Fluid-Structure Interaction Simulation with Multiple Interfaces

Mahmoud Ammar, Guillermo Barraza, Theresa Pasch*

Abstract

This report aims to explain the literature review, work flow, and results of a group project done in the summer semester 2018 at the Chair of Structural Analysis at the Technical University of Munich supervised by Aditya Ghantasala. The tasks were to implement different coupling algorithms for two and three springs systems, for a three coupled beams system, and finally solve an FSI problem with two interfaces. In this present work, the first two tasks were implemented in Python and different relaxation methods were used to compare the results, and the last task was implemented in Python using Kratos Multiphysics for the analysis, and GiD for visualization.

Keywords: Fluid-Structure Interaction, Relaxation, Multiple Interfaces, Kratos Multiphysics.

Introduction

Fluid-structure interaction (FSI) is a multiphysics problem which includes the interaction between a structure and its surrounding fluid. An FSI problem has two coupling approaches [Fig. 1] which are; first, a monolithic approach where the fluid and the structure solvers are combined into one, setting up one system of equations; second, a partitioned approach where the fluid and the structure have their separate solvers and the coupling of the interface occurs by exchanging their solutions with each other as a boundary condition. In the partitioned approach, the same convergence rate as the monolithic approach can be achieved using Newton method if the Jacobian matrix of the fluid and structure systems are already known. Nonetheless, since in most of the cases the Jacobian matrix cannot be directly obtained, it is approximated by using quasi-newton methods which will be discussed in the following chapter.

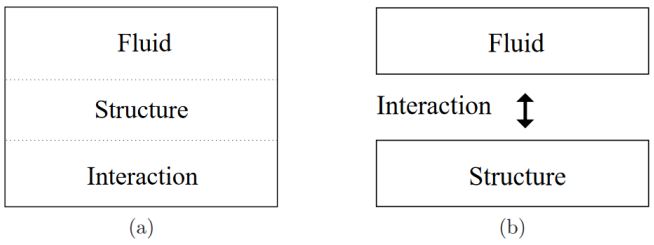


Figure 1: (a) Monolithic approach (b) Partitioned approach [1]

To solve a FSI problem, a partitioned approach has a better flexibility than a monolithic one, but it lacks of robustness. In other words, having a better flexibility means, with the right coupling environment (e.g. mapping the data between the meshes, and taking into consideration the time step's stability

* *Technische Universität München, Chair of Structural Analysis*

requirements), there will be a wider flexibility to choose the solvers of the fluid and structure separately.

There are two main types of coupling in FSI: *weak* and *strong* coupling. One talks of a *weak* coupling if, while solving the whole problem, one of the sub-models is dominant over the other. On the other hand, one talks of a *strong* coupling when both sub-models totally depend on each other. Furthermore, a typical partitioned FSI coupling scheme has a work flow as it is shown in Fig. 2, and it is explained as the following:

1. The computed force is passed from the fluid solver to the structure solver
2. Displacement for the next time step is computed in the structure solver
3. Computed displacement is passed to the fluid solver in the previous time step
4. Force for the next time step is computed in the fluid solver

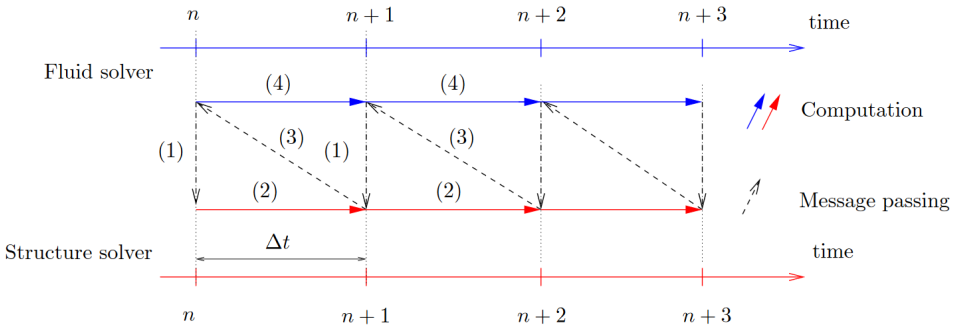


Figure 2: Coupling work flow between structure and fluid [2]

A multiple interface problem is when there are two or more FSI interfaces in which their behaviour is dependent on each other. This can be seen in many real life problems like e.g. the FSI analysis of a wind turbine farm [Fig. 3], since the behavioural effect of wind turbines on each other has a significant effect on their performance. In this report, the multiple interface problem will be discussed with a simple case of two structures with an incompressible fluid flow around them, and the analysis is done using Kratos Multiphysics.

