A Quasi-Newton Accelerated Multirate Coupling Scheme for Partitioned Simulation

Benjamin Rüth[1], Benjamin Uekermann[2], Hans-Joachim Bungartz[3], Miriam Mehl[4], Azahar Monge [5], Philipp Birken [6]

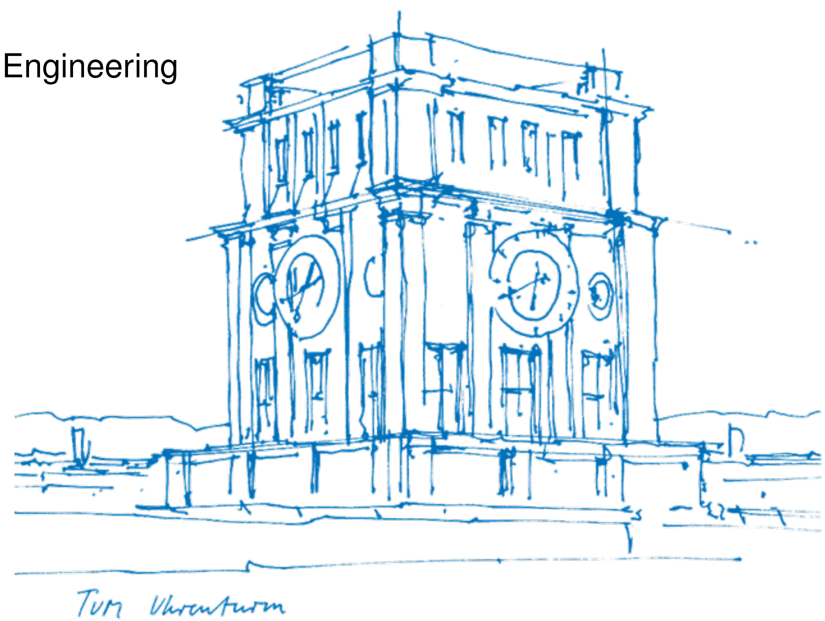
[1,3] Technical University of Munich, Department of Informatics

[2] Eindhoven University of Technology, Department of Mechanical Engineering

[4] University of Stuttgart, Simulation of Large Systems

[5] University of Deusto, Chair of Computational Mathematics

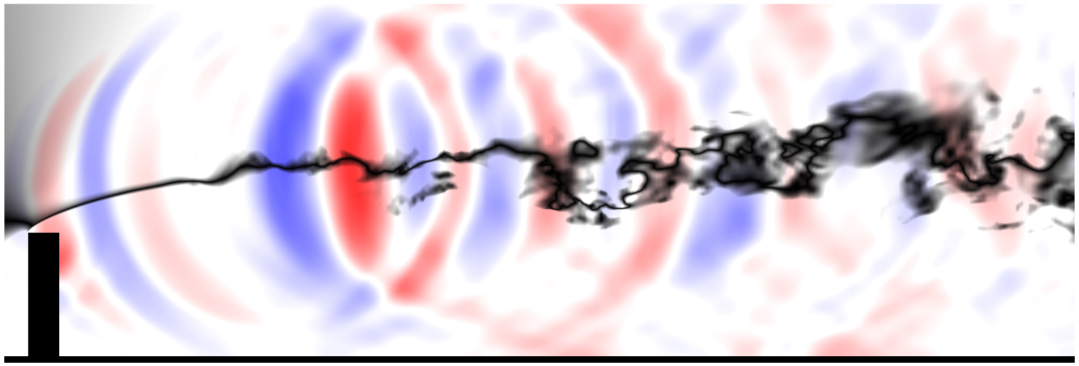
[6] Lund University, Centre for Mathematical Sciences

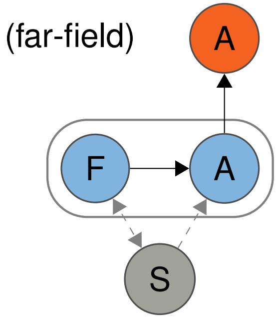


Coupled Problems 2019
Sitges, Spain
June 5, 2019

ExaFSA setup

Why Quasi-Newton? → IQN-ILS





Fluid-acoustics simulation and partitioned setup[1].

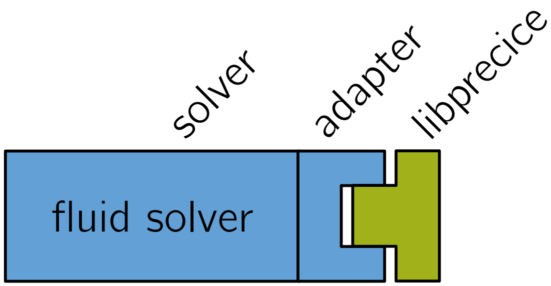
Why multirate? → Waveform relaxation

physics	timescale	solver
(A)	small	Ateles
(A)	small	FASTEST
(F)	medium	FASTEST
(S)	large	FEAP

[1]Reimann, T., et al. (2017). Aspects of FSI with aeroacoustics in turbulent flow. In 7th GACM Colloquium on Computational Mechanics.

