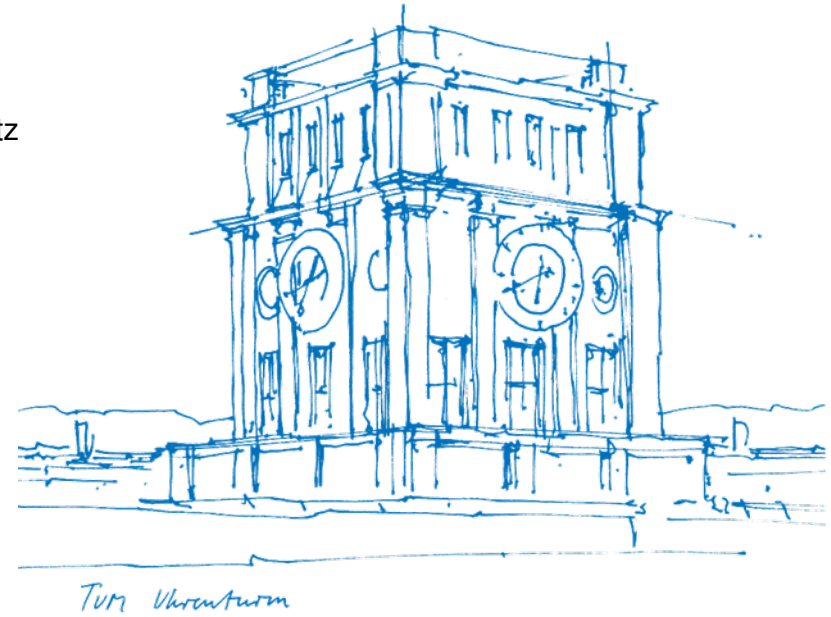
A black-box coupling scheme for higher-order multirate time stepping with preCICE

Benjamin Rodenberg, Benjamin Uekermann, Hans-Joachim Bungartz

COUPLED 2023, June 7, 2023

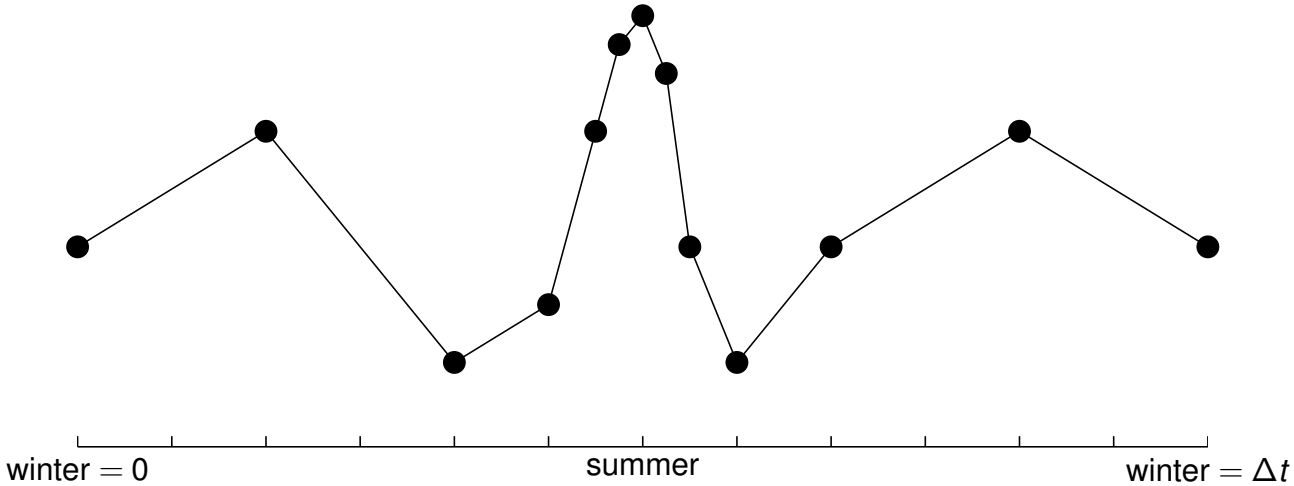


Funding by University of Stuttgart is thankfully acknowledged

Why time interpolation and subcycling?

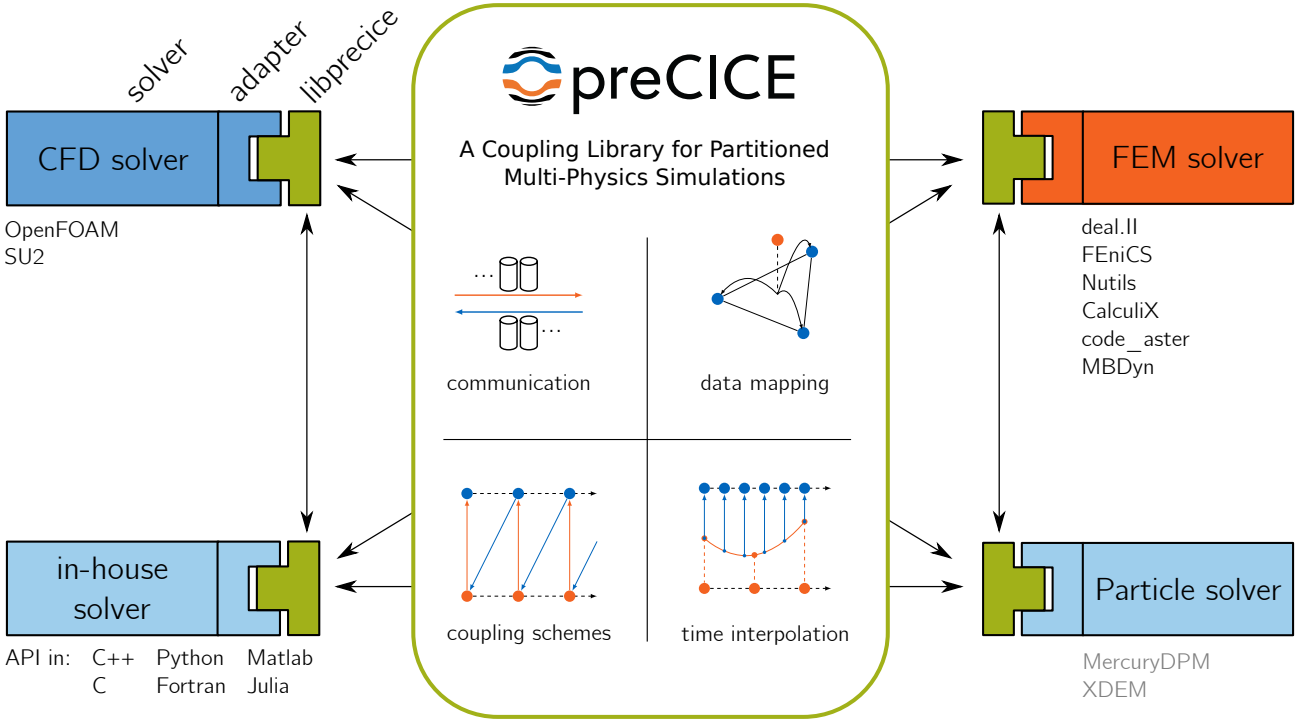
preCICE Workshop 2023: Two "ice sheet talks"[1]

