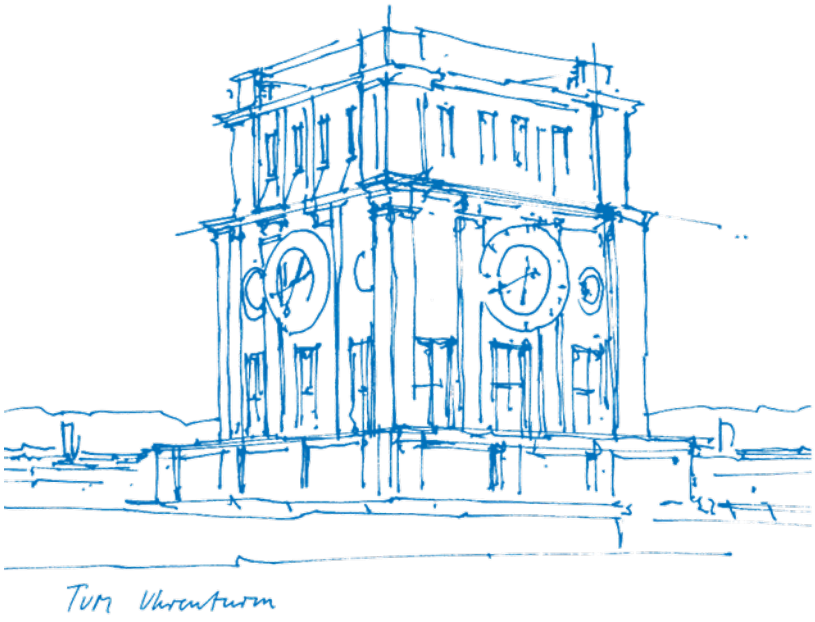
# High-Order Time Stepping in Partitioned FSI with Black-Box Solvers

Benjamin Rüth, Benjamin Uekermann, Miriam Mehl, Hans-Joachim Bungartz

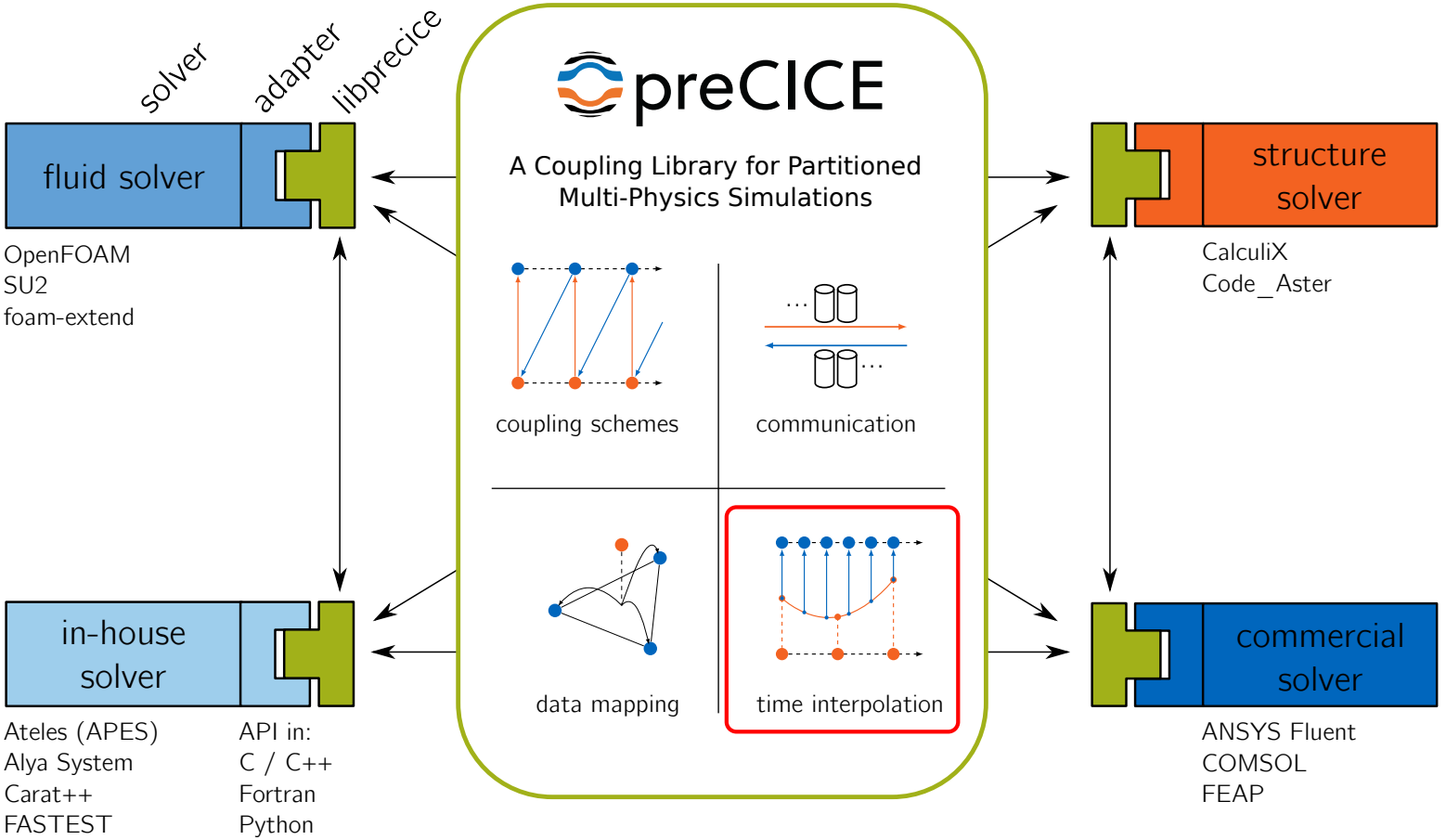
Technical University of Munich
Department of Informatics
Chair of Scientific Computing

WCCM XIII / PANACM II
New York, USA
July 24, 2018



# Partitioned approach

preCICE[1]

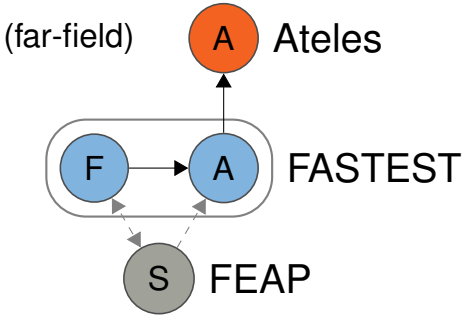



[1] Bungartz, H.-J., et al. (2016). preCICE – A fully parallel library for multi-physics surface coupling.