preCICE

A Plug-and-Play Coupling Library



OpenFOAM
SU2
foam-extend

preCICE

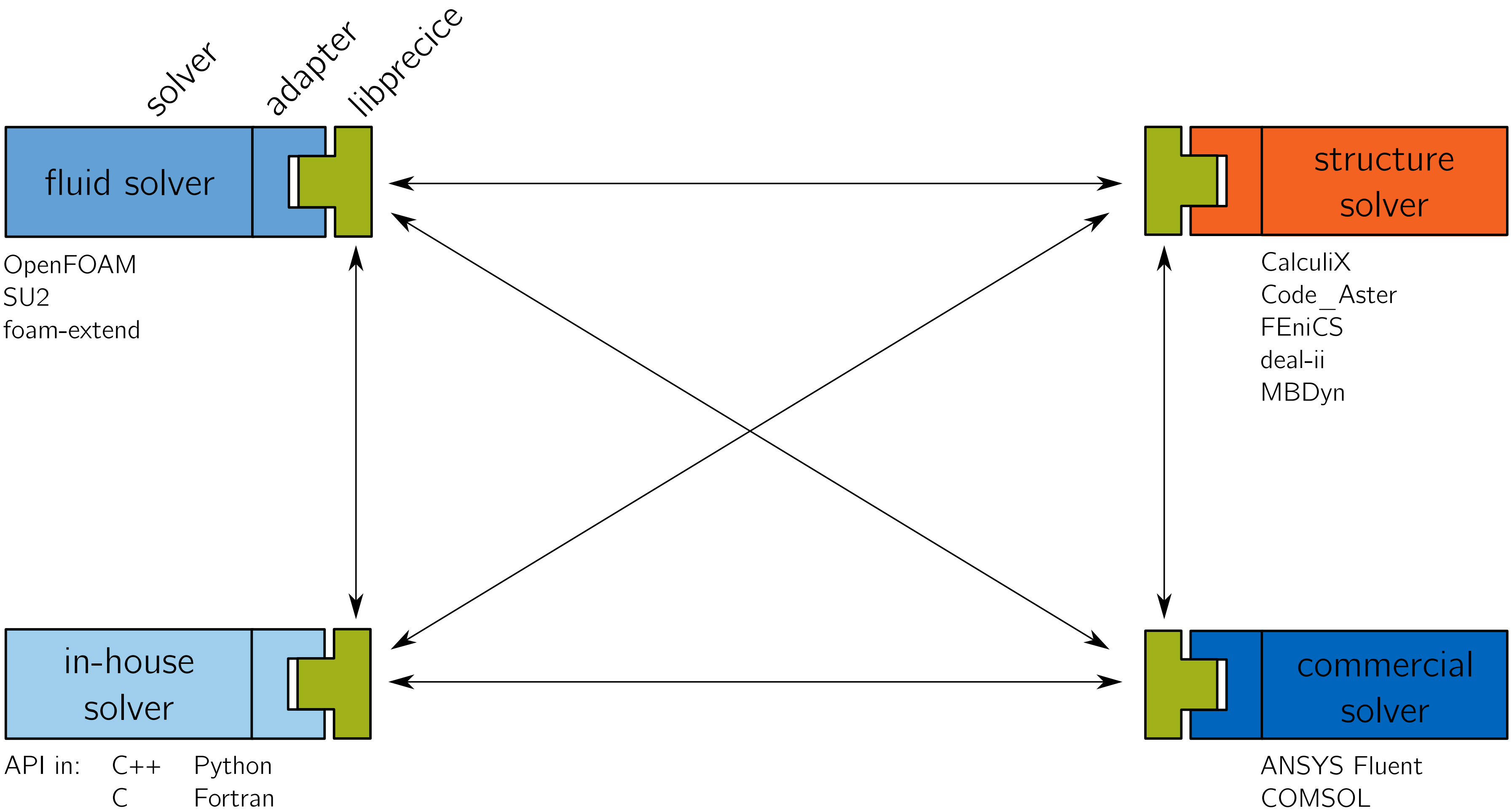
A Plug-and-Play Coupling Library

OpenFOAM
SU2
foam-extend

CalculiX
Code_Aster
FEniCS
deal-ii
MBDyn

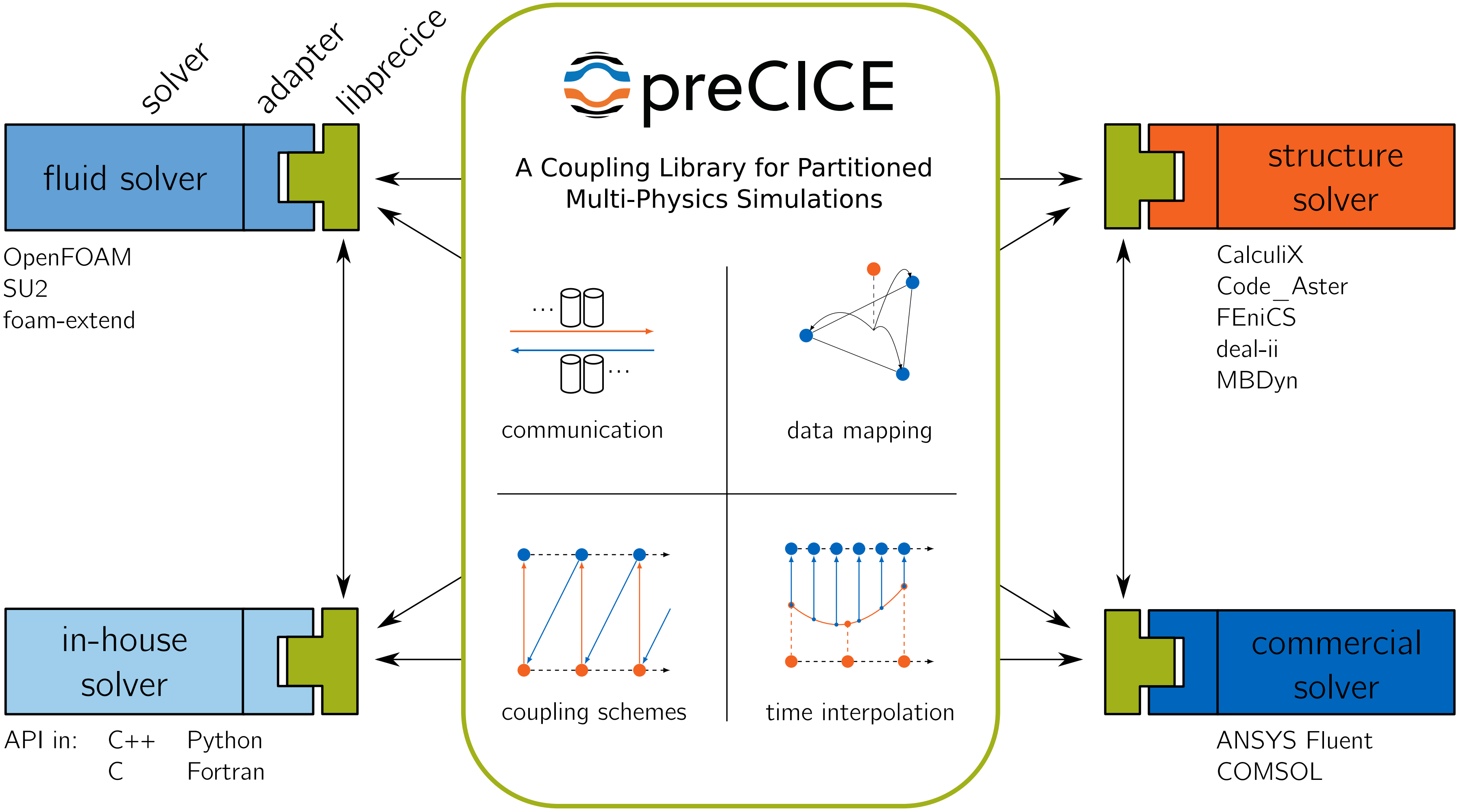
preCICE

A Plug-and-Play Coupling Library



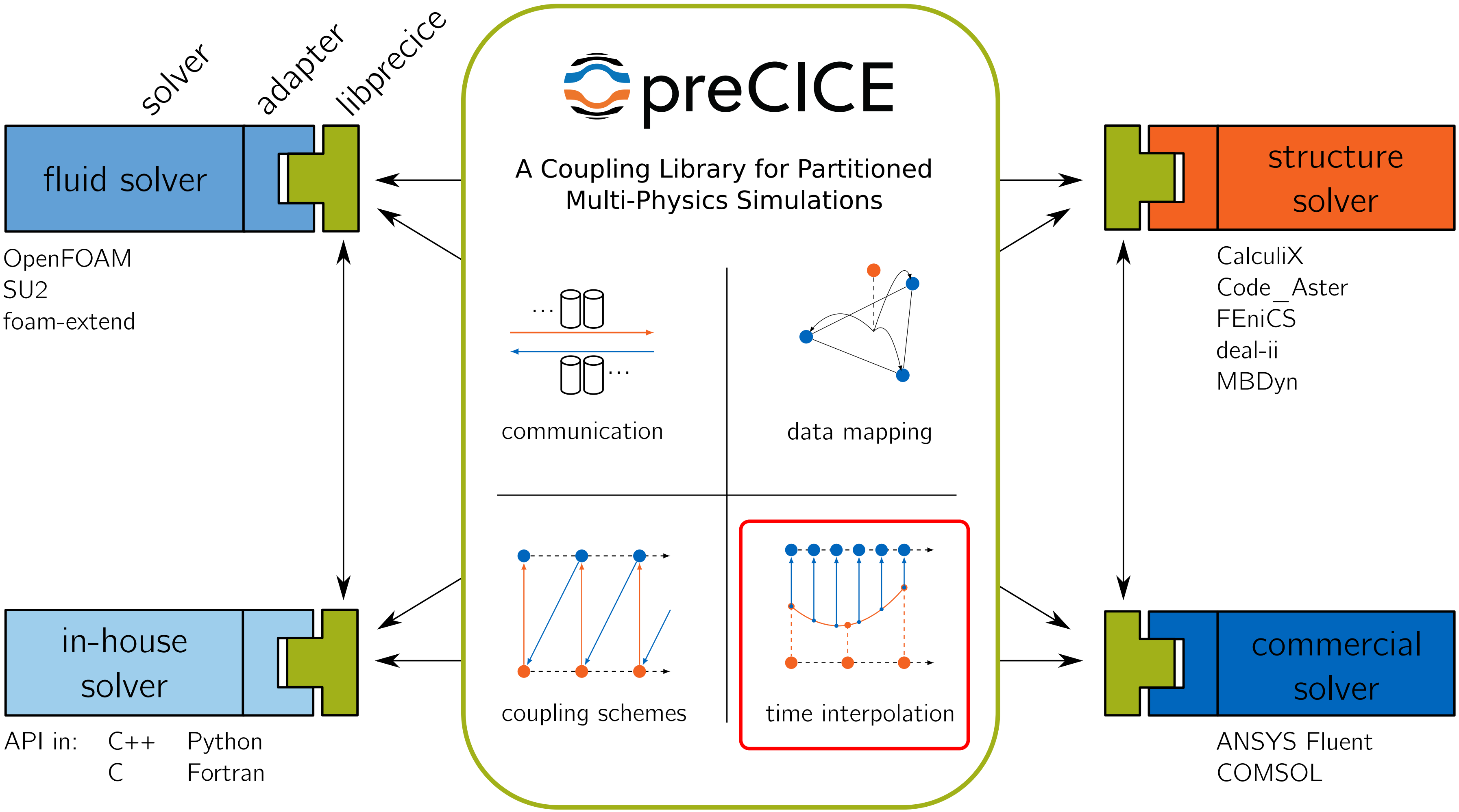
preCICE