- Yannic Fischler: *A preCICE-interface for the ice-sheet and sea-level system model*
- Daniel Abele: *Coupling an ice sheet model with satellite image based simulation of calving fronts*

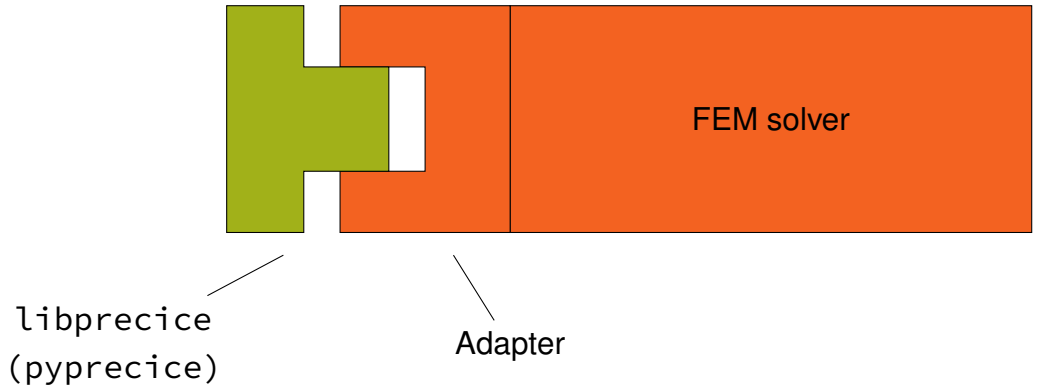


[1] https://precice.org/precice-workshop-2023.html

What is preCICE?



What is important?



A black-box coupling scheme for higher-order multirate time stepping with preCICE

- Numerics: Accuracy, convergence order...
- API Design: Simple user interface!
- Move as much multirate/higher-order/interpolation logic *inside* preCICE as possible
- preCICE v2.5 is the current version, preCICE v3 is planned and will introduce the new API

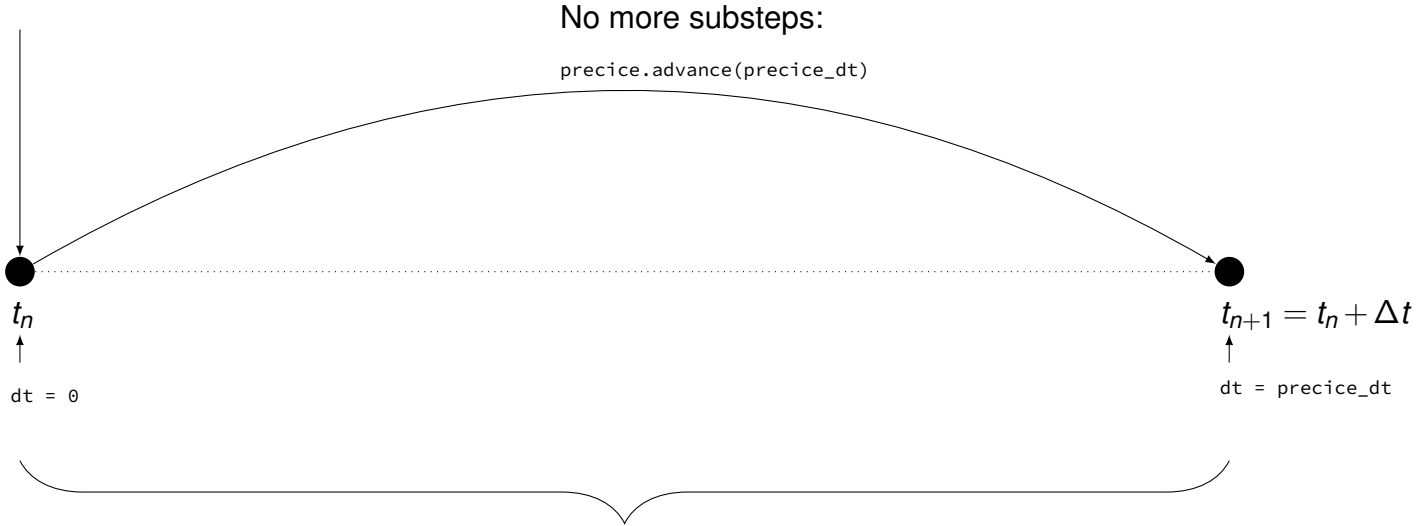
API for simple time stepping

```python
# example: FSI coupling, perspective of fluid solver 𝓕:
participant = precice.Participant("Fluid", "precice-config.xml")

# leaving out coupling mesh and data initialization

participant.initialize()

while participant.is_coupling_ongoing():
  # store checkpoint, if needed
  dt = participant.get_max_time_step_size()
  displacement = participant.read_data(dt)  # Read displacement at t_n + Δt: d_{n+1}
  forces = forces + dt * dfdt(displacements)  # Implicit Euler
  participant.write_data(forces)
  participant.advance(dt)
  # read checkpoint, if needed

participant.finalize()
```

