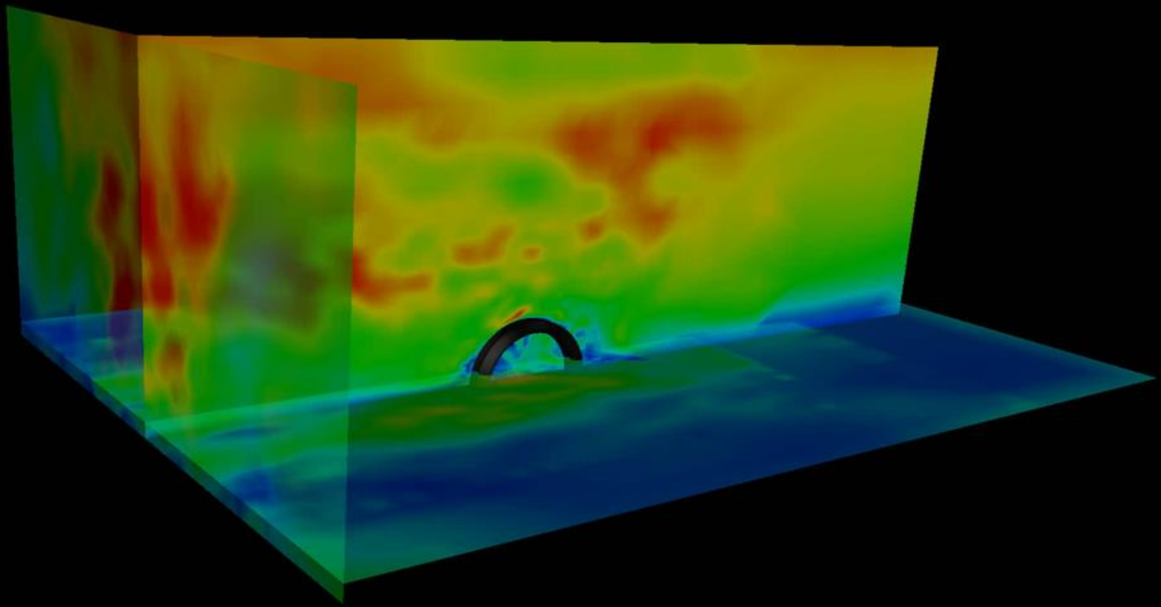
Chair of Structural Analysis
TUM Department of Civil, Geo and Environmental Engineering
Technical University of Munich



STRUCTURAL WIND ENGINEERING

Roland Wüchner, Chair of Structural Analysis, TUM
Máté Péntek, Chair of Structural Analysis, TUM

Presentation material from internal and external sources have been used either directly, modified or adapted to fit the purpose. Effort is continuously being made to accurately reference these. Nonetheless, check referencing in both the script as well as slides for completeness. In case of inconsistencies or mistakes please contact us!

Content

FROM CFD TO FSI

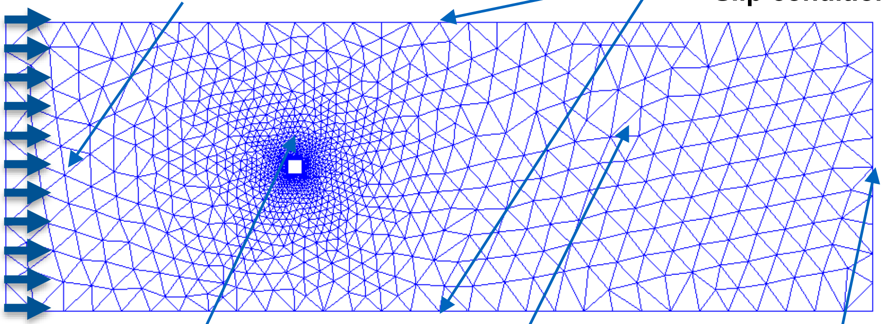
The steps necessery for going from CFD to FSI

2D CFD example ⇒ representing a horizontal cut through a building

CFD setup

Inlet
- **Prescribed velocity**

Wall
- **Slip-condition**

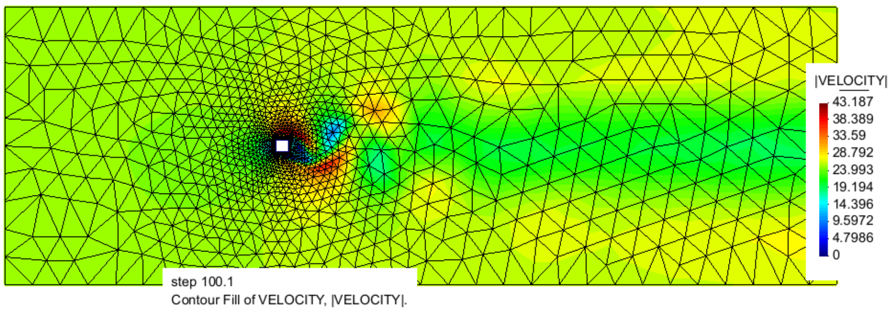


Structure
- **Non-moving**
- **No slip condition**

Fluid
- **Navier-Stokes**
- **Fractional-step solver**

Outlet
- **Assigned pressure**

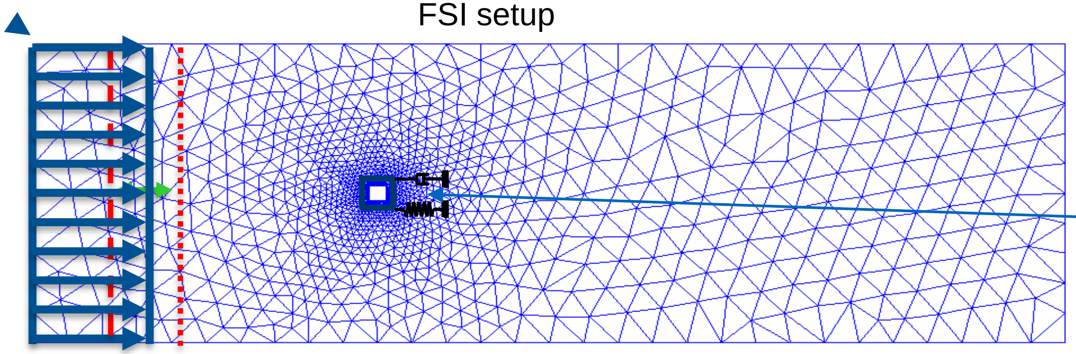
CFD simulation results



2D FSI example ⇒ representing model scenario for gust-induced buffeting

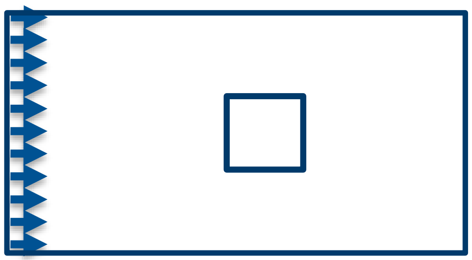
FSI setup

Inlet
- **Prescribed velocity**
- **Varying in time**
- **Oscillating parabola**



Structure
- **No slip condition**
- **Moving**
- **SDoF solver**

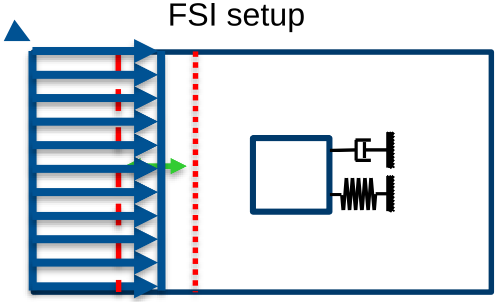
Custom inlet



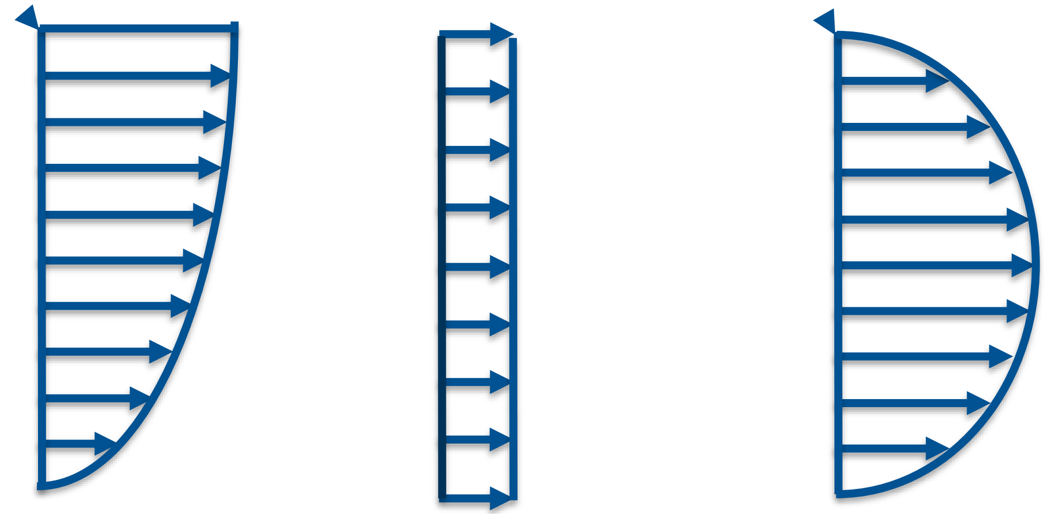
CFD setup

- Custom inlet
- Structural solver
- Mesh mover
- Coupling of physics and numerics

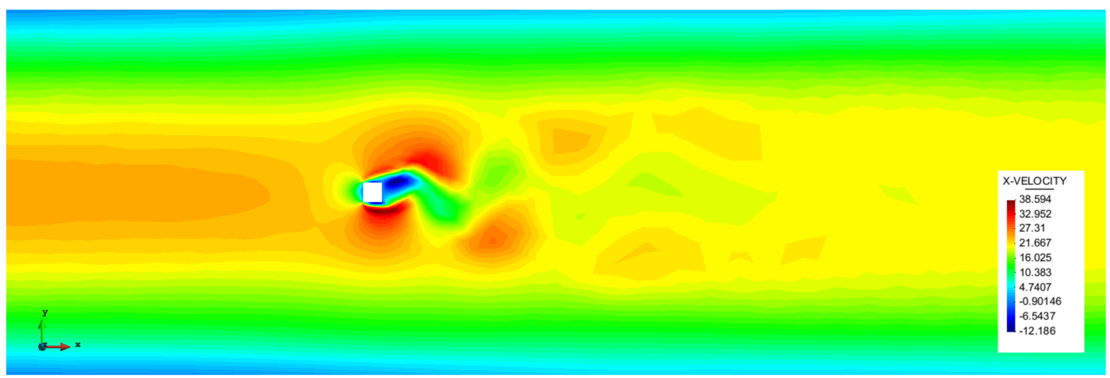
FSI setup



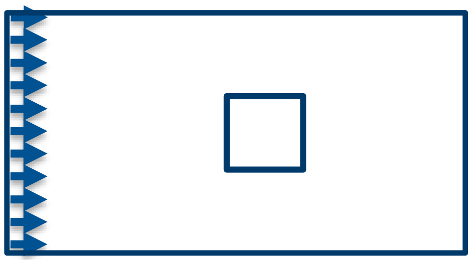
From constant in time and space ⇒ varying in time and space: case of a pulsating inlet.



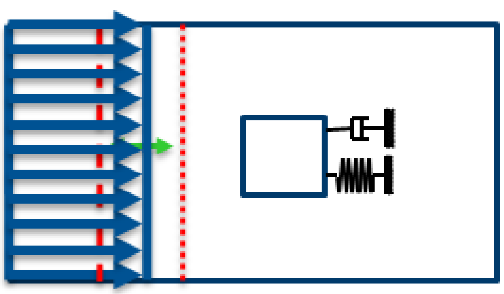
Power law *Constant* *Constant parabolic*



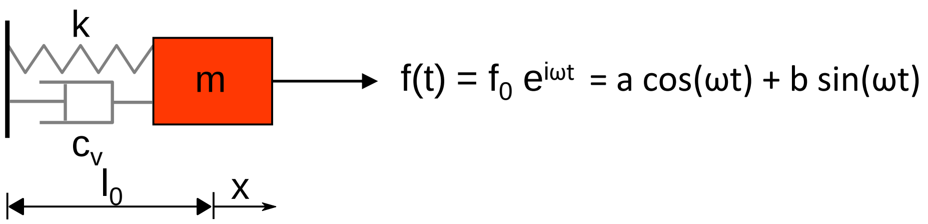
Structural solver

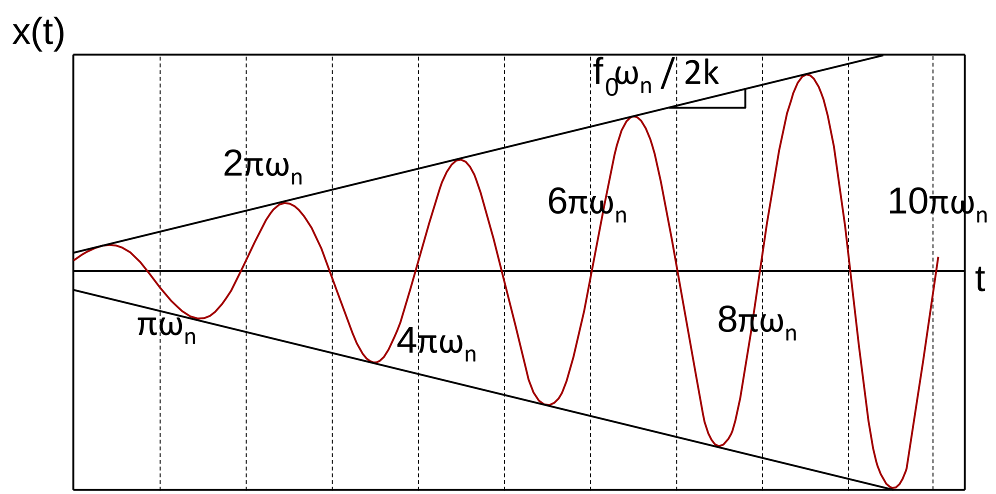


- Custom inlet ✓
- Structural solver
- Mesh mover
- Coupling of physics and numerics

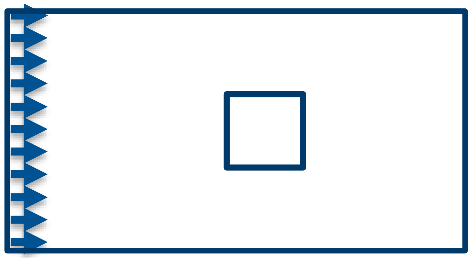


The Generalized Alpha solver for SDoF ⇒ provided in *the* SDoF Python solver





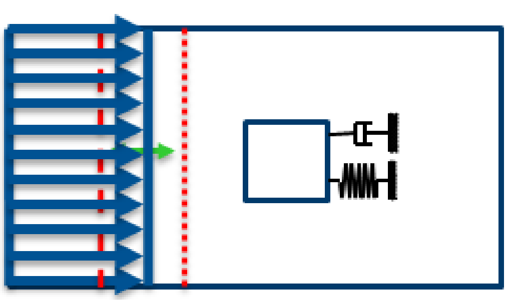
Mesh mover



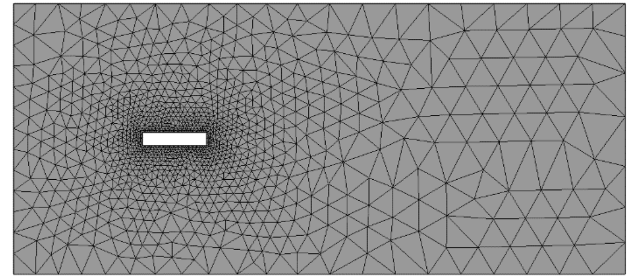
CFD setup

- Custom inlet √
- Structural solver √
- Mesh mover
- Coupling of physics and numerics