# Partitioned approach

Motivation & Requirements

## ExaFSA[1]:



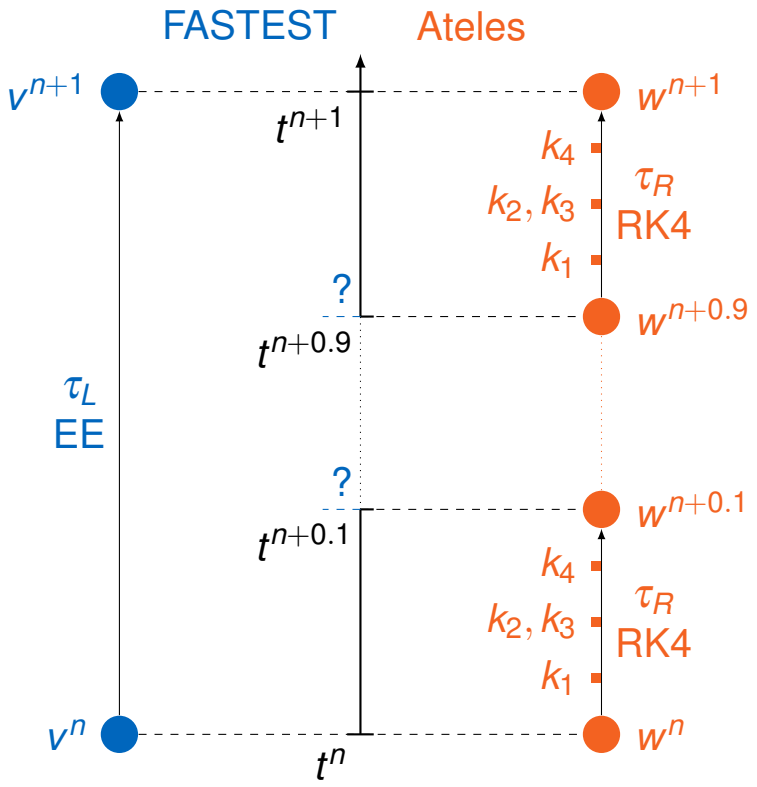

## Engineering:

- use different solvers (EE + RK4)
- use different time discretization
- no degradation of solver performance

## Informatics:

- black-box approach (nodal data)
- parallel (Exa-Scale)

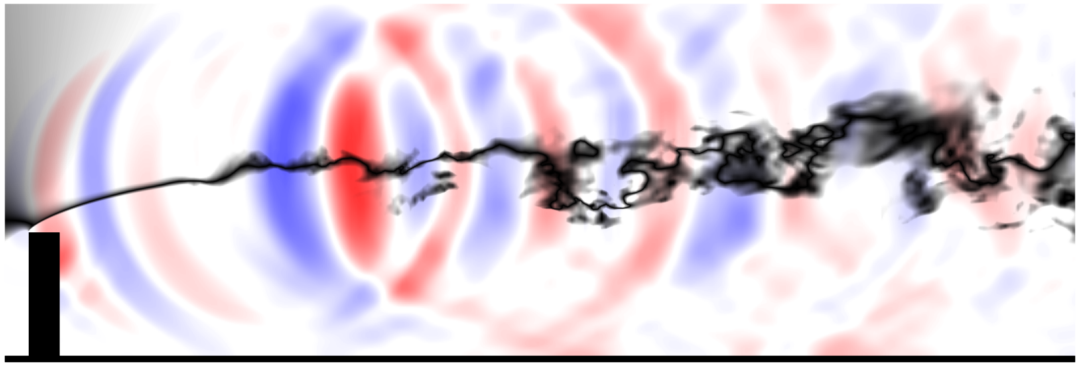
## Multi-Scale Multi-Physics

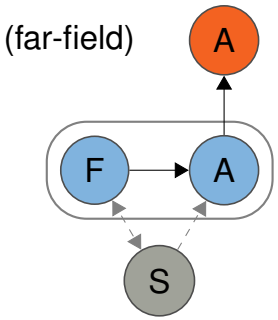



[1] see www.sppexa.de

# Partitioned approach

Model problem: 1D FSI Tube






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

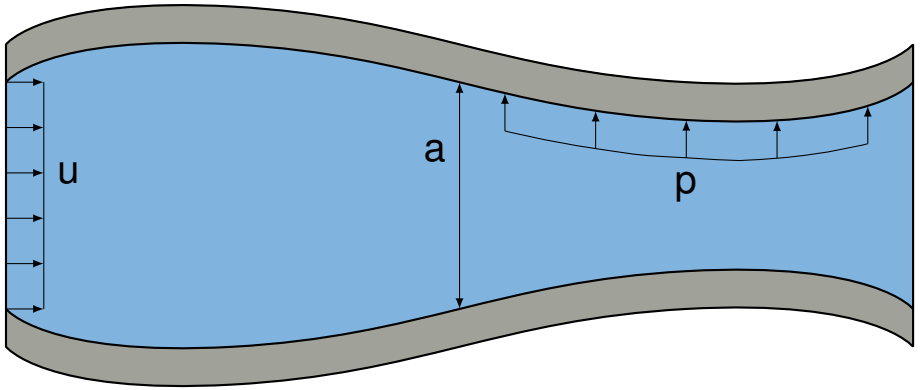
## Future: complex scenarios

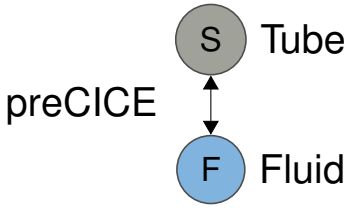
- multirate/local timestepping
- arbitrary meshes
- ND, arbitrary discretization (time & space)
- exa-scale
- preCICE for coupling

$\rightarrow$ real applications

# Partitioned approach

Model problem: 1D FSI Tube




from *Mehl, M., et al. (2016). Parallel coupling numerics for partitioned fluid-structure interaction simulations.*

## Future: complex scenarios

- multirate/local timestepping
- arbitrary meshes
- ND, arbitrary discretization (time & space)
- exa-scale
- preCICE for coupling

→ real applications

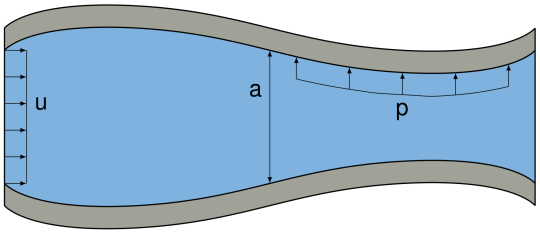
## Today: Make it simple!

