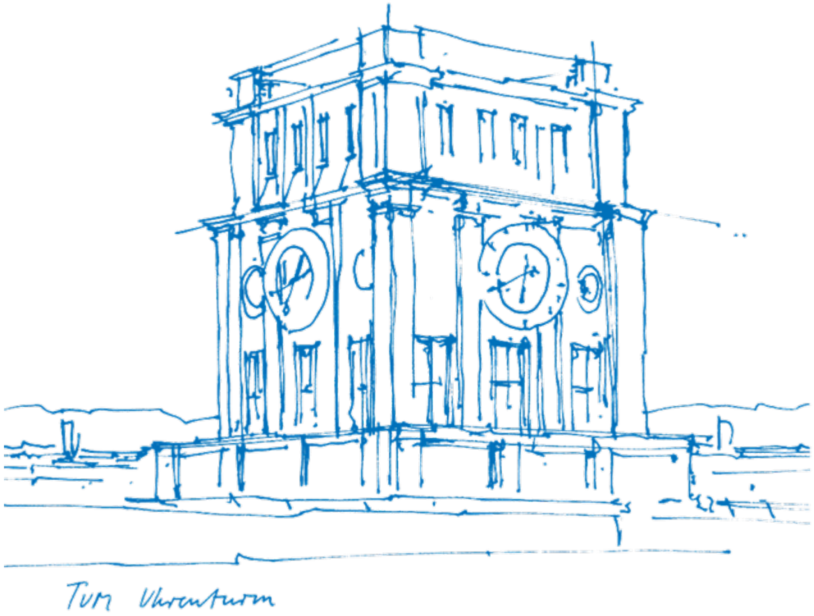
# Design and evaluation of a waveform iteration–based approach for coupling heterogeneous time stepping methods via preCICE

Benjamin Rodenberg[1], Ishaan Desai[2], Benjamin Uekermann[2]

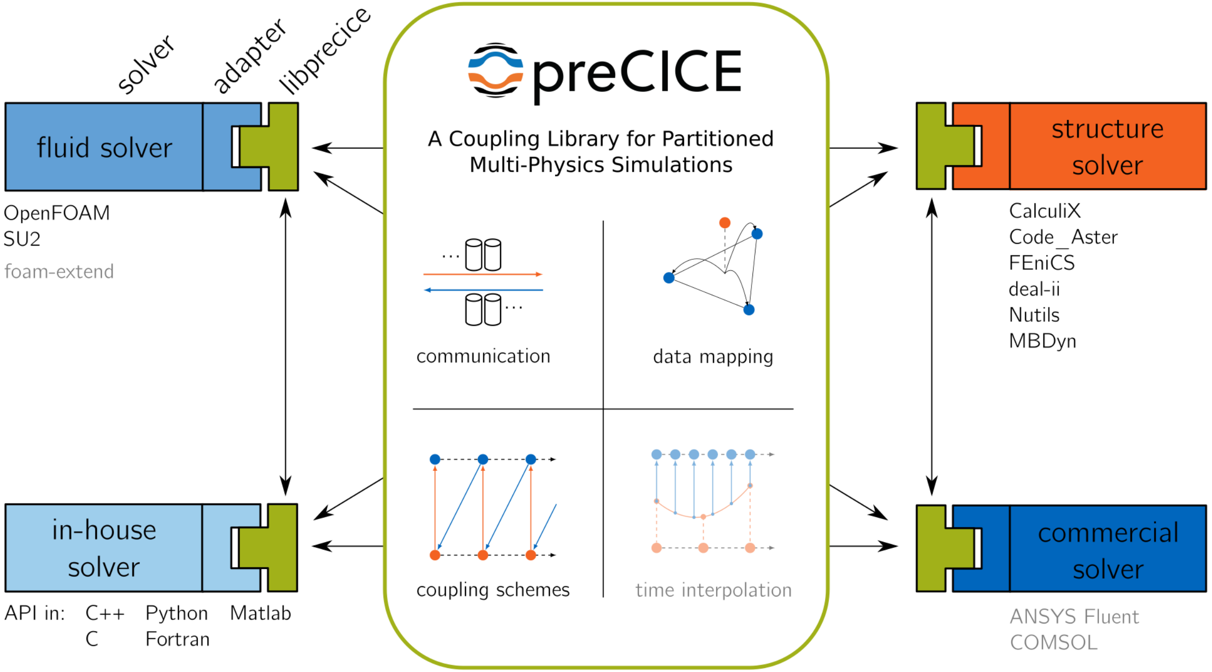
[1]Technical University of Munich, Department of Informatics

