Coupling OpenFOAM and MBDyn with preCICE coupling tool

Mikko Folkersma

Faculty of Aerospace Engineering
Section of Wind Energy
Delft University of Technology
Delft, The Netherlands

TUDelft

AWESCO
Airborne Wind Energy
System Modelling, Control & Optimisation

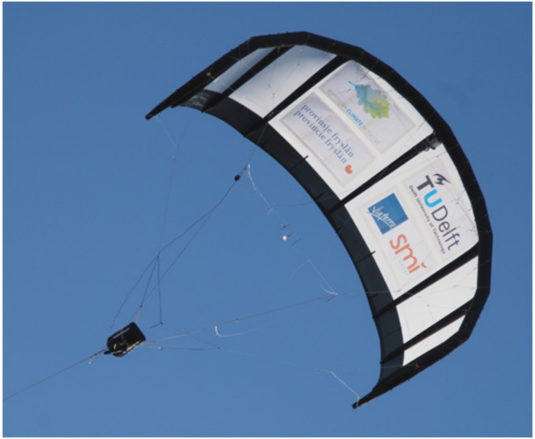
Outline

1. Introduction
2. Fluid-Structure Interaction
3. Quick look into the libraries used for the FSI coupling
4. MBDyn adapter for preCICE
5. Verification tutorial
6. Results
7. Conclusions