A Plug-and-Play Coupling Library

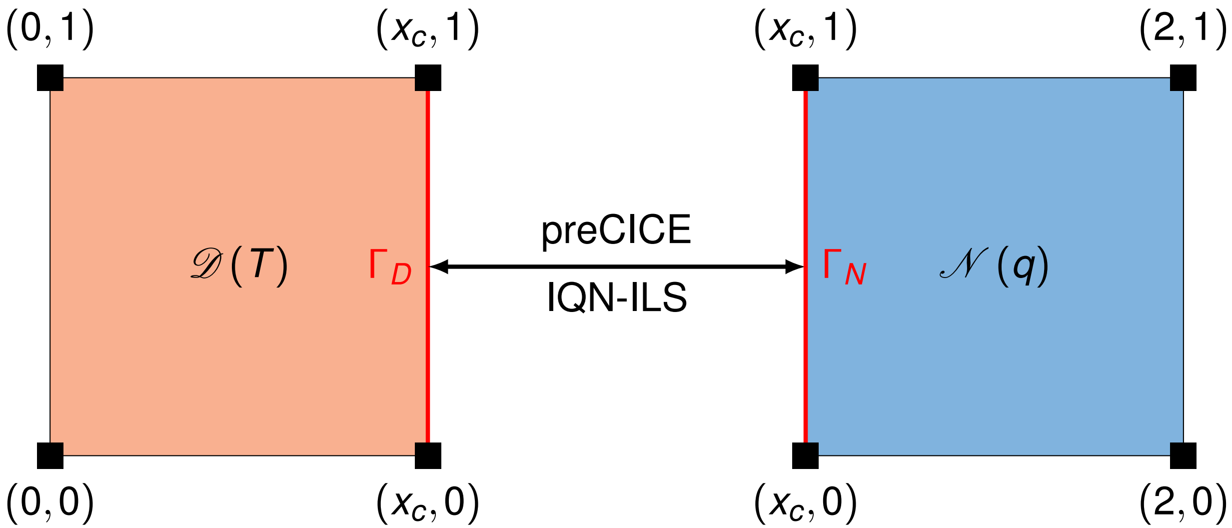


preCICE

A Plug-and-Play Coupling Library



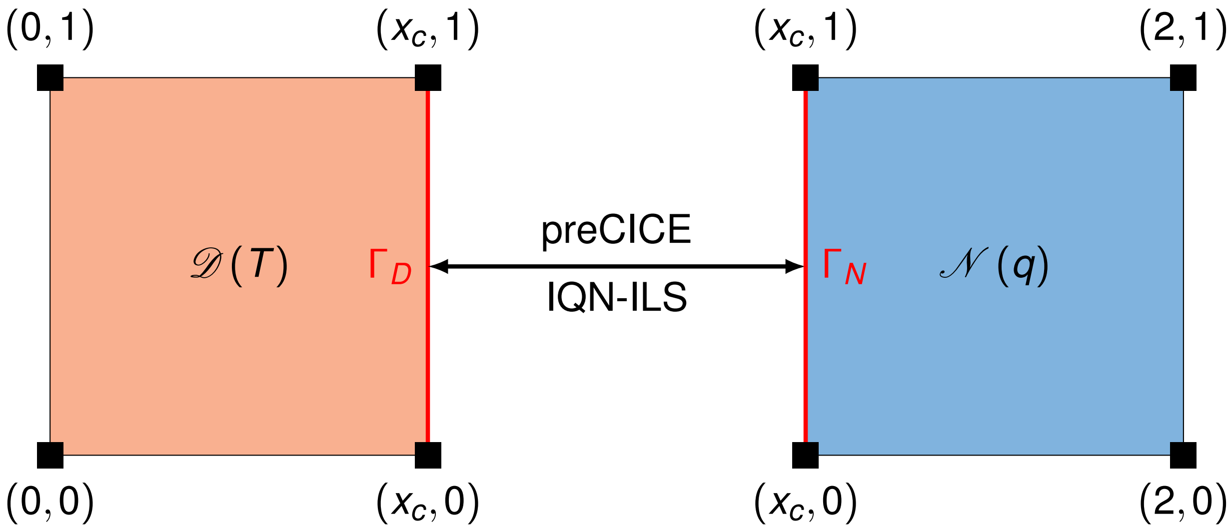
Partitioned Heat Equation



Partitioned heat equation / transmission problem already discussed in literature (e.g.[1] or [2]).

[1] Monge, A. (2018). Partitioned methods for time-dependent thermal fluid-structure interaction. Lund University.
[2] Toselli, A., & Widlund, O. (2005). Domain Decomposition Methods - Algorithms and Theory (1st ed.).
[3] Rüth, B. et al. (2018). Solving the Partitioned Heat Equation Using FEniCS and preCICE. GAMM CSE Workshop 2018.

Partitioned Heat Equation



Partitioned heat equation / transmission problem already discussed in literature (e.g.[1] or [2]).

