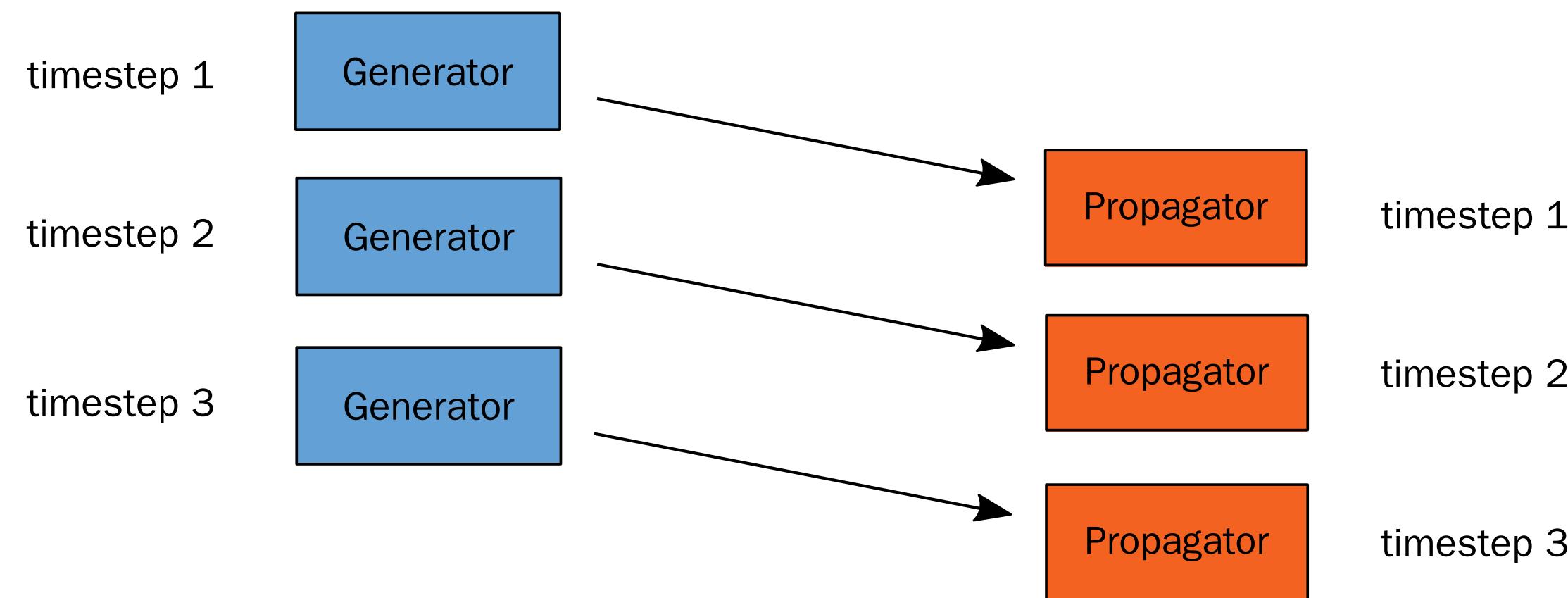


"Generator" and "Propagator"

- `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/skeleton\$ tree

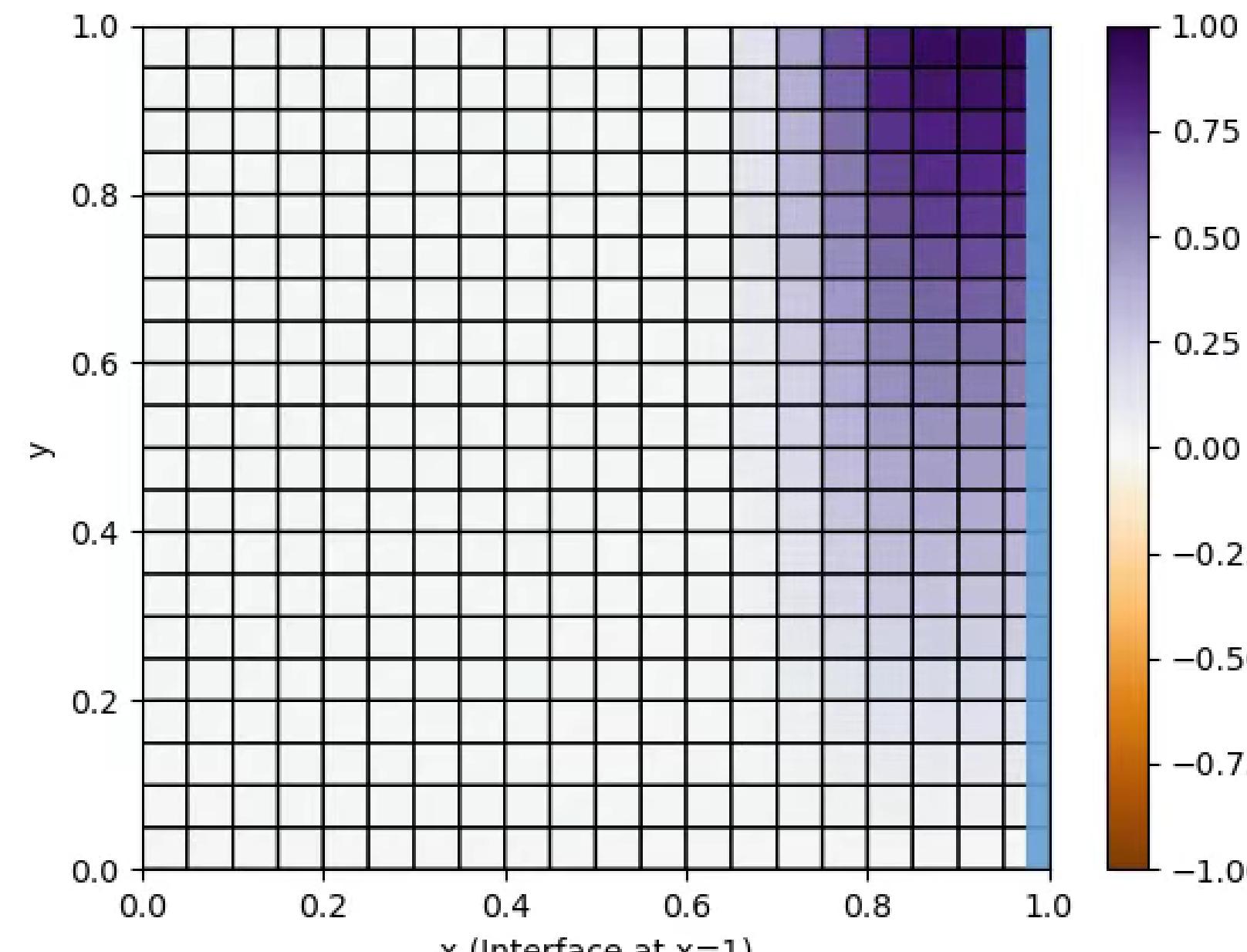
```
├── Allclean
└── generator
    └── generator.py
    ├── precice-config.xml
    └── propagator
        └── propagator.py
    README.txt
```

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 interface =
14     precice.Interface(
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 interface =
14     precice.Interface(
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 interface =
14     precice.Interface(
15         participant_name,
16         config_file_name,
17         solver_process_index,
18         solver_process_size
19     )
20
21 # Get the preCICE mesh id
22 mesh_name = "Generator-Mesh"
```



Define the coupling mesh

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

Initialize and finalize preCICE

```
16             config_file_name,
17             solver_process_index,
18             solver_process_size
19         )
20
21 # Get the preCICE mesh id
22 mesh_name = "Generator-Mesh"
23 mesh_id   = interface.get_mesh_id(mesh_name)
24
25 # Define the coupling mesh
26 vertices  = [[1, y0] for y0 in y[:-1]]
27 vertex_ids = interface.set_mesh_vertices(mesh_id, vertices)
28
29 interface.initialize()
30
31 dt = 0.01
32 t = 0
```



Initialize and finalize preCICE

```
26 vertices    = [[1, y0] for y0 in y[:-1]]
27 vertex_ids = interface.set_mesh_vertices(mesh_id, vertices)
28
29 interface.initialize()
30
31 dt = 0.01
32 t = 0
33
34 while True:
35     print("Generating data")
36     u = 1 - 2 * numpy.random.rand(n)
37
38     t = t + dt
39     if(t > 0.1):
40         break
41
42 interface.finalize()
```



Advance the coupling

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

Advance the coupling

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

Advance the coupling

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

Advance the coupling

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

Advance the coupling

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



Advance the coupling

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



Advance the coupling

```
27 vertex_1as = interface.set_mesh_vertices(mesh_1a, vertices)
28
29 precice_dt = interface.initialize()
30
31 dt = 0.01
32 t = 0
33
34 while interface.is_coupling_ongoing():
35     dt = np.minimum(dt, precice_dt)
36
37     print("Generating data")
38     u = 1 - 2 * numpy.random.rand(n)
39
40     precice_dt = interface.advance(dt)
41
42     t = t + dt
43
44 interface.finalize()
```

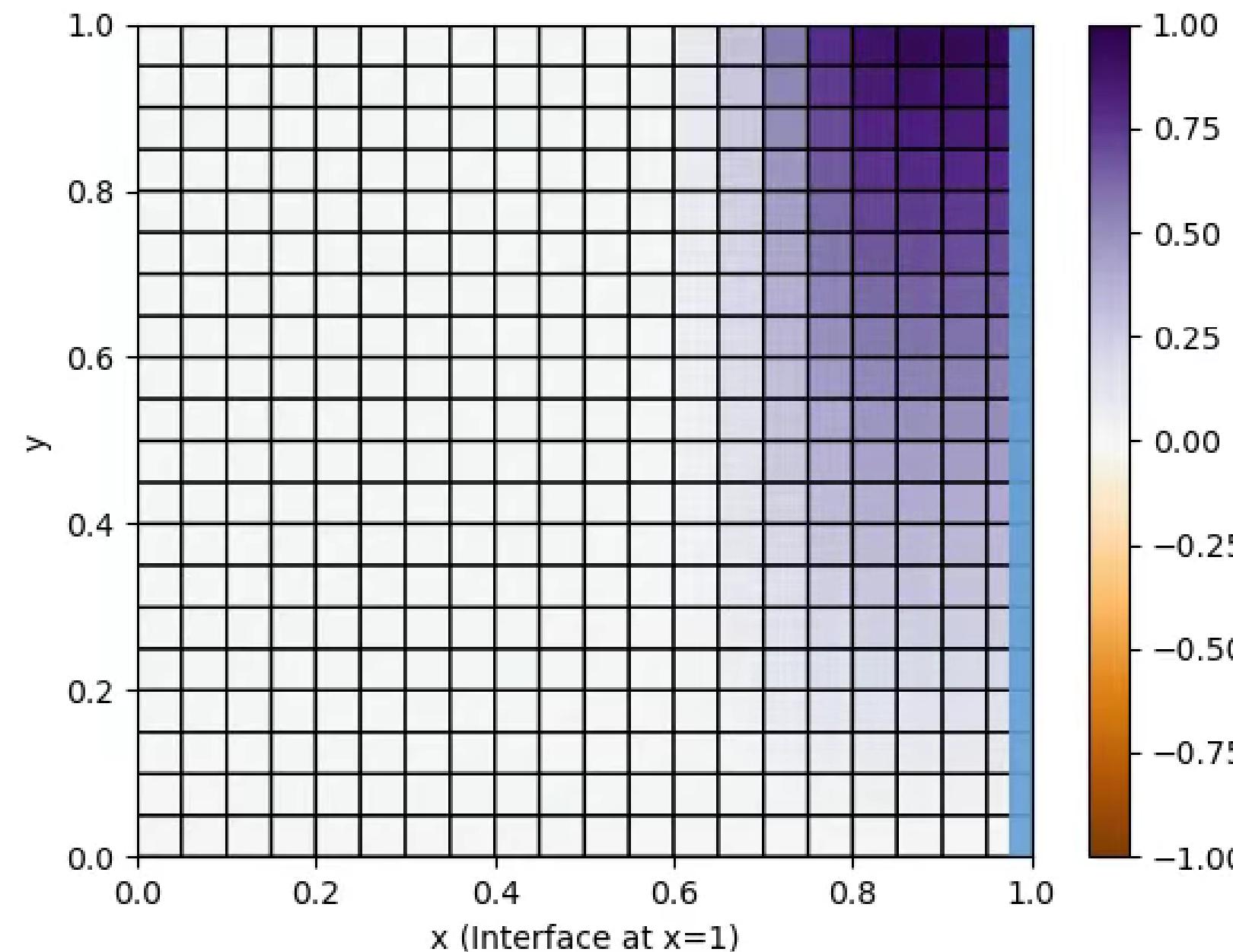


Not there yet, but let's run it
(similar changes in propagator.py - try it at home)