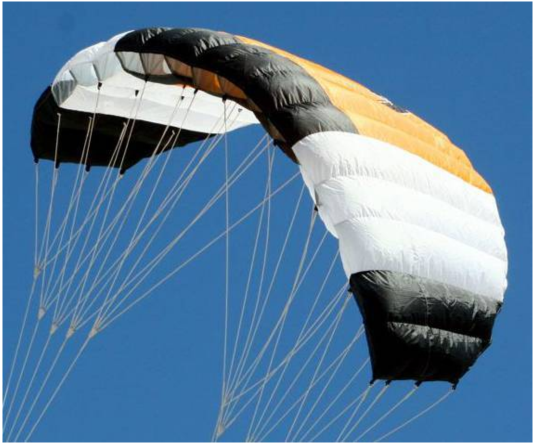
Introduction

Soft kites

LEI kite



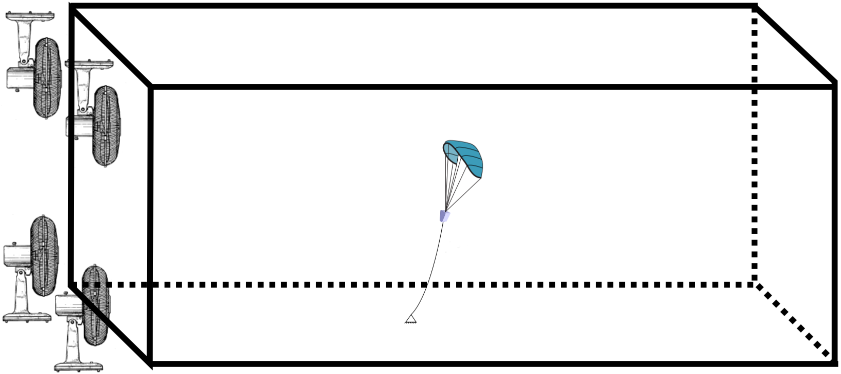
Ram air kite



- Flexible structures → FSI problem

Introduction

Objectives



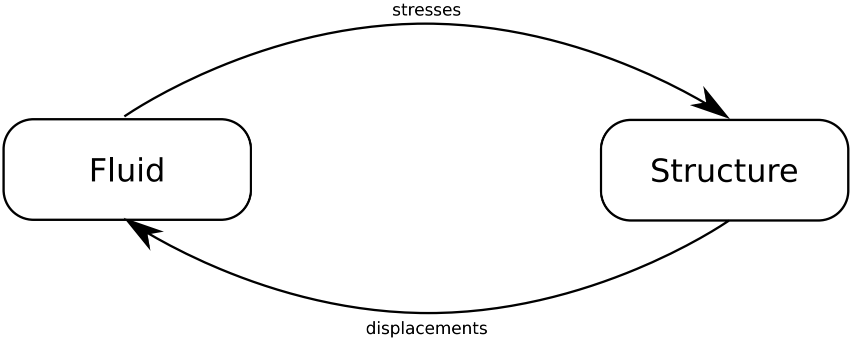
- Develop a virtual wind tunnel for soft kites
- Motivations
  1. evaluate and improve kite designs
  2. develop lower fidelity aerodynamic models (look-up tables)
  3. get insight into the physics

Fluid-Structure Interaction

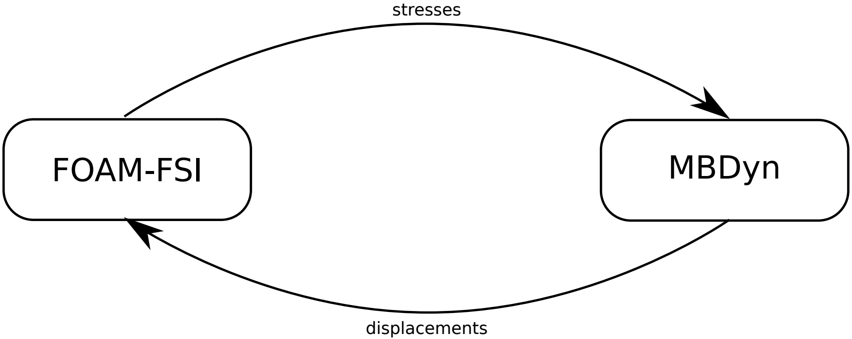
1. Multiphysics problem
2. Equations for the flow and the displacement
3. Partitioned approach: two distinct solvers are used

Fluid-Structure Interaction

1. Multiphysics problem
2. Equations for the flow and the displacement
3. Partitioned approach: two distinct solvers are used



Fluid-Structure Interaction



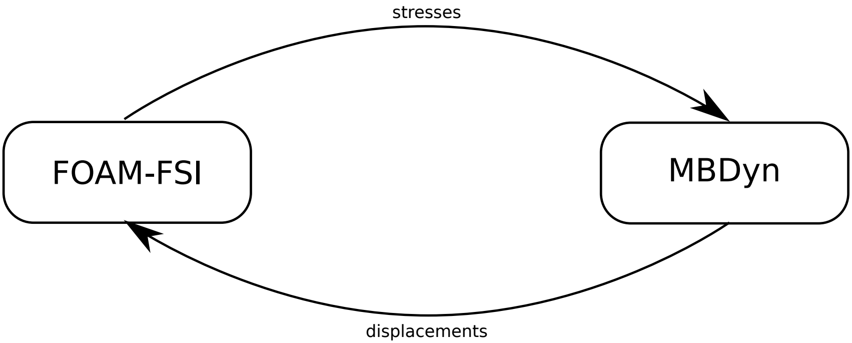
stresses

FOAM-FSI

MBDyn

displacements

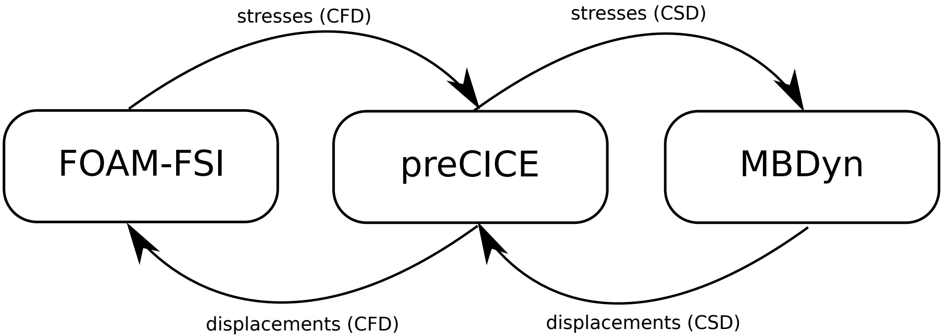
Fluid-Structure Interaction



Challenges

1. Interfacing between different programming languages
2. Non-conforming meshes at the interface
3. Stability of the coupling
4. Efficiency (Scalability, parallelization)