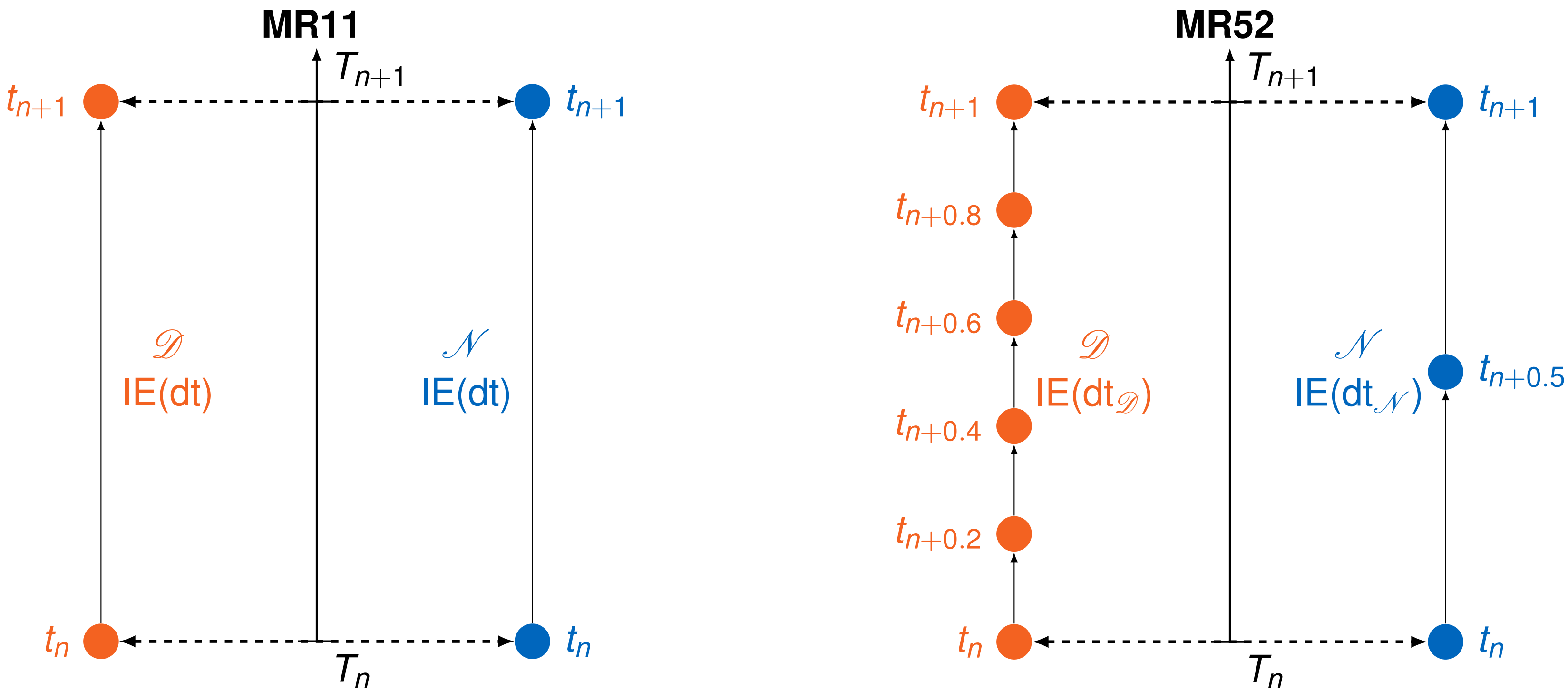
- Problem: $\frac{\partial u}{\partial t} = \Delta u - f$ with $f = \beta - 2 - 2\alpha$ and Dirichlet BC on outer boundary
- Analytical Solution: $u(x,y,t) = 1 + x^2 + \alpha y^2 + \beta t$ (can be obtained with implicit Euler + linear FEM)
- FEniCS used for FEM
- tutorial in `precice/tutorials`
- for more details see [3].

[1] Monge, A. (2018). Partitioned methods for time-dependent thermal fluid-structure interaction. Lund University.
[2] Toselli, A., & Widlund, O. (2005). Domain Decomposition Methods - Algorithms and Theory (1st ed.).
[3] Rüth, B. et al. (2018). Solving the Partitioned Heat Equation Using FEniCS and preCICE. GAMM CSE Workshop 2018.

Extension: Multirate

Can we obtain the exact solution for all setups?

$$u(x,y,t) = 1 + x^2 + \alpha y^2 + \beta t$$



timestepping dt $=$ coupling dT

5 $dt_{\mathscr{D}} = 2\ dt_{\mathscr{N}} = dT$

Extension: Multirate

First experiments

- *vanilla* preCICE supports multirate (+ python bindings)
- coupling happens only at T_n (not at $t_{n+0.2}, t_{n+0.4}, \ldots$)
- no coupling at substeps! No exchange of substeps!
- only MR11 recovers exact solution
- other scenarios (MR55, MR52...) introduce error

[1] Rüth, B., et al (2018). Time Stepping Algorithms for Partitioned Multi-Scale Multi-Physics. Proceedings of ECCM VI / ECFD VII, (June).