FSI setup

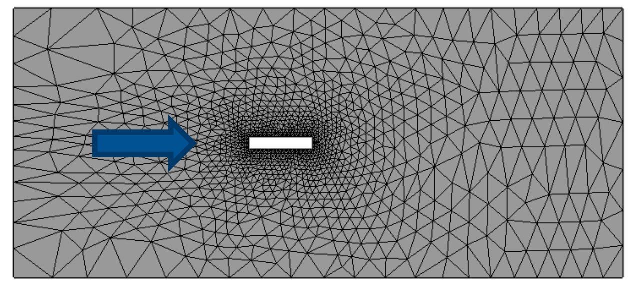


Example initial CFD mesh

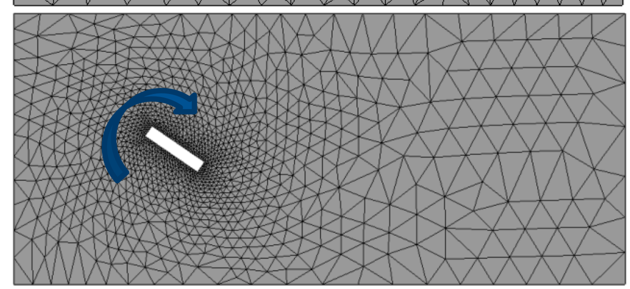


Possible movement

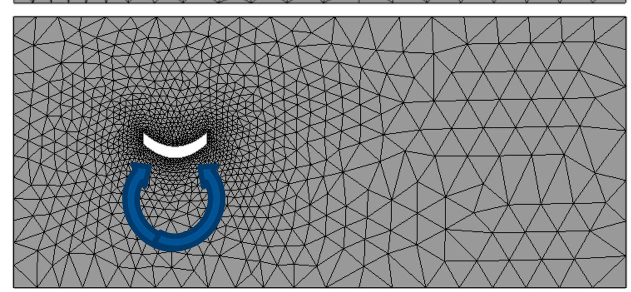
- Translation



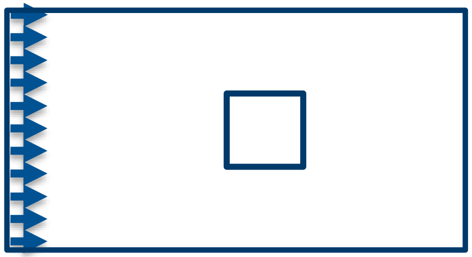
- Rotation



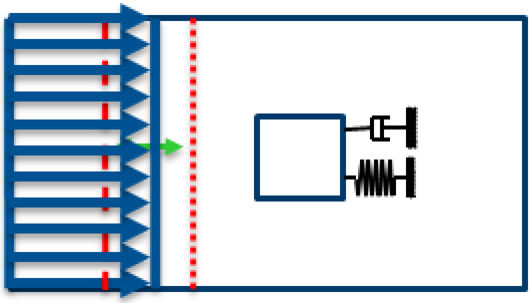
- Bending



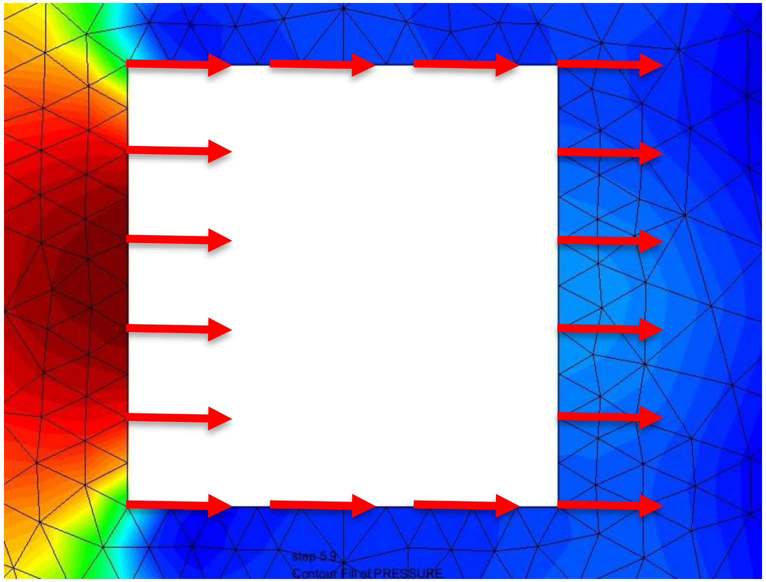
Coupling of physics D-N



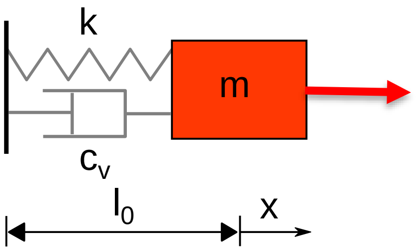
- Custom inlet √
- Structural solver √
- Mesh mover √
- Coupling of physics and numerics



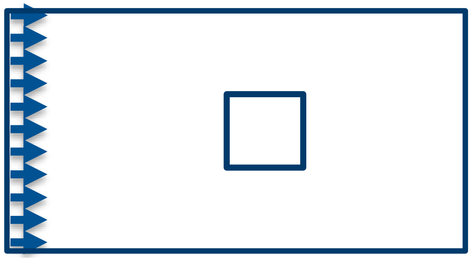
Fluid solver ⇒ solves at timestep ⇒ pressure and friction on structure ⇒ forces on surface nodes ⇒ passed to structural solver as input



Structural solver ⇒ SDoF solver for given forces reduced into center of gravity – Neumann condition for structure



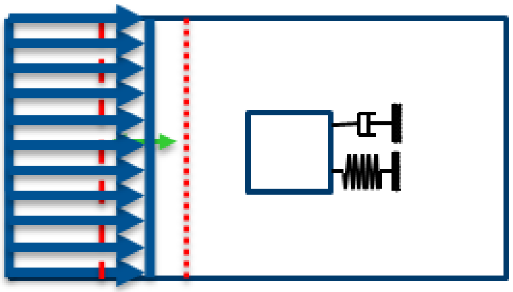
Coupling of physics D-N



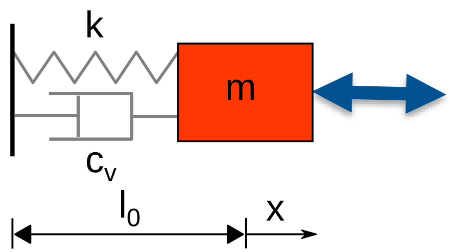
CFD setup

- Custom inlet √
- Structural solver √
- Mesh mover √
- Coupling of physics and numerics

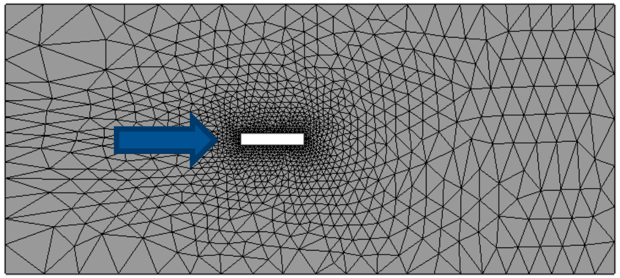
FSI setup



Structural solver ⇒ SDoF solver results a displacement X



Mesh solver ⇒ receives displacement as input from structural solver ⇒ solves ⇒ send the calculated mesh velocity to the fluid solver



Fluid solver ⇒ takes the velocity input as Dirichlet condition ⇒ solves

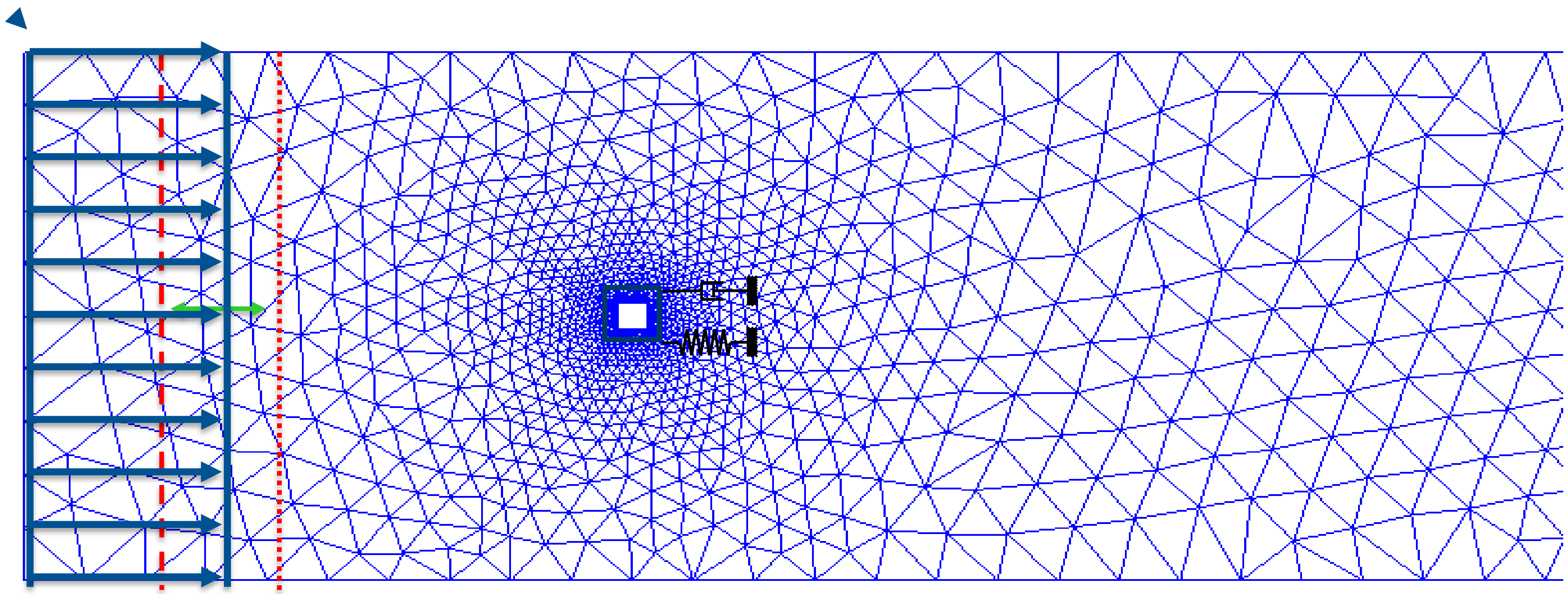
⇒ The iteration loop goes on until a convergence criteria is met ⇒ refer back to the stong coupling for FSI problems

FSI – EIE: BUFFETING-INDUCED VIBRATION

FSI – buffeting tutorial

Based up on Tutorial 4 ⇒ *StructWindEng_WS1920_GiDKratosTutorial1_2D_CFD*

- 2D FSI example representing a horizontal cut through a building
- Please make the modifications to the files in this example

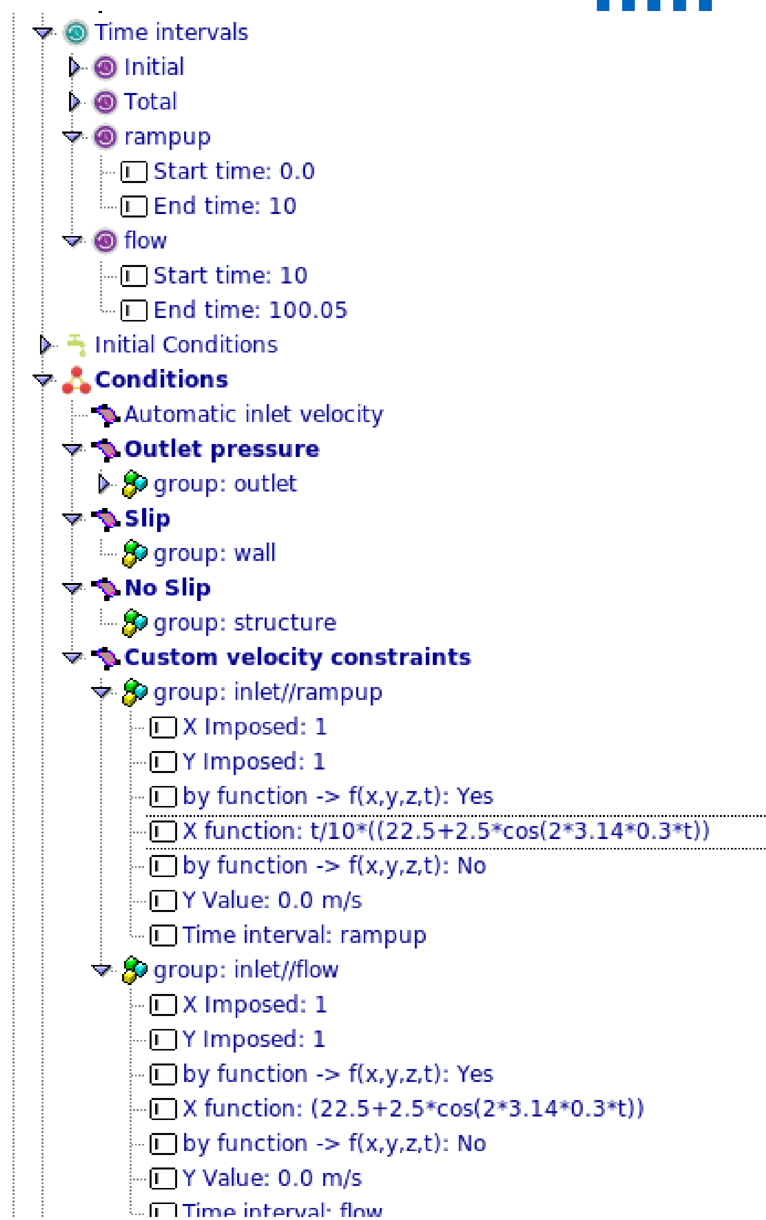


FSI – buffeting tutorial

1. Open with GiD the file you created on *SWE_WS1920_GiDKratosTutorial1_2D_CFD*
2. Apply an oscillating inlet velocity and ramp up time of 10 sec
3. Save as a new file
4. copy the contents of the folder “FSISDoFBuffeting” from “AuxiliaryFiles”
5. Rename “ProjectParameters.json” to “ProjectParametersCFD.json”
6. Copy “solver_settings” block from the “ProjectParametersCFD.json” Template

   (check the next slide for picture)
7. open “ProjectParametersFSI.json” and check “interface_submodel_part” matches the name given to your structural model part

8. run “MainKratosFSI.py”



FSI – buffeting tutorial

ProjectParametersFSIBuffeting.json

ProjectParameters.json

```
    },
    "solver_settings"  : {
        "model_part_name"                : "FluidModelPart",
        "domain_size"                    : 2,
        "solver_type"                    : "FractionalStep",
        "model_import_settings"          : {
            "input_type"      : "mdpa",
            "input_filename"  : "Tutorial1_2D_CFD"
        },
        "material_import_settings"       : {
            "materials_filename" : "FluidMaterials.json"
        },
        "echo_level"                     : 1,
        "compute_reactions"              : true,
        "dynamic_tau"                    : 0.1,
        "predictor_corrector"            : false,
        "pressure_tolerance"             : 0.001,
        "maximum_pressure_iterations"    : 4,
        "velocity_tolerance"             : 0.001,
        "maximum_velocity_iterations"    : 15,
```

ProjectParametersCFD.json

> 4_FSI > 1_CustomSc

ProjectParametersCFD.json Template

```
                "save_output_files_in_folder"             : true,
                "nodal_solution_step_data_variables"      : ["VELOCITY","PRESSURE","MESH_DISPLACEMENT"],
                "nodal_data_value_variables"              : [],
                "element_data_value_variables"            : [],
                "condition_data_value_variables"          : [],
                "gauss_point_variables_extrapolated_to_nodes" : []
            }
        }]
    },
    "solver_settings"  : {
        "solver_type": "ale_fluid",
        "ale_boundary_parts": [
            "NoSlip2D_structure"
        ],
        "mesh_motion_solver_settings": {
            "solver_type": "mesh_solver_structural_similarity",
            "mesh_motion_linear_solver_settings": {
                "solver_type": "amgcl",
                "smoother_type": "spai0",
                "krylov_type": "gmres",
                "coarsening_type": "aggregation",
                "max_iteration": 200,
                "provide_coordinates": false,
                "gmres_krylov_space_dimension": 100,
                "verbosity": 0,
                "tolerance": 1e-7,
                "scaling": false,
                "block_size": 1,
                "use_block_matrices_if_possible": true,
                "coarse_enough": 5000
            },
            "reform_dofs_each_step": false,
            "compute_reactions": false
        },
        "fluid_solver_settings": {
            "model_part_name"                : "FluidModelPart",
            "domain_size"                    : 2,
            "solver_type"                    : "FractionalStep",
            "model_import_settings"          : {
                "input_type"     : "mdpa",
                "input_filename" : "Tutorial1_2D_CFD"
            },
            "material_import_settings"       : {
                "materials_filename" : "FluidMaterials.json"
```