[2]University of Stuttgart, Usability and Sustainability of Simulation Software

WCCM-XV & APCOM-VIII

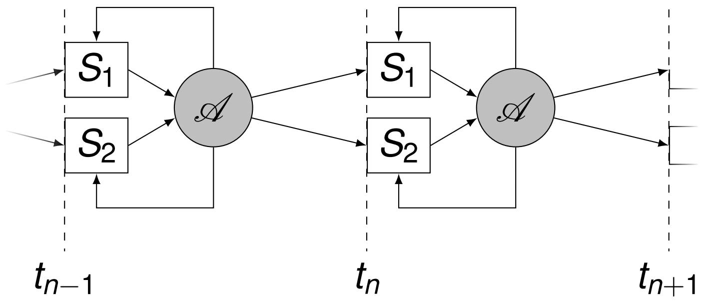



recorded on July 13, 2022

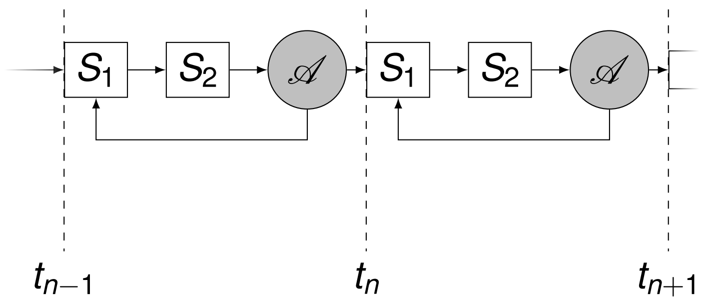

solver
adapter
libprecice
fluid solver
OpenFOAM
SU2
foam-extend
in-house solver
API in: C++ Python Matlab
C Fortran
preCICE
A Coupling Library for Partitioned Multi-Physics Simulations
communication
data mapping
coupling schemes
time interpolation
structure solver
CalculiX
Code_Aster
FEniCS
deal-ii
Nutils
MBDyn
commercial solver
ANSYS Fluent
COMSOL