Extension: Multirate

First experiments

- *vanilla* preCICE supports multirate (+ python bindings)
- coupling happens only at T_n (not at $t_{n+0.2}, t_{n+0.4}, \ldots$)
- no coupling at substeps! No exchange of substeps!
- only MR11 recovers exact solution
- other scenarios (MR55, MR52...) introduce error

Possible explanation

- first order IE time-stepping is spoiled by zeroth order constant interpolation at coupling interface

$\rightarrow$ Our hope: Using higher order interpolation (and exchanging subsamples) recovers exact solution[1]

[1] Rüth, B., et al (2018). Time Stepping Algorithms for Partitioned Multi-Scale Multi-Physics. Proceedings of ECCM VI / ECFD VII, (June).

Extension: Multirate

First experiments

- *vanilla* preCICE supports multirate (+ python bindings)
- coupling happens only at T_n (not at $t_{n+0.2}, t_{n+0.4}, \ldots$)
- no coupling at substeps! No exchange of substeps!
- only MR11 recovers exact solution
- other scenarios (MR55, MR52...) introduce error

Possible explanation

- first order IE time-stepping is spoiled by zeroth order constant interpolation at coupling interface

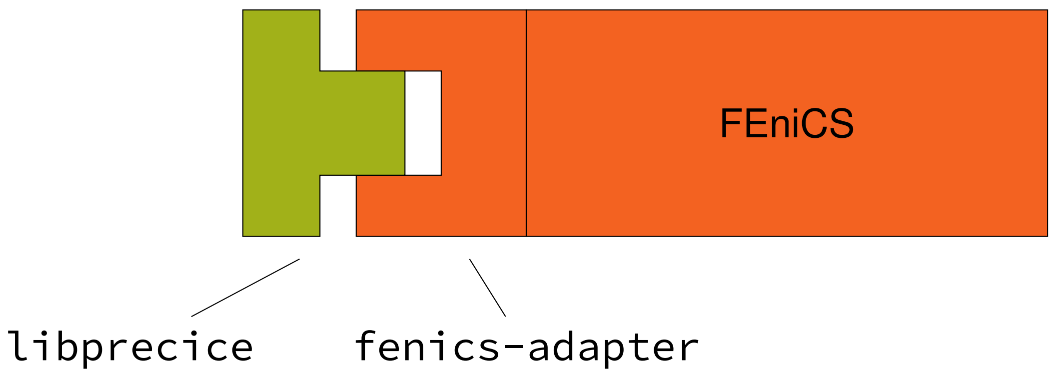
$\rightarrow$ Our hope: Using higher order interpolation (and exchanging subsamples) recovers exact solution[1]

Wishlist

- Minimal changes in preCICE, FEniCS solver and adapter
- Exchange blackbox data (nodal data + timestamps)
- Use Quasi-Newton
- Arbitrary multirate setups (MR11, MR12, MR21, ...)
- No degradation of quality of solution

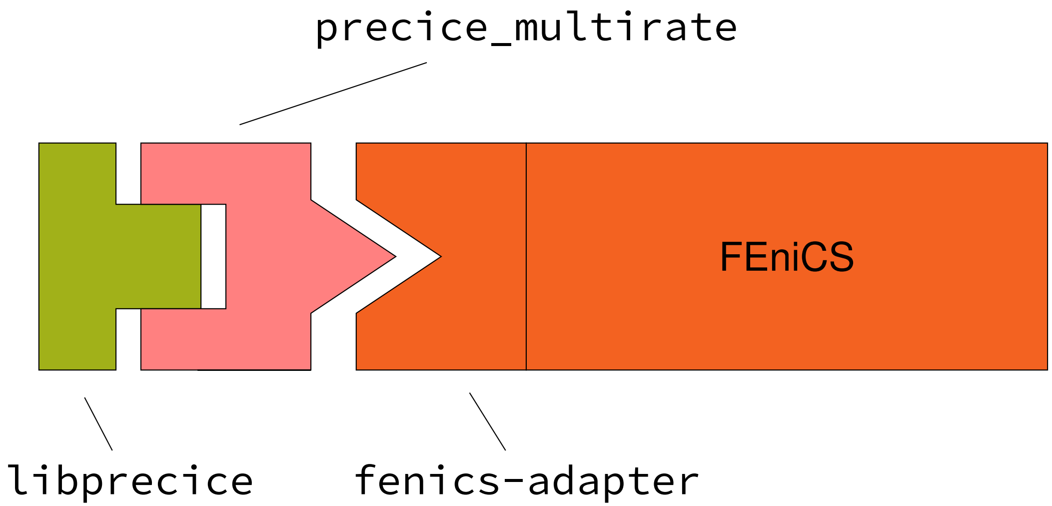
[1] Rüth, B., et al (2018). Time Stepping Algorithms for Partitioned Multi-Scale Multi-Physics. Proceedings of ECCM VI / ECFD VII, (June).

Implementation




```
<precice-configuration>
    <data:scalar name="Temperature"/>
    <data:scalar name="Flux"/>...
    <post-processing:IQN-ILS>...
        <data name="Temperature" mesh="NeumannNodes"/>
    </post-processing:IQN-ILS>
</precice-configuration>
```