```
vagrant@precicevm:~/Desktop/solution/T4$ cd generator/
vagrant@precicevm:~/Desktop/solution/T4$ ls
generator.py
vagrant@precicevm:~/Desktop/solution/T4$ python3 generator.py
---[preCICE] This is preCICE version 2.2.0
---[preCICE] Revision info: v2.2.0
---[preCICE] Configuration: Debug
---[preCICE] Configuring preCICE with configuration "../precice-config.xml"
---[preCICE] I am participant "Generator"
---[preCICE] Setting up master communication to coupling partner/s
```



Nothing happening?



Did we forget something?



Write & read data

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

Write & read data

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



Write & read data

```
34
35 dt = 0.01
36 t = 0
37
38 while interface.is_coupling_ongoing():
39     dt = np.minimum(dt, precice_dt)
40
41     print("Generating data")
42     u = 1 - 2 * numpy.random.rand(n)
43
44     interface.write_block_scalar_data(data_id, vertex_ids, u)
45     precice_dt = interface.advance(dt)
46     # interface.read_block_scalar_data(data_id, vertex_ids, u)
47
48     t = t + dt
49
50 interface.finalize()
```

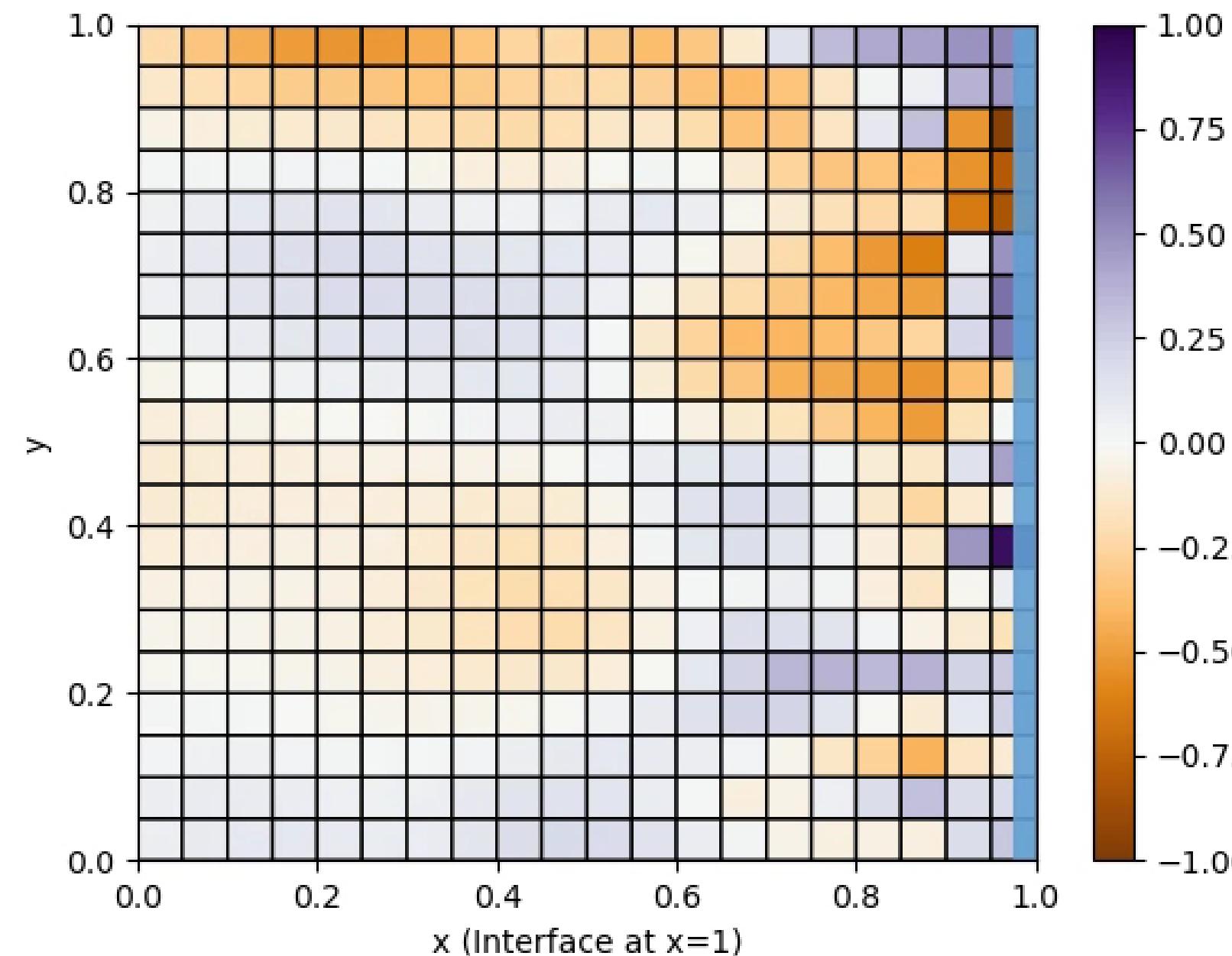


It should work now!

The screenshot shows a terminal window titled "vagrant@precicevm: ~/Desktop/solution/T5". The window has a red header bar. The terminal output is as follows:

```
vagrant@precicevm:~/Desktop/solution/T5$ ls
Allclean generator precice-config.xml propagator
vagrant@precicevm:~/Desktop/solution/T5$ cd generator/
```

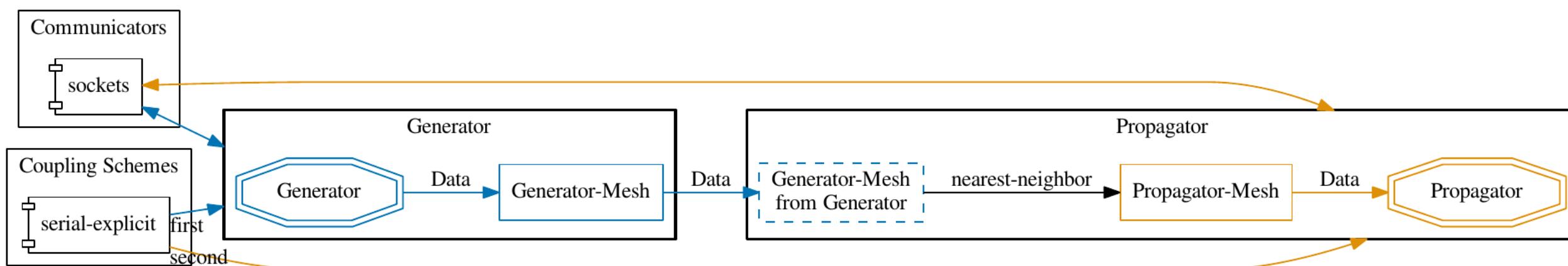
Data is being transferred!



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



preCICE configuration



Visual representation of `precice-config.xml` using the [config visualizer](#).

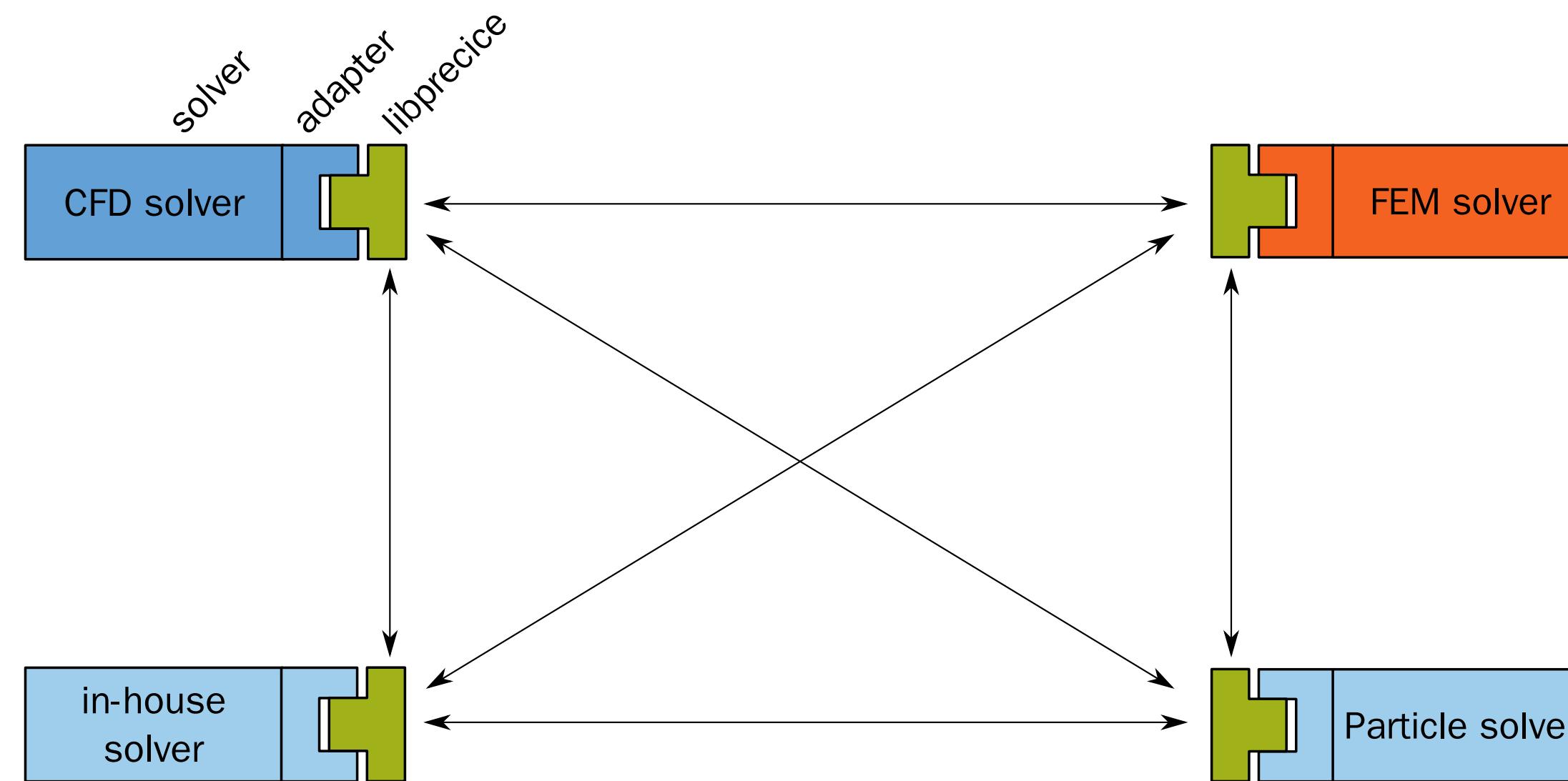
Using the preCICE ecosystem

Example: Coupling OpenFOAM and CalculiX

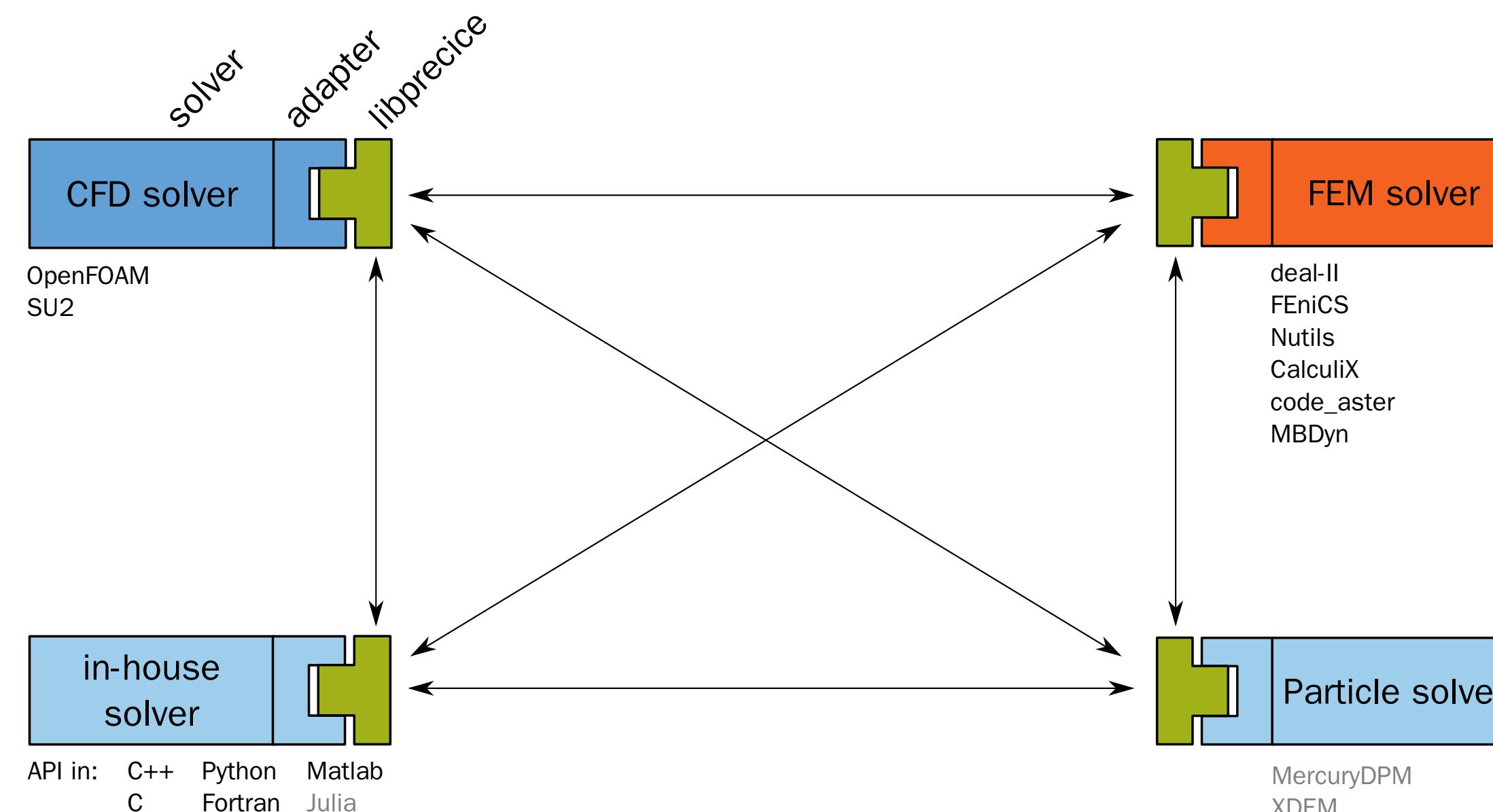
The big picture



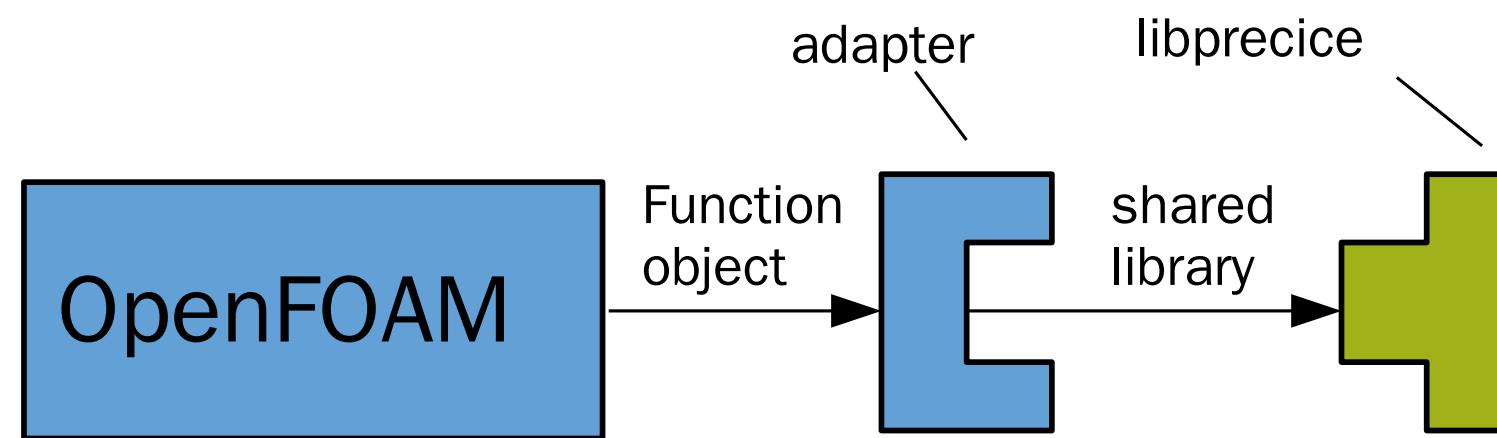
The big picture



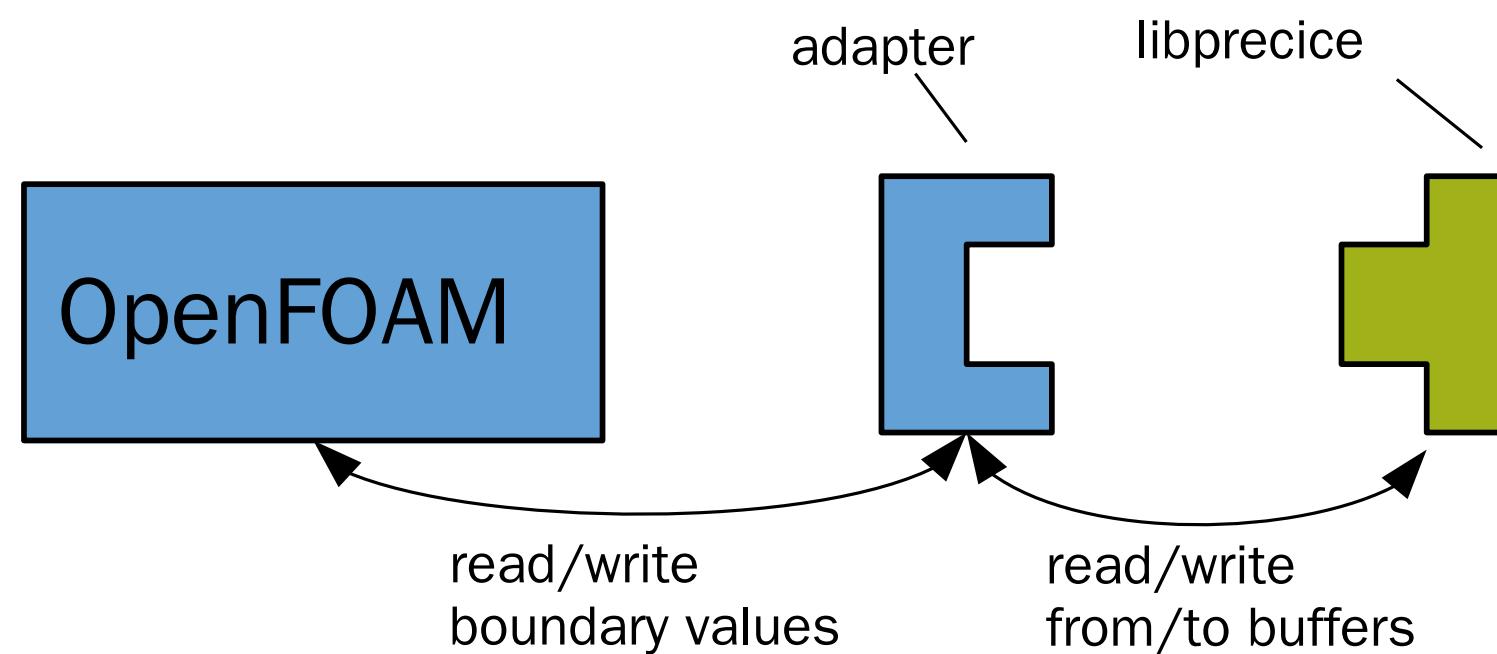
The big picture



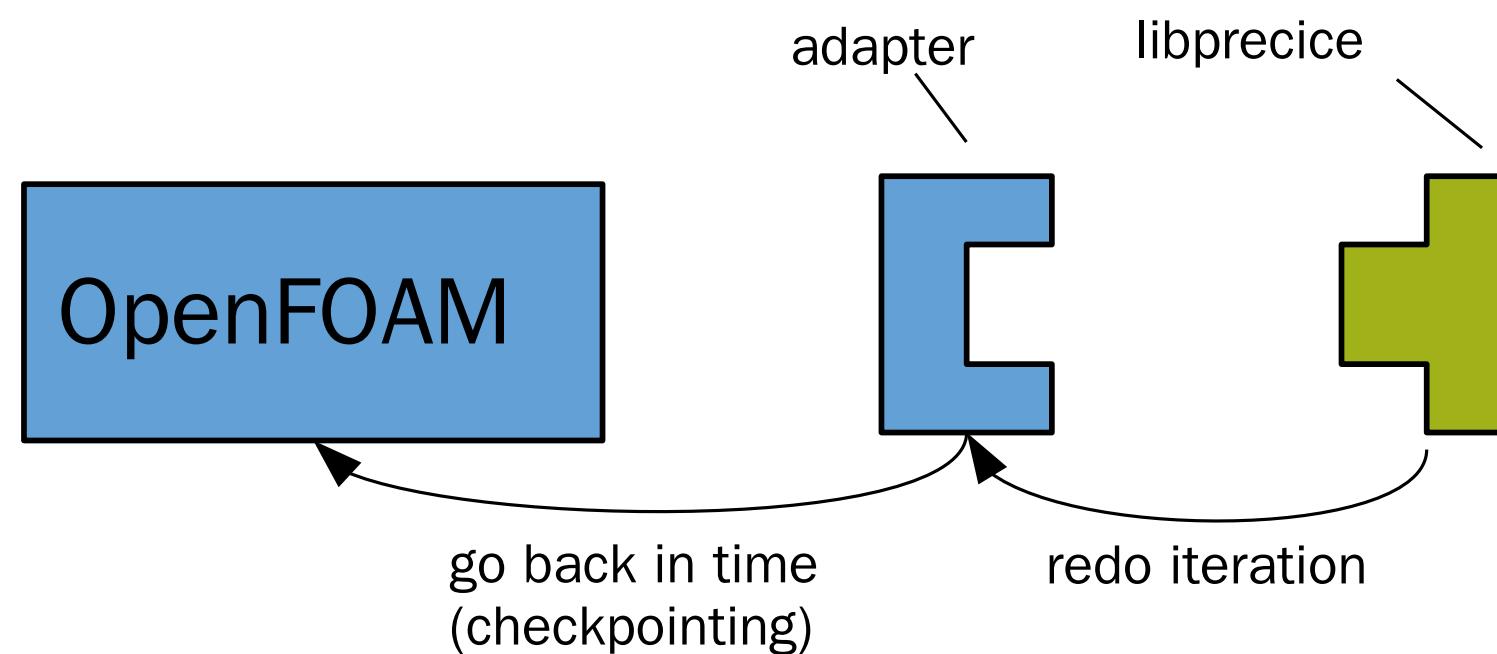
What does the adapter do?



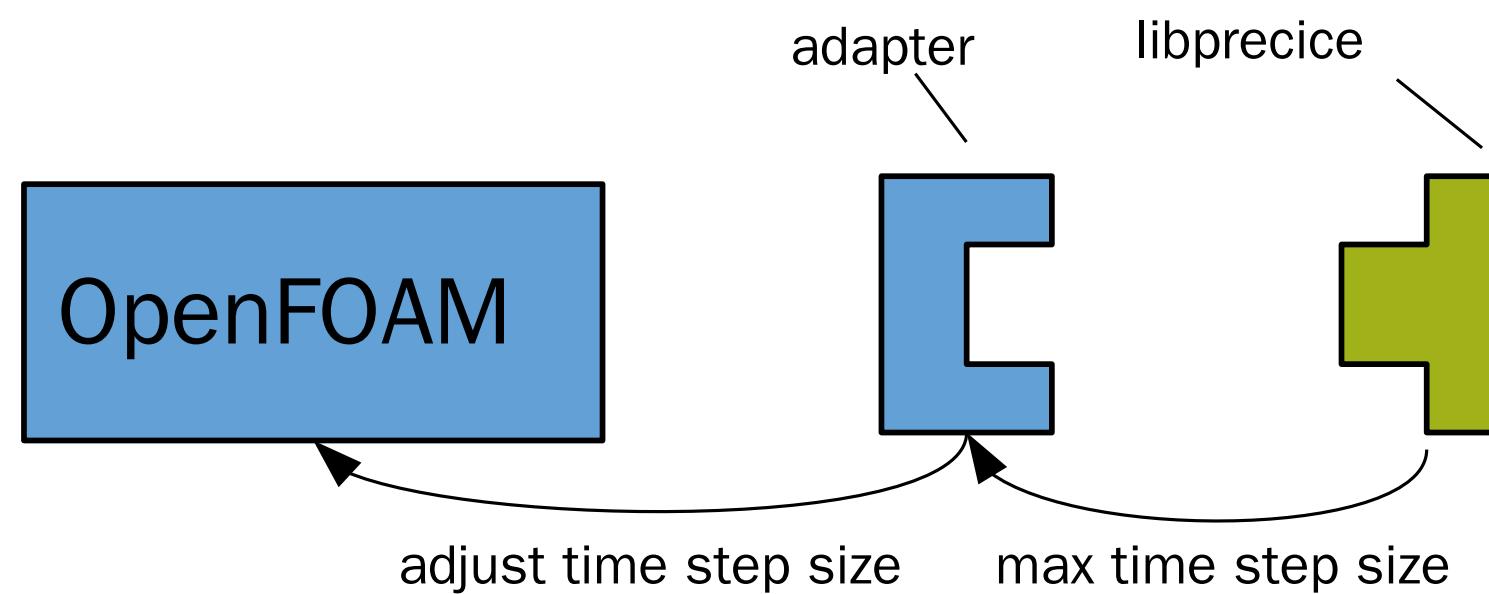
What does the adapter do?



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 ↗](#)

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 ↗](#), a [longer talk on the fundamentals ↗](#), or [click through a tutorial in your browser ↗](#).

Installation

1. Get and install preCICE. For Ubuntu 20.04 (Focal Fossa), this is pretty easy: [download ↗](#) and install our binary package by clicking on it or using the following commands:

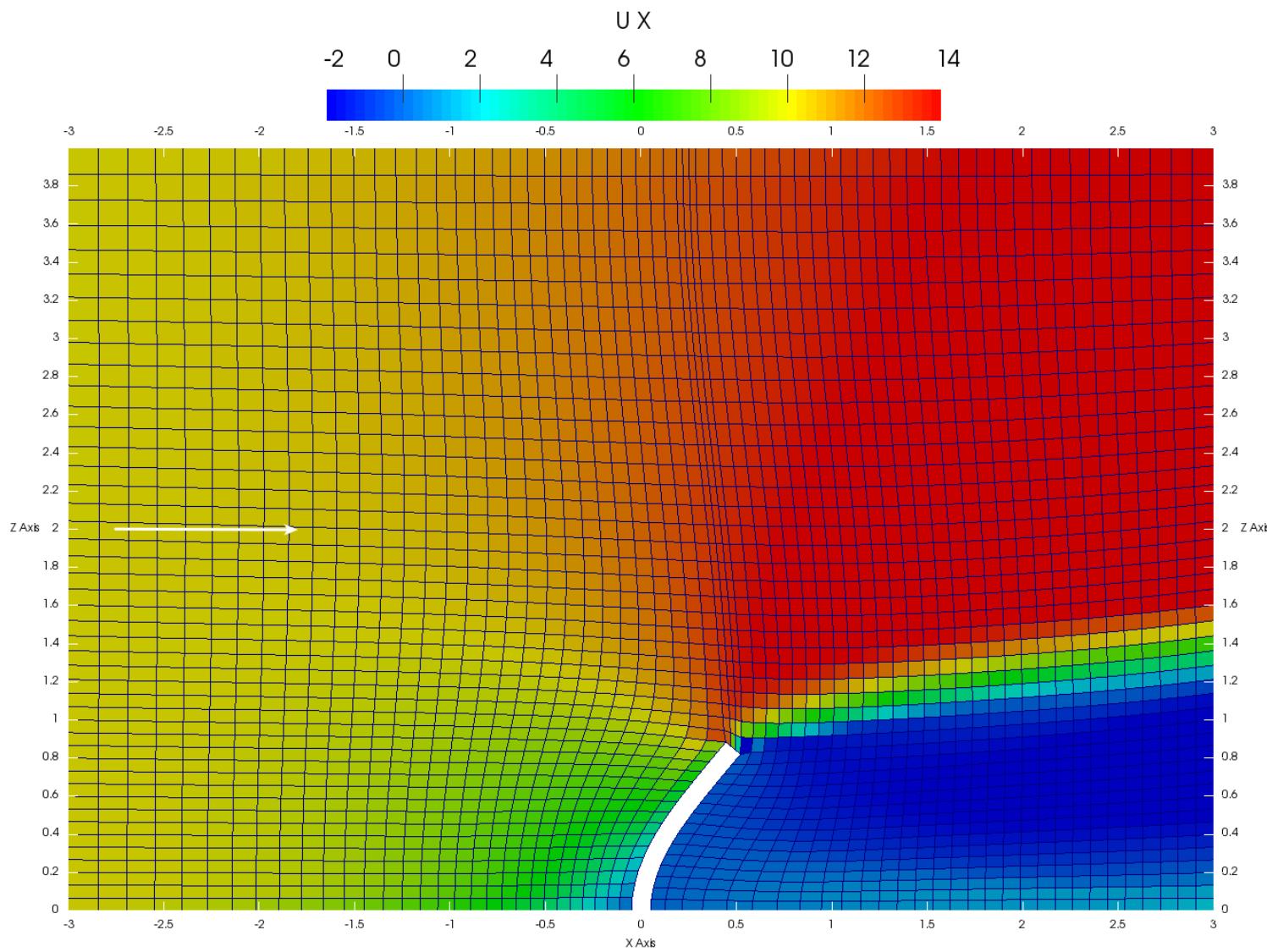
```
wget https://github.com/precice/precice/releases/download/v2.5.0/libprecice2_2.5.0_focal.deb  