# Implicit coupling schemes

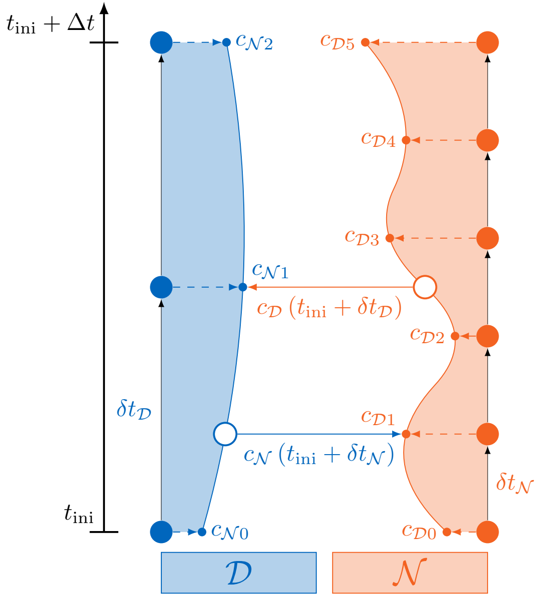
parallel




serial



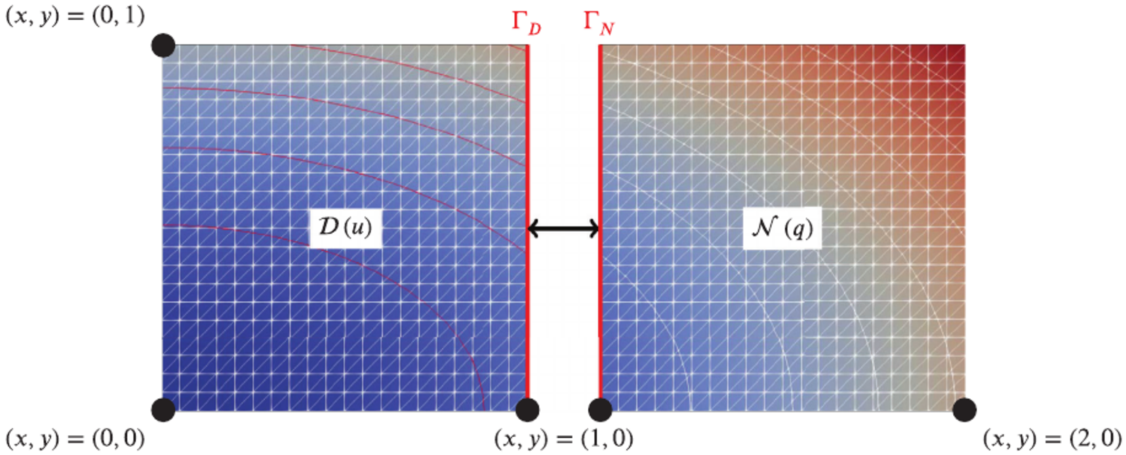
# Idea: Use waveforms




For details: See QNWI paper[1]

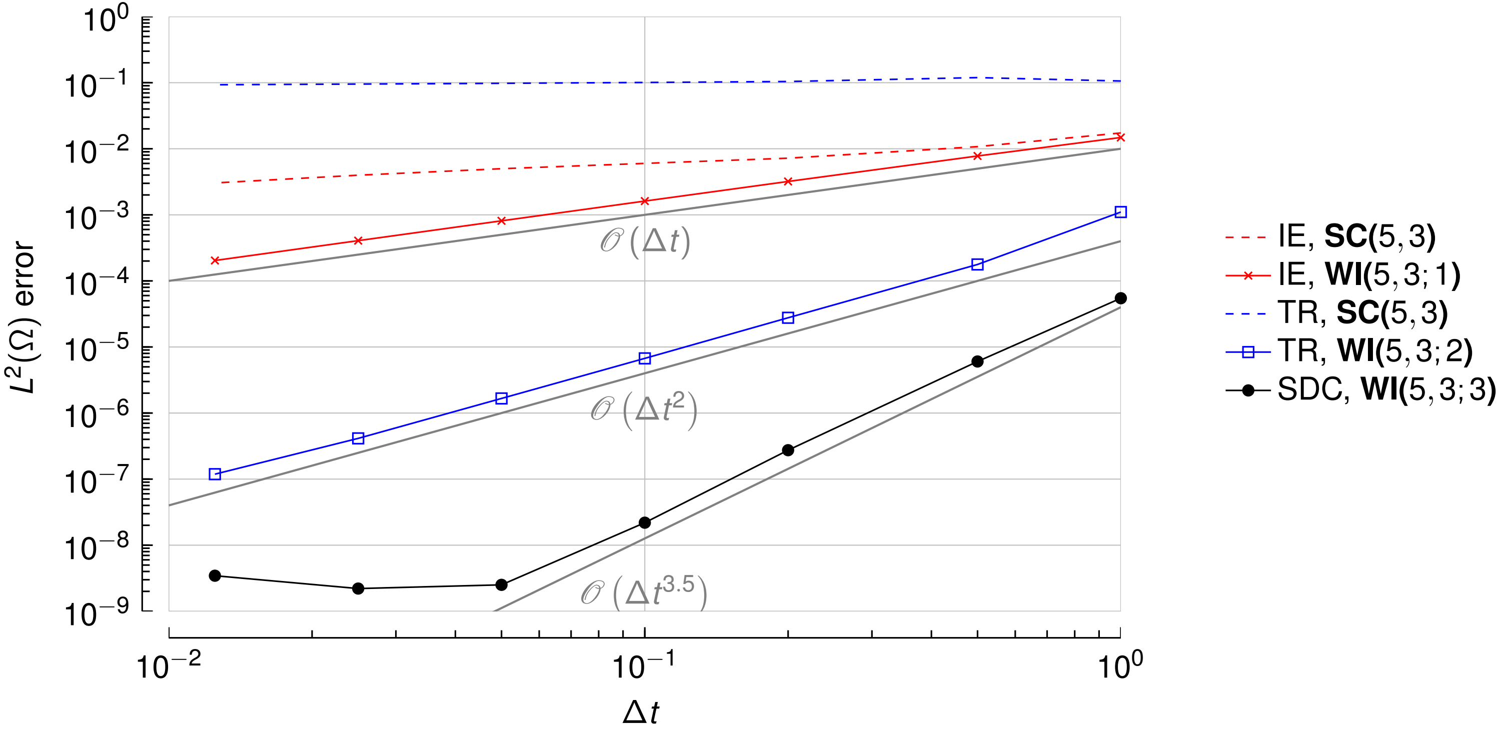
[1] *Quasi-Newton waveform iteration for partitioned surface-coupled multiphysics applications. Int J Numer Methods Eng. 2021. https://doi.org/10.1002/nme.6443*