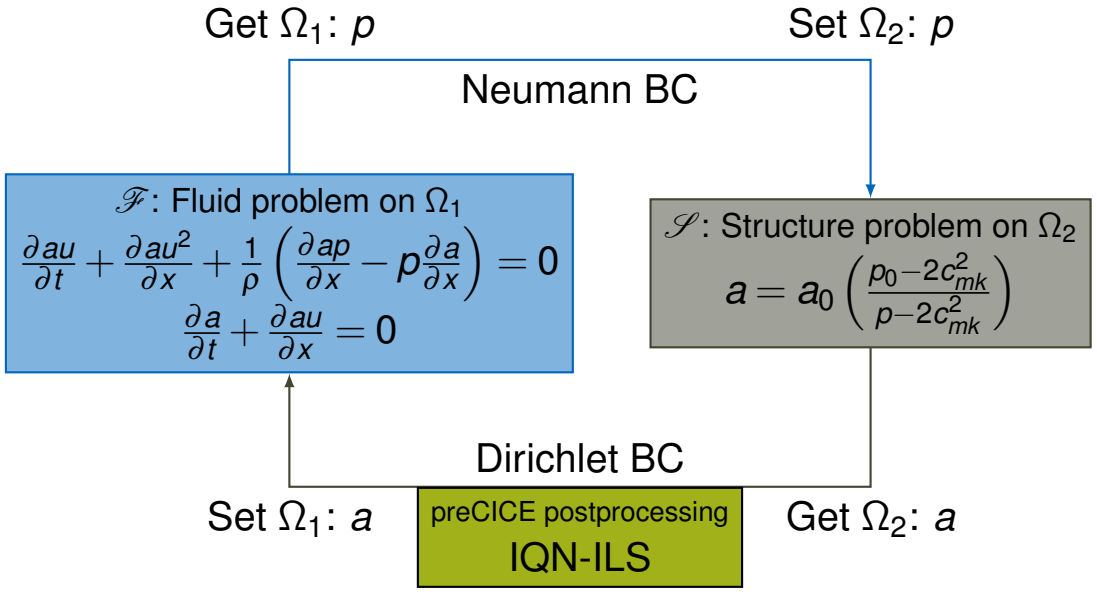
- identical timestep size
- matching meshes
- only 1D, FD, IE/TR
- small-scale run
- preCICE for coupling

→ investigate high-order coupling

# Partitioned approach

Model problem: 1D FSI Tube



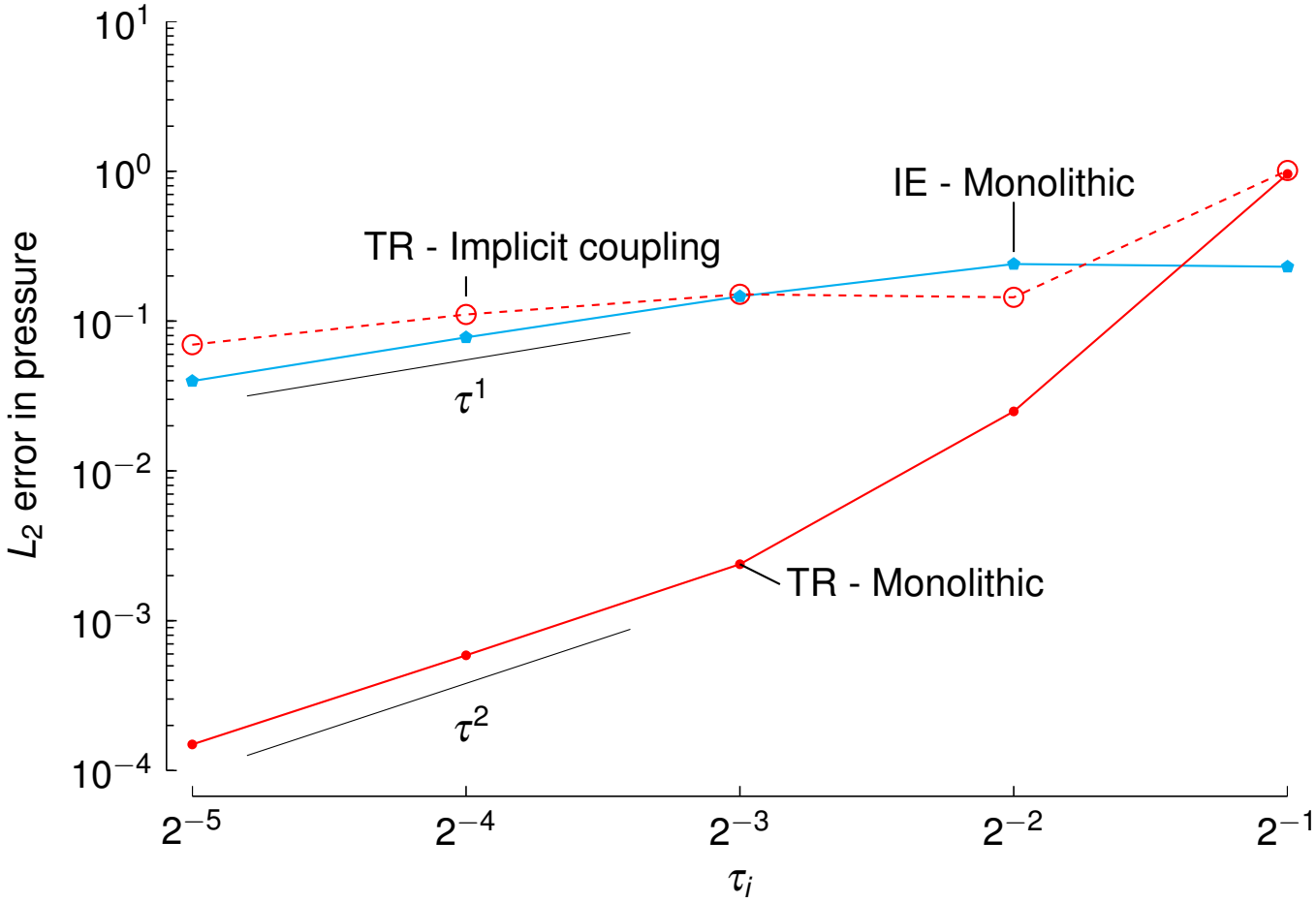




- partitioned FSI
- Dirichlet-Neumann coupling
- Fluid: $\mathscr{F}(\boldsymbol{a}) = (\boldsymbol{u}, \boldsymbol{p})$
- Structure: $\mathscr{S}(\boldsymbol{p}) = \boldsymbol{a}$
- Black-box python solvers[1]