Fluid-Structure Interaction



stresses (CFD)

stresses (CSD)

FOAM-FSI

preCICE

MBDyn

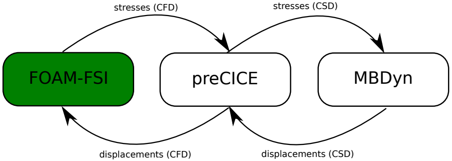
displacements (CFD)

displacements (CSD)

FOAM-FSI

https://github.com/davidsblom/FOAM-FSI.

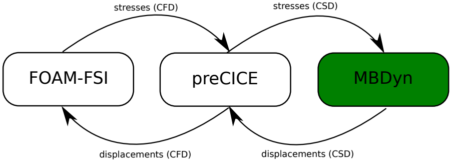
- foam-extend 3.2 extension developed by Blom et al.
- Efficient RBF based mesh deformation with coarsening
- preCICE adapter



MBDyn

https://www.mbdyn.org/

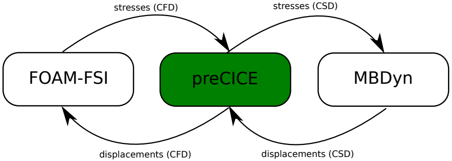
- Multibody dynamics software to study the behavior of interconnected rigid and flexible bodies
- Flexible body capabilities: beam, shell and **membrane** elements
- Python interface (through sockets)



preCICE coupling tool

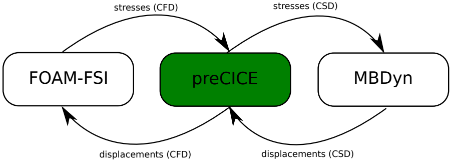
https://www.precice.org/

- High-level interface in C/C++, Fortran and Python
- Parallel execution and communication
- Mapping for non-matching meshes
- Coupling algorithms
- Minimally invasive (No/minimal modifications to the coupled solvers)



preCICE requirements

- Configuration file (XML) which defines the interfaces, the mapping and the coupling schemes
- Adapter from OpenFOAM to preCICE (FOAM-FSI)
  (Adapter will be soon available for the latest OpenFOAM versions)
- Adapter from MBDyn to preCICE (this work)



MBDyn adapter

1. Initialization
    - Import preCICE interface
    - Pass the node coordinates to preCICE (displacements)
    - Pass the cell center coordinates to preCICE (stresses)
    - initialize the arrays for displacements and stresses

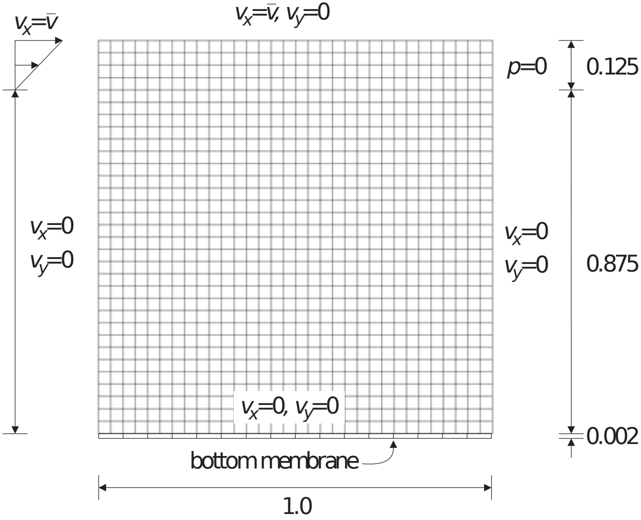
MBDyn adapter

1. Initialization
   - Import preCICE interface
   - Pass the node coordinates to preCICE (displacements)
   - Pass the cell center coordinates to preCICE (stresses)
   - initialize the arrays for displacements and stresses
2. Solving
   - Solve the problem
   - pass the displacements to preCICE
   - Read stresses from preCICE to MBDyn
   - Check if the solvers have converged