API for multirate

Read interpolated data from current time t_n:

```
displacement = precice.read_data(dt)
```



API for multirate

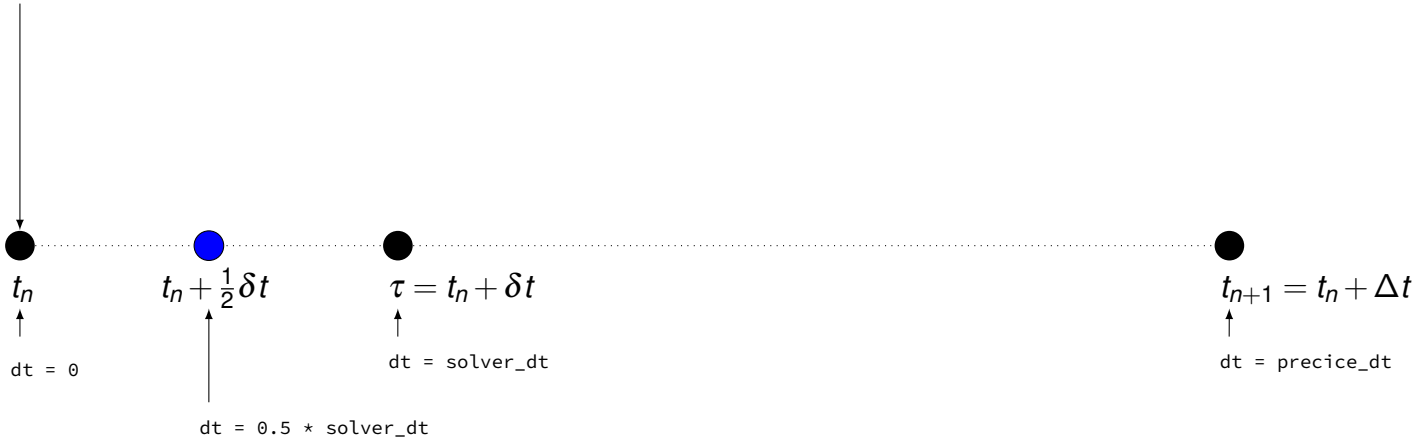
Read interpolated data from current time t_n:

```
displacement = precice.read_data(dt)
```

API for multirate

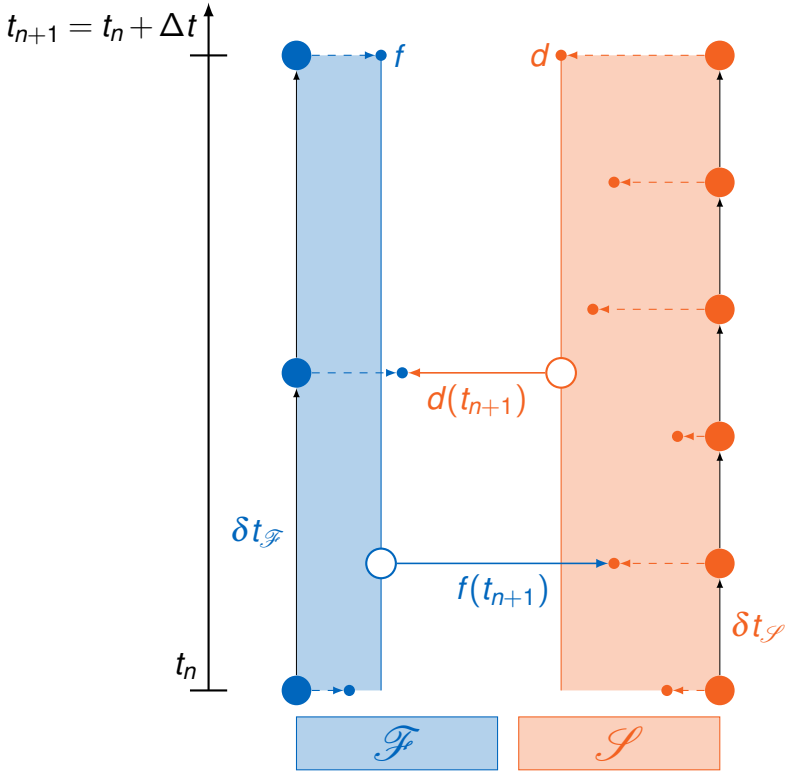
Read interpolated data from current time t_n:

```
displacement = precice.read_data(dt)
```