ProjectParametersFSI.json

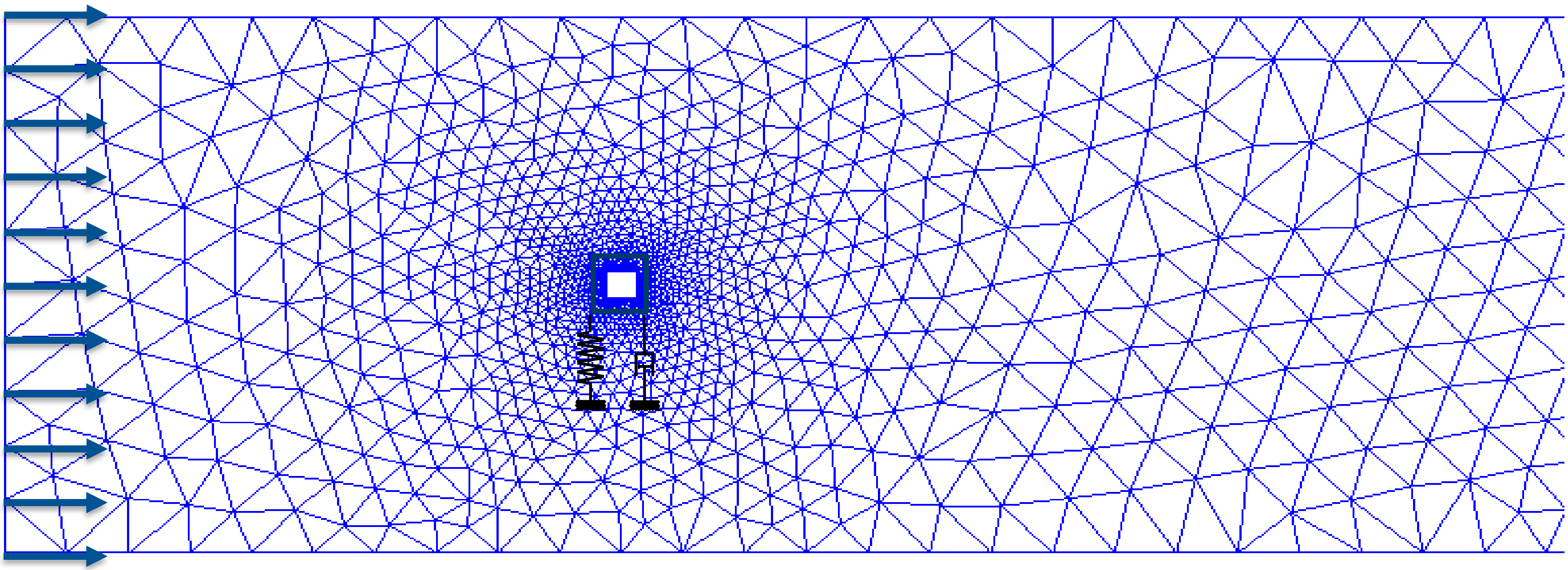
```
    "coupling_settings":{
        "mapper_settings":{
        "mapper_type": "sdof",
        "interface_submodel_part_destination": "NoSlip2D_structure",
        "interface_submodel_part_origin": "SDoF_origin"
        },
        "convergence_accelerator_settings": {
            "type": "aitken",
            "max_iterations": 5,
            "residual_relative_tolerance": 1e-5,
            "residual_absolute_tolerance": 1e-9,
            "relaxation_coefficient_initial_value": 0.25
        }
    }
}
```

FSI – IIE: VORTEX SHEDDING-INDUCED VIBRATION

FSI – vortex shedding-induced vibration tutorial

Based up on Tutorial 4 ⇒ *SWE_WS1920_GiDKratosTutorial1_2D_CFD*

- 2D FSI example representing a horizontal cut through a building
- Please make the modifications to the files in this example

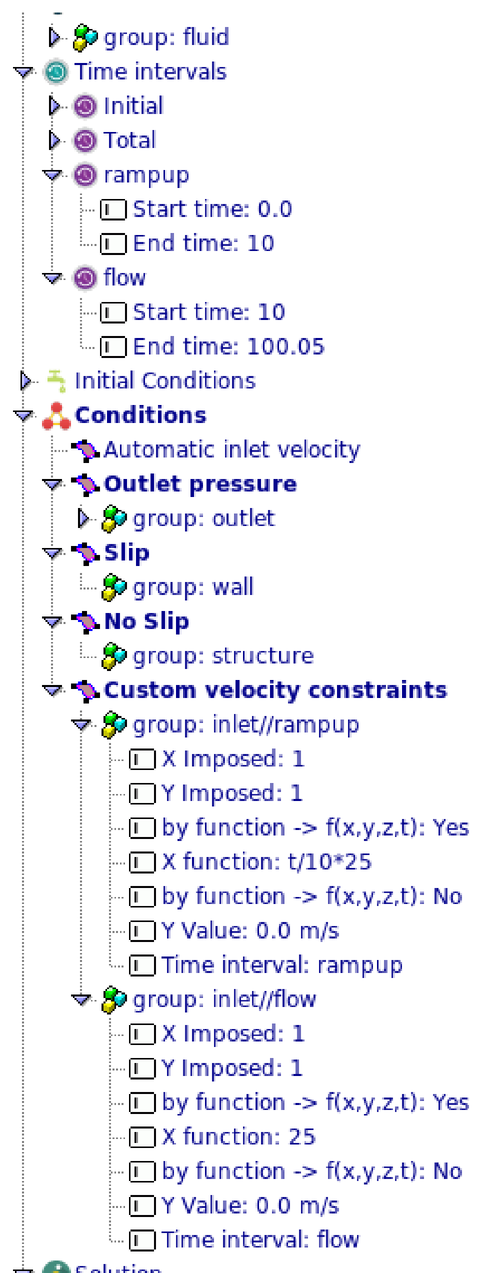


FSI – vortex shedding-induced vibration tutorial

1. Open with GiD the file you created on *SWE_WS1920_GiDKratosTutorial1_2D_CFD*
2. Apply a constant inlet velocity and ramp up time of 10 sec
3. Save as a new file
4. copy the contents of the folder “FSISDoFVortexShedding” from “AuxiliaryFiles”
5. Rename “ProjectParameters.json” to “ProjectParametersCFD.json”
6. Copy “solver_settings” block from the “ProjectParametersCFD.json” Template

   (check the next slide for picture)
7. open “ProjectParametersFSI.json” and check “interface_submodel_part” matches the name given to your structural model part

8. run “MainKratosFSI.py”



FSI – buffeting tutorial

ProjectParametersFSIBuffeting.json

ProjectParameters.json

```
"solver_settings"  : {
    "model_part_name"                  : "FluidModelPart",
    "domain_size"                      : 2,
    "solver_type"                      : "FractionalStep",
    "model_import_settings"            : {
        "input_type"     : "mdpa",
        "input_filename" : "Tutorial1_2D_CFD"
    },
    "material_import_settings"         : {
        "materials_filename" : "FluidMaterials.json"
    },
    "echo_level"                       : 1,
    "compute_reactions"                : true,
    "dynamic_tau"                      : 0.1,
    "predictor_corrector"              : false,
    "pressure_tolerance"               : 0.001,
    "maximum_pressure_iterations"      : 4,
    "velocity_tolerance"               : 0.001,
    "maximum_velocity_iterations"      : 15,
```

ProjectParametersFSI.json

```
"coupling_settings":{
        "mapper_settings":{
        "mapper_type": "sdof",
        "interface_submodel_part_destination": "NoSlip2D_structure",
        "interface_submodel_part_origin": "SDoF_origin"
    },
    "convergence_accelerator_settings": {
        "type": "aitken",
        "max_iterations": 5,
        "residual_relative_tolerance": 1e-5,
        "residual_absolute_tolerance": 1e-9,
        "relaxation_coefficient_initial_value": 0.25
    }
  }
}
```

ProjectParametersCFD.json

> 1_CustomS

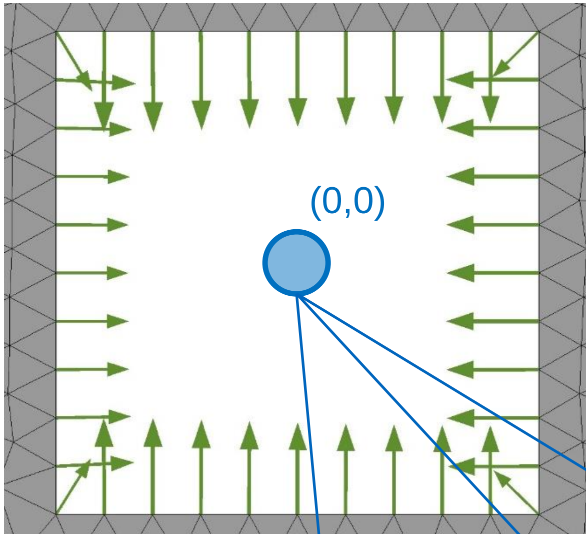
ProjectParametersCFD.json Template

```
                "save_output_files_in_folder"              : true,
                "nodal_solution_step_data_variables"       : ["VELOCITY","PRESSURE","MESH_DISPLACEMENT"],
                "nodal_data_value_variables"               : [],
                "element_data_value_variables"             : [],
                "condition_data_value_variables"           : [],
                "gauss_point_variables_extrapolated_to_nodes" : []
            }
        }]
    },
    "solver_settings"  : {
        "solver_type": "ale_fluid",
        "ale_boundary_parts": [
            "NoSlip2D_structure"
        ],
        "mesh_motion_solver_settings": {
            "solver_type": "mesh_solver_structural_similarity",
            "mesh_motion_linear_solver_settings": {
                "solver_type": "amgcl",
                "smoother_type": "spai0",
                "krylov_type": "gmres",
                "coarsening_type": "aggregation",
                "max_iteration": 200,
                "provide_coordinates": false,
                "gmres_krylov_space_dimension": 100,
                "verbosity": 0,
                "tolerance": 1e-7,
                "scaling": false,
                "block_size": 1,
                "use_block_matrices_if_possible": true,
                "coarse_enough": 5000
            },
            "reform_dofs_each_step": false,
            "compute_reactions": false
```

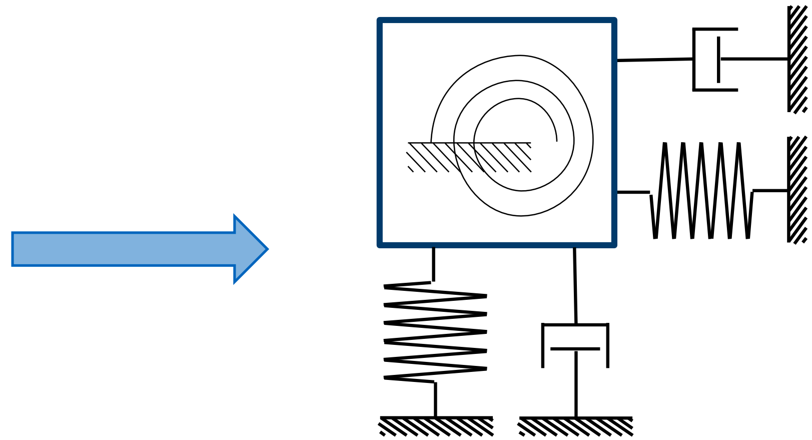
FROM FSI WITH SDOF TOWARDS MDOF

Case of 3 x SDoF = MDoF

Reaction magnitude

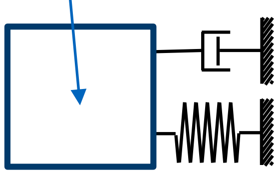


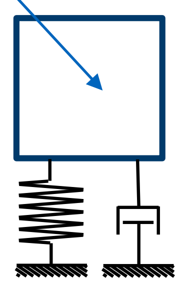
Nodal reactions from the CFD

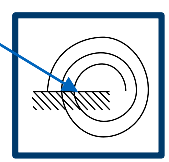


Concentrated action on the SDoF ⇒ here the MDoF can be seen and simplified into a superposition of SDoFs ⇒ a 3SDoF

A superposition of SDoFs ⇒ displacement_X + displacement_Y + rotation







Each SDoF

⇒ **concentrated** into a material point (**to *(0,0)***)

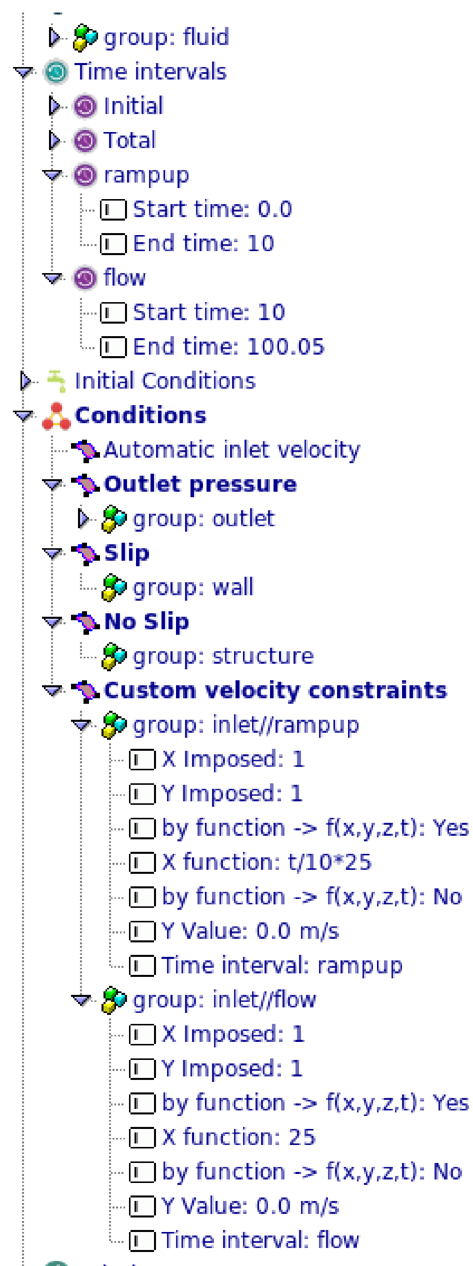
⇒ **no shape + no dimension!**

FSI – 3SDoF tutorial

Based up on Tutorial 4 ⇒
SWE_WS1920_GiDKratosTutorial1_2D_CFD

1. Open with GiD the file you created on *SWE_WS1920_GiDKratosTutorial1_2D_CFD*
2. Apply a constant inlet velocity and ramp up time of 10 sec
3. Save as a new file
4. copy the contents of the folder “FSI3SDoF from “AuxiliaryFiles”
5. Rename “ProjectParameters.json” to “ProjectParametersCFD.json”
6. Copy “solver_settings” block from the “ProjectParametersCFD.json” Template
7. open “ProjectParametersFSI.json” and check “interface_submodel_part” matches the name given to your structural model part

8. run “MainKratosFSI.py”



FSI – 3SDoF tutorial

ProjectParametersFSIBuffeting.json ×

ProjectParameters.json

```
"solver_settings"  : {
    "model_part_name"                 : "FluidModelPart",
    "domain_size"                     : 2,
    "solver_type"                     : "FractionalStep",
    "model_import_settings"           : {
        "input_type"      : "mdpa",
        "input_filename"  : "Tutorial1_2D_CFD"
    },
    "material_import_settings"        : {
        "materials_filename" : "FluidMaterials.json"
    },
    "echo_level"                      : 1,
    "compute_reactions"               : true,
    "dynamic_tau"                     : 0.1,
    "predictor_corrector"             : false,
    "pressure_tolerance"              : 0.001,
    "maximum_pressure_iterations"     : 4,
    "velocity_tolerance"              : 0.001,
    "maximum_velocity_iterations"     : 15,
```

