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

A Coupling Library for Partitioned Multi-Physics Simulations

- coupling schemes
- communication
- data mapping
- time interpolation

solver
adapter
libprecice

fluid solver
in-house solvent
adapter
structure
solver
libprecice

OpenFOAM
SU2
foam-extend

in-house

Ateles (APES)
Alya System
Carat++
FASTEST

API in:
C / C++
Fortran
Python

commercial solver

ANSYS Fluent
COMSOL
FEAP

structure
solver

CalculiX
Code_Aster


Benjamin Rüth (TUM) | High-Order Time Stepping in Partitioned FSI with Black-Box Solvers
Partitioned approach
Motivation & Requirements

ExaFSA\(^1\):

- (far-field) Ateles
- FASTEST
- FEAP

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

\( v^{n+1} \)
\( t^{n+1} \)
\( k_4 \)
\( \tau_R \)
\( k_1 \)
\( k_2, k_3 \)
\( w^{n+1} \)

\( v^n \)
\( t^n \)
\( w^n \)

\( w^{n+0.1} \)
\( w^{n+0.9} \)

\(^1\)see www.sppexa.de
Partitioned approach
Model problem: 1D FSI Tube


Future: complex scenarios

- multirate/local timestepping
- arbitrary meshes
- ND, arbitrary discretization (time & space)
- exa-scale
- preCICE for coupling
→ real applications
Partitioned approach
Model problem: 1D FSI Tube


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

\[ \frac{\partial a}{\partial t} + \frac{\partial a u}{\partial x} + \frac{1}{\rho} \left( \frac{\partial a p}{\partial x} - p \frac{\partial a}{\partial x} \right) = 0 \]

\[ \frac{\partial a}{\partial t} + \frac{\partial a u}{\partial x} = 0 \]

\[ a = a_0 \left( \frac{p_0 - 2c_{mk}^2}{p - 2c_{mk}^2} \right) \]

Get \( \Omega_1: p \)

Set \( \Omega_2: p \)

Neumann BC

\( t = F(a) = (u, p) \)

set

Dirichlet BC

\( \mathcal{S}(p) = a \)

Black-box python solvers¹

partitioned FSI

Dirichlet-Neumann coupling

Fluid: \( F(a) = (u, p) \)

Structure: \( S(p) = a \)

preCICE postprocessing

IQN-ILS

¹see https://github.com/precice/elasticTube1d/tree/master/PythonTube

Benjamin Rüth (TUM) | High-Order Time Stepping in Partitioned FSI with Black-Box Solvers
High order FSI
Convergence study for 1D FSI Tube coupled via preCICE

\[ L_2 \text{error in pressure} \]

TR - Implicit coupling
IE - Monolithic
TR - Monolithic

\( \tau^1 \)
\( \tau^2 \)

\( 2^{-5} \) to \( 2^{-1} \)
High order FSI
Convergence study for 1D FSI Tube coupled via preCICE