[1] see `https://github.com/precice/elastictube1d/tree/master/PythonTube`

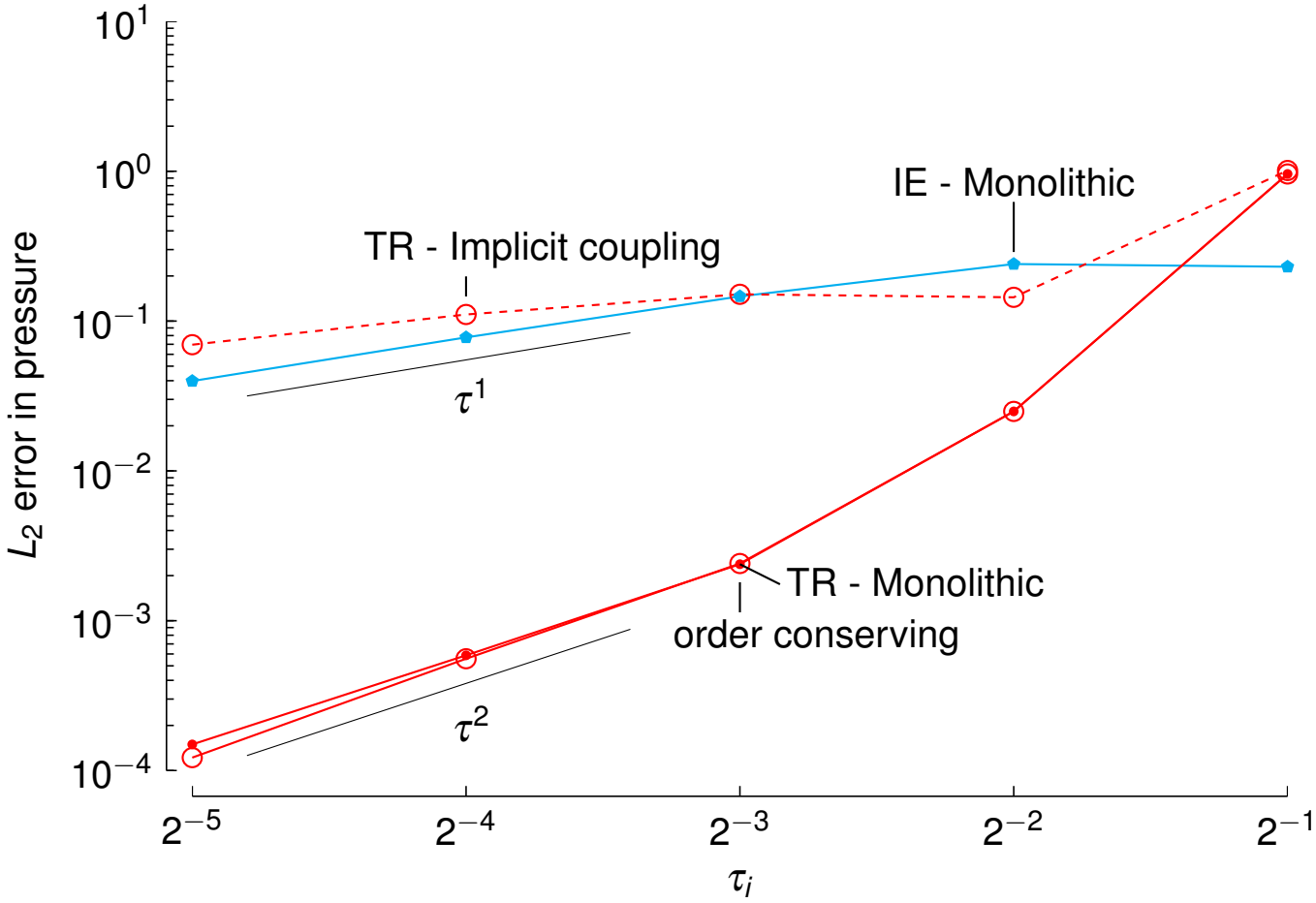
# High order FSI

Convergence study for 1D FSI Tube coupled via preCICE



# High order FSI

Convergence study for 1D FSI Tube coupled via preCICE



# High order FSI

## Order degradation

FluidSolver.py

```
import ... # fluid solver...
import PySolverInterface  # preCICE
# initialization
interface = PySolverInterface("FLUID")

while interface.isCouplingOngoing():
    v,p = fluid_step(a)
    interface.writeBlockScalarData(p)
    interface.advance()
    interface.readBlockScalarData(a)

    if interface.couplingConverged():
        # goto next timestep
    else:
        # repeat timestep
```

StructureSolver.py

```
import ... # structure solver...
import PySolverInterface  # preCICE
#initialization
interface = PySolverInterface("SOLID")

while interface.isCouplingOngoing():
    a = solid_step(p)
    interface.writeBlockScalarData(a)
    interface.advance()
    interface.readBlockScalarData(p)

    if interface.couplingConverged():
        # goto next timestep
    else:
        # repeat timestep
```

# High order FSI

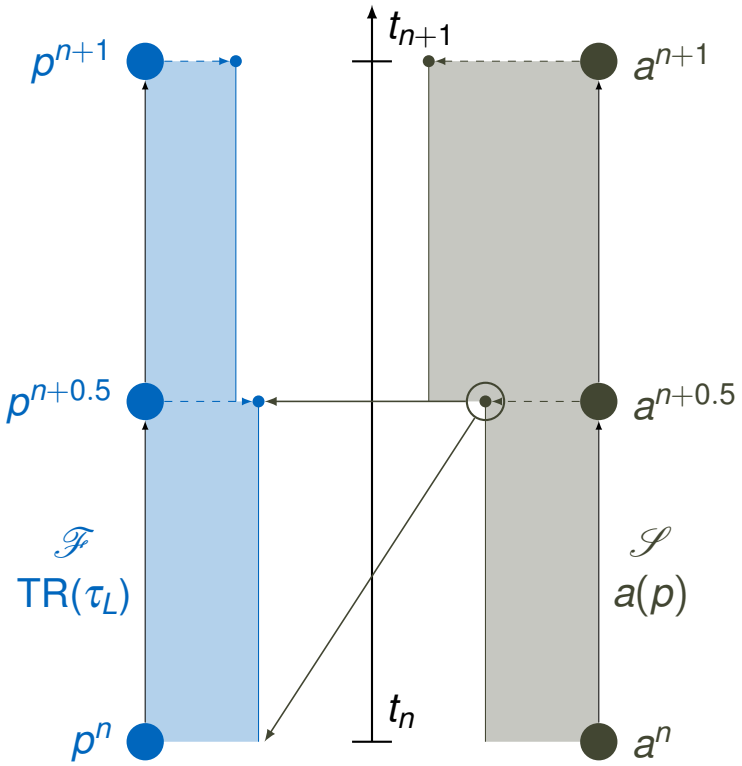
## Order degradation

FluidSolver.py

```
import ... # fluid solver...
import PySolverInterface  # preCICE
# initialization
interface = PySolverInterface("FLUID")

while interface.isCouplingOngoing():
    v,p = fluid_step(a)
    interface.writeBlockScalarData(p)
    interface.advance()
    interface.readBlockScalarData(a)

    if interface.couplingConverged():
        # goto next timestep
    else:
        # repeat timestep
```