ProjectParametersCFD.json ×

ProjectParametersCFD.json Template

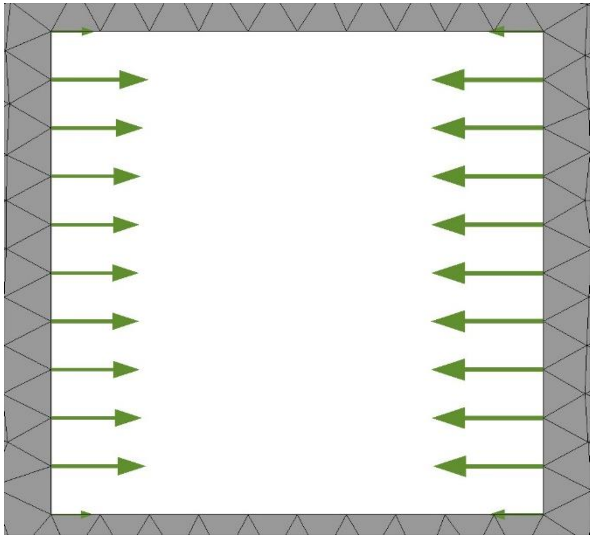
```
                "save_output_files_in_folder"                 : true,
                "nodal_solution_step_data_variables"          : ["VELOCITY","PRESSURE","MESH_DISPLACEMENT"],
                "nodal_data_value_variables"                  : [],
                "element_data_value_variables"                : [],
                "condition_data_value_variables"              : [],
                "gauss_point_variables_extrapolated_to_nodes" : []
            }
        }]
    },
    "solver_settings"  : {
        "solver_type": "ale_fluid",
        "ale_boundary_parts": [
            "NoSlip2D_structure"
        ],
        "mesh_motion_solver_settings": {
            "solver_type": "mesh_solver_structural_similarity",
            "mesh_motion_linear_solver_settings": {
                "solver_type": "amgcl",
                "smoother_type": "spai0",
                "krylov_type": "gmres",
                "coarsening_type": "aggregation",
                "max_iteration": 200,
                "provide_coordinates": false,
                "gmres_krylov_space_dimension": 100,
                "verbosity": 0,
                "tolerance": 1e-7,
                "scaling": false,
                "block_size": 1,
                "use_block_matrices_if_possible": true,
                "coarse_enough": 5000
            },
            "reform_dofs_each_step": false,
            "compute_reactions": false
```

ProjectParametersFSI.json

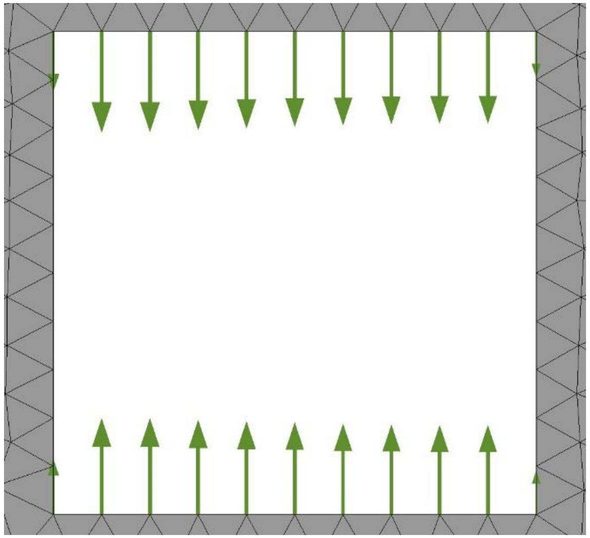
```
    "coupling_settings":{
            "mapper_settings":{
            "mapper_type": "sdof",
            "interface_submodel_part_destination": "NoSlip2D_structure",
            "interface_submodel_part_origin": "SDoF_origin"
        },
        "convergence_accelerator_settings": {
            "type": "aitken",
            "max_iterations": 5,
            "residual_relative_tolerance": 1e-5,
            "residual_absolute_tolerance": 1e-9,
            "relaxation_coefficient_initial_value": 0.25
        }
    }
}
```

Case of 3 x SDoF = MDoF

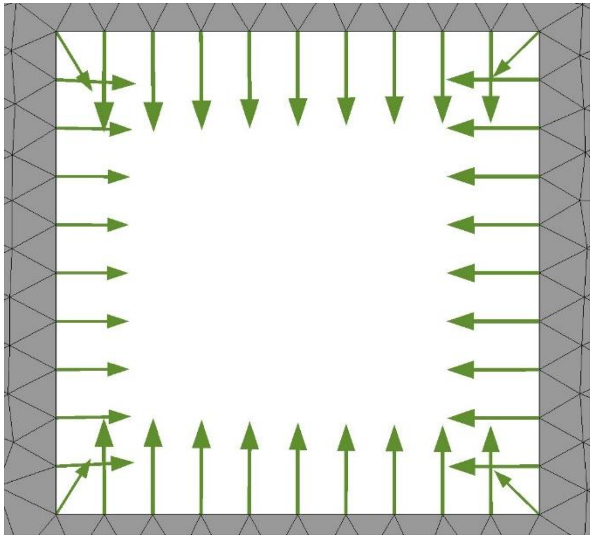
Reaction *X*



Reaction *Y*



Reaction magnitude

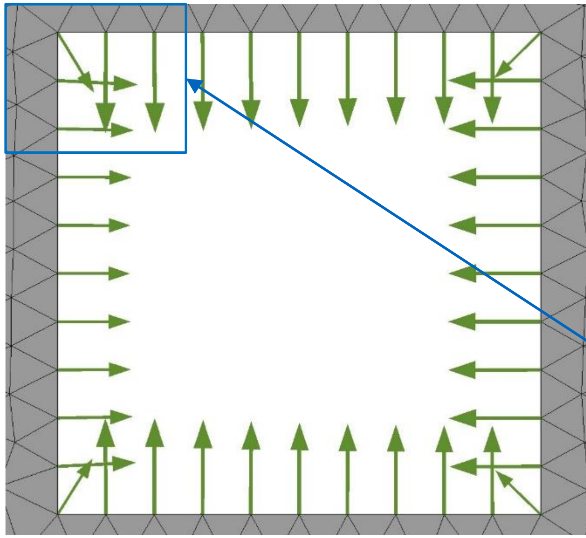


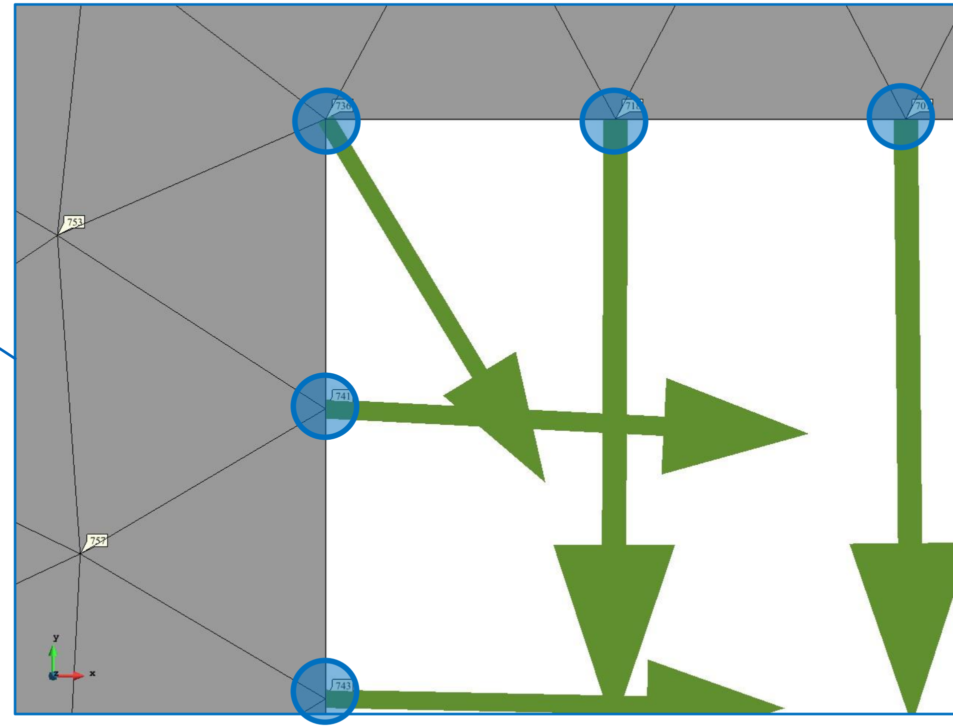
Reaction ⇒ result of the CFD simulation, here 2D case of horizontal cut ⇒ reaction extraction and „preparing/sending“ to the SDoF solver

- Locate structure interface in fluid domain
- Loop over the nodes on the interface
- Extract reactions from the nodes

Case of 3 x SDoF = MDoF

Reaction magnitude





Node

- Coordinates: *node.X, node.Y, node.Z*
- Id number: *node.Id*
- Nodal data: pressure, velocity, reaction, displacement
- Retrieving nodal solutions – from the current time step using the commands:
  - `node.GetSolutionStepValue(PRESSURE, 0)`
  - `node.GetSolutionStepValue(VELOCITY, 0)`
  - `node.GetSolutionStepValue(REACTION, 0)`
  - `node.GetSolutionStepValue(MESH_DISPLACEMENT, 0)`
- Interested in nodes on the interface of the structure ⇒ nodes part of a certain *Sub Model Part*

```
for node in main_model_part.GetSubModelPart(structure_submodel).Nodes
```

Case of 3 x SDoF = MDoF

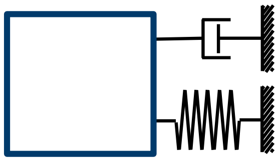
Some selected code snippets

Compare: fsi_utilities.py: lines 84-111

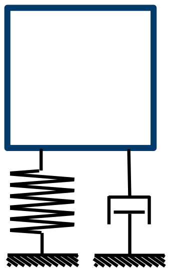
```
def NeumannToStructure(mapper, structure_solver, flag):
    multiplicator = 1.0
    # if swap_sign:
    if flag:
        multiplicator = -1.0

    fx = 0.0
    fy = 0.0
    mz = 0.0

    for node in mapper.destination_interface.Nodes:
        reaction = node.GetSolutionStepValue(
             KratosMultiphysics.REACTION, 0)
        fx += multiplicator * reaction[0]

        fy += multiplicator * reaction[1]

        fx_n = multiplicator * reaction[0]
        fy_n = multiplicator * reaction[1]
        rx_n = node.X - current_center_x
        ry_n = node.Y - current_center_y
        mz += ry_n * fx_n - rx_n * fy_n

    structure_solver.SetExternalForce([fx, fy, mz])
```

Extract and apply appropriate action force for

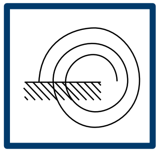
⇒ displacement_X



⇒ displacement_Y



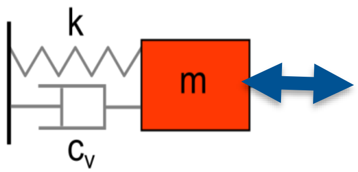
⇒ rotation



Case of 3 x SDoF = MDoF

Solve for each structural solver separately

Compare : MainKratosFSI.py: lines 225-234




```
fsi_utilities.NeumannToStructure(mapper, structural_solver, True)

# # Solver Structure
structural_solver.SolveSolutionStep()
```

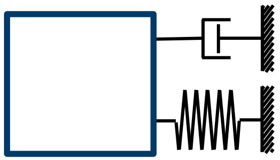
Set displacement to the nodes on the surface of the structure ⇒ as a ***superposition*** of SDoFs

Compare: fsi_utilities.py: lines 114-134

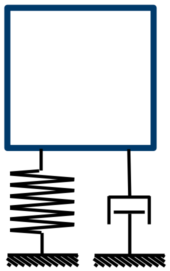
```
def DisplacementToMesh(mapper, displacement, structure_solver):
    disp_x = displacement[0]
    disp_y = displacement[1]
    theta = displacement[2]

    for node in mapper.destination_interface.Nodes:
        rx0 = node.X0 - current_center_x
        ry0 = node.Y0 - current_center_y
        rx = math.cos(theta) * rx0 + math.sin(theta) * ry0
        ry = - math.sin(theta) * rx0 + math.cos(theta) * ry0
        dx = rx - rx0 + disp_x
        dy = ry - ry0 + disp_y
        node.SetSolutionStepValue(
             KratosMultiphysics.MESH_DISPLACEMENT_X, dx)
        node.SetSolutionStepValue(
             KratosMultiphysics.MESH_DISPLACEMENT_Y, dy)
```

Calculate the DoFs using the respective structural solver for

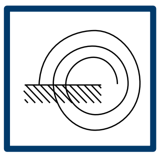
⇒ displacement_X



⇒ displacement_Y



⇒ rotation

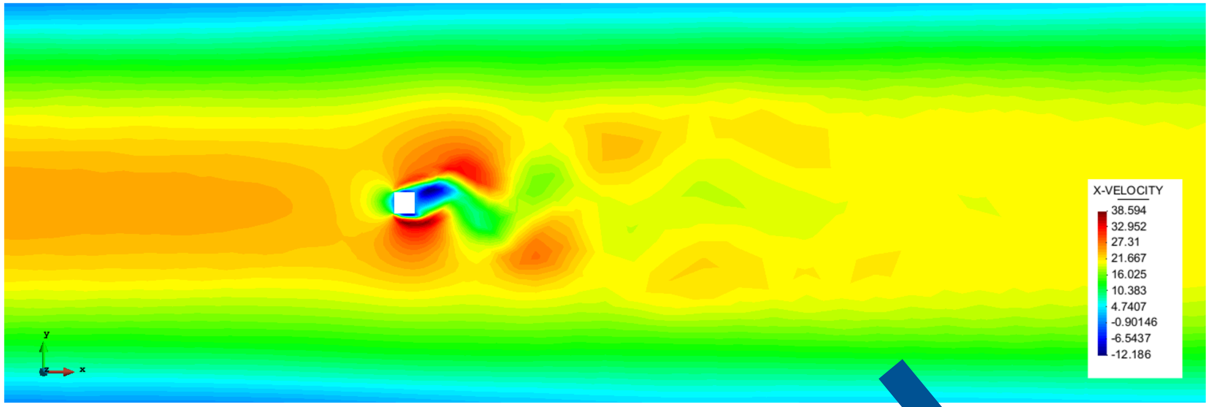


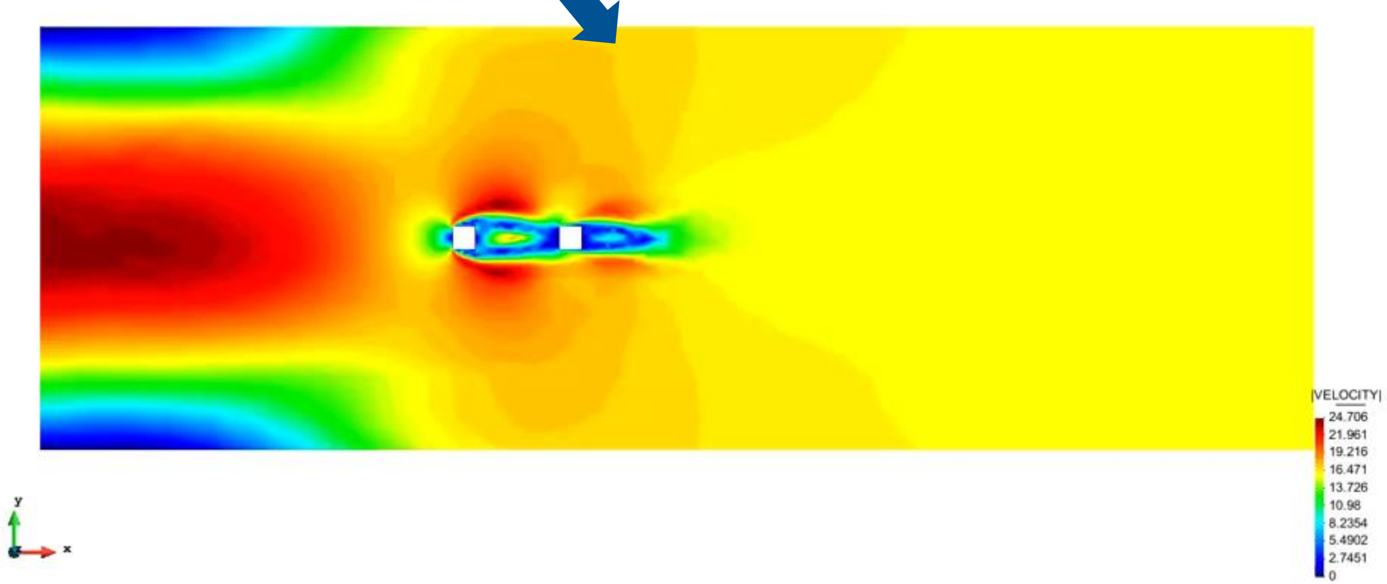
FSI OF MULTIPLE INTERFERING BODIES

[B. Chandra, H. L. Weng, S.v. Wenczowski, M. Péntek, A. Winterstein:
Partitioned Multi-Physics Simulation of Interfering Square Cylinders in a Flow,
FEM for FSI project, Statik TUM, 2017]