# QNWI paper: What can we expect?

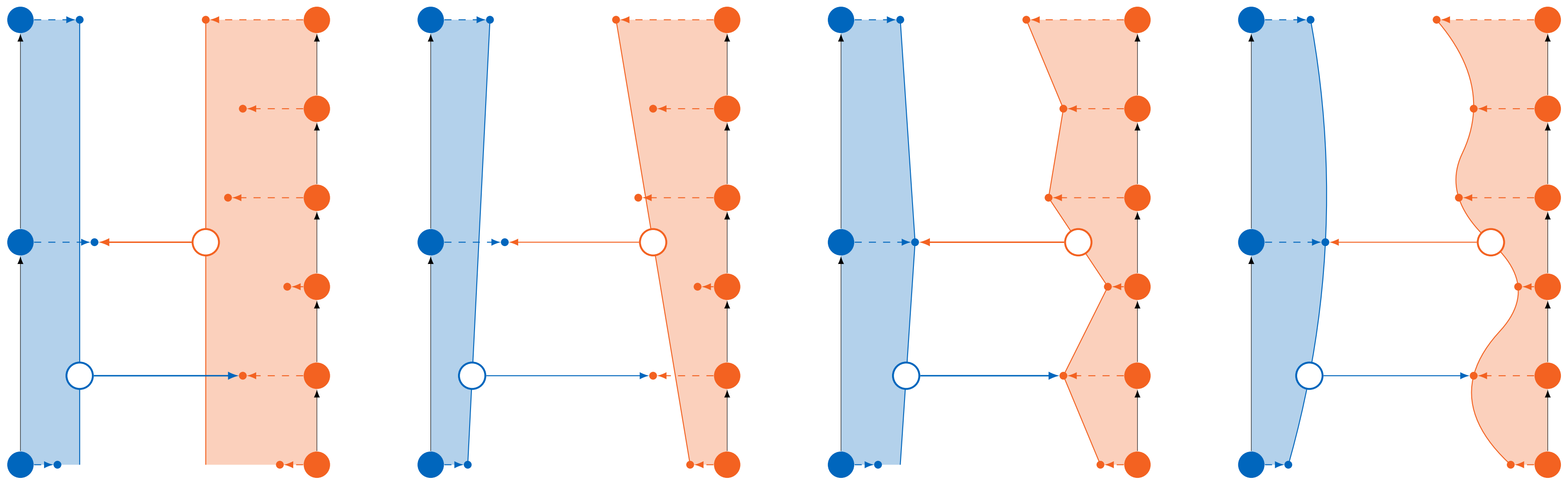



- Manufactured solution allows us to quickly check for errors: $u(x,y,t) = 1 + g(t)x^2 + 3y^2 + 1.2t$
- linear FEM are exact in space. Only error in time, if any.
- $g(t)$ is used to check
  - for high order: $g(t) = (1+t)^\alpha$
  - for convergence (time & quasi-Newton): $g(t) = \sin(t)$

# QNWI paper: What can we expect?



# Bringing waveforms to preCICE



v1.0 | v2.4 & v3.0 | v3.x? | QNWI paper[1]

[1] *Quasi-Newton waveform iteration for partitioned surface-coupled multiphysics applications. Int J Numer Methods Eng. 2021. https://doi.org/10.1002/nme.6443*