sudo apt install ./libprecice2_2.5.0_focal.deb
```

- Are you using something else? Just pick what suits you best on [this overview page ↗](#).
- Facing any problems? [Ask for help ↗](#).

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

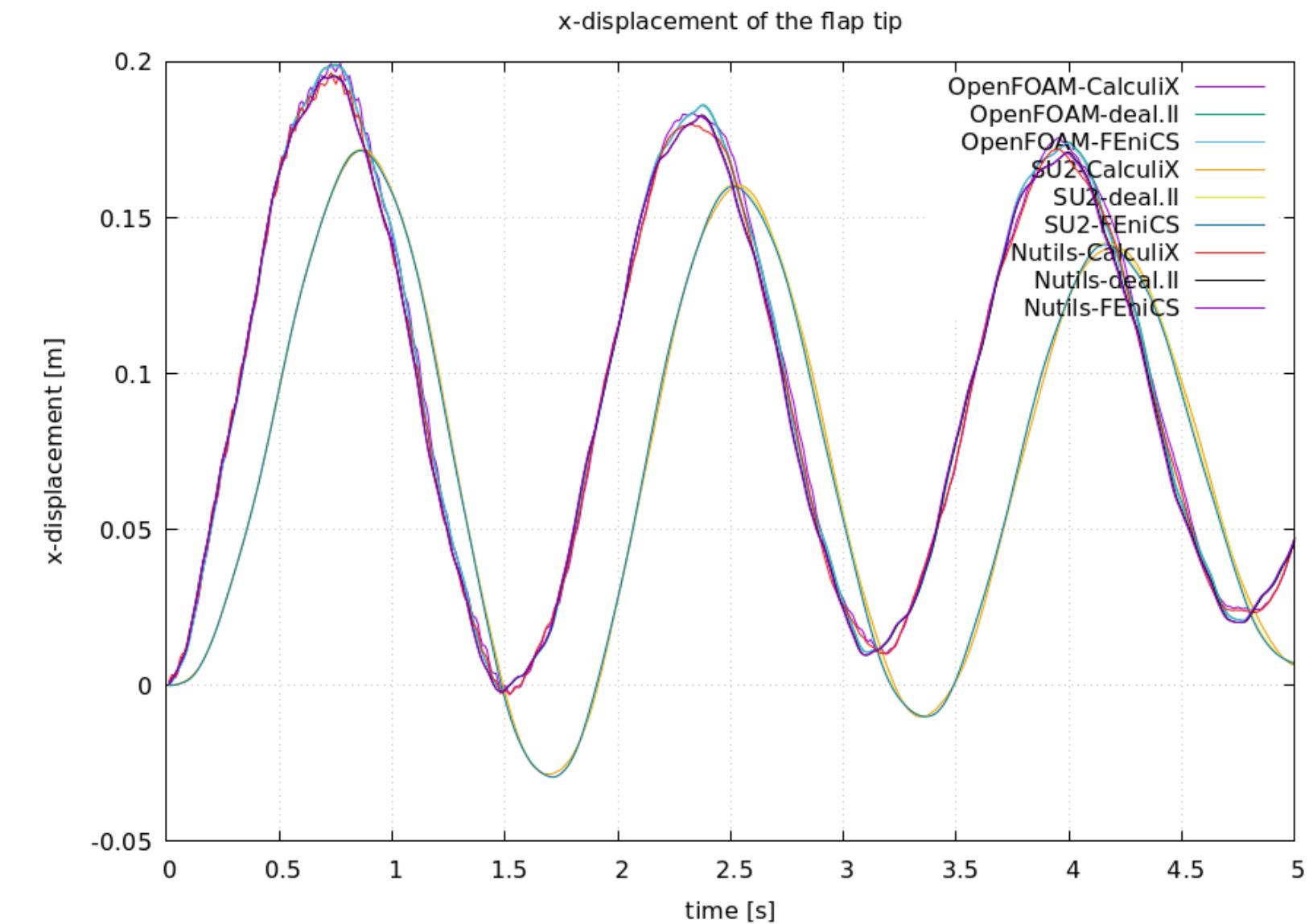
```
# Add the signing key, add the repository, update (check this):  
wget -q -O - https://dl.openfoam.com/add-debian-repo.sh | sudo bash  
# Install OpenFOAM v2206:  
sudo apt install openfoam2206-dev  
# Enable OpenFOAM by default in your system and apply now:  
echo "source /usr/lib/openfoam/openfoam2206/etc/bashrc" >> ~/.bashrc  
source ~/.bashrc
```

Tutorial: Channel with a perpendicular flap



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

Arbitrary solver combinations



Dependencies

- preCICE v2 (e.g. packages for Ubuntu)
- Recent OpenFOAM (e.g., v2306)
- OpenFOAM-preCICE adapter v1
- CalculiX 2.20
- CalculiX-preCICE adapter 2.20.0



File structure

- precice-config.xml
- fluid-openfoam/
 - 0/U ...
 - constant/dynamicMeshDict ...
 - system/
 - controlDict
 - preciceDict
 - ...
- solid-calculix/
 - flap.inp
 - config.yml
 - ...

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-calculix/flap.inp:
 - Lower the solid density from 3000 to 30



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 }
```