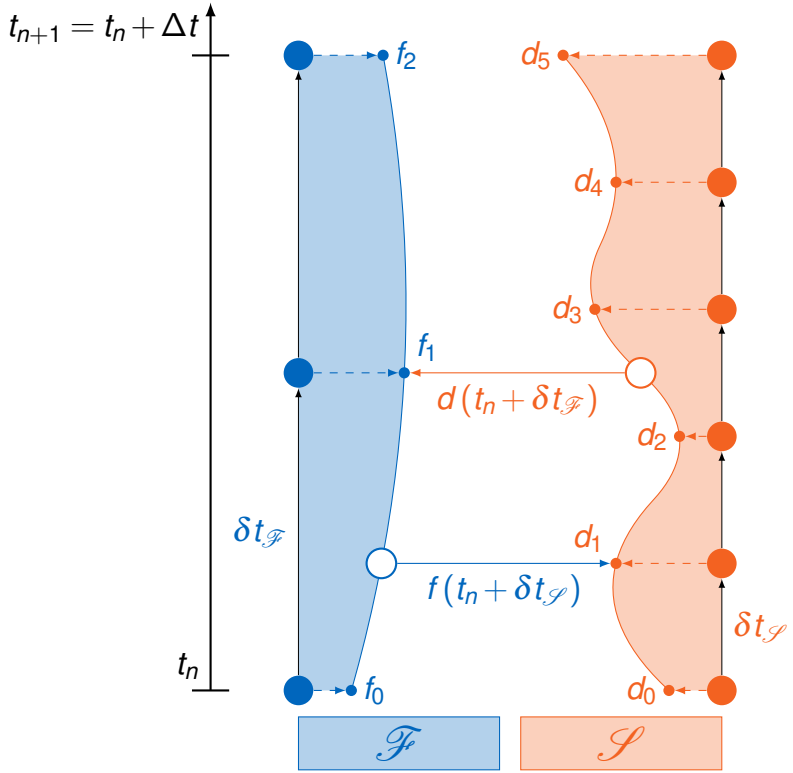
preCICE v2: Constant in time window

```
<participant name="Fluid">
  <write-data name="Force" />
  <read-data  name="Displacement" />
</participant>

<participant name="Solid">
  <write-data name="Displacement" />
  <read-data  name="Force" />
</participant>
```



preCICE v3: Interpolation & waveform iteration

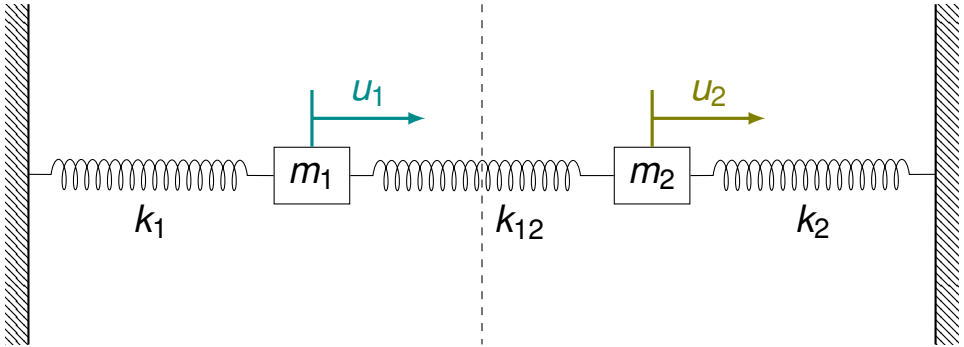
```
<participant name="Fluid">
  <write-data name="Force" />
  <read-data  name="Displacement"
   ↪  waveform-order="3" />
</participant>

<participant name="Solid">
  <write-data name="Displacement" />
  <read-data  name="Force"
   ↪  waveform-order="3" />
</participant>
```



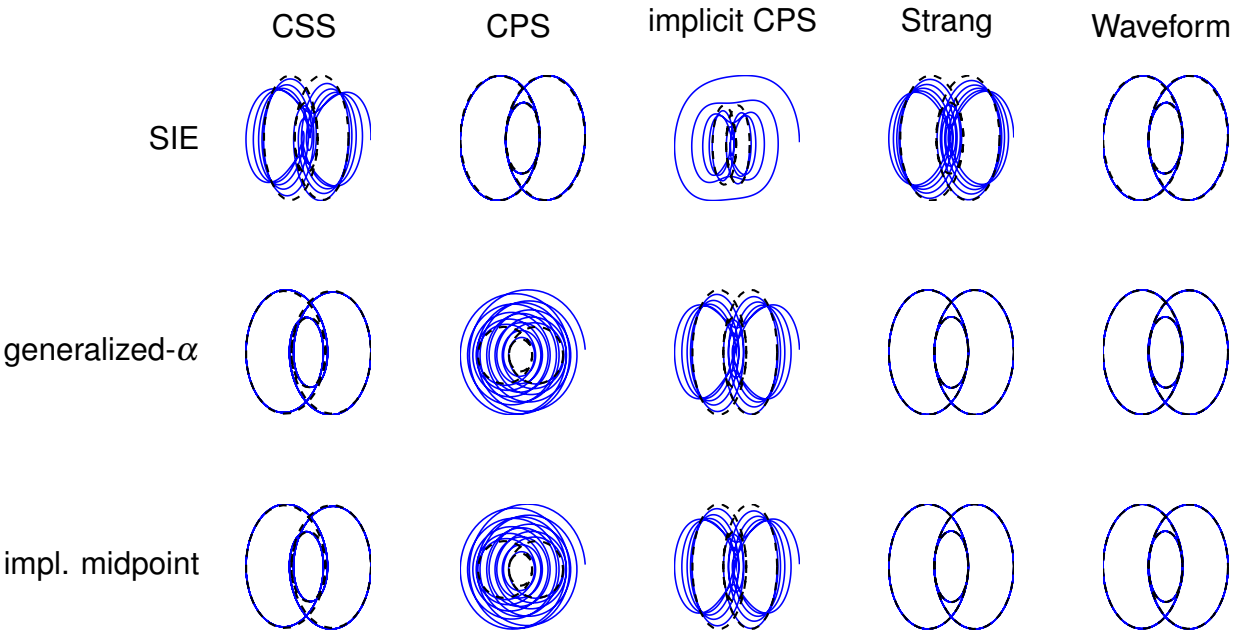
API for higher-order multirate time stepping

```python
# example: FSI coupling, perspective of fluid solver 𝓕:
# ...
participant.initialize()

while participant.is_coupling_ongoing():
  # store checkpoint, if needed
  precice_dt = participant.get_max_time_step_size()  # until end of window
  solver_dt = time_stepper.get_max_dt()  # stability, adaptivity
  dt = np.min([precice_dt, solver_dt])  # actual time step size δt
  # time_stepper represents s-stage RK scheme
  ts = time_stepper.rhs_eval_points(dt)  # t_n + c_1, t_n + c_2 ..., t_n + c_s
  displacements = [participant.read_data(t) for t in ts]  # d(t_n + c_1), d(t_n + c_2) ..., d(t_n + c_s)
  forces = time_stepper.do_step(displacements, dt)  # f_{n+1} = f_n + δt Σ_{i=1}^{s} b_i k_i
  participant.write_data(forces)
  participant.advance(dt)
  # read checkpoint, if needed

participant.finalize()
```