# Bringing waveforms to preCICE

## Define interpolation order:

```xml
<participant name="SolverOne">
  ...
  <write-data name="Force" mesh="MeshOne" />
  <read-data name="Velocities" mesh="MeshOne" waveform-order="1" />
</participant>
```

## Sample from interpolant:

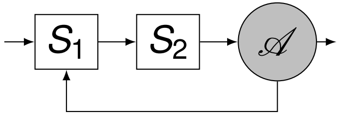
```cpp
// old API
precice.readScalarData(readDataID, vertexID, readData);

// new API has optional argument
precice.readScalarData(readDataID, vertexID, sampleDt, readData);
```

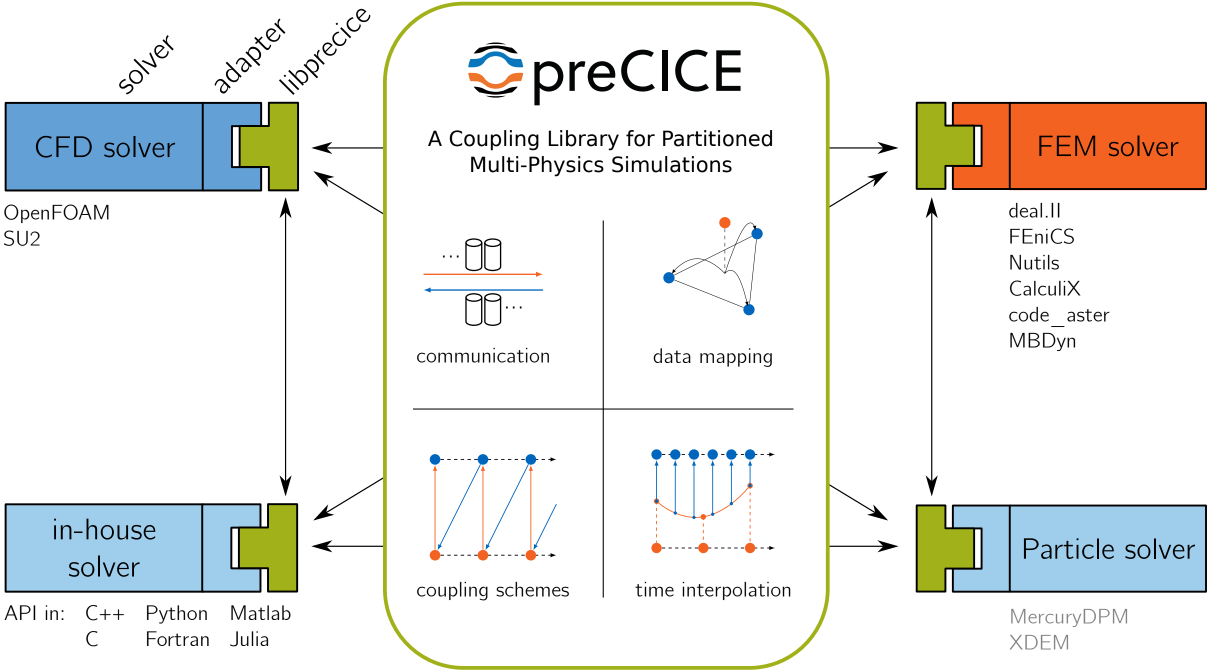
# Bringing waveforms to preCICE

```xml
<coupling-scheme:serial-implicit>
  <participants first="SolverOne" second="SolverTwo" />
  <max-time-windows value="10" />
  <time-window-size value="1.0" />
  ...
  <exchange data="Forces" from="SolverOne" to="SolverTwo" initialize="true"/>
  <exchange data="Velocities" from="SolverTwo" to="SolverOne" initialize="true"/>
</coupling-scheme:serial-implicit>
```

Data initialization for data sent from first to second participant in serial coupling is now possible.