FSI: modeling of multiple interfering bodies

Open in GiD postprocess Version1 and Version2 results from “ FSIInterfering“





TOWARDS A GENERIC FSI CASE

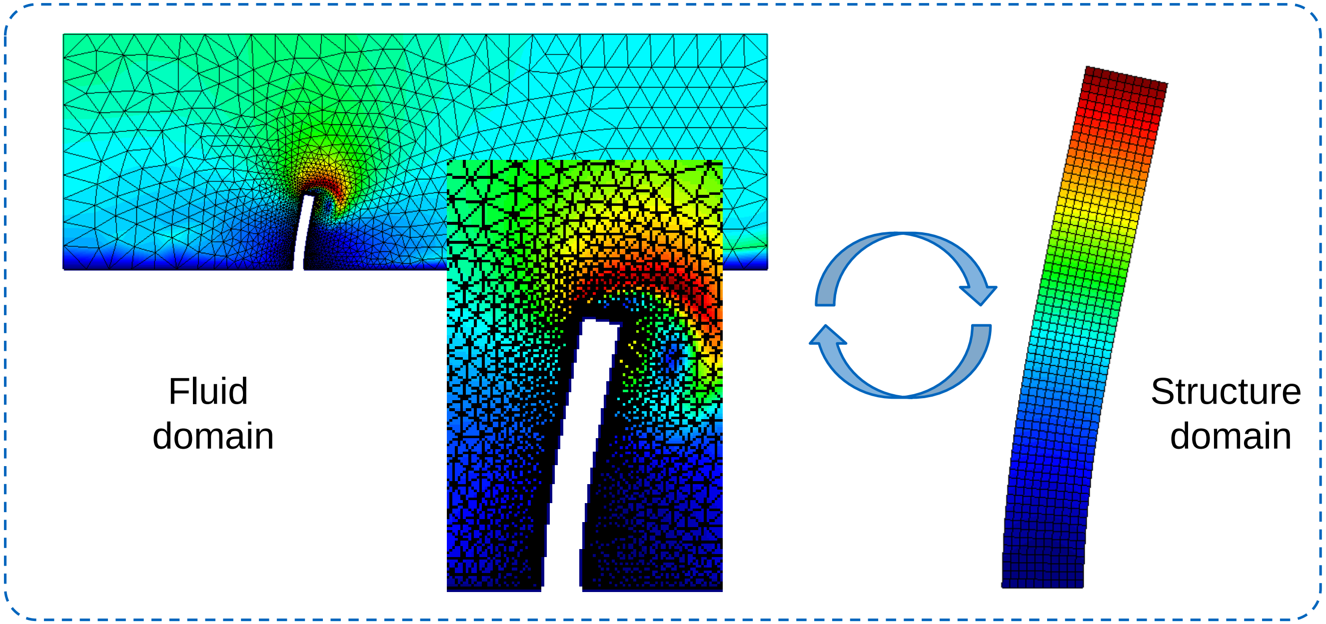
Differences compared to the example of the horizontal cut

CFD setup

- Custom inlet ✓
- Structural solver ✓
- Mesh mover ✓
- Coupling of physics and numerics ✓

FSI setup

Structural solver ⇒ A generalized MDoF solver (*X* and *Y* displacement possible for each node of the structure) ⇒ FEM formulation



Coupling of physics and numerics

⇒ using *matching grids*: each node of the interface of the structure has a corresponding node (with the same coordinates) in the fluid domain

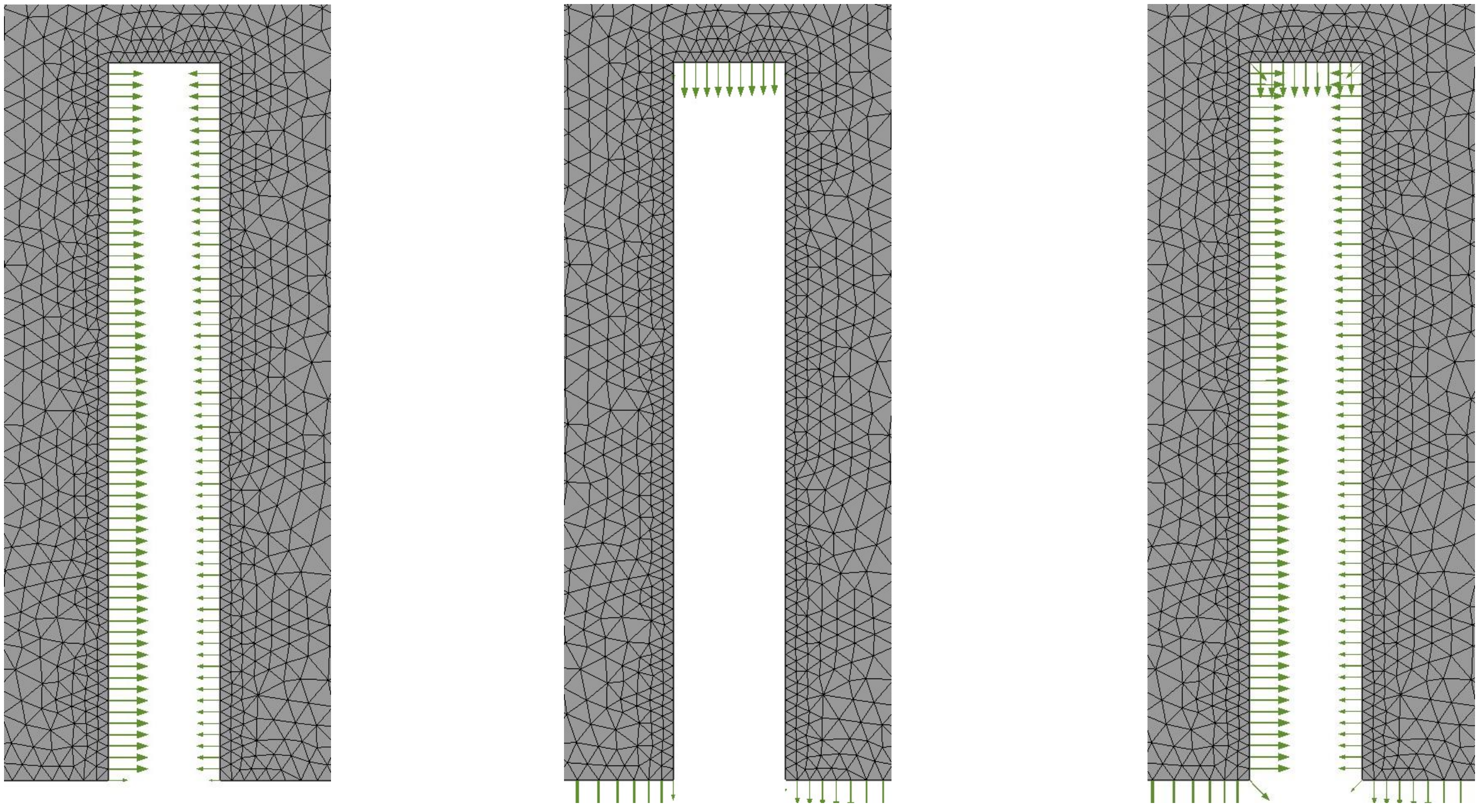
⇒ the forces of the fluid are not reduced to a certain point (previously center of gravity), but passed to the corresponding node in the structural solver

⇒ vectorial formulation of the residual for convergence

⇒ *Fix-point* and *Aitken* relaxation

FSI: Kratos CFD + Kratos CSD coupling – MDoF/general FEM case

Reaction *X* Reaction *Y*

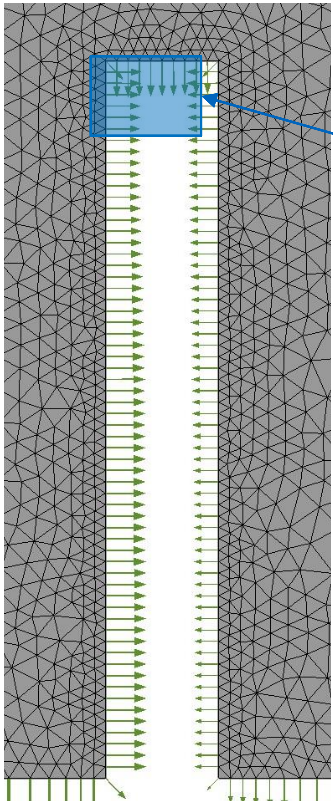


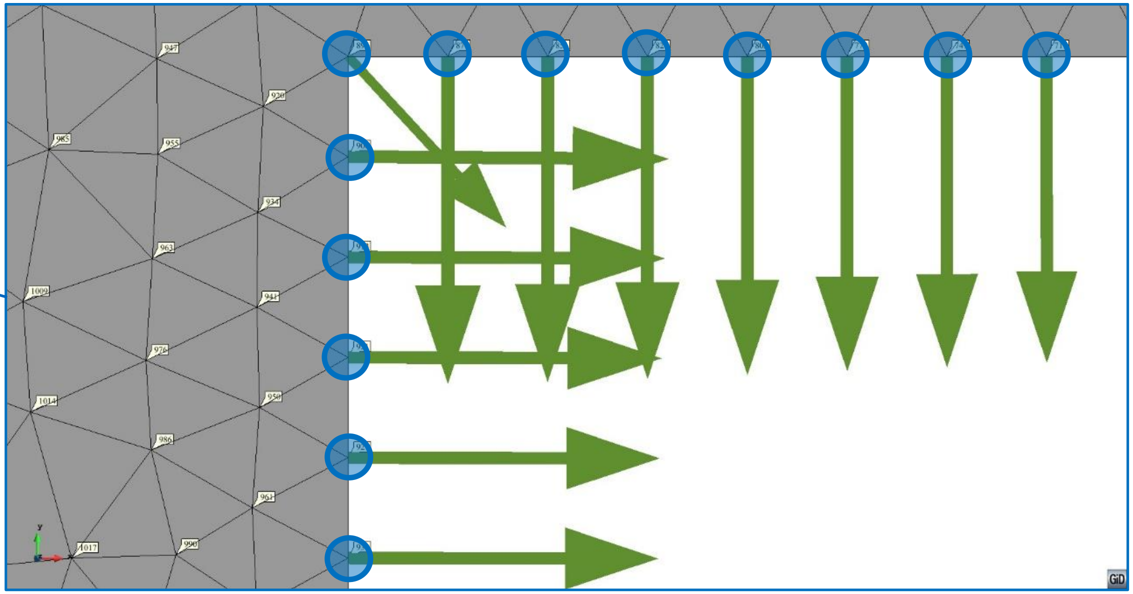
Reaction ⇒ result of the CFD simulation, here 2D case of vertical cut ⇒ reaction extraction and „preparing/sending“ to the structural (generalized MDoF/FEM) solver

- Locate structure interface in fluid domain
- Loop over the nodes on the interface
- Extract reactions from the nodes

Kratos CFD + Kratos CSD coupling – MDoF/general FEM case

Reaction magnitude



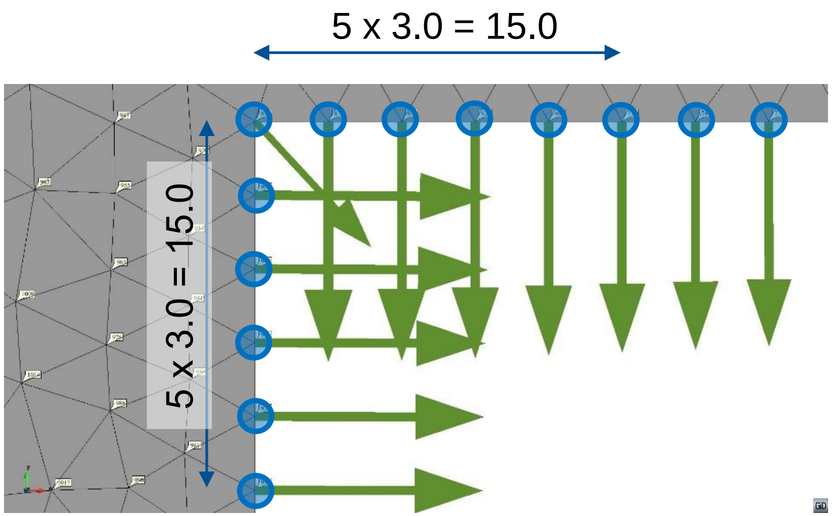


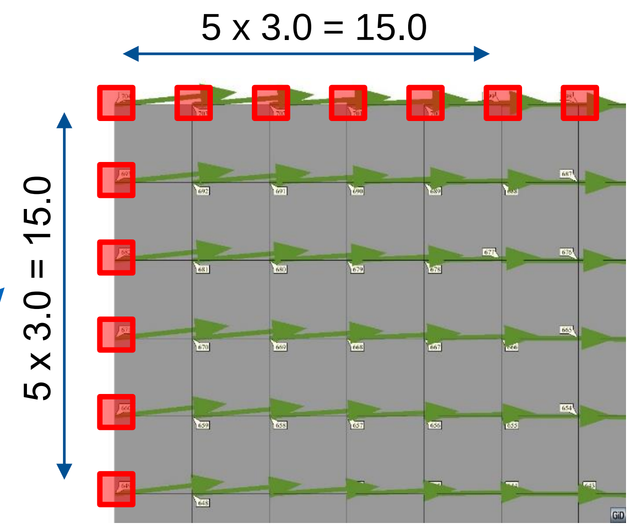
Node

- Coordinates: *node.X, node.Y, node.Z*
- Id number: *node.Id*
- Nodal data: pressure, velocity, reaction, displacement
- Retrieving nodal solutions – from the current time step using the commands:
  - `node.GetSolutionStepValue(PRESSURE, 0)`
  - `node.GetSolutionStepValue(VELOCITY, 0)`
  - `node.GetSolutionStepValue(REACTION, 0)`
  - `node.GetSolutionStepValue(DISPLACEMENT, 0)`
- Interested in nodes on the interface of the structure ⇒ nodes part of a certain *Submodel Part*

```
for node in main_model_part_fluid.GetSubModelPart(structure)
.Nodes:
```