Figure 3: Example: Wind turbine farm (source: www.carbonbrief.org)

1 Relaxation Methods

Relaxation methods are used in order to enforce strong coupling into the system to reach equilibrium. In this section, fixed point iteration, Aitken, and quasi-newton relaxation methods will be discussed.

1.1 Fixed point iteration

Fixed point iteration method is used to iterate in the same time step until the solution converges to a pre-specified criteria. An important factor in this approach is the chosen value of the relaxation factor ω which is a number between 0 and 1, and a value of 1 means there is no relaxation applied in the system. It is defined by Eq. 1.

$$x_{k+1} = \omega * \tilde{x}_{k+1} + (1-\omega) * x_k \tag{1}$$

1.2 Aitken

The Aitken method is an improvement of the fixed point iteration method where vector extrapolation is performed to achieve a faster convergence. In other words, the value of the relaxation factor ω is not constant anymore since there is an extra step after every fixed point iteration in order to compute a new ω for the next iteration step, which consists on making use of the residual of the previous and the current iteration (r_k and r_{k+1}) as seen in Eq. 2.

$$\omega_{k+1} = \omega_k * \frac{r_k}{r_k - r_{k+1}} \tag{2}$$

1.3 IQN-ILS

IQN-ILS stands for Interface Quasi-Newton method with Inverse jacobian from a Least-Squares model. This method approximates the inverse of the Jacobian matrix of the nonlinear system from a vector of the residuals and predictions computed from the previous iterations, making the quasi-newton iterations look like as in Eq. 3 [3]. The fact that it contains more information than Aitken, should lead to better results.

$$x^{k+1} = x^k + \widehat{(\frac{dr}{dx})^{-1}} (- r^k) \tag{3}$$

An initial guess of x^0 is calculated from an extrapolation of the position of the interface from previous time steps. Also, The residual r^k in Eq. 3 is calculated using Eq. 4, such that $\tilde{x}^{k+1}$ is the prediction of the next iteration, and x^k is the converged solution of the previous iteration [3].

$$r^k = \tilde{x}^{k+1} - x^k \tag{4}$$

Two more important parameters are the iteration horizon and time step horizon. Iteration horizon is the maximum number of vectors that are used in each time step. The time step horizon is the number of time steps whose vectors will be used. But since the time step horizon cannot be determined easily, a different vector extrapolation method called MVQN [section 1.4] can be used which has faster convergence [4].

1.4 MVQN

MVQN stands for Multi-Vector update Quasi-Newton method. This method updates the Jacobian matrix from a vector of the residuals and approximated solutions from the previous iterations which is illustrated in Eq. 5.

$$J_k^{n+1} = J^n + (\Delta R_k^n - J^n \Delta X_k^n) [(\Delta X_k^n)^T \Delta X_k^n]^{-1} \tag{5}$$

Knowing the differencing matrices for both the residuals from the previous iterations ΔR_k^{n+1} [Eq. 6], and the approximated solutions of the previous iterations ΔX_k^{n+1} [Eq. 7] yields the following equations.

$$\Delta R_k^{n+1} = [r_k - r_{k-1},\ r_{k-1} - r_{k-2},\ ...] \tag{6}$$

$$\Delta X_k^{n+1} = [u_k - u_{k-1},\ u_{k-1} - u_{k-2},\ ...] \tag{7}$$

Finally, the quasi-newton iteration looks like Eq. 8 where the residual r_k is the same as in Eq. 4.

$$x_k^{n+1} = x_k^n - (J_k^{n+1})^{-1} r_k \tag{8}$$

It is important to note that since MVQN needs to start after two iterations in order to approximate the Jacobian, a relaxation method is used for the first iteration, and the initial Jacobian matrix of the first iteration of the first time step is assumed to be the identity matrix. Also, the number of differencing matrices should not be more than the degrees of freedom of the interface [5].