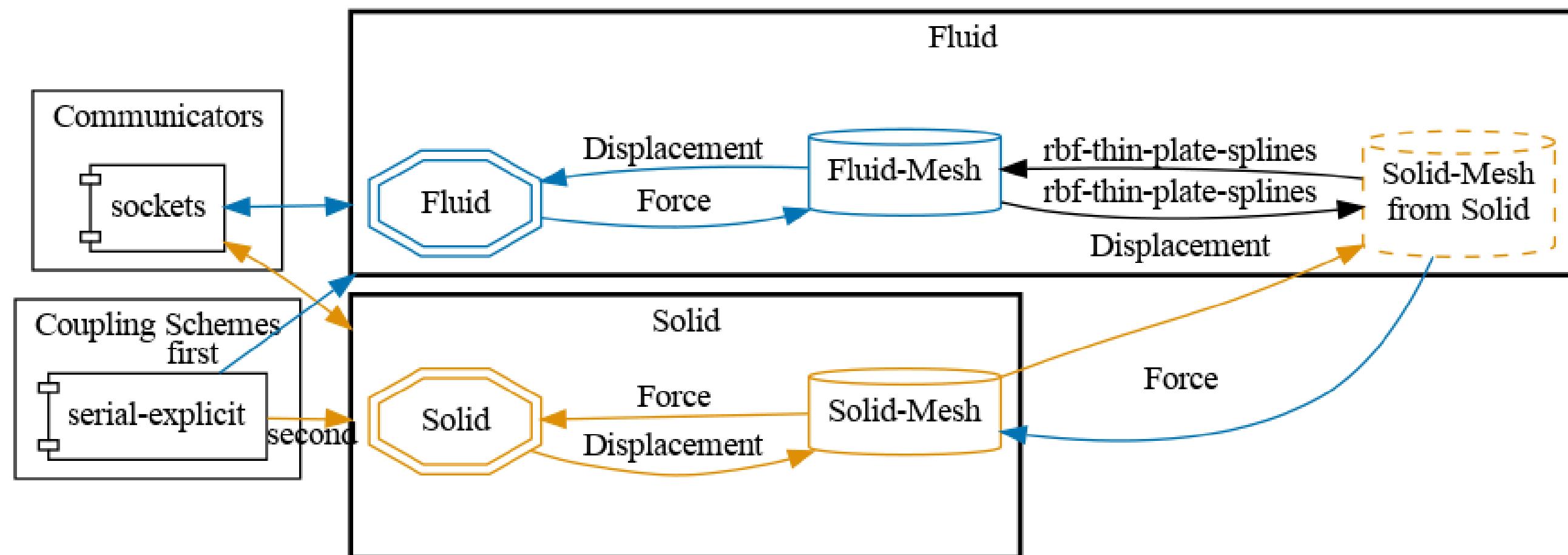


Configure the CalculiX adapter

```
1 // solid-calculix/config.yml
2
3 participants:
4
5   Solid:
6     interfaces:
7       - nodes-mesh: Solid-Mesh
8         patch: surface
9         read-data: [Force]
10        write-data: [Displacement]
11
12   precice-config-file: ../precice-config.xml
```



Configure preCICE



Configure preCICE

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

Notice the serial-explicit coupling scheme



Configure preCICE

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

Notice the serial-explicit coupling scheme



Configure preCICE

```
15 <participant name="Fluid">
16   <export:vtk directory="results" />
17   <use-mesh name="Fluid-Mesh" provide="yes" />
18   <use-mesh name="Solid-Mesh" from="Solid" />
19   <write-data name="Force" mesh="Fluid-Mesh" />
20   <read-data name="Displacement" mesh="Fluid-Mesh" />
21   <mapping:rbf-thin-plate-splines
22     direction="write"
23     from="Fluid-Mesh"
24     to="Solid-Mesh"
25     constraint="conservative" />
26   <mapping:rbf-thin-plate-splines
27     direction="read"
28     from="Solid-Mesh"
29     to="Fluid-Mesh"
30     constraint="consistent" />
31 </participant>
```

Notice the serial-explicit coupling scheme



Configure preCICE

```
15 <participant name="Fluid">
16   <export:vtk directory="results" />
17   <use-mesh name="Fluid-Mesh" provide="yes" />
18   <use-mesh name="Solid-Mesh" from="Solid" />
19   <write-data name="Force" mesh="Fluid-Mesh" />
20   <read-data name="Displacement" mesh="Fluid-Mesh" />
21   <mapping:rbf-thin-plate-splines
22     direction="write"
23     from="Fluid-Mesh"
24     to="Solid-Mesh"
25     constraint="conservative" />
26   <mapping:rbf-thin-plate-splines
27     direction="read"
28     from="Solid-Mesh"
29     to="Fluid-Mesh"
30     constraint="consistent" />
31 </participant>
```

Notice the serial-explicit coupling scheme



Configure preCICE

```
15 <participant name="Fluid">
16   <export:vtk directory="results" />
17   <use-mesh name="Fluid-Mesh" provide="yes" />
18   <use-mesh name="Solid-Mesh" from="Solid" />
19   <write-data name="Force" mesh="Fluid-Mesh" />
20   <read-data name="Displacement" mesh="Fluid-Mesh" />
21   <mapping:rbf-thin-plate-splines
22     direction="write"
23     from="Fluid-Mesh"
24     to="Solid-Mesh"
25     constraint="conservative" />
26   <mapping:rbf-thin-plate-splines
27     direction="read"
28     from="Solid-Mesh"
29     to="Fluid-Mesh"
30     constraint="consistent" />
31 </participant>
```

Notice the serial-explicit coupling scheme



Configure preCICE

```
32
33 <participant name="Solid">
34   <use-mesh name="Solid-Mesh" provide="yes" />
35   <write-data name="Displacement" mesh="Solid-Mesh" />
36   <read-data name="Force" mesh="Solid-Mesh" />
37   <watch-point mesh="Solid-Mesh" name="Flap-Tip" coordinate="0.0;1"
38 </participant>
39
40 <m2n:sockets from="Fluid" to="Solid" exchange-directory=".." />
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="
48 </coupling-scheme:serial-explicit>
```

Notice the serial-explicit coupling scheme



Configure preCICE

```
34    <use-mesh name="Solid-Mesh" provide="yes" />
35    <write-data name="Displacement" mesh="Solid-Mesh" />
36    <read-data name="Force" mesh="Solid-Mesh" />
37    <watch-point mesh="Solid-Mesh" name="Flap-Tip" coordinate="0.0;1"
38    </participant>
39
40    <m2n:sockets from="Fluid" to="Solid" exchange-directory=".." />
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="Fluid"
48    </coupling-scheme:serial-explicit>
49
50 </solver-interface>
```