Verification tutorial

Cavity with membrane

$$\overline{v} = 1 - cos(\frac{2\pi t}{5})$$



FOAM-FSI settings

0/U

```
inlet
{
        type groovyBC; // swak4foam boundary condition
        valueExpression "vector((pos().y - 0.875)/0.125 *\
                        (1.0-cos(2.0*pi*time()/5.0)),0.0,0.0)";
        value           uniform (0 0 0);
}
movingWall
{
        type groovyBC;
        valueExpression "vector(1.0-cos(2.0*pi*time()/5.0),0.0,0.0)";
        value           uniform (0 0 0);
}
bottomWall
{
        type            myMovingWallVelocity; // FOAM-FSI boundary condition
        value           uniform (0 0 0);
}
```

MBDyn adapter

cavityFSI.py

```
from mbdynAdapter import MBDynHelper, MBDynAdapter
mbd = MBDynHelper()
mbd.readMsh(’membrane.msh’)
mbd.controlDict = {’initialTime’:0,’finalTime’:100,’output frequency’:10}
mbd.materialDict = {’E’:250, ’nu’:0,’t’:0.002,’rho’:500,’C’:0.000001}
adapter = MBDynAdapter(mbd)
adapter.runPreCICE()
```

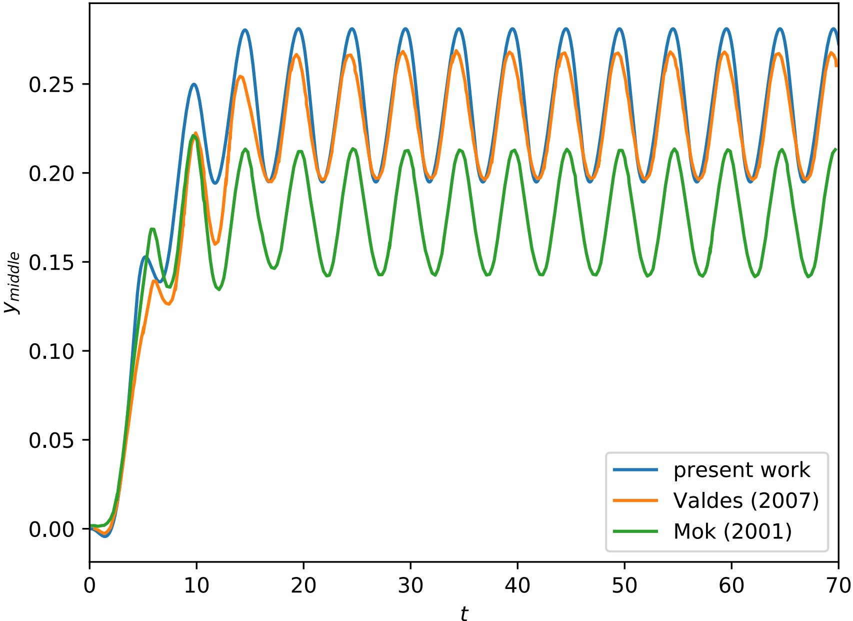
Execution

Allrun

```
fsiFluidFoam &> log.fluidFoam &
python cavityFSI.py &> log.structureSolver &
```

Results

Cavity with membrane



Conclusions

Conclusions

- Coupled MBDyn and OpenFOAM with preCICE.
- No changes needed to MBDyn source code.
- Verified the implementation.