Kratos CFD + Kratos CSD coupling – MDoF/general FEM case

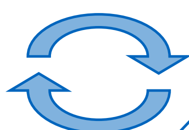


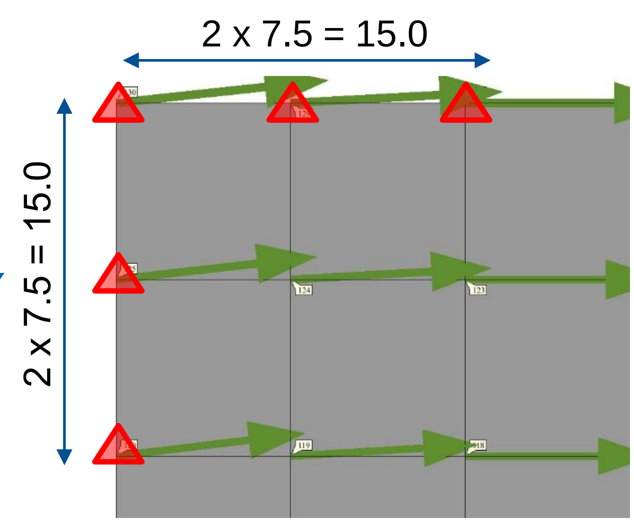


CFD – Reaction magnitude

⇒ needs to be „communicated/sent" to the structural solver

⇒ mesh size on the *structure interface of the CFD domain* is *3.0*



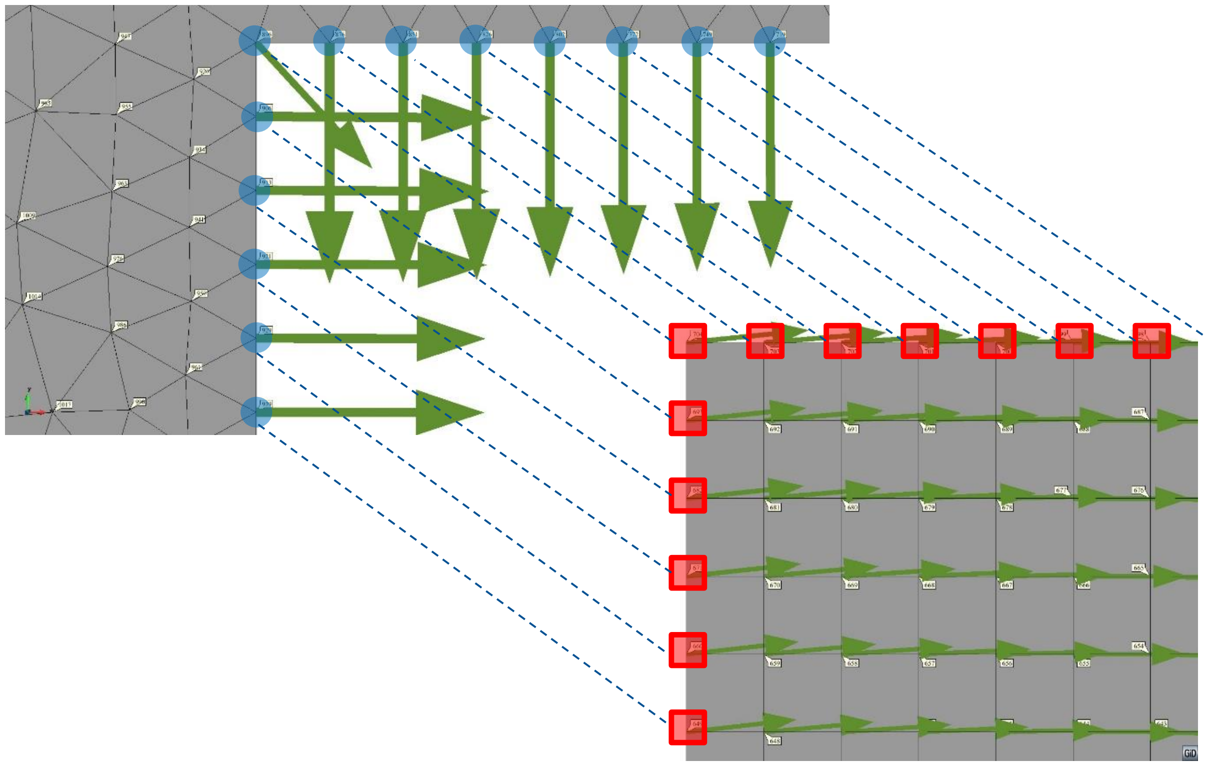


CSD – Displacement magnitude

⇒ needs to be „communicated/sent" to the fluid (or mesh) solver

⇒ mesh size on the *structure interface of the CSD domain* is *3.0 (**matching**)* and *7.5 (**non-matching**)*

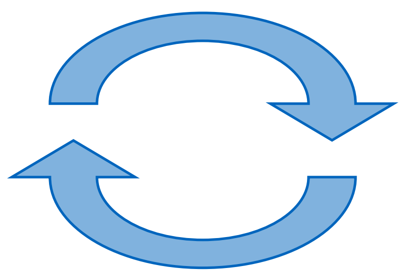
Kratos CFD + Kratos CSD coupling ⇒ matching grid



CFD – Reaction magnitude

⇒ needs to be „communicated/sent” to the structural solver

⇒ mesh size on the *structure interface of the CFD domain* is *3.0*



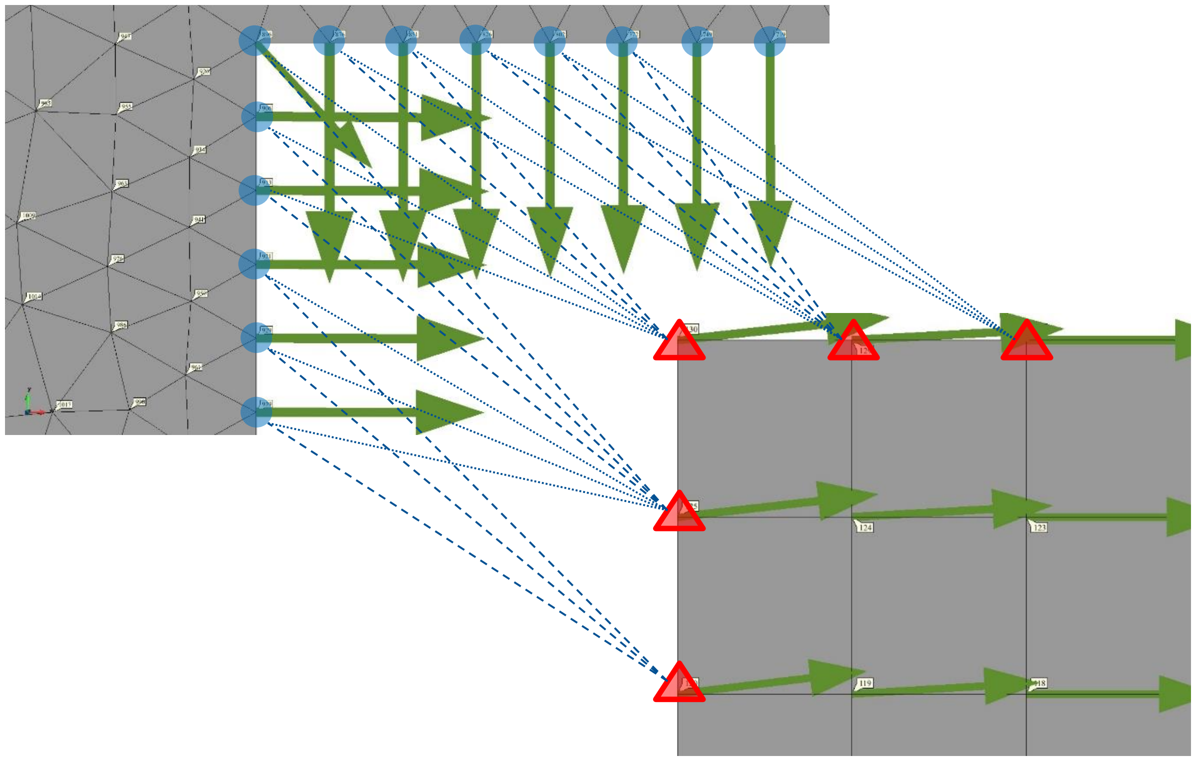
CSD – Displacement magnitude

⇒ needs to be „communicated/sent” to the fluid (or mesh) solver

⇒ mesh size on the *structure interface of the CSD domain* is *3.0* (***matching the one on the CFD side***)

Matching nodes on interface ⇒ needs a searching, matching and storing algorithm for pairing nodes

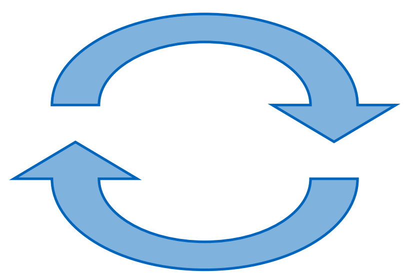
Kratos CFD + Kratos CSD coupling ⇒ <u>non-matching</u> grid



CFD – Reaction magnitude

⇒ needs to be „communicated/sent" to the structural solver

⇒ mesh size on the *structure interface of the CFD domain* is *3.0*



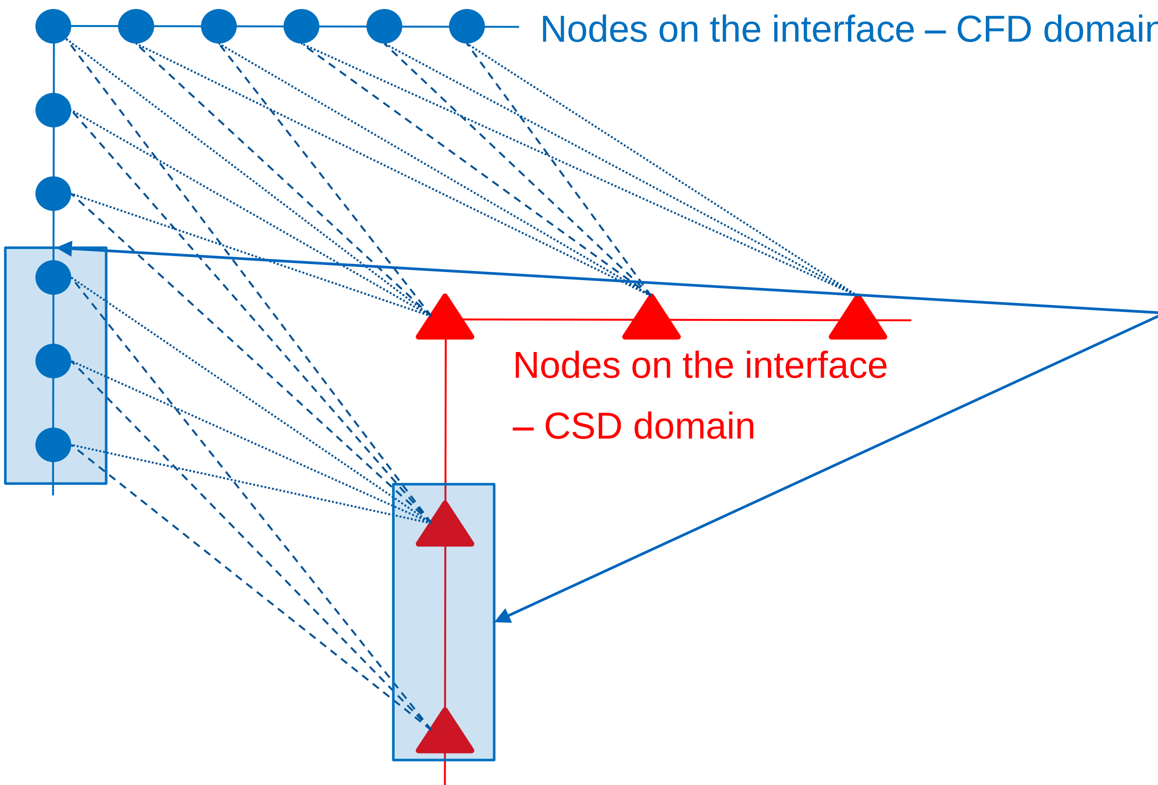
CSD – Displacement magnitude

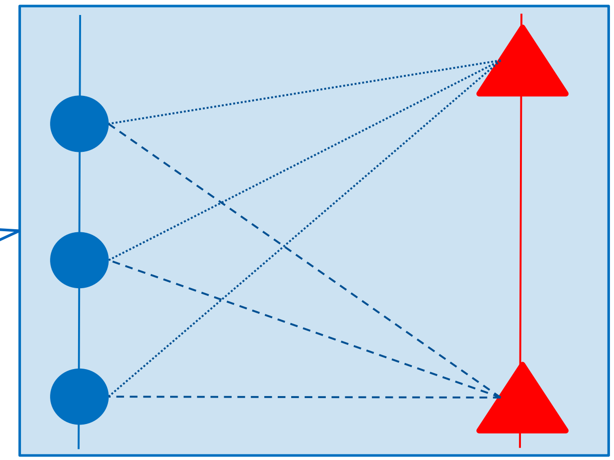
⇒ needs to be „communicated/sent" to the fluid (or mesh) solver

⇒ mesh size on the *structure interface of the CSD domain* is *3.0* (***matching the one on the CFD side***)

Matching nodes on interface ⇒ needs a searching, matching and storing algorithm for pairing nodes

Kratos CFD + Kratos CSD coupling ⇒ <u>non-matching</u> grid





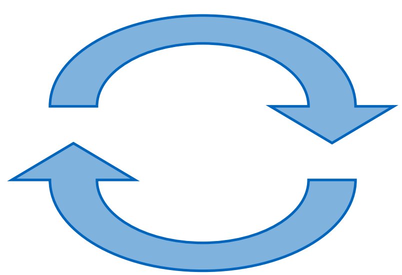
Assumption:

CFD mesh finer than
CSD on the interface

CFD – Reaction magnitude

⇒ needs to be „communicated/sent" to the structural solver

⇒ mesh size on the *structure interface of the CFD domain* is *3.0*



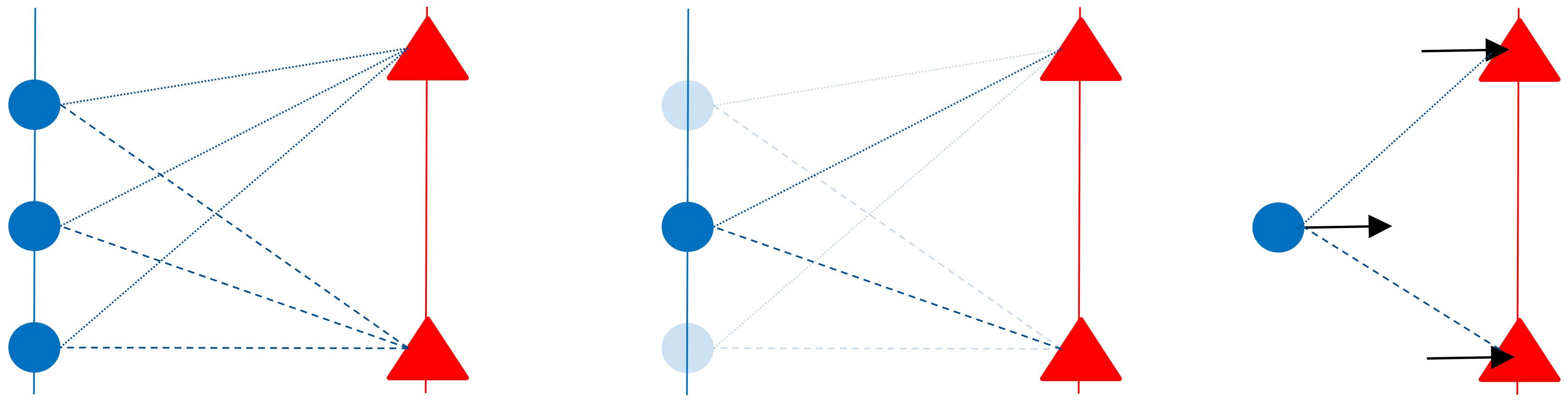
CSD – Displacement magnitude

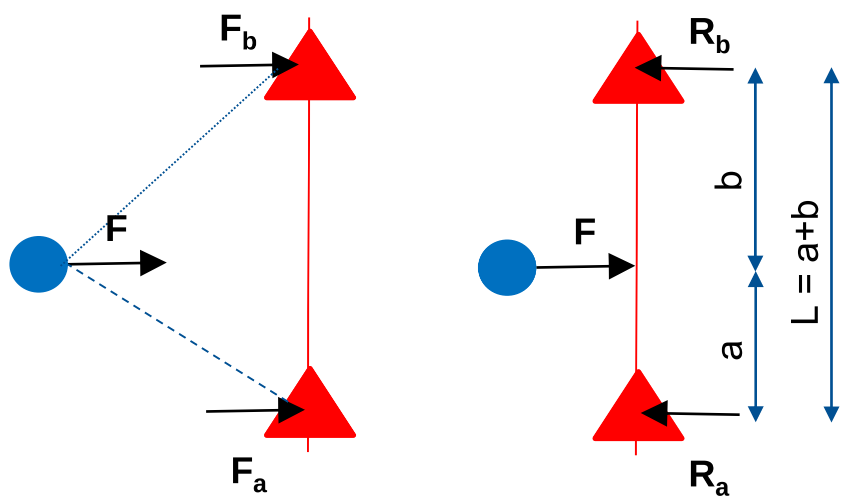
⇒ needs to be „communicated/sent" to the fluid (or mesh) solver