Notice the serial-explicit coupling scheme

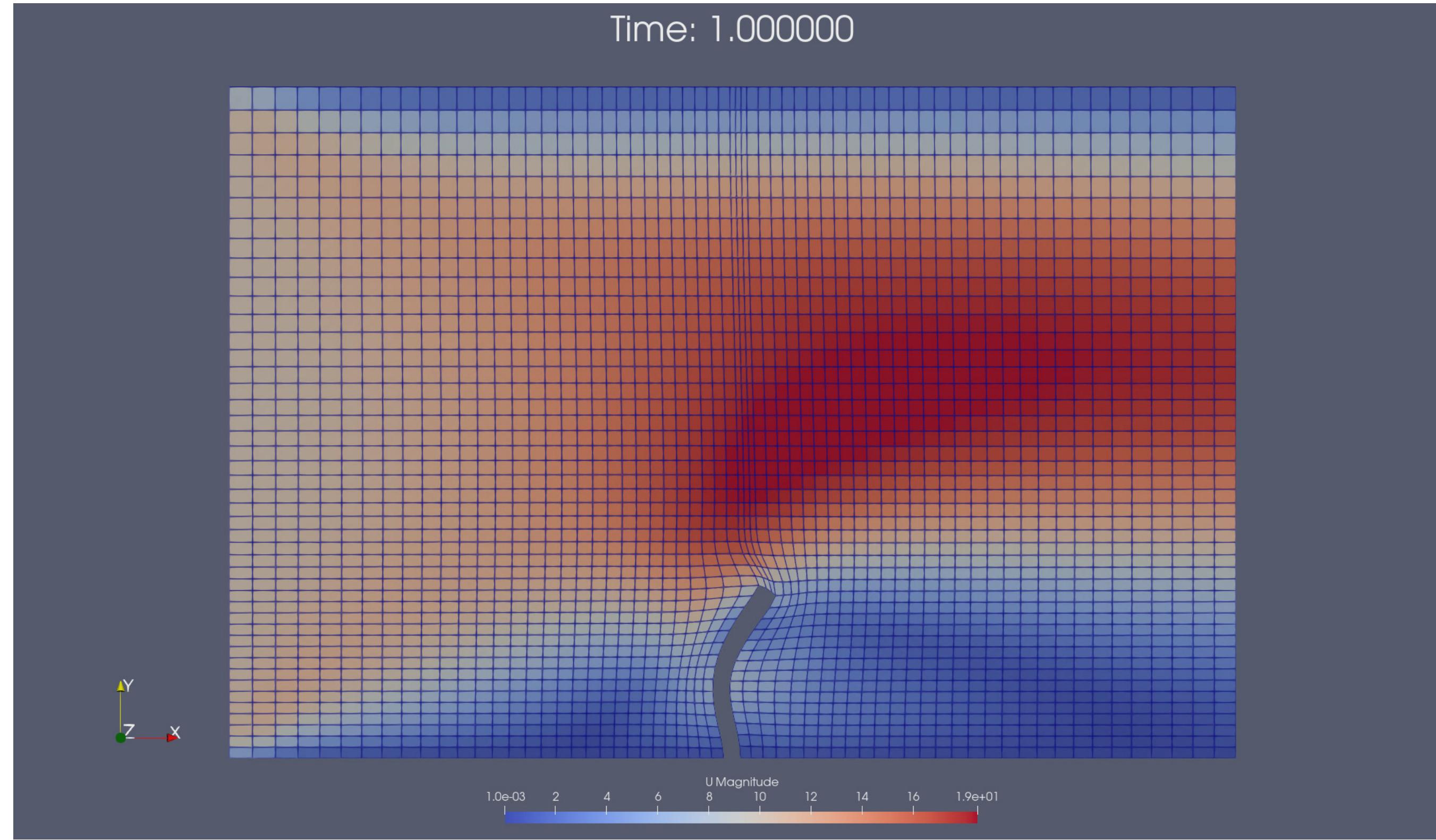


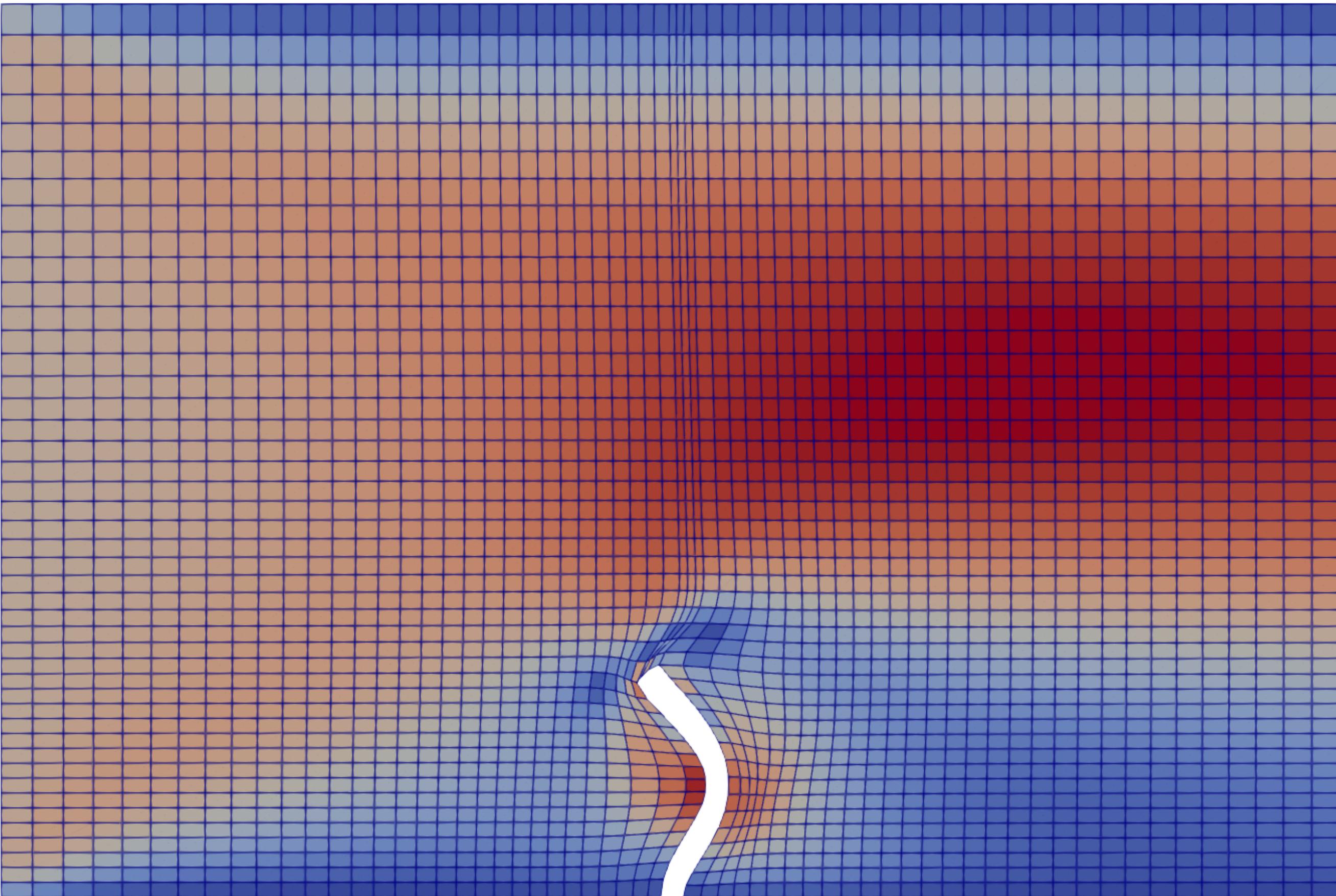
```
vagrant@precicevm:~/tutor... fsi-course - File Manager
vagrant@precicevm:~/tutorials/fsi-course/fluid-openfoam
vagrant@precicevm:~/tutorials/fsi-course 160x3
vagrant@precicevm:~/tutorials/fsi-course$ ls
clean-tutorial.sh fluid-openfoam plot-displacement.sh precice-config.xml solid-calculix
vagrant@precicevm:~/tutorials/fsi-course$ 

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

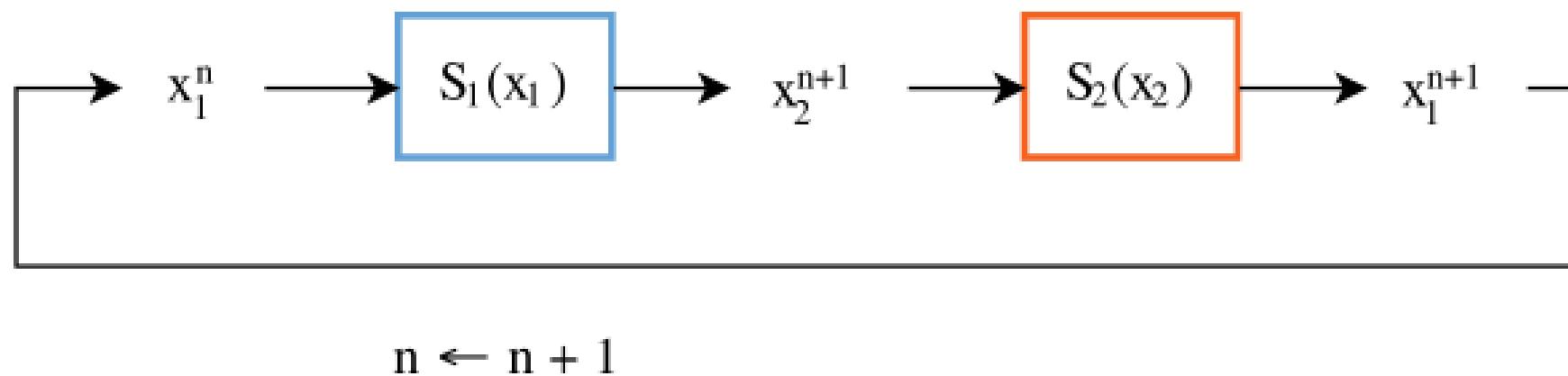
3 directories, 15 files
vagrant@precicevm:~/tutorials/fsi-course/fluid-openfoam$ ./run.sh

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

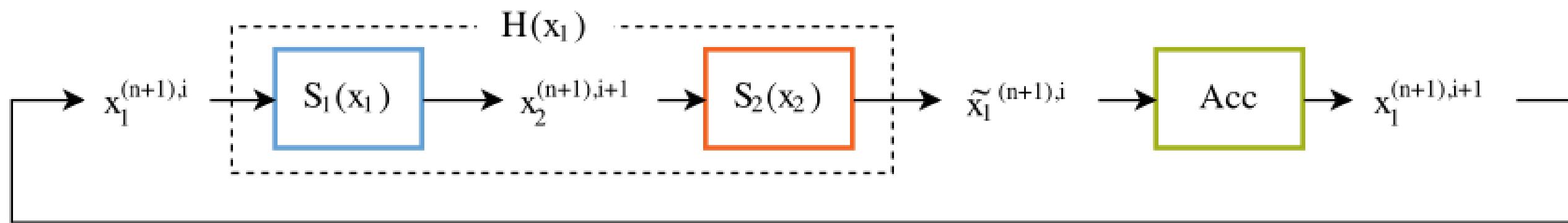




What we did so far: serial-explicit scheme



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  <acceleration:constant>
11    <relaxation value="0.5" />
12  </acceleration:constant>
13 </coupling-scheme:serial-implicit>
```

Simplest case: constant under-relaxation



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 case: constant under-relaxation



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 case: constant under-relaxation



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 case: constant under-relaxation



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 case: constant under-relaxation