Proof of concept: Test case

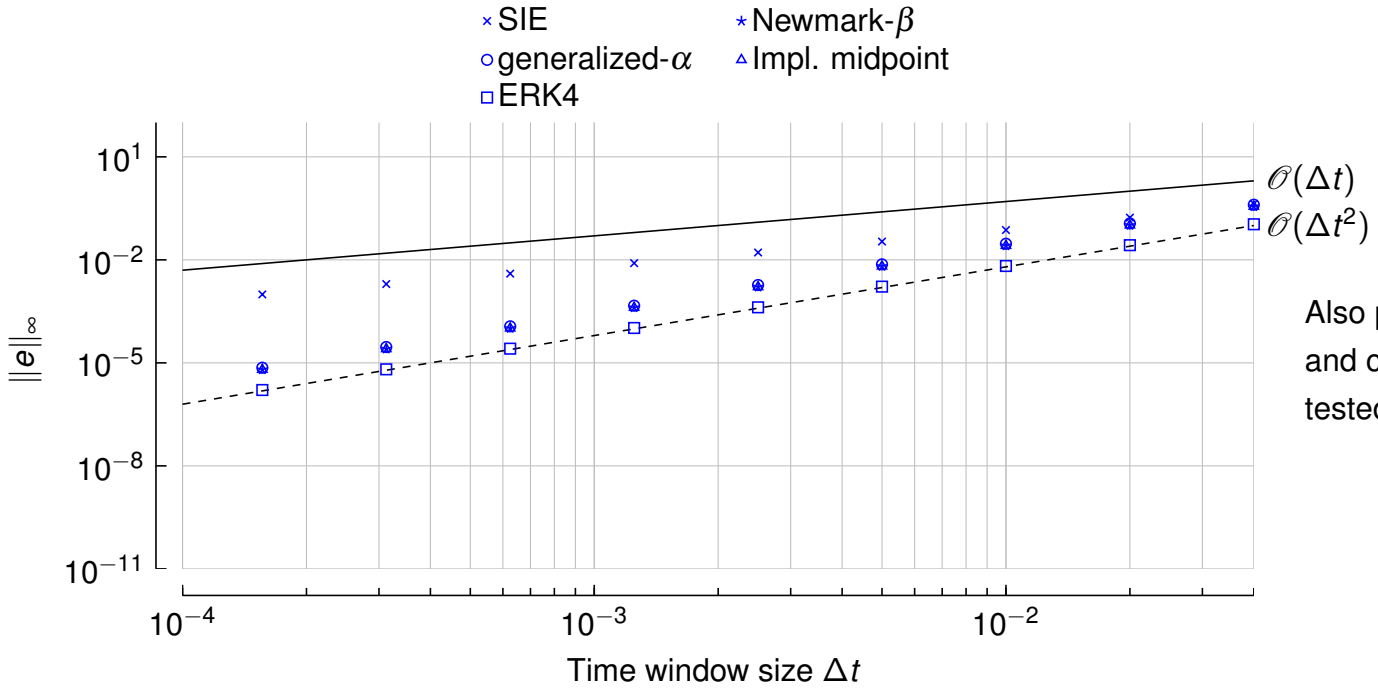


Proof of concept: Energy conservation



A Simple Test Case for Convergence Order in Time and Energy Conservation of Black-Box Coupling Schemes. 2022. `https://doi.org/10.23967/wccm-apcom.2022.038`

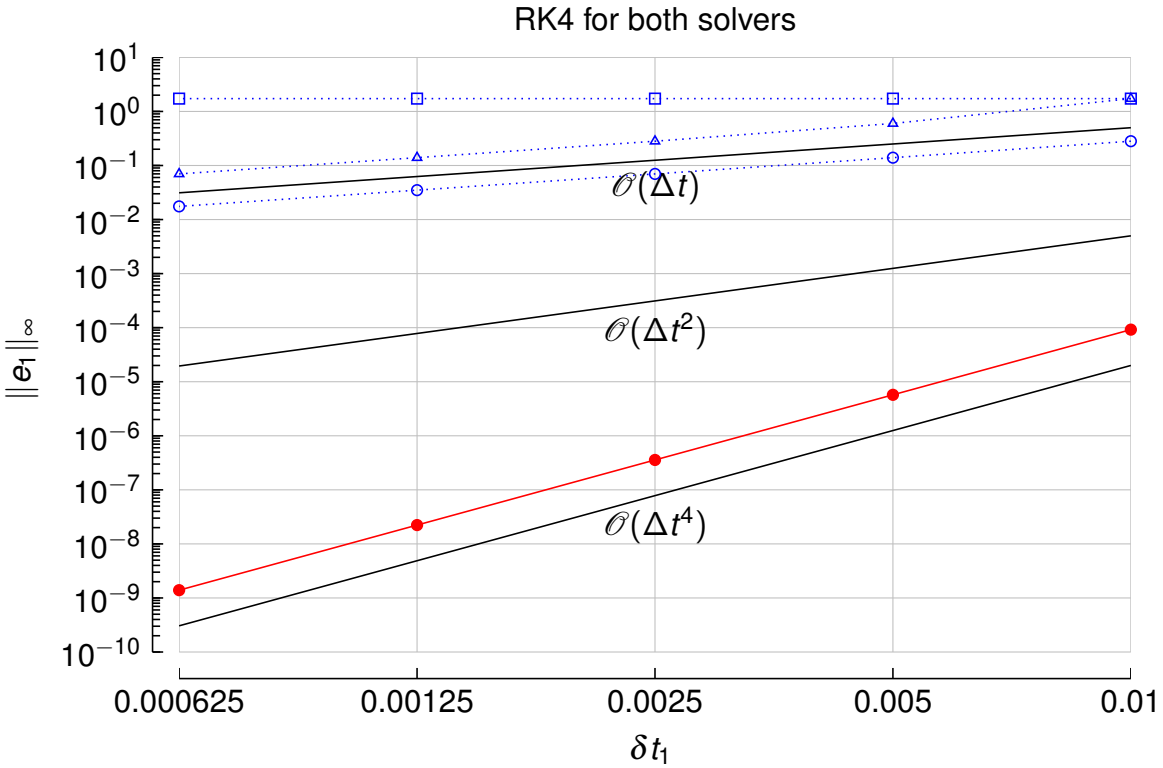
Proof of concept: Order conservation



Also piecewise linear and cubic interpolation tested in other publication.