⇒ mesh size on the *structure interface of the CSD domain* is *7.5* (***not matching the one on the CFD side***)

Non-matching nodes on interface ⇒ each fluid node is connected to the closest 2 structural nodes ✓

Kratos CFD + Kratos CSD coupling ⇒ non-matching grid





Action/point load on the structure seen as a distance weighted value of the reaction of the fluid

$$F_b = F * w_b = F * a / (a+b)$$
$$F_a = F * w_a = F * b / (a+b)$$

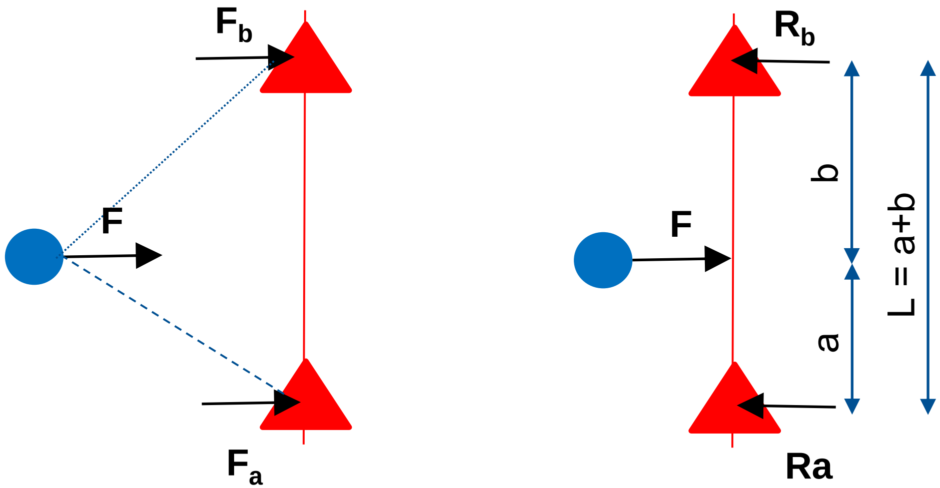
⇒ analogy with the point load on a simply supported beam

⇒ structure nodes and respective weights (*wa*, *wb*) for each fluid node on the interface have to be searched for, computed and stored

⇒ one way of data transformation between different discretizations

Non-matching nodes on interface ⇒ each fluid node is connected to the closest 2 structural nodes ✓

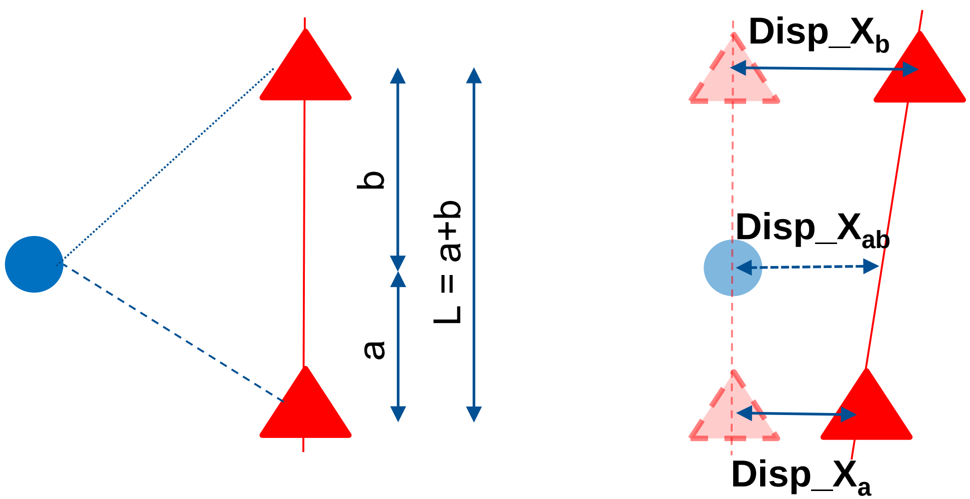
Kratos CFD + Kratos CSD coupling ⇒ non-matching grid



Action/point load on the structure seen as a distance weighted value of the reaction of the fluid

$$F_b = F * w_b = F * a / (a+b)$$
$$F_a = F * w_a = F * b / (a+b)$$

⇒ analogy with the point load on a simply supported beam



Displacement of the nodes on the structural interface of the CSD solver sent to the paired node on the fluid side

⇒ linear interpolation of the displacements $Disp\_X_b$ and $Disp\_X_a$ for the node of the fluid mesh

$$Disp\_X_{ab} = Disp\_X_b * w_b + Disp\_X_a * w_a$$

⇒ using the respective weights (*wa*, *wb*) already calculated and stored

Non-matching nodes on interface ⇒ each fluid node is connected to the closest 2 structural nodes ✓

Kratos CFD + Kratos CSD coupling ⇒ non-matching grid: important code snippets highlighted

Apply forces from fluid on structure

Compare: fsi_utilities.py lines 235-431

```
def InverseMap(self, origin_variable, destination_variable, swap_sign):

    origin_var_name = origin_variable.Name()
    origin_var_comp_x = KratosMultiphysics.KratosGlobals.GetVariable(origin_var_name + "_X")
    origin_var_comp_y = KratosMultiphysics.KratosGlobals.GetVariable(origin_var_name + "_Y")
```

Get point_loads

```
    for destination_node in self.destination_interface.Nodes:
        coupling_for_destination_node = self.coupling_matrix[destination_node.Id]
        node_1_id = coupling_for_destination_node[0]
        node_2_id = coupling_for_destination_node[1]
        node_1_w = coupling_for_destination_node[2]
        node_2_w = coupling_for_destination_node[3]
```

get corresponding two structure nodes and weights (stored in CouplingMatrix)

```
        val_comp_x = multiplicator * destination_node.GetSolutionStepValue(destination_var_comp_x, 0)
        val_comp_y = multiplicator * destination_node.GetSolutionStepValue(destination_var_comp_y, 0)
```

get the reaction force from the fluid node

```
        origin_node_1_val_comp_x = val_comp_x * node_1_w
        origin_node_1_val_comp_y = val_comp_y * node_1_w
        origin_node_2_val_comp_x = val_comp_x * node_2_w
        origin_node_2_val_comp_y = val_comp_y * node_2_w
```

compute the resulting reaction forces on the structure nodes

add the forces already subjected from other fluid nodes

```
        origin_node_1_val_comp_x += self.origin_interface.Nodes[node_1_id].GetSolutionStepValue(origin_var_comp_x, 0)
        origin_node_1_val_comp_y += self.origin_interface.Nodes[node_1_id].GetSolutionStepValue(origin_var_comp_y, 0)
        origin_node_2_val_comp_x += self.origin_interface.Nodes[node_2_id].GetSolutionStepValue(origin_var_comp_x, 0)
        origin_node_2_val_comp_y += self.origin_interface.Nodes[node_2_id].GetSolutionStepValue(origin_var_comp_y, 0)
```

set the summmed up forces on the nodes

```
        self.origin_interface.Nodes[node_1_id].SetSolutionStepValue(origin_var_comp_x, 0, origin_node_1_val_comp_x)
        self.origin_interface.Nodes[node_1_id].SetSolutionStepValue(origin_var_comp_y, 0, origin_node_1_val_comp_y)
        self.origin_interface.Nodes[node_2_id].SetSolutionStepValue(origin_var_comp_x, 0, origin_node_2_val_comp_x)
        self.origin_interface.Nodes[node_2_id].SetSolutionStepValue(origin_var_comp_y, 0, origin_node_2_val_comp_y)
```

Kratos CFD + Kratos CSD coupling ⇒ non-matching grid: important code snippets highlighted

Apply displacements of structure on fluid mesh

Compare: fsi_utilities.py: lines 294-330

```
def Map(self, origin_variable, destination_variable):

    for destination_node in self.destination_interface.Nodes:
```
loop over all fluid nodes at interface

```
        coupling_for_destination_node = self.coupling_matrix[destination_node.Id]
        node_1_id = coupling_for_destination_node[0]
        node_2_id = coupling_for_destination_node[1]
        node_1_w = coupling_for_destination_node[2]
        node_2_w = coupling_for_destination_node[3]
```
get corresponding two structure nodes and weights (stored in CouplingMatrix)

get the displacements of both structure nodes

```
        origin_node_1_val_comp_x = self.origin_interface.Nodes[node_1_id].GetSolutionStepValue(origin_var_comp_x, 0)
        origin_node_1_val_comp_y = self.origin_interface.Nodes[node_1_id].GetSolutionStepValue(origin_var_comp_y, 0)
        origin_node_2_val_comp_x = self.origin_interface.Nodes[node_2_id].GetSolutionStepValue(origin_var_comp_x, 0)
        origin_node_2_val_comp_y = self.origin_interface.Nodes[node_2_id].GetSolutionStepValue(origin_var_comp_y, 0)
```

compute the fluid node displacement by interpolation using the weights

```
        destination_node_val_comp_x = origin_node_1_val_comp_x * node_1_w + origin_node_2_val_comp_x * node_2_w
        destination_node_val_comp_y = origin_node_1_val_comp_y * node_1_w + origin_node_2_val_comp_y * node_2_w
```

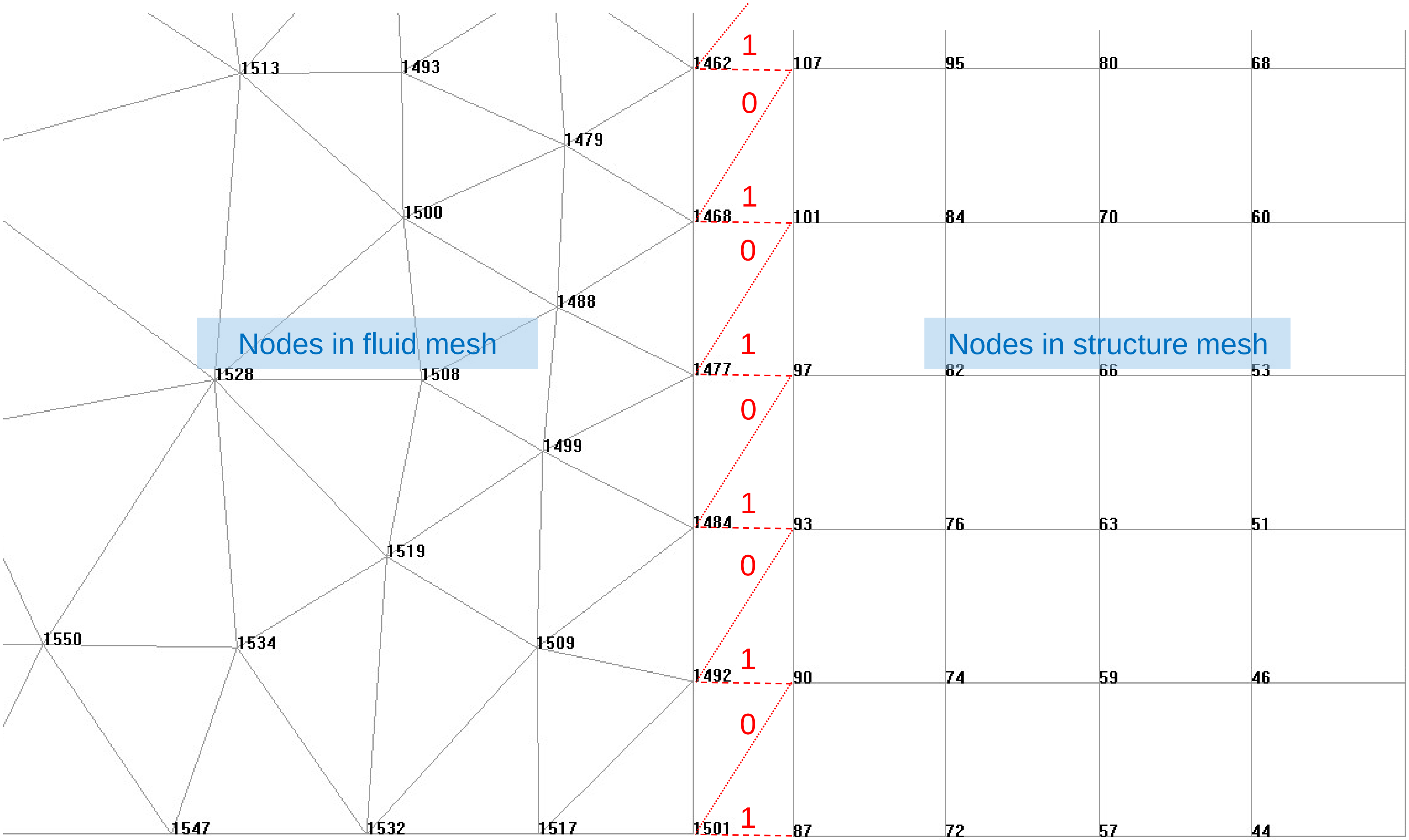
set the displacement of the fluid node

```
        destination_node.SetSolutionStepValue(destination_var_comp_x, 0, destination_node_val_comp_x)
        destination_node.SetSolutionStepValue(destination_var_comp_y, 0, destination_node_val_comp_y)
```

Kratos CFD + Kratos CSD coupling ⇒ non-matching grid *map*

Mapping algorithm for a matching grid

⇒ each fluid node is linked to the closest 2 structural nodes (for this case weigths *1* and *0*)



Kratos CFD + Kratos CSD coupling ⇒ non-matching grid *map*

Mapping algorithm for a matching grid

⇒ each fluid node is linked to the closest 2 structural nodes (for this case weights *1.0* or *0.0*)