L₂ error in pressure

TR - Implicit coupling
IE - Monolithic
TR - Monolithic
order conserving

τᵢ

2⁻⁵  2⁻⁴  2⁻³  2⁻²  2⁻¹
High order FSI
Order degradation

FluidSolver.py

```python
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

```python
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**

```python
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

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

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

We want to solve the coupled problem

\[
\mathcal{F}(p, u, a) = p, \quad \mathcal{I}(p) = a.
\]

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 \(\mathcal{F}, \mathcal{I}\) 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 \(T_{n+1}\)

---

\(^1\)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

4th order Heat Transport
Convergence study with partitioned heat transport equation

\[ L_2 \text{ error on } \Omega_h \]

\[ \tau^2 \]

\[ \text{TR}(\tau) - \text{Monolithic Waveform Relaxation} \]

\[ \tau \]

\[ 10^{-3} \quad 10^{-2} \quad 10^{-1} \]

\[ 10^{-3} \quad 10^{-5} \quad 10^{-7} \quad 10^{-9} \quad 10^{-11} \quad 10^{-13} \quad 10^{-15} \]

\[ \text{see Rüth, B., et al. (2018). Time Stepping Algorithms for Partitioned Multi-Scale Multi-Physics.} \]
4th order Heat Transport

Convergence study with partitioned heat transport equation

4th order Heat Transport

Convergence study with partitioned heat transport equation

\[ \tau_i^4 \]

\[ \tau_i^2 \]

\[ \tau_i \]

\[ L_2 \text{ error on } \Omega_h \]

\[ TR(\tau) - \text{Monolithic Waveform Relaxation} \]

\[ RK4(\tau)/RK4(0.1\tau) - \text{WR} \]

\[ RK4(\tau) - \text{WR} \]

\[ RK4(\tau) - \text{Mono} \]

\[ RK4(0.1\tau) - \text{Mono} \]

---

1\text{see Rüth, B., et al. (2018). Time Stepping Algorithms for Partitioned Multi-Scale Multi-Physics.}

Benjamin Rüth (TUM) | High-Order Time Stepping in Partitioned FSI with Black-Box Solvers
4th order Heat Transport
Convergence study with partitioned heat transport equation\(^1\)

\[ L_2 \text{ error on } \Omega_h \]

\[ \tau \]

\[ \tau^2 \]

\[ \tau^4 \]

RK4(\(\tau\)) - WR
RK4(\(\tau\) - Mono
RK4(0.1\(\tau\)) - Mono
RK4(\(\tau\))/RK4(0.1\(\tau\)) - WR
RK4(\(\tau\))/TR(\(\tau\)) - WR
TR(\(\tau\)) - Monolithic Waveform Relaxation

---

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
Appendix
ExaFSA - Exascale Simulation of Fluid-Structure-Acoustics Interactions

Fluid-acoustics simulation and partitioned setup$^1$.

<table>
<thead>
<tr>
<th>physics</th>
<th>timescale</th>
<th>solver</th>
<th>scheme</th>
<th>order</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A)</td>
<td>small</td>
<td>Ateles</td>
<td>RK</td>
<td>2 or 4</td>
</tr>
<tr>
<td>(A)</td>
<td>small</td>
<td>FASTEST</td>
<td>LW</td>
<td>2</td>
</tr>
<tr>
<td>(F)</td>
<td>medium</td>
<td>FASTEST</td>
<td>CN</td>
<td>2</td>
</tr>
<tr>
<td>(S)</td>
<td>large</td>
<td>FEAP</td>
<td>HHT$^2$</td>
<td>2</td>
</tr>
</tbody>
</table>


$^2$Hilber-Hughes-Taylor Method
### Appendix

#### Coupling Details

<table>
<thead>
<tr>
<th>Currently preCICE</th>
<th>( p^{n+1} = p^n + \frac{\tau}{2} [\mathcal{F}(p^n, t_n) + \mathcal{F}(p^{n+1}, t_{n+1})] )</th>
<th>( \mathcal{O}(\tau) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a^{n+1} = a^n + \frac{\tau}{2} [\mathcal{I}(a^n, t_n) + \mathcal{I}(a^{n+1}, t_{n+1})] )</td>
<td>( \mathcal{O}(\tau) )</td>
<td></td>
</tr>
</tbody>
</table>
### Appendix

**Coupling Details**

<table>
<thead>
<tr>
<th>currently preCICE</th>
<th>update scheme</th>
<th>order</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_{n+1} = p^n + \frac{\tau}{2} [ \mathcal{F}(p^n, a^{n+1}, t_n) + \mathcal{F}(p^{n+1}, a^{n+1}, t_{n+1}) ] )</td>
<td>( a_{n+1} = a^n + \frac{\tau}{2} [ \mathcal{I}(p^{n+1}, a^{n+1}, t_{n+1}) ] )</td>
<td>( \mathcal{O}(\tau) )</td>
</tr>
</tbody>
</table>
## Appendix

### Coupling Details

<table>
<thead>
<tr>
<th></th>
<th>update scheme</th>
<th>order</th>
</tr>
</thead>
<tbody>
<tr>
<td>currently preCICE</td>
<td>$p^{n+1} = p^n + \frac{\tau}{2} \left[ \mathcal{F}(p^n, a^{n+1}, t_n) + \mathcal{F}(p^{n+1}, a^{n+1}, t_{n+1}) \right]$</td>
<td>$O(\tau)$</td>
</tr>
<tr>
<td></td>
<td>$a^{n+1} = a^n + \frac{\tau}{2} \left[ \mathcal{I}(p^{n+1}, a^{n+1}, t_{n+1}) \right]$</td>
<td></td>
</tr>
<tr>
<td>fixed version</td>
<td>$p^{n+1} = p^n + \frac{\tau}{2} \left[ \mathcal{F}(p^n, a^n, t_n) + \mathcal{F}(p^{n+1}, a^{n+1}, t_{n+1}) \right]$</td>
<td>$O(\tau^2)$</td>
</tr>
<tr>
<td></td>
<td>$a^{n+1} = a^n + \frac{\tau}{2} \left[ \mathcal{I}(p^n, a^n, t_n) + \mathcal{I}(p^{n+1}, a^{n+1}, t_{n+1}) \right]$</td>
<td></td>
</tr>
</tbody>
</table>

![Diagram](attachment:image.png)

Benjamin Rüth (TUM) | High-Order Time Stepping in Partitioned FSI with Black-Box Solvers