A Simple Test Case for Convergence Order in Time and Energy Conservation of Black-Box Coupling Schemes. 2022. `https://doi.org/10.23967/wccm-apcom.2022.038`

Experiments with preCICE v3 (development version)

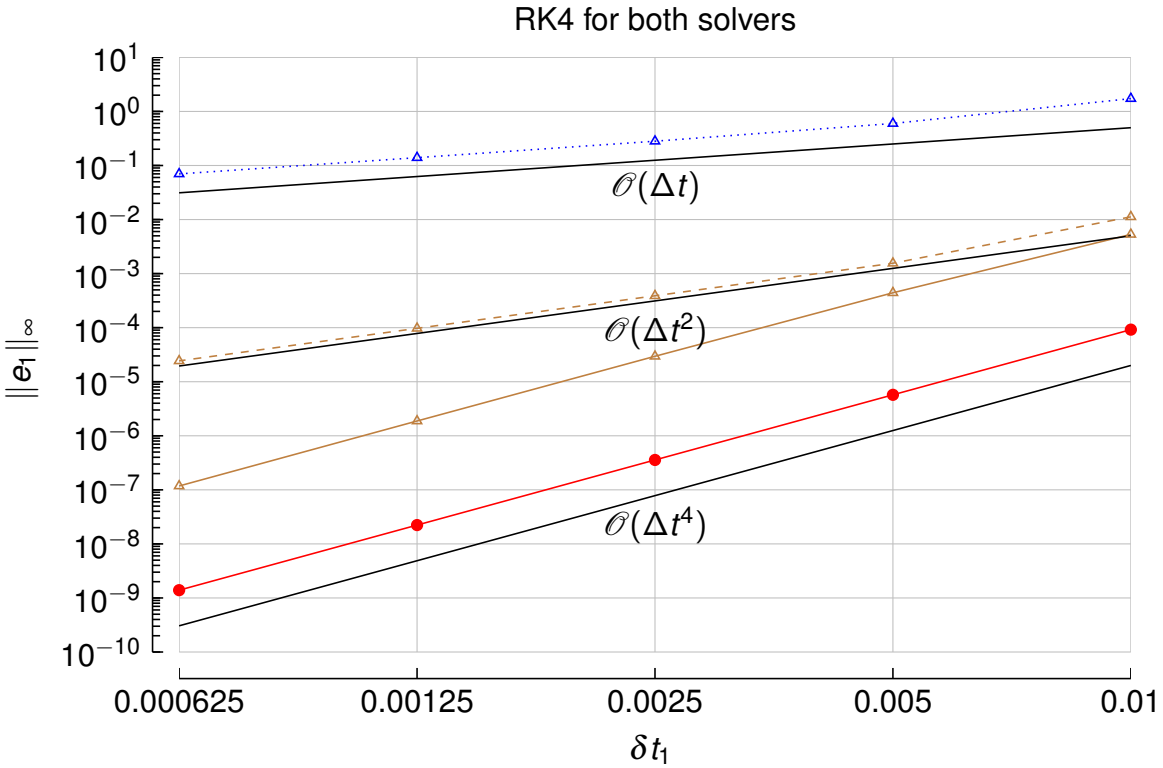


Summary:

- Decrease step size δt
- Convergence study w.r.t. δt

- monolithic, $\Delta T = \Delta t_{1,2}$
- v2.5.0, $\Delta T = \Delta t_{1,2}$
- v2.5.0, $\Delta T = 4\Delta t_{1,2}$
- v2.5.0, $\Delta T = 0.04$

Experiments with preCICE v3 (development version)