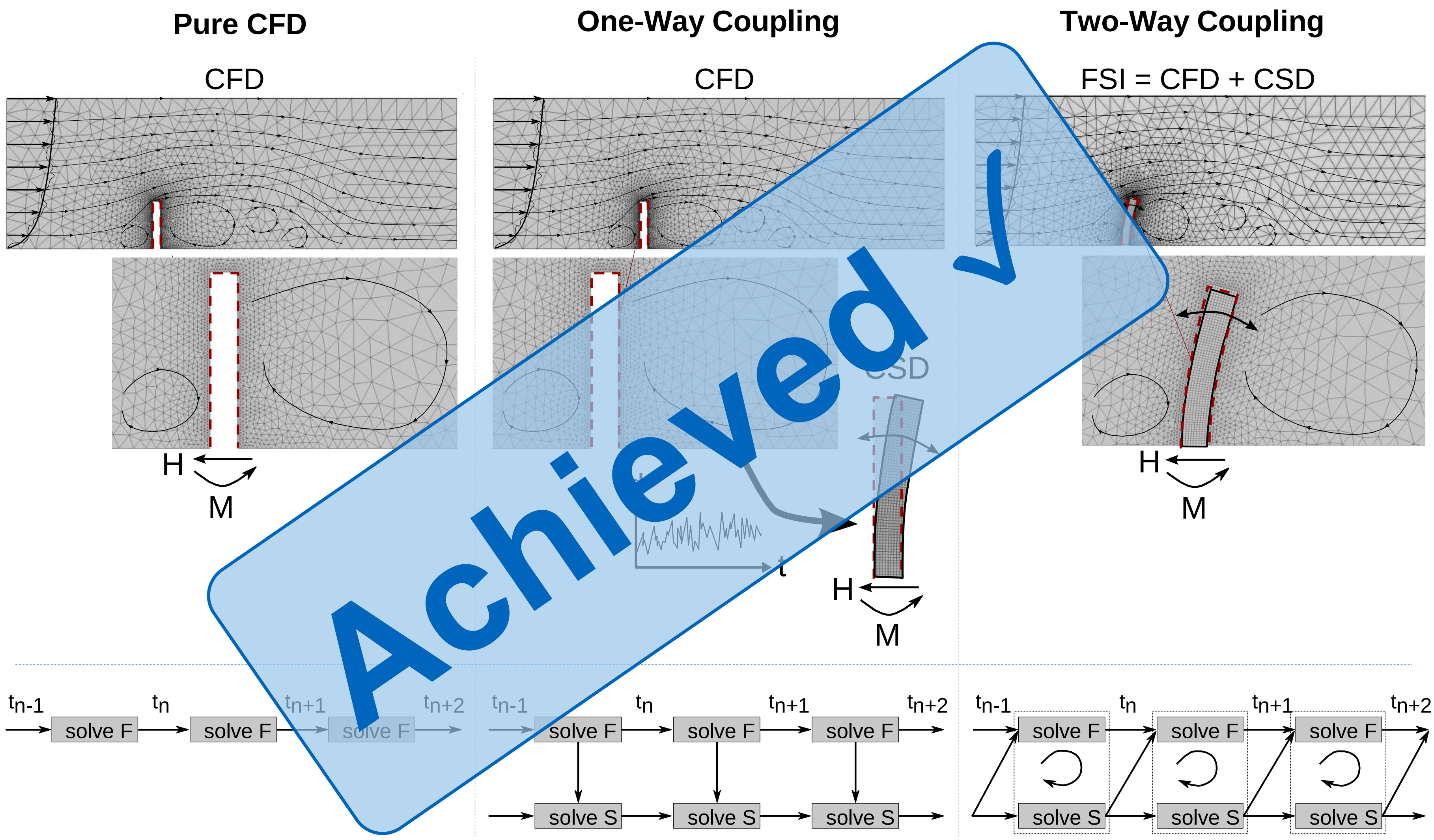
```
Fluid Node ID            StructNode1 ID          StructNode2 ID
StructNode1 weight    StructNode2 weight
…
1421        146         138         1.0         0.0
1432        138         146         1.0         0.0
1438        129         122         1.0         0.0
1446        122         129         1.0         0.0
1452        114         107         1.0         0.0
1462        107         114         1.0         0.0
1468        101         97          1.0         0.0
1477        97          93          1.0         0.0
1484        93          97          1.0         0.0
1492        90          87          1.0         0.0
1501        87          90          1.0         0.0
```

Mapping algorithm for a non-matching grid

⇒ each fluid node is linked to the closest 2 structural nodes (for this case weights take values from the *[0,1]* interval)

```
Fluid Node ID            StructNode1 ID          StructNode2 ID
StructNode1 weight     StructNode2 weight
…
1421        18          20          0.984       0.016
1432        18          15          0.714       0.286
1438        15          18          0.587       0.413
1446        15          18          0.889       0.111
1452        15          13          0.81        0.19
1462        15          13          0.508       0.492
1468        13          15          0.794       0.206
1477        13          10          0.905       0.095
1484        13          10          0.603       0.397
1492        10          13          0.698       0.302
1501        10          13          1.0         0.0
```

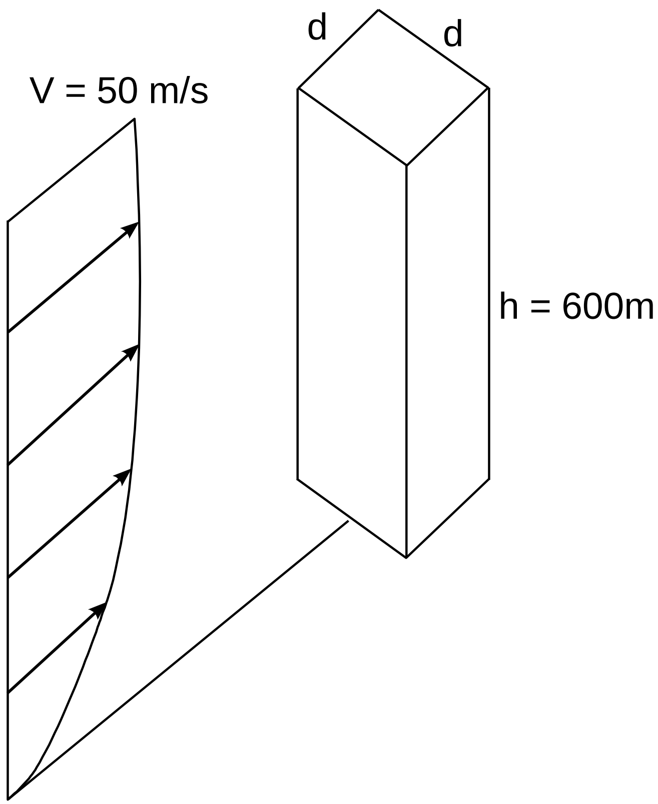
The goal ⇒ to you get familiar with the crucial components of CWSI



ADDITIONAL NOTES AND REMARKS

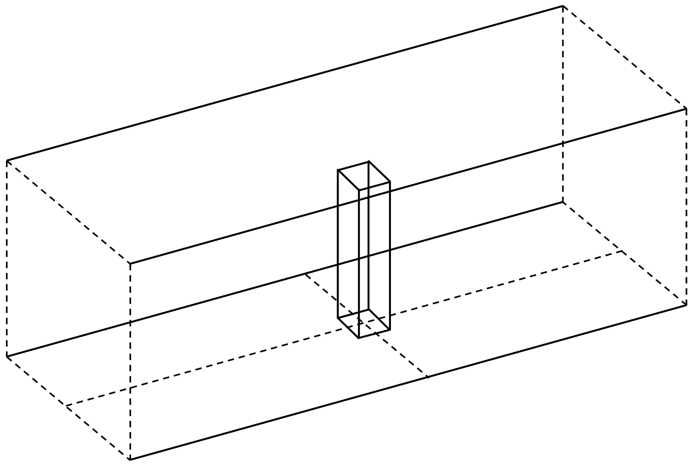
When using a simplified (2D) setup of a skyscraper (3D)…

Real case / setup

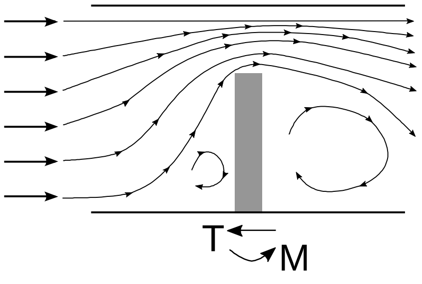


assume h/d =10
square cross section

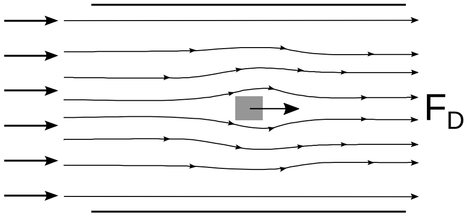
Numerical wind-tunnel setup



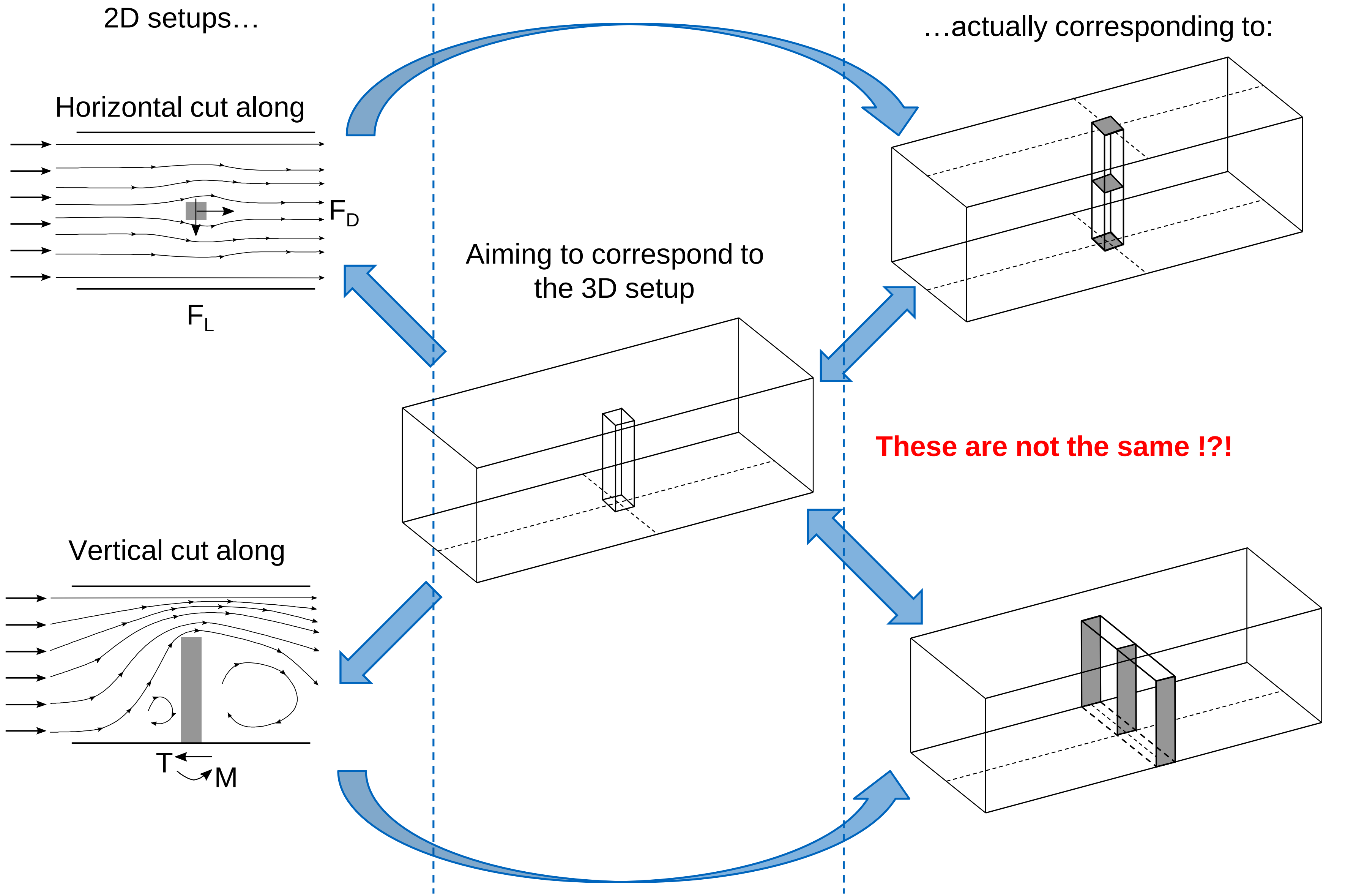
Vertical cut along



Horizontal cut along



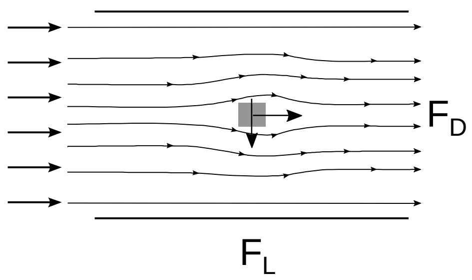
…be aware of the consequences of dimensional reduction

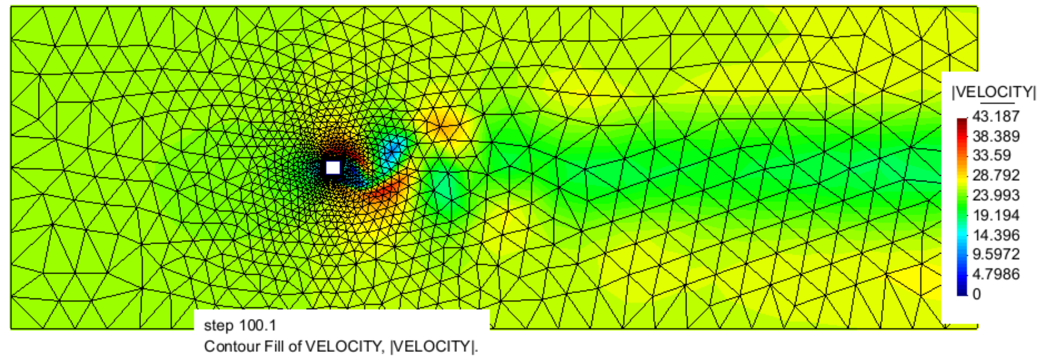


These are not the same !?!

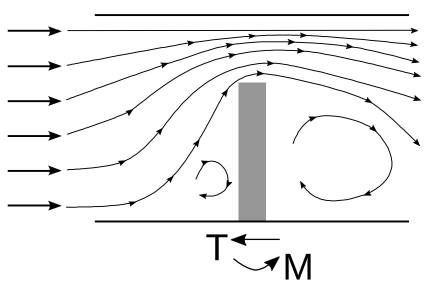
…be aware of the consequences due to modeling aspects („disclaimer“)

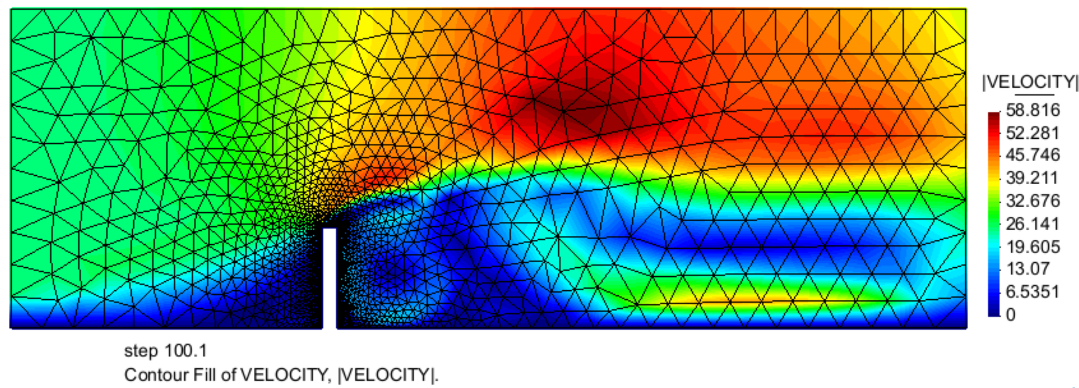
Horizontal cut along





Vertical cut along





We are here running (mostly) 2D simulations of an inherently 3D physical phenomenon

⇒ recall that turbulence is a 3D phenomenon

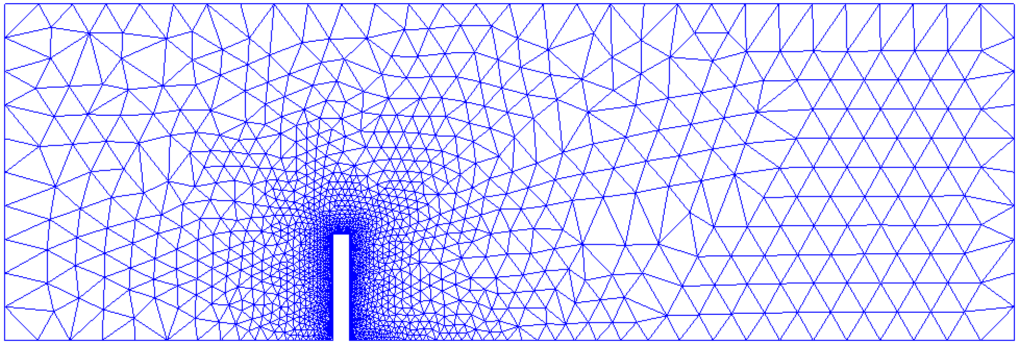
- Also large 3D phenomena in flow field are not captured, like e.g. „end effects“, „flow around sides“, …
- Concerning turbulence: RANS model for 2D or running the LES(-type) simulations in 3D
- Methodology, presented components and workflow are nonetheless valid
- Presented tutorials are adequate for educational and prototyping purposes (limits the computational necessities: hardware, time, etc!)

OUTLOOK

Advanced topics: FSI in 3D & respective challenges ⇒ for e.g w.r.t. mapping…

How to map to a structure modeled and discretized by e.g. beam elements (dimensional reduction!)?
→ several application cases in wind engineering: bridges, towers, skyscrapers, wind turbine blades, …

CFD – 2D



CFD – 3D