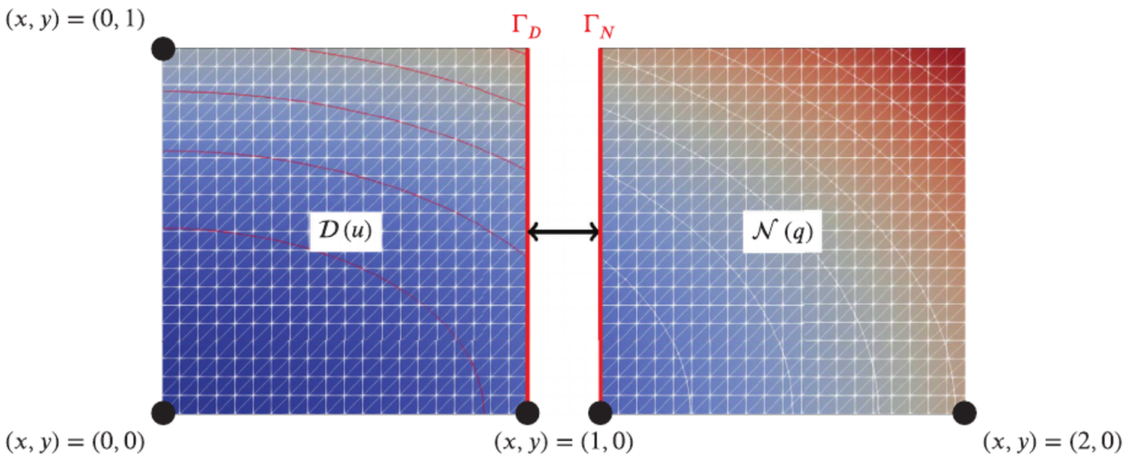
# Remaining components: bindings, adapter, tutorials



# Remaining components: bindings, adapter, tutorials

- python bindings (`https://github.com/precice/python-bindings/pull/147`)
- FEniCS adapter (`https://github.com/precice/fenics-adapter/pull/153`)
- update tutorial (`https://github.com/precice/tutorials/pull/281`)
- All these updates were just a few hours of work.
- Conjugate heat transfer with multirate works as expected!

# Evaluation: Conjugate heat transfer




- Manufactured solution allows us to quickly check for errors: $u(x, y, t) = 1 + g(t)x^2 + 3y^2 + 1.2t$
- linear FEM are exact in space. Only error in time, if any.
- $g(t)$ is used to check
  - for high order: $g(t) = (1 + t)^\alpha$
  - for convergence (time & quasi-Newton): $g(t) = \sin(t)$

# tutorials/partitioned-heat-conduction

6 partitioned-heat-conduction/precice-config.xml

Viewed

```diff
@@ -7,7 +7,7 @@
       enabled="true" />
   </log>

-  <solver-interface dimensions="2">
+  <solver-interface dimensions="2" experimental="true">
     <data:scalar name="Temperature" />
     <data:scalar name="Heat-Flux" />

@@ -25,7 +25,7 @@
       <use-mesh name="Dirichlet-Mesh" provide="yes" />
       <use-mesh name="Neumann-Mesh" from="Neumann" />
       <write-data name="Heat-Flux" mesh="Dirichlet-Mesh" />
-      <read-data name="Temperature" mesh="Dirichlet-Mesh" />
+      <read-data name="Temperature" mesh="Dirichlet-Mesh" waveform-order="1"/>
       <mapping:rbf-thin-plate-splines
         direction="read"
         from="Neumann-Mesh"
@@ -38,7 +38,7 @@
       <use-mesh name="Neumann-Mesh" provide="yes" />
       <use-mesh name="Dirichlet-Mesh" from="Dirichlet" />
       <write-data name="Temperature" mesh="Neumann-Mesh" />
-      <read-data name="Heat-Flux" mesh="Neumann-Mesh" />
+      <read-data name="Heat-Flux" mesh="Neumann-Mesh" waveform-order="0"/>
       <mapping:rbf-thin-plate-splines
         direction="read"
         from="Dirichlet-Mesh"
```

# tutorials/partitioned-heat-conduction

4 partitioned-heat-conduction/fenics/heat.py

Viewed

```
@@ -62,7 +62,7 @@ def determine_gradient(V_g, u, flux):

args = parser.parse_args()

- fenics_dt = .1  # time step size
+ fenics_dt = .01  # time step size
# Error is bounded by coupling accuracy. In theory we would obtain the analytical solution.
error_tol = args.error_tol

@@ -178,7 +178,7 @@ def determine_gradient(V_g, u, flux):
    if precice.is_action_required(precice.action_write_iteration_checkpoint()):
        precice.store_checkpoint(u_n, t, n)

-   read_data = precice.read_data()
+   read_data = precice.read_data(dt(0))

    # Update the coupling expression with the new read data
    precice.update_coupling_expression(coupling_expression, read_data)
```