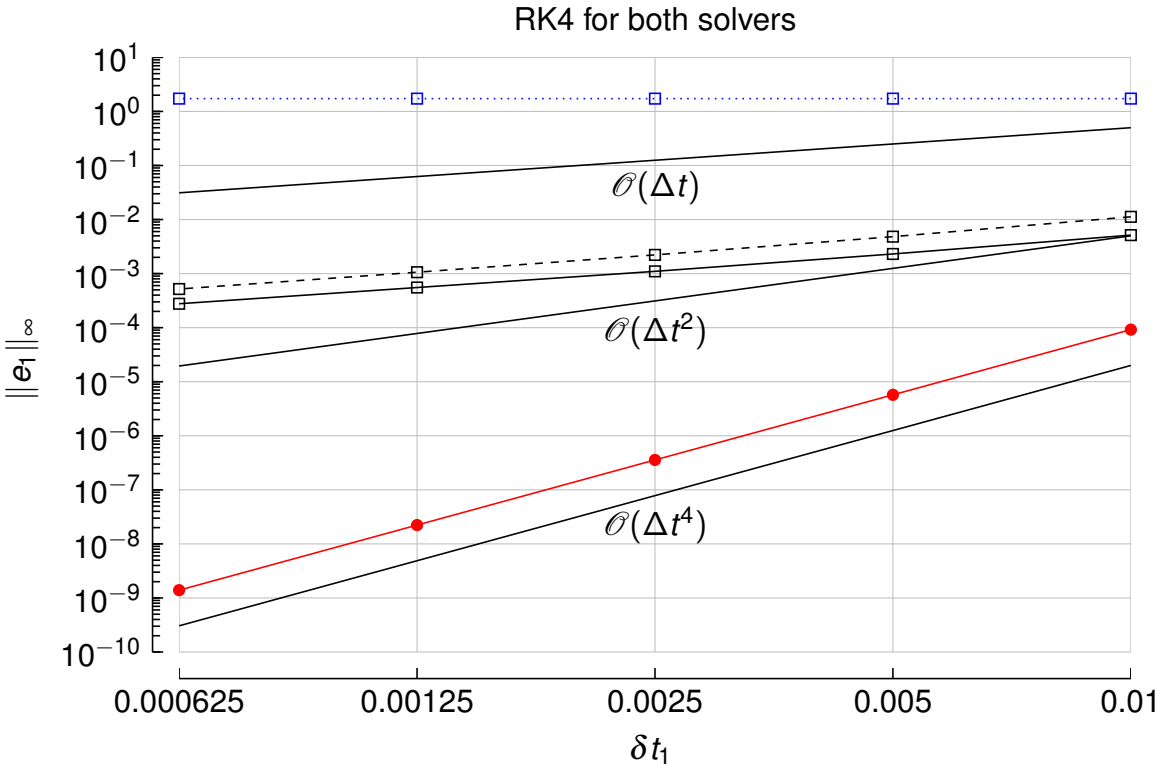


Summary:

- Decrease step size δt
- Convergence study w.r.t. δt
- Gap to monolithic? Wrong convergence measurement?

- monolithic, $\Delta T = \Delta t_{1,2}$
- `v2.5.0`, $\Delta T = 4\Delta t_{1,2}$
- `v3.0.0`, $\Delta T = 4\Delta t_{1,2}$, linear B-Spline
- `v3.0.0`, $\Delta T = 4\Delta t_{1,2}$, cubic B-Spline

Experiments with preCICE v3 (development version)



Summary:

- Decrease step size δt
- Convergence study w.r.t. δt
- Gap to monolithic? Wrong convergence measurement?
- Does higher-order interpolation work?
- $\Delta t >> \delta t$?

monolithic, $\Delta T = \Delta t_{1,2}$
`v2.5.0`, $\Delta T = 0.04$
`v3.0.0`, $\Delta T = 0.04$, linear B-Spline
`v3.0.0`, $\Delta T = 0.04$, $p = 10$ B-Spline

Conclusion

State of development

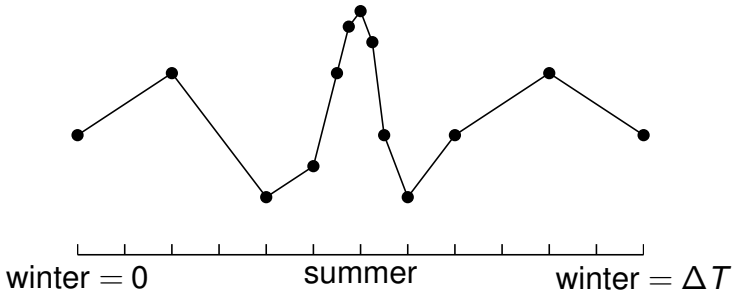
- Achievements: 4th order multirate time-stepping + black-box + usable API
- Goal: Finalize implementation of all this for preCICE v3. (we are very close)
- If you want to try experimental version now: Talk to me!
- Restriction: High order interpolation needs subcycling

Where to be careful:

- BSpline interpolation works well for up to 100s of samples per window. From 1000+ samples it becomes very slow.
- Only very simple oscillator case so far!

Many questions:

- Does less frequent synchronization due to subcycling improve performance?
- Are more Quasi-Newton iterations harmful?
- Convergence order & B-spline interpolation?



Community

preCICE Workshop 2023

Stay in touch?

- `precice.org/community`
- `precice.discourse.group`

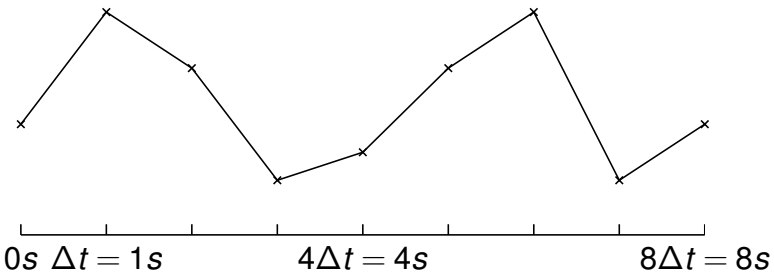
Conferences

- preCICE Workshop
- ECCOMAS Congress
- COUPLED

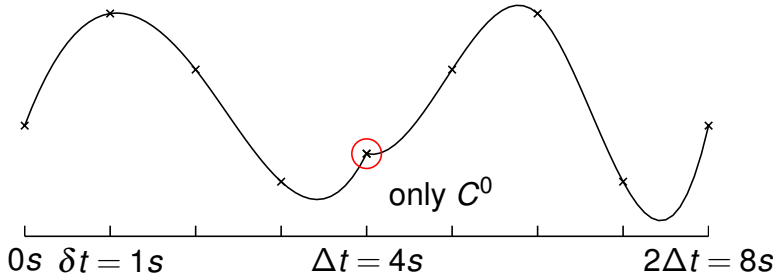
Subcycling

- Time window size $\Delta t \geq$ time step size δt_1 and δt_2.
- Do n time steps in window: $\Delta t = n_1 \delta t_1 = n_2 \delta t_2$
- Allows to create BSpline of degree $n - 1$. (Goal reached: Something better than linear interpolation)
- Restriction: Only use data of current window!
- Larger window + subcycling has impact on number of QN iterations[1]