Implementation



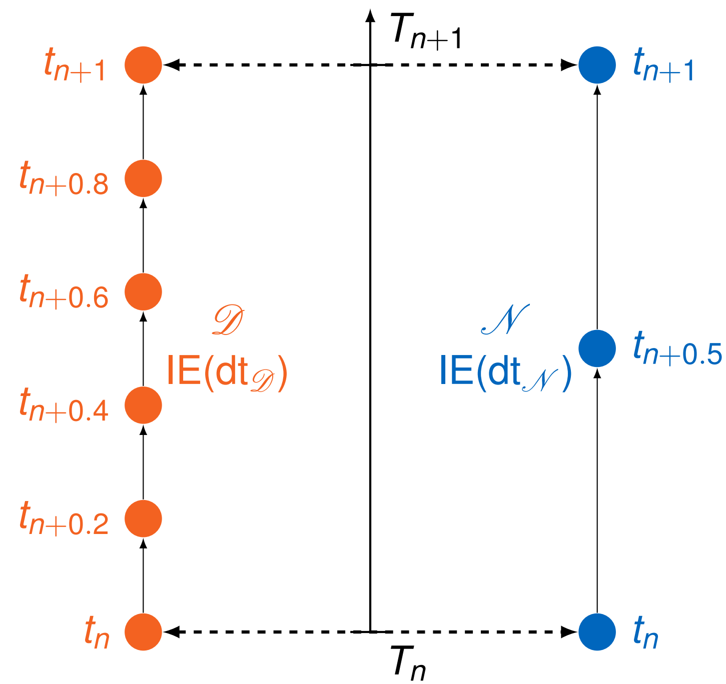

```
<precice-configuration>
    <data:scalar name="Temperature1"/>
    <data:scalar name="Temperature2"/>
    <data:scalar name="Flux1"/>...
    <post-processing:IQN-ILS>...
        <data name="Temperature1" mesh="NeumannNodes"/>
        <data name="Temperature2" mesh="NeumannNodes"/>
    </post-processing:IQN-ILS>
</precice-configuration>
```

Results

QN Iterations / window; $T = 10$

dT	1.0	0.5	0.2	0.1
MR11	15.9	16.3	16.7	17.3
MR12	6.7	4.4	3.8	3.4
MR15	5.7	4.8	3.7	3.4
MR110	6.7	4.9	3.8	3.4
MR21	17.0	16.9	16.9	17.3
MR22	7.2	4.6	3.7	3.2
MR25	7.5	4.7	3.9	3.5
MR210	6.9	5.0	3.9	3.5
MR51	16.9	17.0	17.5	17.7
MR52	6.9	5.1	3.7	3.3
MR55	6.6	4.3	3.7	3.3
MR510	7.0	4.8	3.7	3.5
MR1010	6.6	4.8	3.7	3.3

Example Setup: MR52



$5\ dt_{\mathscr{D}} = 2\ dt_{\mathscr{N}} = dT$

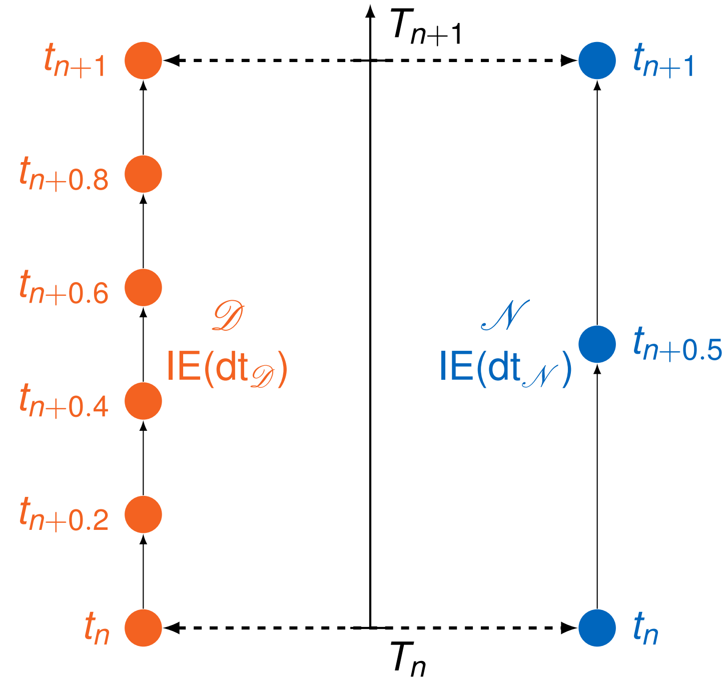
10 dT = T = 10

Results

QN Iterations / window; $T = 10$

dT	1.0	0.5	0.2	0.1
MR11	15.9	16.3	16.7	17.3
MR12	6.7	4.4	3.8	3.4
MR15	5.7	4.8	3.7	3.4
MR110	6.7	4.9	3.8	3.4
MR21	17.0	16.9	16.9	17.3
MR22	7.2	4.6	3.7	3.2
MR25	7.5	4.7	3.9	3.5
MR210	6.9	5.0	3.9	3.5
MR51	16.9	17.0	17.5	17.7
MR52	6.9	5.1	3.7	3.3
MR55	6.6	4.3	3.7	3.3
MR510	7.0	4.8	3.7	3.5
MR1010	6.6	4.8	3.7	3.3

Example Setup: MR52



$5\ dt_{\mathscr{D}} = 2\ dt_{\mathscr{N}} = dT$
10 dT = T = 10

- strict convergence limit $\rightarrow$ many iterations
- QN iterations constant for fixed dT
- QN Iterations reduced for smaller dT
- MRx1 bug or feature?