Implicit/strong coupling

$$TR(\tau): u^{n+1} = u^n + \frac{1}{2}\left(f(u^n) + f(u^{n+1})\right)$$

# High order FSI

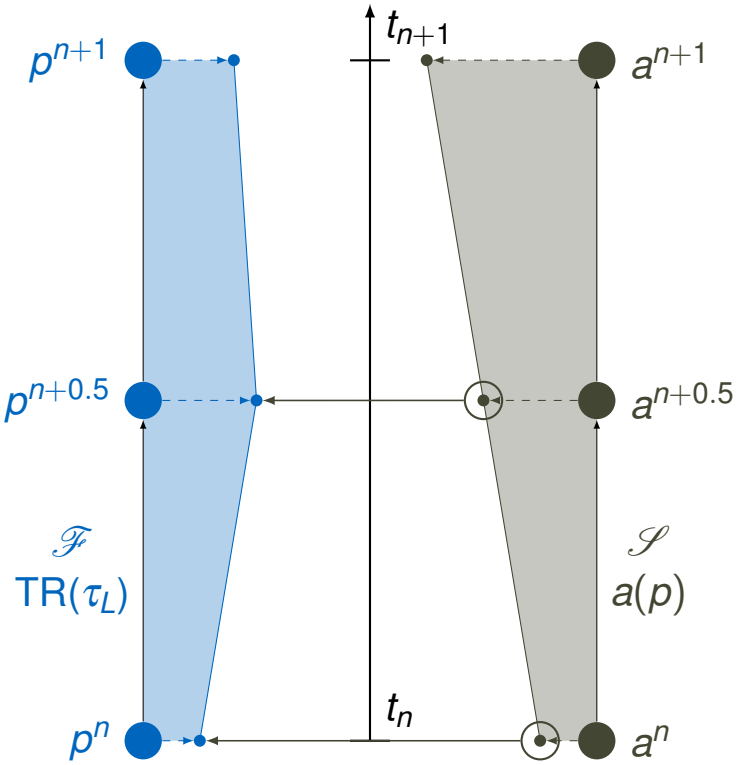
How to fix it

FluidSolver.py

```
# initialization like before
a_new = np.copy(a_old)

while interface.isCouplingOngoing():
    a = linear_interp(a_old, a_new, dt)
    v,p = fluid_step(a)
    interface.writeBlockScalarData(p)
    interface.advance(dt)
    interface.readBlockScalarData(a_new)

    if interface.couplingConverged():
        # goto next timestep
        a_old = np.copy(a_new)
    else:
        # repeat timestep
```




Improved coupling

$$TR(\tau): u^{n+1} = u^n + \frac{1}{2}\left(f(u^n) + f(u^{n+1})\right)$$

# High order FSI

Generalization: Waveform relaxation

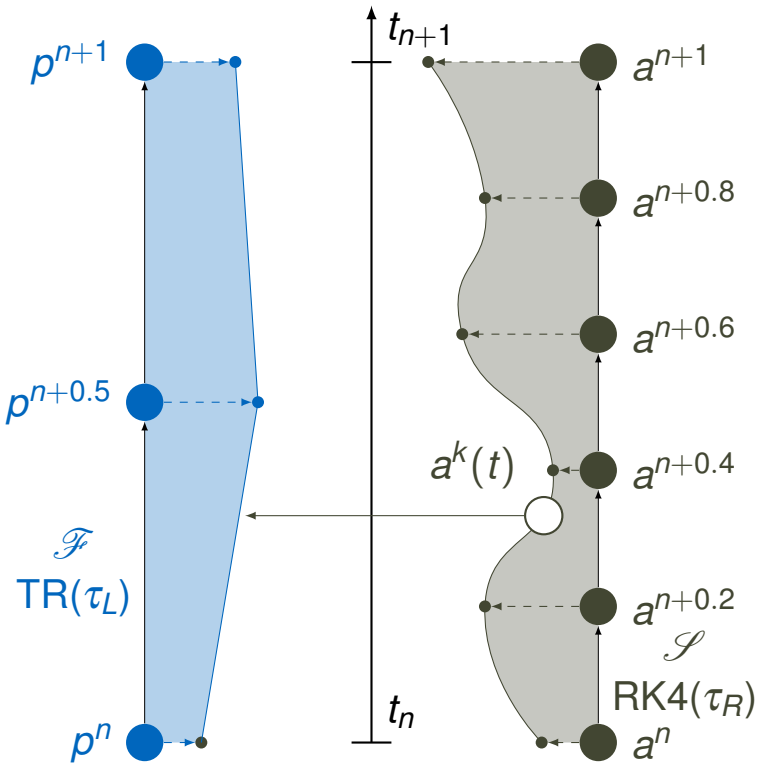
## Algorithm[1]

We want to solve the coupled problem

$$\mathscr{F}(p,u,a) = p,\ \mathscr{S}(p) = a.$$

with $p, u, a$ known for $t < t_n$ on the window $T_n = [t_n, t_{n+1}]$.

1. set $k = 0$ and extrapolate $a^0(t) = a_n$ and $p^0(t) = p_n$ for $t \in T$
2. solve decoupled $\mathscr{F}, \mathscr{S}$ using $a^k(t), p^k(t)$ to obtain $p^{k+1}, a^{k+1}$ for $t \in T$
3. if not converged:
   a. set $k = k+1$ and go to step 2,
   b. otherwise proceed to next window $T_{n+1}$




Waveform Relaxation

[1] Adapted from *Schöps, S., et al. (2017). Application of the Waveform Relaxation Technique to the Co-Simulation of Power Converter Controller and Electrical Circuit Models. https://doi.org/10.1109/MMAR.2017.8046937*