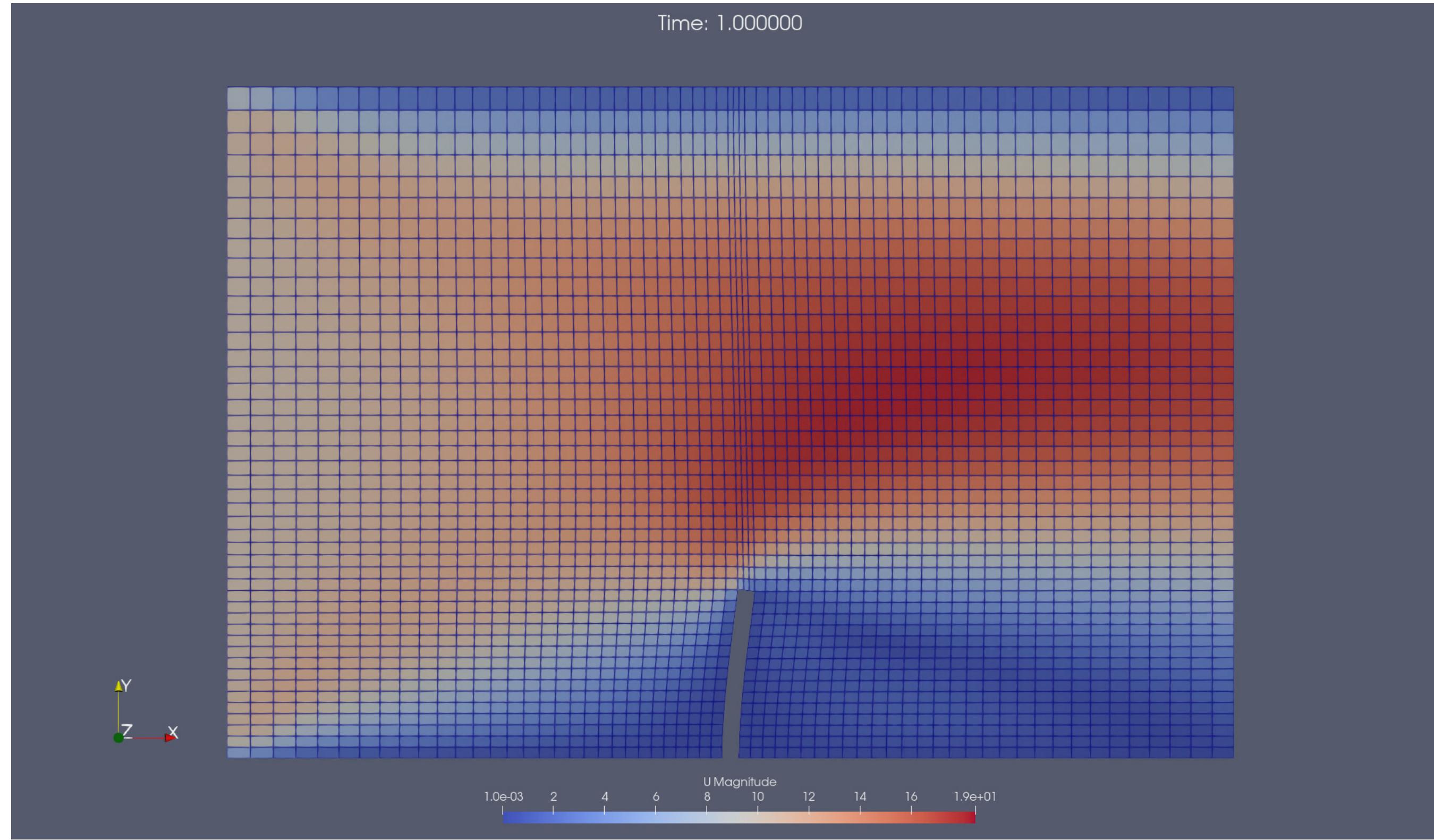


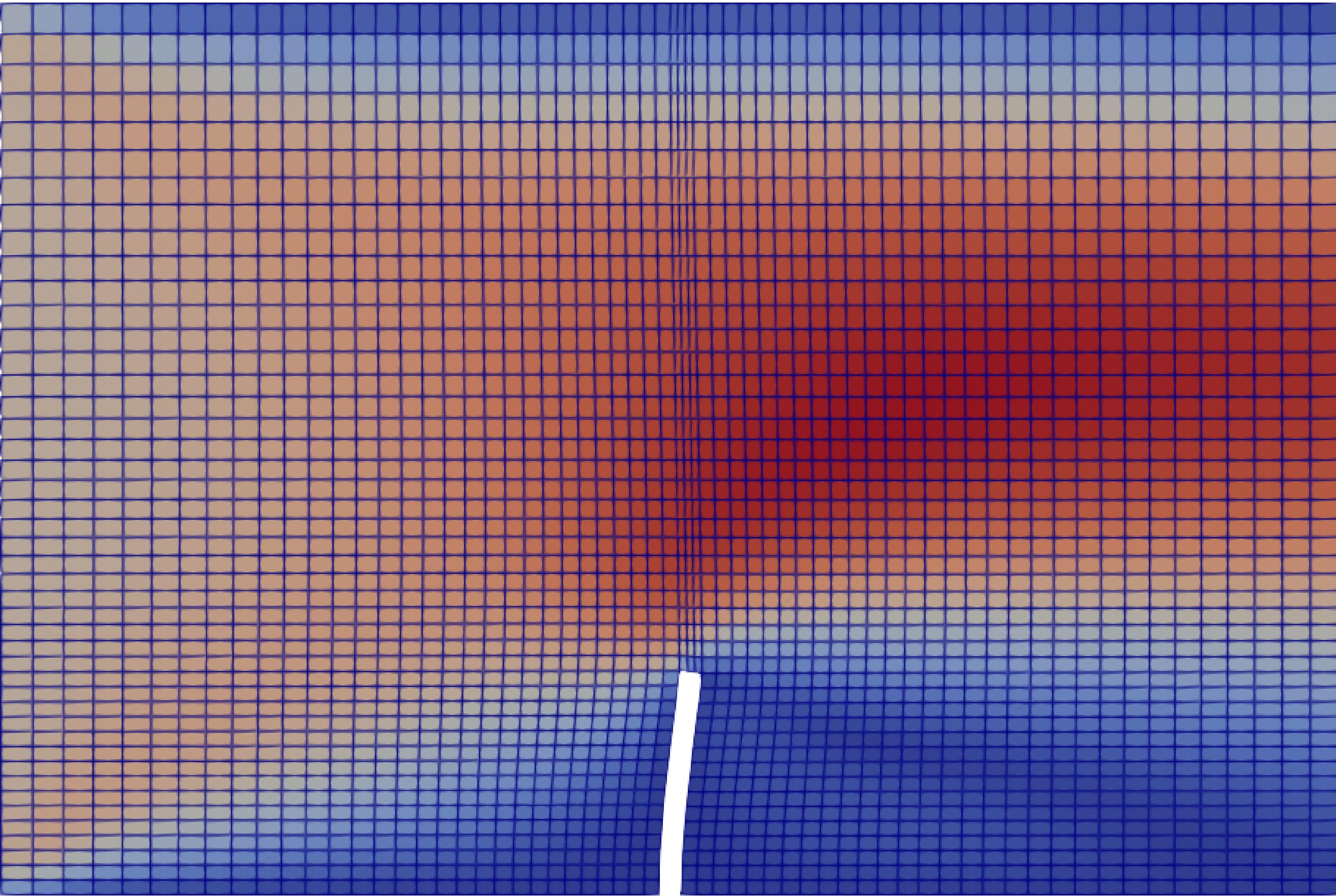
Configure the coupling scheme

```
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 case: constant under-relaxation







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>
```

Average iterations drop from 3.85 (constant) to 3.49 (Aitken)

Note: This simulation is reaching a steady-state, thus differences diffuse

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>
```

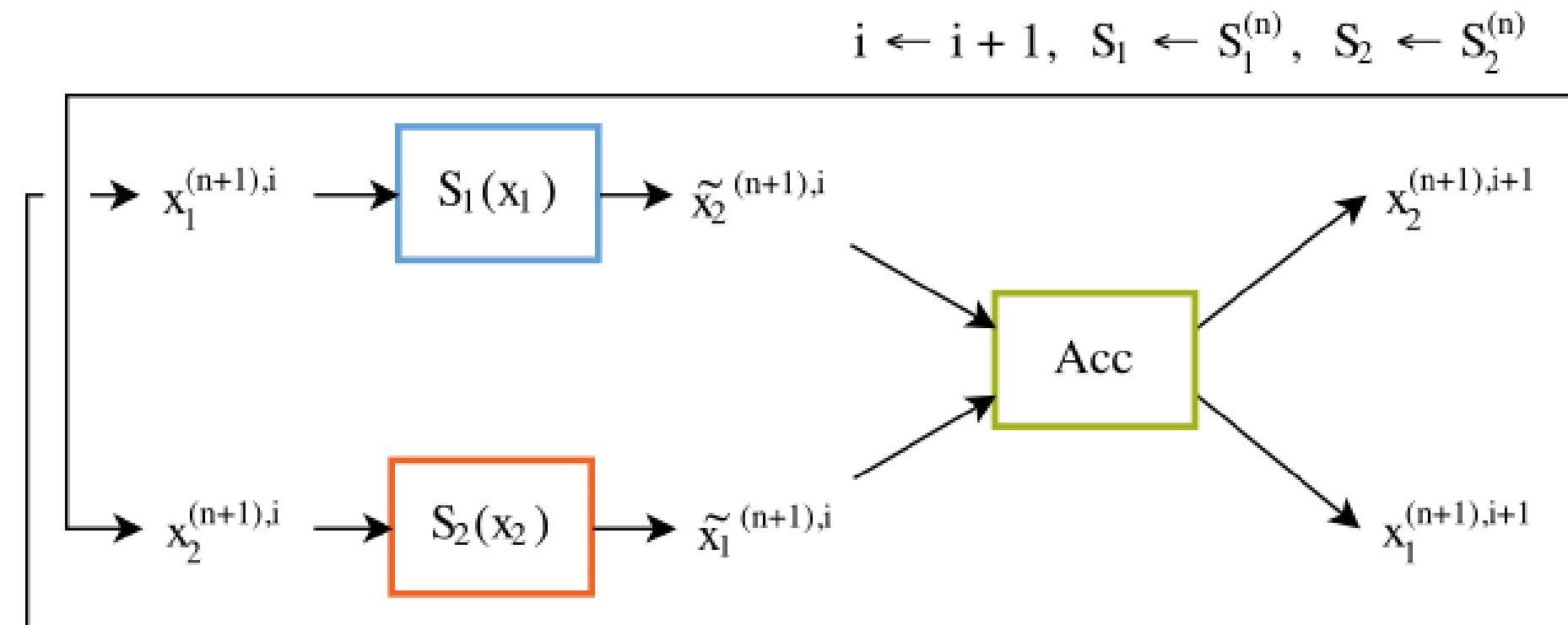
Average iterations drop from 3.49 (Aitken) to 2.91 (IQN)

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>
```

Average iterations drop from 3.49 (Aitken) to 2.91 (IQN)

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" />
```

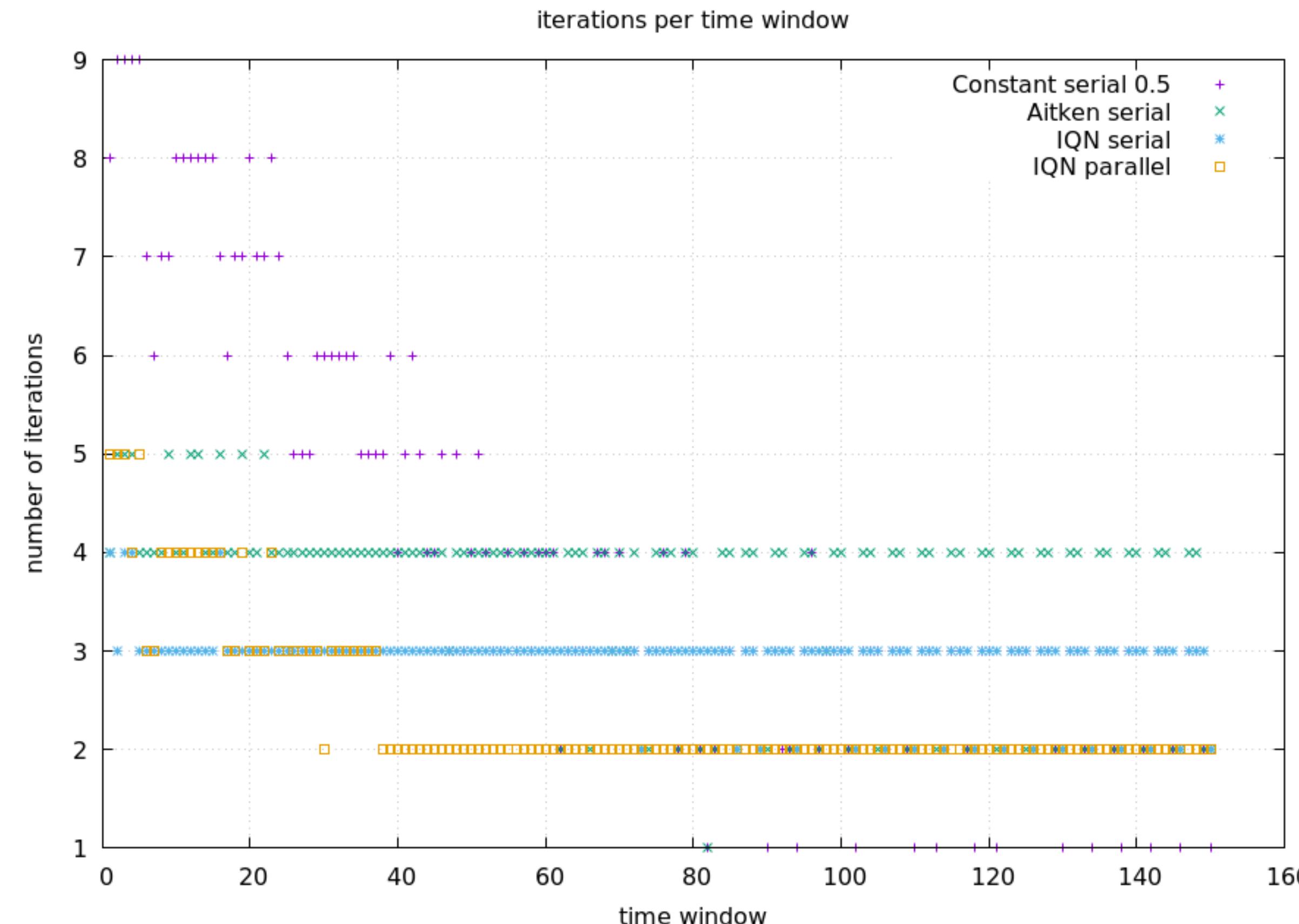
Average iterations drop from 2.91 (serial-implicit IQN) to 2.37 (parallel-implicit IQN)



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>
```

Average iterations drop from 2.91 (serial-implicit IQN) to 2.37 (parallel-implicit IQN)



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



Resources

Start here: precice.org



Welcome to preCICE

The coupling library for partitioned multi-physics simulations.

[Star on Github](#) ★ 573

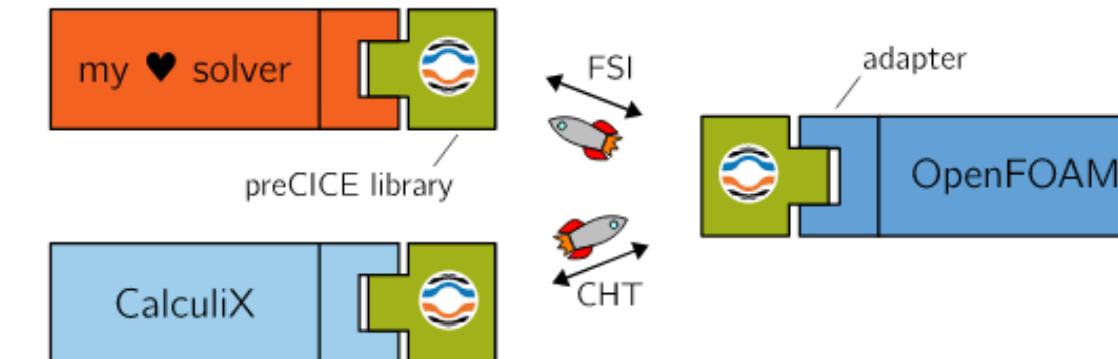
[Latest \(Aug 10, 2022\)](#)

[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

**Docs v2.5.0****Fundamentals**[Overview](#)[Terminology](#)[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 ↗](#)

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

 Home GitHub Twitter YouTube   

all categories ► all tags ► Categories Latest Top Bookmarks  + New Topic

Category	Topics	Latest
News News, announcements, "blog"-like posts	26	 MMCP'23 Workshop at HPC Asia 2023 News 1 13h
Is preCICE for me? General questions regarding preCICE as a coupling solution.	41	 Turek-Hron FSI 2 Benchmark Community projects 2 8d calculix fsi inactive openfoam
Installing preCICE Any issues with getting the preCICE library installed	58	 FSI with interFoam and CalculiX: Convergence 11 9d Using preCICE fsi inactive multiphase openfoam
Using preCICE Using the preCICE API, configuring a new simulation	195	 Turek-Hron FSI with OpenFOAM and CalculiX 16 9d Official adapters and tutorials calculix fsi openfoam
Community projects Share your simulation cases for everybody to admire and try.	3	 Fluid Fluid Coupling 9 10d Using preCICE
Jobs & theses market Jobs and thesis projects related to preCICE.	0	 Multiple Coupling Variable for OpenFOAM 5 11d