Example case

T=10	MR22	MR210	MR1010
$dt_{\mathscr{D}}/dt_{\mathscr{N}}$	0.1/0.1	0.5/0.1	0.1/0.1
dT	0.2	1.0	1.0
iterations per window	3.7	6.9	6.6
total windows	50	10	10
total iterations	185	69	66
timesteps $\mathscr{D}$	370	138	660
timesteps $\mathscr{N}$	370	690	660

Example case

T=10	MR22	MR210	MR1010
$dt_{\mathscr{D}}/dt_{\mathscr{N}}$	0.1/0.1	0.5/0.1	0.1/0.1
dT	0.2	1.0	1.0
iterations per window	3.7	6.9	6.6
total windows	50	10	10
total iterations	185	69	66
timesteps $\mathscr{D}$	370	138	660
timesteps $\mathscr{N}$	370	690	660
good:	few iterations	ideal $dt_{\mathscr{D}}$	high parallelism
bad:	low parallelism	many $\mathscr{N}$ iterations	many iterations

Conclusion and future work

Conclusion

- partitioned black-box solvers can use multirate + QN
- functionality can be hidden inside preCICE
  - no changes in solver
  - minimal changes in adapter
- all code on github!
  - preCICE `v1.5.0` + python bindings[1]
  - `fenics-adapter:CoupledProblems2019` [2]
  - `tutorials:CoupledProblems2019/HT/partitioned-heat/fenics-fenics` [3]

[1] `github.com/precice/precice/releases/tag/v1.5.0`
[2] `github.com/precice/fenics-adapter/tree/CoupledProblems2019`
[3] `github.com/precice/tutorials/tree/CoupledProblems2019/HT/partitioned-heat/fenics-fenics`

Conclusion and future work

Conclusion

- partitioned black-box solvers can use multirate + QN
- functionality can be hidden inside preCICE
  - no changes in solver
  - minimal changes in adapter
- all code on github!
  - preCICE `v1.5.0` + python bindings[1]
  - `fenics-adapter:CoupledProblems2019` [2]
  - `tutorials:CoupledProblems2019/HT/partitioned-heat/fenics-fenics` [3]

Future work

- higher order (+ API extensions)
- go beyond heat equation
- real preCICE implementation
- IQN-ILS + reuse
- help user choosing the ideal setup

[1] `github.com/precice/precice/releases/tag/v1.5.0`
[2] `github.com/precice/fenics-adapter/tree/CoupledProblems2019`
[3] `github.com/precice/tutorials/tree/CoupledProblems2019/HT/partitioned-heat/fenics-fenics`

Thank you!

preCICE

Flexible:	Couple your own solver with any other
Easy:	Add a few lines to your code
Ready:	Out-of-the box support for many solvers
Fast:	Fully parallel, peer-to-peer, designed for HPC
Stable:	Implicit coupling, accelerated with Quasi-Newton
Multi-coupling:	Couple more than two solvers
Free:	LGPL3, source on GitHub

- www.precice.org
- github.com/precice
- @preCICE_org
- Mailing-list, Gitter
- Literature Guide on wiki

How does this look in code?

Solves heat equation in FEniCS: `heat.py`

```
u_n, u_np1 = ... # solution
bcs = ... # boundary conditions
a, L = ... u_n ... u_np1 # weak form
while adapter.is_coupling_ongoing():
    solve(a == L, u_np1, bcs)
    dt, bcs = adapter.advance(u_np1, u_n, t, ...)
```

`fenics-adapter`: `advance(...)`

```
import precice
...
data = sample(u_np1)
precice.write(data, "Flux")
next_dt = precice.advance(dt)
data = precice.read("Temperature")
bcs = interpolate(data)
...
```

- solver has state $s = (u, t)$
- `heat.py` solves $s_{n+1} = \mathscr{D}(s_n, b)$.
- boundary conditions b are exchanged and updated via advance

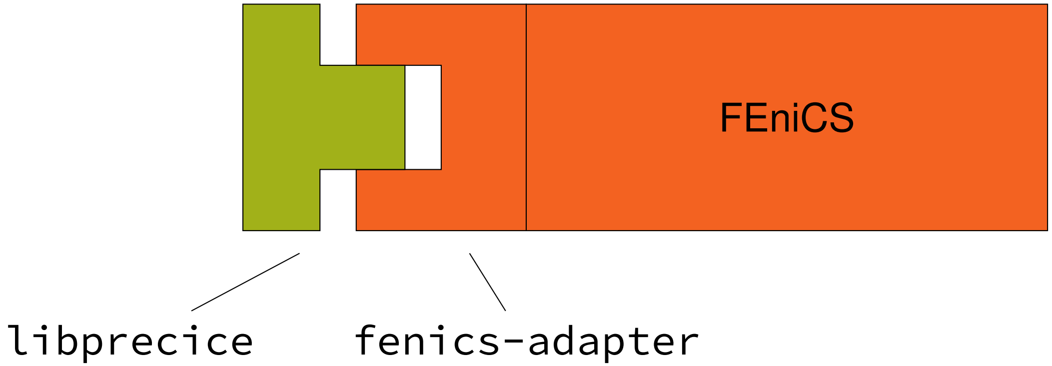
How does this look in code?

Two small changes in advance

- `precice.write(data, t, "Flux")`
- `data = precice.read(t, "Temperature")`

Use `import precice_multirate as precice`

- creates interpolant to provide `data = precice.read(t, "Temperature")`
- maps `precice.write(data, t, "Flux")` to `precice.write(data, "Flux1")`, `...`
- controls stepsize and window



How does this look in code?

Two small changes in advance

- `precice.write(data, t, "Flux")`
- `data = precice.read(t, "Temperature")`

Use `import precice_multirate as precice`

- creates interpolant to provide `data = precice.read(t, "Temperature")`
- maps `precice.write(data, t, "Flux")` to `precice.write(data, "Flux1"), ...`
- controls stepsize and window