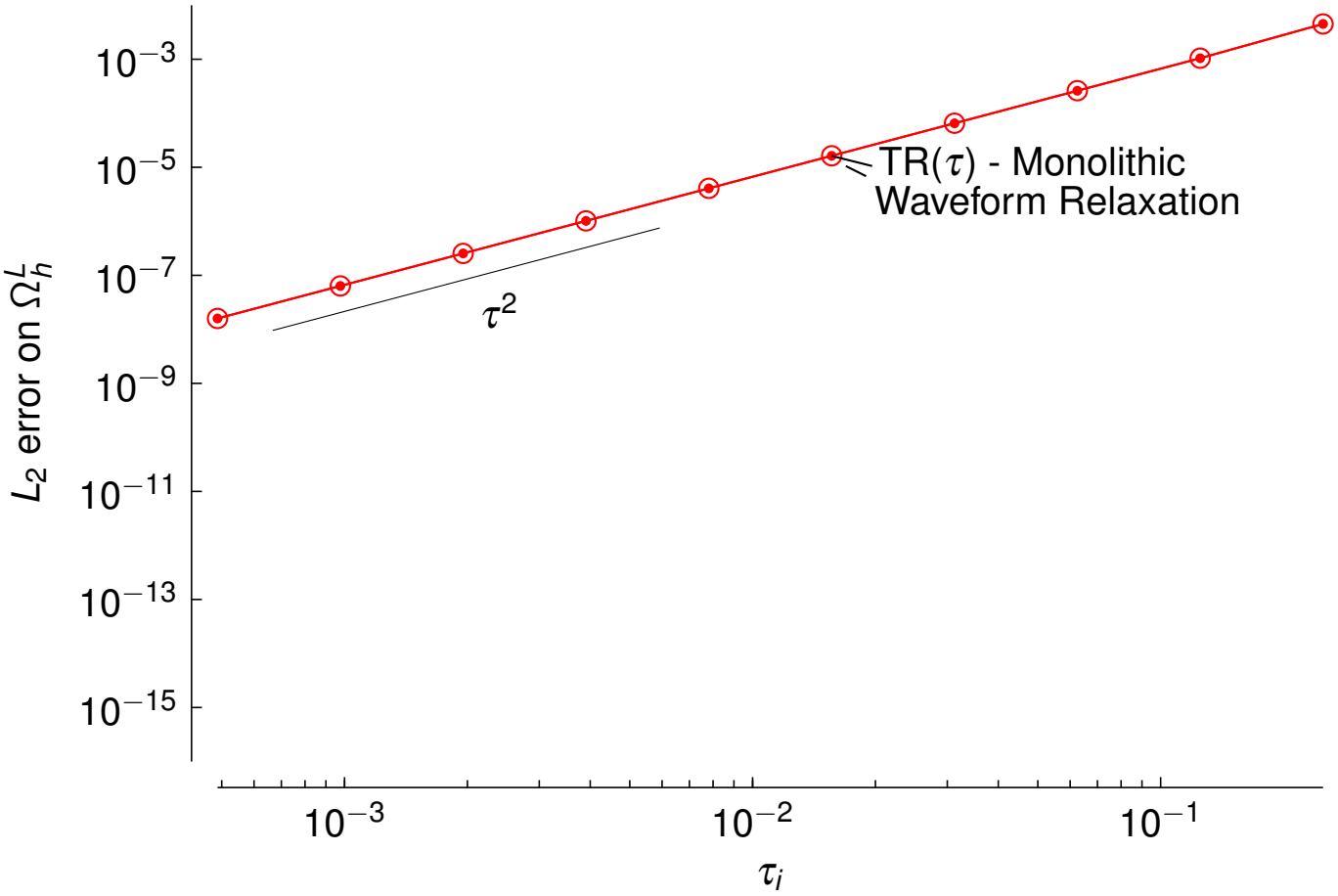
# 4th order Heat Transport

Convergence study with partitioned heat transport equation[1]

[1] see Rüth, B., et al. (2018). Time Stepping Algorithms for Partitioned Multi-Scale Multi-Physics.

# 4th order Heat Transport

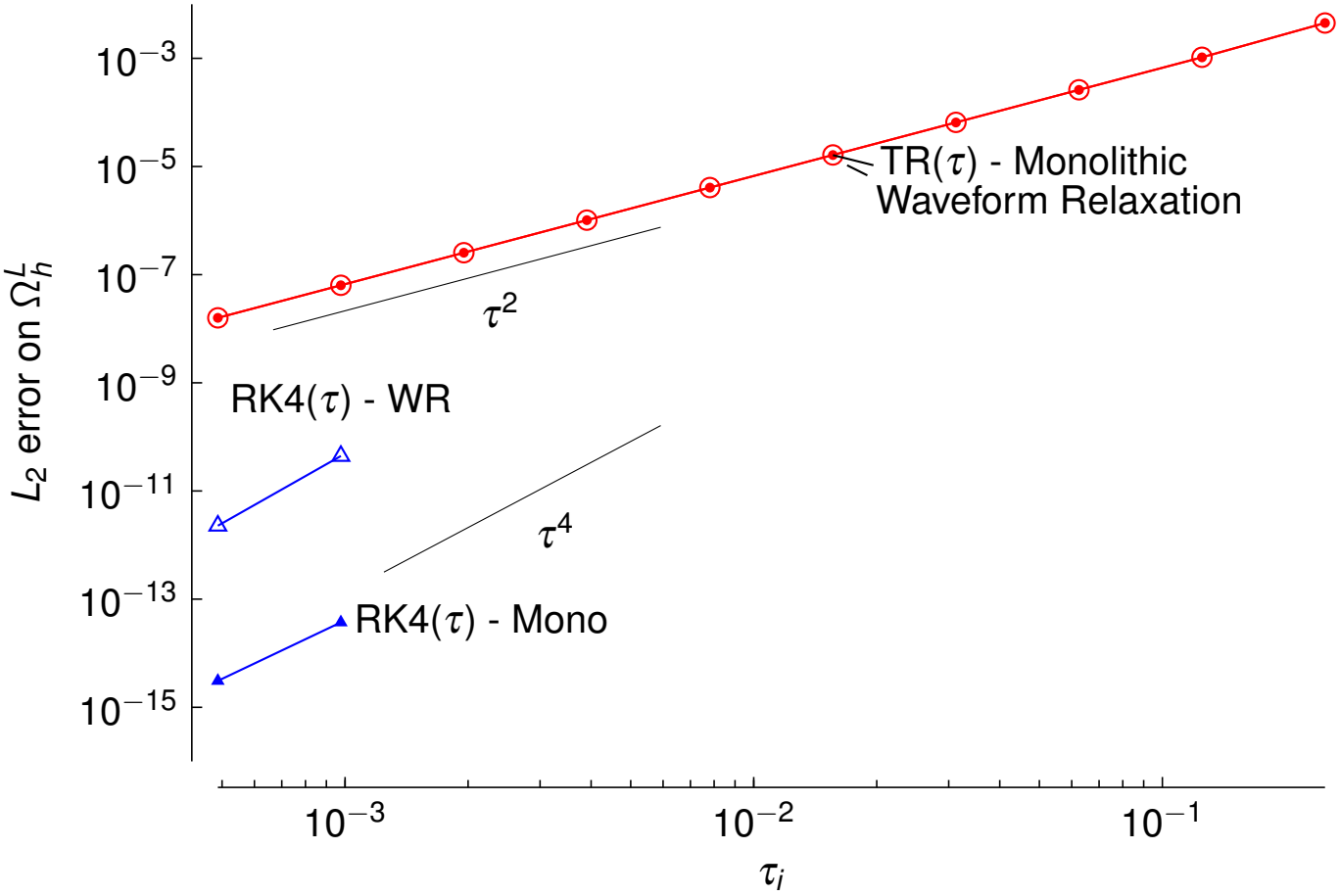
Convergence study with partitioned heat transport equation[1]




[1] see Rüth, B., et al. (2018). Time Stepping Algorithms for Partitioned Multi-Scale Multi-Physics.