# Are we there?

## Parallel implicit

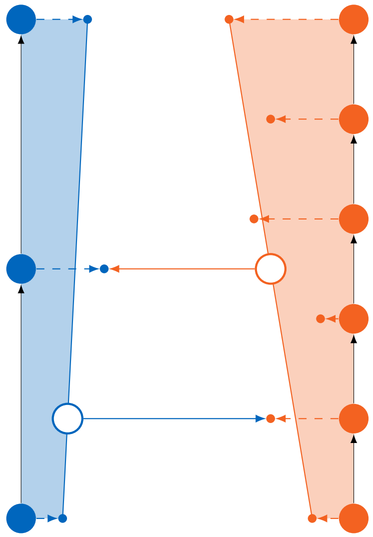
Released in preCICE v2.4.0 (`https://github.com/precice/precice/releases/tag/v2.4.0`)

## Serial implicit

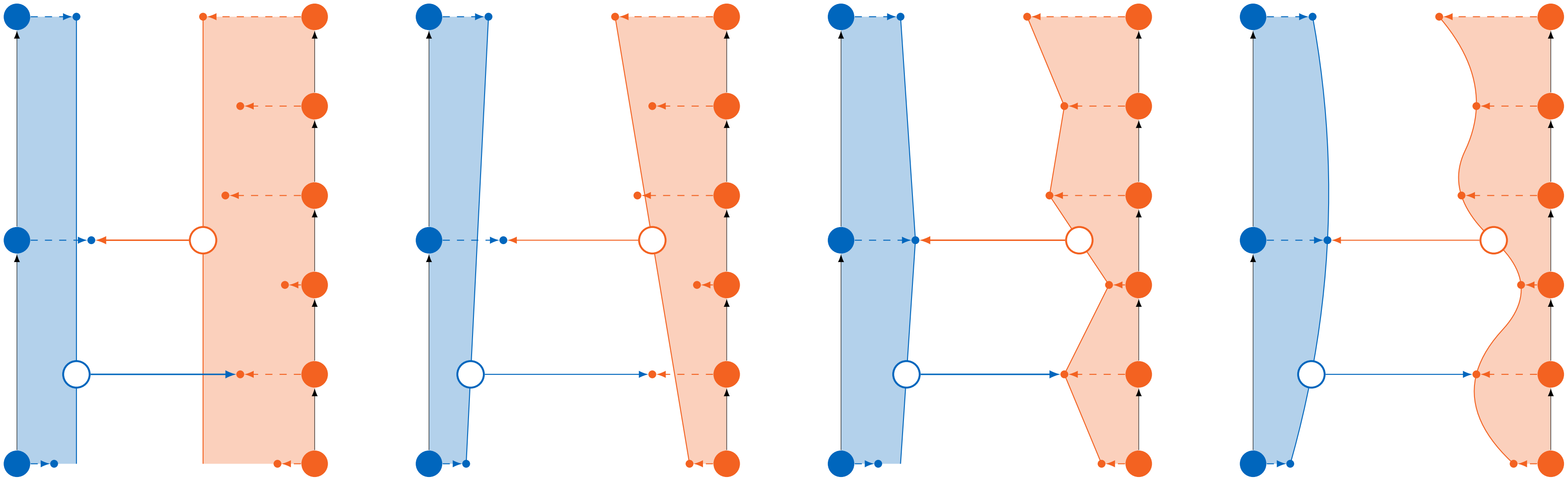
Ready for preCICE v3.0.0

## Restrictions

First order, no real multirate support.



# Outlook: Multirate, Higher order, Quasi-Newton



v1.0 v2.4 & v3.0 v3.x? QNWI paper[1]

[1] *Quasi-Newton waveform iteration for partitioned surface-coupled multiphysics applications. Int J Numer Methods Eng. 2021. https://doi.org/10.1002/nme.6443*