Active since October 2019, 380+ users, 5k posts in 600+ topics



Learn: YouTube

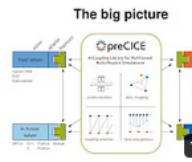
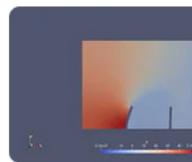
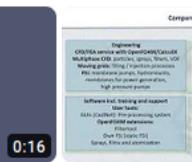
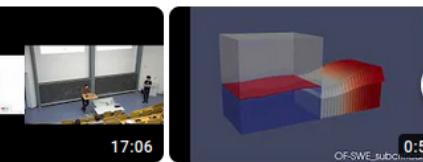
 **preCICE Coupling**
@preCICECoupling 455 subscribers 42 videos

A coupling library for partitioned multi-physics simulations, including, but n... >

[Subscribe](#)

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

Popular videos ► [Play all](#)

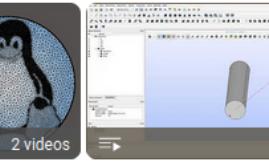
 Multiphysics Modeling with the preCICE Coupling Library 2.3K views • 2 years ago	 Fundamentals of preCICE (Benjamin Uekermann,...) 1.4K views • 2 years ago	 Multicoupling with preCICE: two flaps in a channel... 1K views • 2 years ago	 Transfer of FSI coupling with preCICE, OpenFOAM and... 868 views •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

Talks about preCICE ► [Play all](#)

Conference talks about the preCICE coupling library

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

Community

 FSI coupling with OpenFOAM, CalculiX and preCICE Simulations using preCICE Playlist · preCICE Coupling View full playlist	 Open-Source FINITE ELEMENT Tools Users talking about preCICE Playlist · preCICE Coupling View full playlist	 FSI for vascular flows using OpenFOAM, preCICE, and... Playlist · Torsten Schenkel View full playlist
---	---	--

Active since 2020, 450+ subscribers, 40+ videos

Get news: Twitter

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



 ... ⌂ Following

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

📍 Germany 🌐 precice.org 📅 Joined April 2018

0 Following 462 Followers

Active since 2018, 400+ followers
also on @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

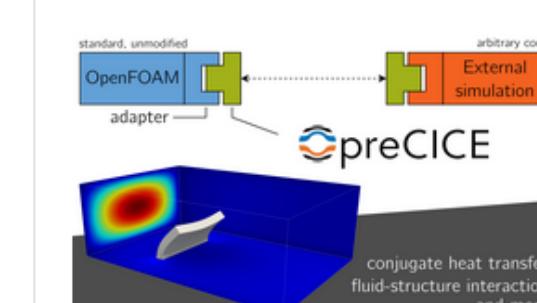
OpenFOAM-preCICE: Coupling OpenFOAM with External Solvers for Multi-Physics Simulations

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 a 'standard, unmodified' **OpenFOAM** block connected via an 'adapter' to an 'arbitrary code' block labeled **External simulation**. A 3D visualization below shows a flow field over a wing-like body, with text indicating 'conjugate heat transfer', 'fluid-structure interaction', and 'and more'. The **preCICE** logo is positioned between the two main components.

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

Sponsors

GOMPUTE
A GRIDCORE COMPANY

upstreamCFD



Read more: New v2 reference paper

Open Research Europe

Search

SUBMIT YOUR RESEARCH

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

104 Views | 52 Downloads | 7 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

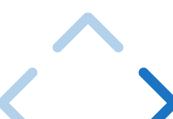
Open Peer Review

Approval Status ✓ ✓ ⓘ

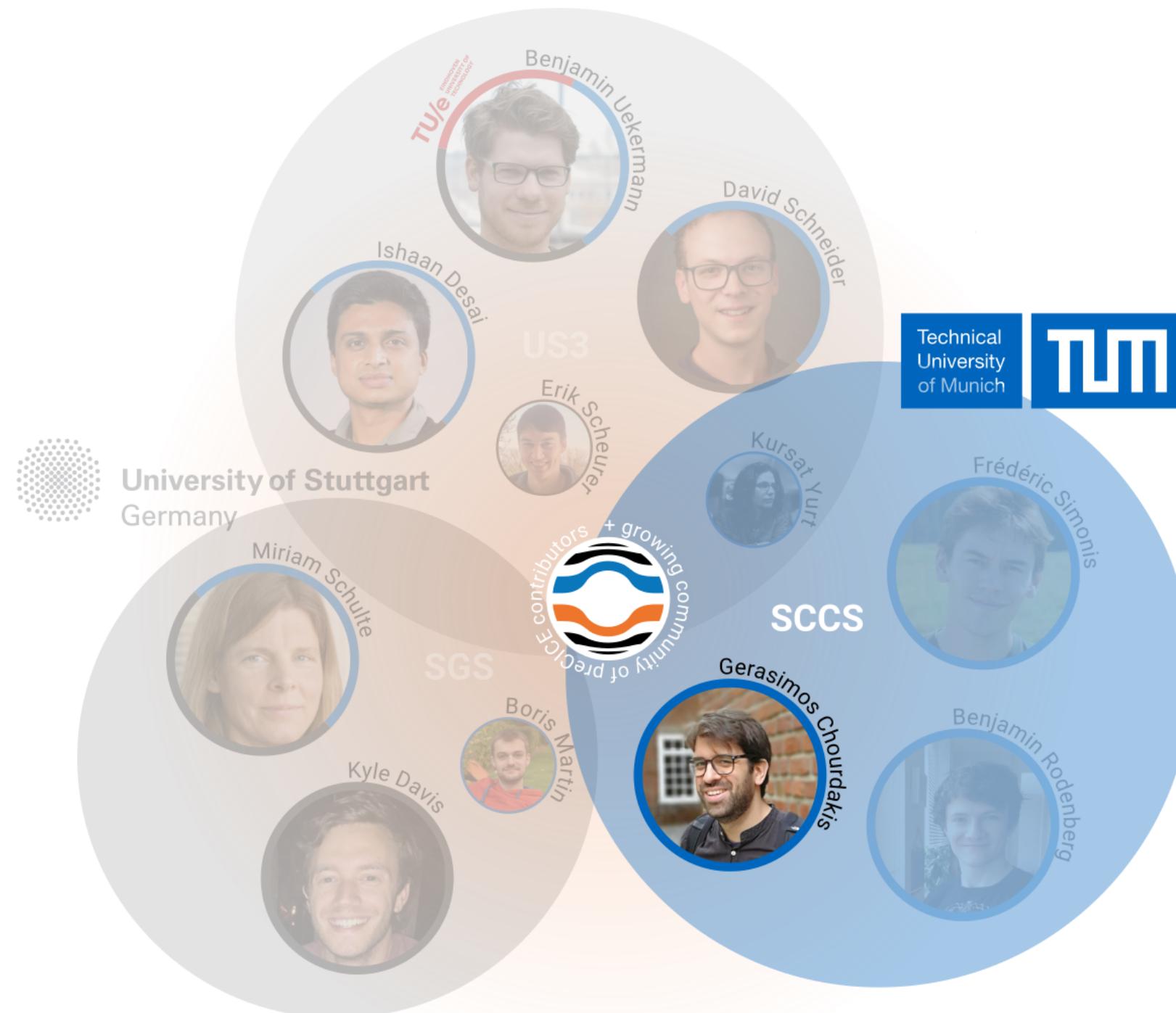
	1	2
Version 2 (Revision) 30 Sep 22		
Version 1 29 Apr 22	view	view

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

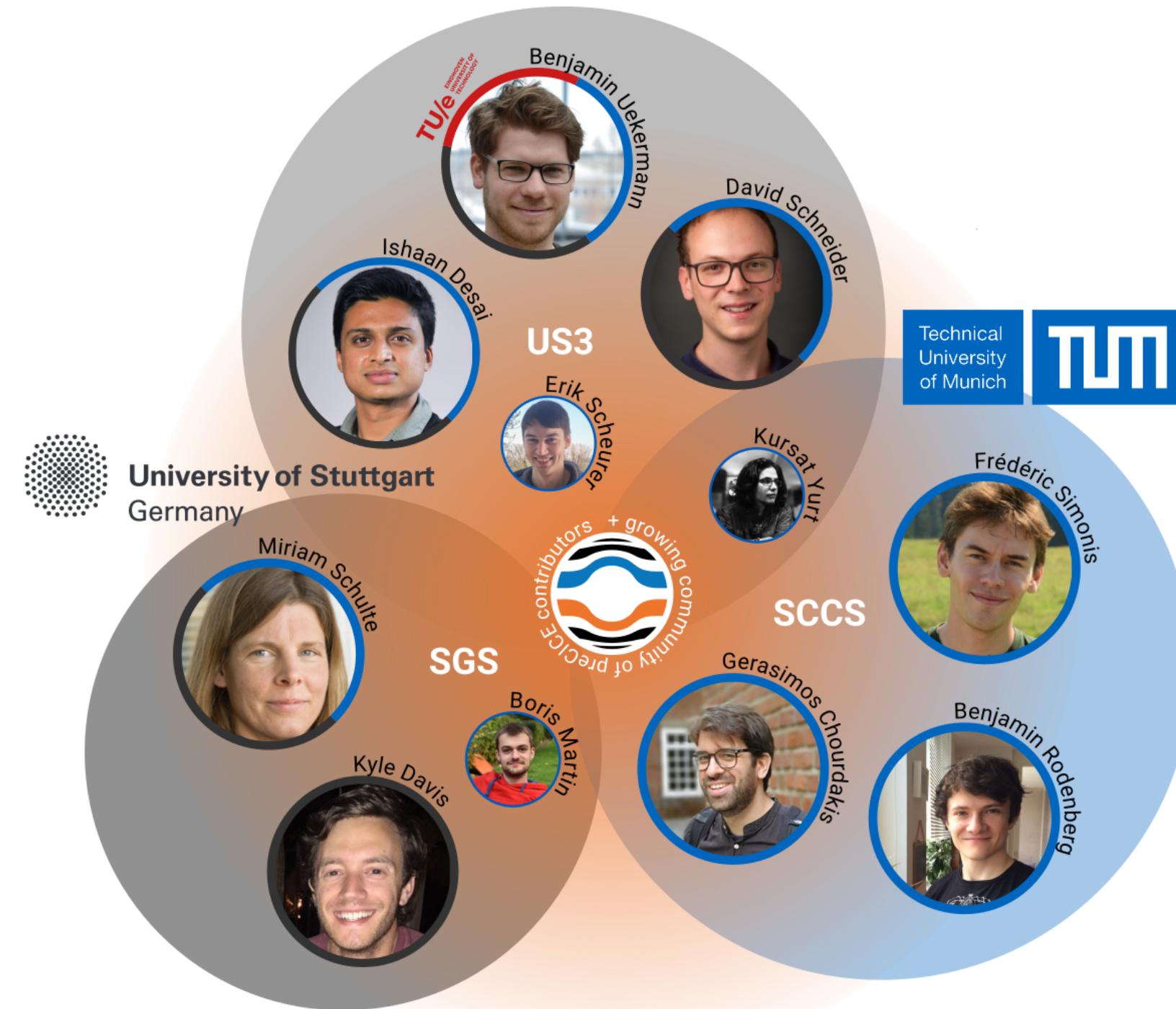
2. Garth Wells ⓘ, University of Cambridge, Cambridge, UK



The people behind preCICE



The people behind preCICE



Meet the people



preCICE is free because of



Research Software
Sustainability



EXC 2075
SimTech



Bundesministerium
für Wirtschaft
und Energie



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

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

Summary

Both for developers and for users,
many efficient algorithms to choose from via just config

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

Please give me feedback on this talk: go.tum.de/530822



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