# 4th order Heat Transport

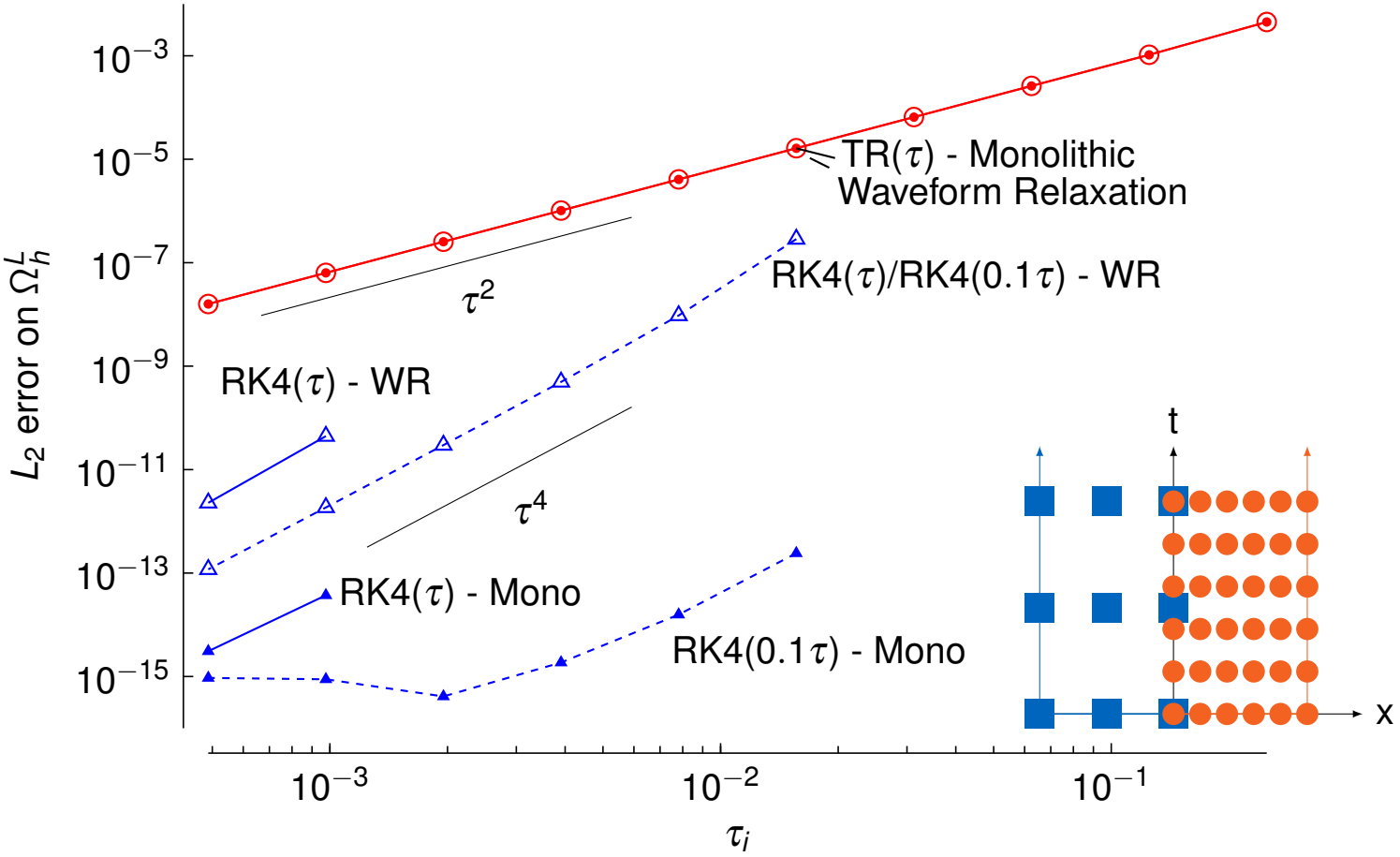
Convergence study with partitioned heat transport equation[1]




[1] see Rüth, B., et al. (2018). Time Stepping Algorithms for Partitioned Multi-Scale Multi-Physics.

# 4th order Heat Transport

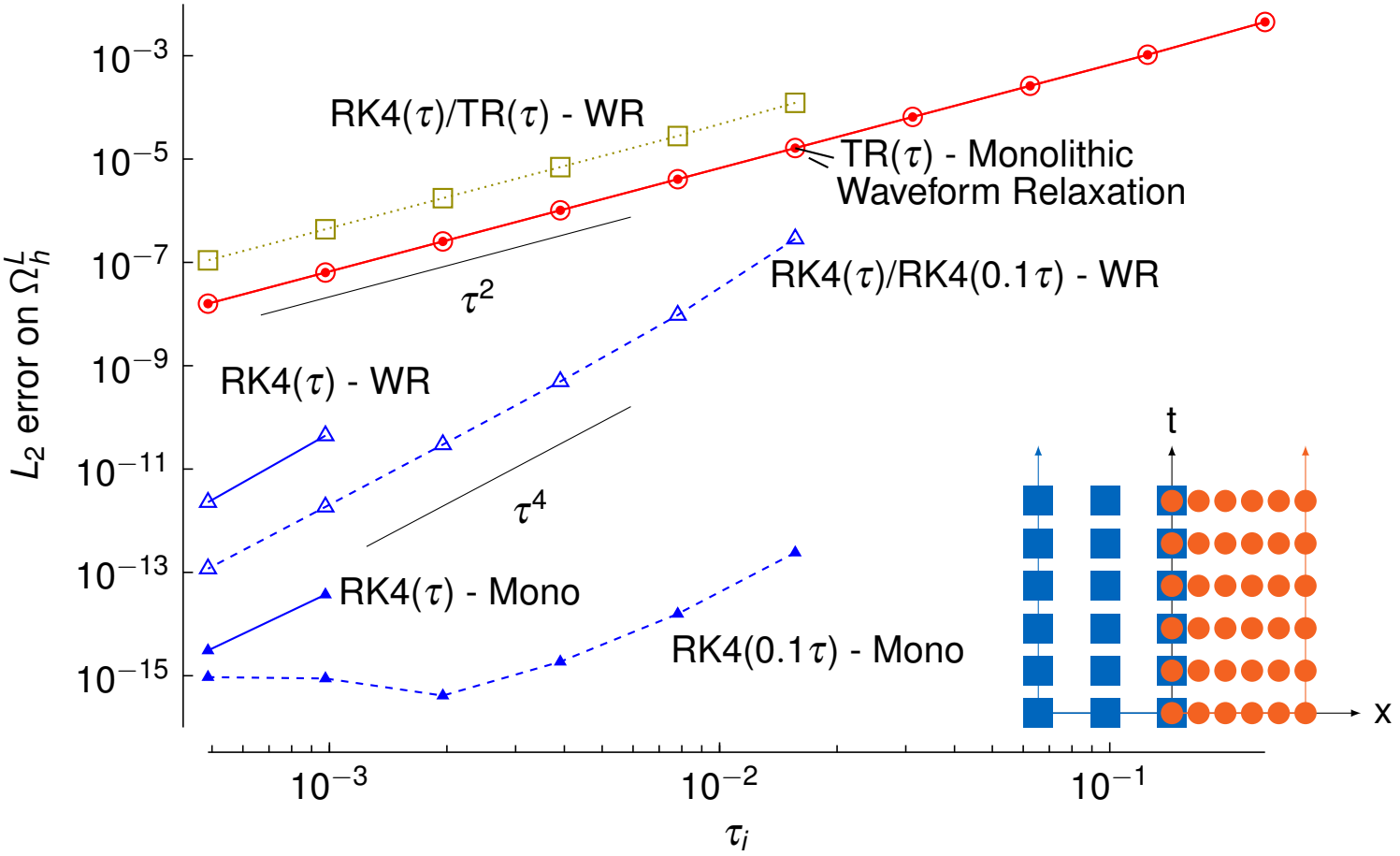
Convergence study with partitioned heat transport equation[1]




[1] see Rüth, B., et al. (2018). Time Stepping Algorithms for Partitioned Multi-Scale Multi-Physics.