Without subcycling

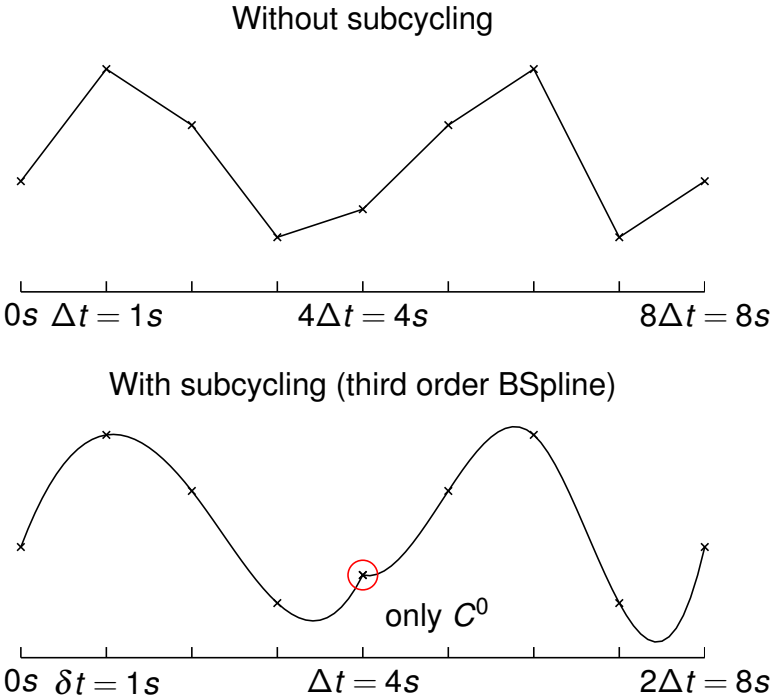


With subcycling (third order BSpline)



[1] Rüth, B, Uekermann, B, Mehl, M, Birken, P, Monge, A, Bungartz, H-J. Quasi-Newton waveform iteration for partitioned surface-coupled multiphysics applications. Int J Numer Methods Eng. 2021; 122: 5236– 5257. https://doi.org/10.1002/nme.6443

QN iterations

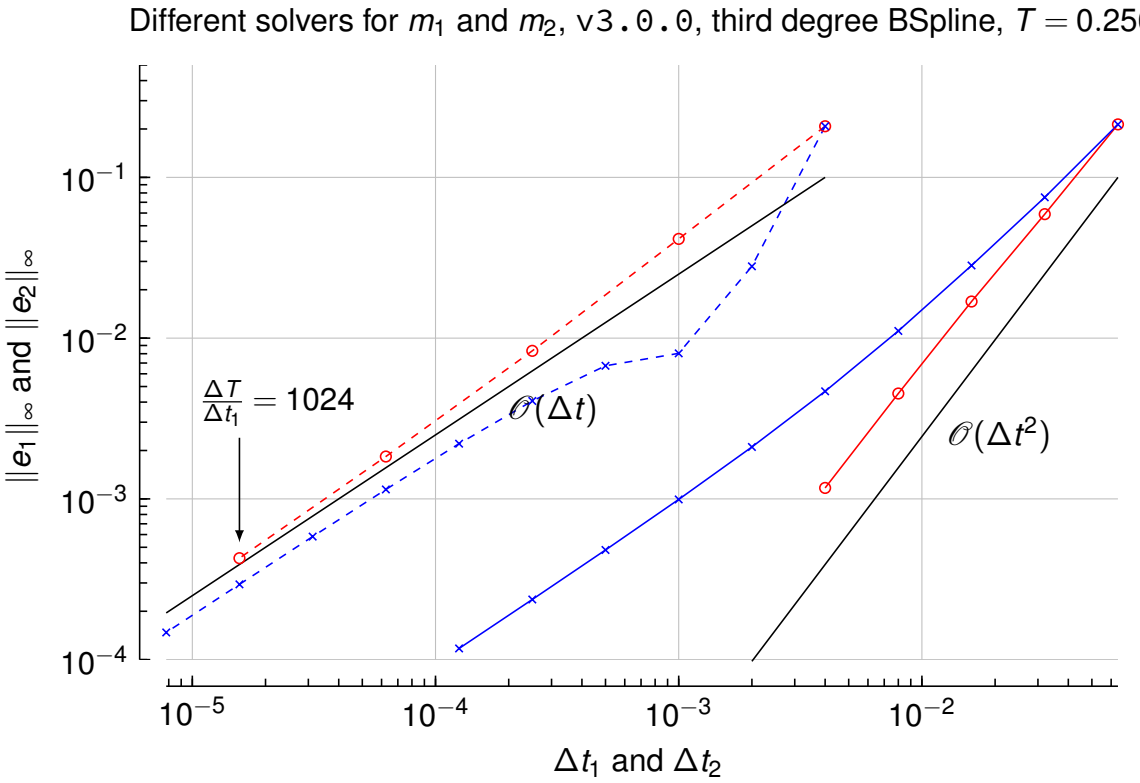


rQN-WI | Δt	0.5	0.1
WI(1, 1; 1)	7.85	5.45
WI(5, 5; 1)	10.95	7.48

rQN-WI means we only use the data at the end of the window for Quasi-Newton. Different example case, but similar implementation. More possibilities shown in[1].

[1] Rüth, B, Uekermann, B, Mehl, M, Birken, P, Monge, A, Bungartz, H-J. Quasi-Newton waveform iteration for partitioned surface-coupled multiphysics applications. Int J Numer Methods Eng. 2021; 122: 5236– 5257. https://doi.org/10.1002/nme.6443

Possible real-world setup? (preCICE Workshop 2023)



Summary:

- Combine TS schemes of different order
- Compensate low order with small δt_1
- Decrease Δt to avoid $\Delta t / \delta t_1 > 1000$
- Does this strategy work?

- 1st order EE with $\Delta t_1 = \Delta T / 64 / 2^n$
- 2nd order H with $\Delta t_2 = \Delta T / 4$
- 1st order EE with $\Delta t_1 = \Delta T / 64$
- 2nd order H with $\Delta t_2 = \Delta T / 4$