# 4th order Heat Transport

Convergence study with partitioned heat transport equation[1]




[1] see Rüth, B., et al. (2018). Time Stepping Algorithms for Partitioned Multi-Scale Multi-Physics.

# Final words

## Conclusion

- Order degradation occurs in FSI
- Can be fixed by communication of the "correct" data
- Waveform relaxation generalizes this idea

## Outlook

- High order interpolation schemes? (Dense output, ADER, ...)
- Black-box?
- Interpolation inside preCICE?
- 2D, 3D, FSI, FSA?
- Quasi Newton (i.e. IQN-ILS) + WR?

# Thank you!

Website: `precice.org`
Source/Wiki: `github.com/precice`

Mailing list: `precice.org/resources`
My e-mail: `rueth@in.tum.de`

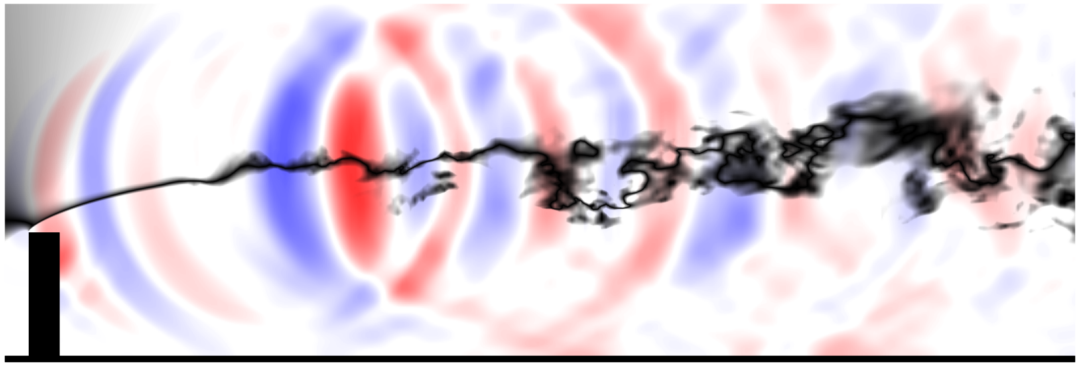
Homework:

- Follow a tutorial
- Join our mailing list
- Star on GitHub
- Send us feedback
- Ask me for stickers

preCICE

# Appendix

ExaFSA - Exascale Simulation of Fluid-Structure-Acoustics Interactions




Fluid-acoustics simulation and partitioned setup[1].

| physics | timescale | solver | scheme | order |
|---|---|---|---|---|
| (A) | small | Ateles | RK | 2 or 4 |
| (A) | small | FASTEST | LW | 2 |
| (F) | medium | FASTEST | CN | 2 |
| (S) | large | FEAP | HHT[2] | 2 |

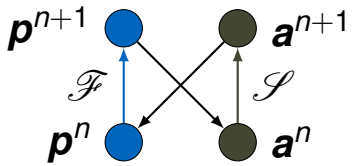
[1] Reimann, T., et al. (2017). Aspects of FSI with aeroacoustics in turbulent flow. In 7th GACM Colloquium on Computational Mechanics.

[2] Hilber-Hughes-Taylor Method

# Appendix

## Coupling Details

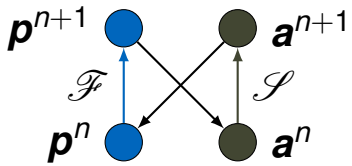
| | update scheme | order |
|---|---|---|
| currently preCICE | $\boldsymbol{p}^{n+1} = \boldsymbol{p}^{n} + \frac{\tau}{2}\left[\mathscr{F}\left(\boldsymbol{p}^{n} \quad, t_n\right) + \mathscr{F}\left(\boldsymbol{p}^{n+1} \quad, t_{n+1}\right)\right]$<br>$\boldsymbol{a}^{n+1} = \boldsymbol{a}^{n} + \frac{\tau}{2}\left[\mathscr{S}\left(\quad \boldsymbol{a}^{n}, t_n\right) + \mathscr{S}\left(\quad \boldsymbol{a}^{n+1}, t_{n+1}\right)\right]$ | $\mathscr{O}(\tau)$ |



# Appendix

## Coupling Details

| | update scheme | order |
|---|---|---|
| currently preCICE | $\boldsymbol{p}^{n+1} = \boldsymbol{p}^{n} + \frac{\tau}{2}\left[\mathscr{F}\left(\boldsymbol{p}^{n}, \boldsymbol{a}^{n+1}, t_n\right) + \mathscr{F}\left(\boldsymbol{p}^{n+1}, \boldsymbol{a}^{n+1}, t_{n+1}\right)\right]$<br>$\boldsymbol{a}^{n+1} = \boldsymbol{a}^{n} + \frac{\tau}{2}\left[\mathscr{S}\left(\boldsymbol{p}^{n+1}, \boldsymbol{a}^{n}, t_n\right) + \mathscr{S}\left(\boldsymbol{p}^{n+1}, \boldsymbol{a}^{n+1}, t_{n+1}\right)\right]$ | $\mathscr{O}(\tau)$ |



# Appendix

## Coupling Details

| | update scheme | order |
|---|---|---|
| currently preCICE | $\boldsymbol{p}^{n+1} = \boldsymbol{p}^{n} + \frac{\tau}{2}\left[\mathscr{F}\left(\boldsymbol{p}^{n}, \boldsymbol{a}^{n+1}, t_n\right) + \mathscr{F}\left(\boldsymbol{p}^{n+1}, \boldsymbol{a}^{n+1}, t_{n+1}\right)\right]$<br>$\boldsymbol{a}^{n+1} = \boldsymbol{a}^{n} + \frac{\tau}{2}\left[\mathscr{S}\left(\boldsymbol{p}^{n+1}, \boldsymbol{a}^{n}, t_n\right) + \mathscr{S}\left(\boldsymbol{p}^{n+1}, \boldsymbol{a}^{n+1}, t_{n+1}\right)\right]$ | $\mathscr{O}(\tau)$ |
| fixed version | $\boldsymbol{p}^{n+1} = \boldsymbol{p}^{n} + \frac{\tau}{2}\left[\mathscr{F}\left(\boldsymbol{p}^{n}, \boldsymbol{a}^{n}, t_n\right) + \mathscr{F}\left(\boldsymbol{p}^{n+1}, \boldsymbol{a}^{n+1}, t_{n+1}\right)\right]$<br>$\boldsymbol{a}^{n+1} = \boldsymbol{a}^{n} + \frac{\tau}{2}\left[\mathscr{S}\left(\boldsymbol{p}^{n}, \boldsymbol{a}^{n}, t_n\right) + \mathscr{S}\left(\boldsymbol{p}^{n+1}, \boldsymbol{a}^{n+1}, t_{n+1}\right)\right]$ | $\mathscr{O}\left(\tau^2\right)$ |