2 Two Coupled Linear Springs System

In order to validate the different relaxation methods mentioned in section 1, a benchmark of these is performed on a simple two springs - one mass system. The governing equation for a dynamic linear spring is Eq. 9. The coupling algorithm for the two springs is explained in algorithm 1, and the used relaxation methods were explained in section 1. The coupling, and relaxation methods used in this section are implemented in Python.

$$M \ddot{x} + k x = f \tag{9}$$

for *time in range end time* **do**
  while $u_B[n]$ *!= converged* **do**
    Calculate $F_A[n]$ using $u_B[n-1]$
    Calculate $u_B[n]$ using $F_A[n]$
    Residual = $u_B[n]$ - $u_B[n-1]$
    Perform relaxation on $u_B[n]$
  end
end

Algorithm 1: Coupling two springs system with relaxation

2.1 Time Integration

The time discretization used is the Second Order Backward Differencing Scheme (BDF2). The time derivative for this scheme is shown in Eq. 10 with the coefficients defined in Eq. 11.

$$\frac{\partial x}{\partial t} = c_0 x_{n+1} - c_1 x_n + c_2 x_{n-1} \tag{10}$$

$$c_0 = \frac{3}{2\Delta t}, \quad c_1 = \frac{-4}{2\Delta t}, \quad c_0 = \frac{1}{2\Delta t} \tag{11}$$

2.2 Optimum relaxation factor for fixed point iteration

In this problem, the constant relaxation factor ω is changed to check how increasing it affects the maximum number of iterations per time step. From the results in Fig.4 it can be observed that as ω increases till 0.9, the maximum number of iterations decreases significantly from 80 to less than 5 iterations per time step. On the other hand, if ω is chosen to be equal to 1 this implies there is no relaxation in the system since the result will be $x_{k+1} = \tilde{x}_{k+1}$ for every iteration. The optimum relaxation factor depends on the problem and has to be evaluated individually.

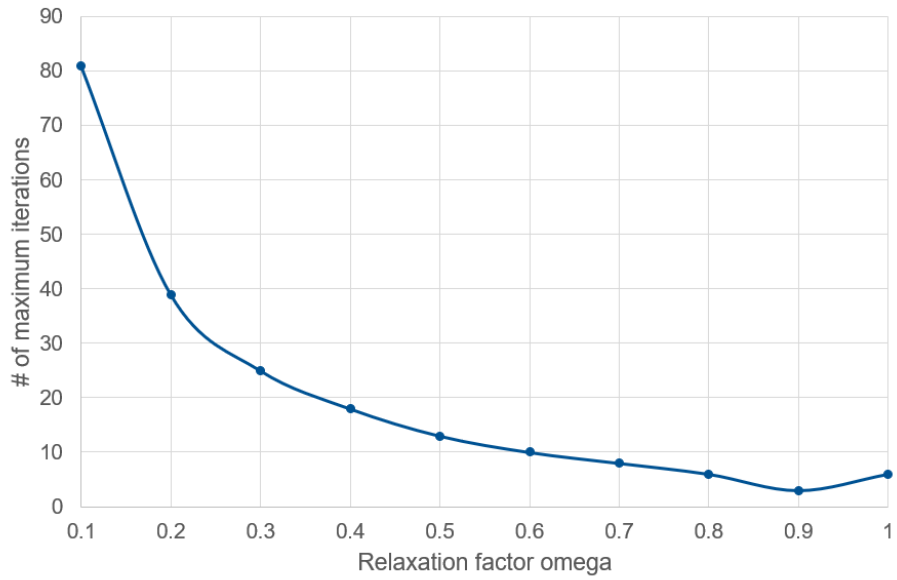


Figure 4: Variation of the maximum number of iterations per time step with relaxation factor ω

2.3 Comparison between different relaxation approaches

Figure 5 shows the results for the two coupled springs systems using all of the discussed relaxation methods. To have a better understanding of the loosely coupled and Dirichlet Neumann solutions, both are explained in algorithms 2 and 3, respectively. What makes Dirichlet Neumann different than loosely coupled is that in Dirichlet Neumann the solution of the spring is being iterated without relaxation until equilibrium is reached, but in loosely coupled there is only one iteration for each time step and no equilibrium is reached.

for *time in range end time* **do**
  Calculate $F_A[n]$ using $u_B[n-1]$
  Calculate $u_B[n]$ using $F_A[n]$
end

Algorithm 2: Loosely coupled

for *time in range end time* **do**
  while $u_B[n] \neq$ *converged* **do**
    Calculate $F_A[n]$ using $u_B[n-1]$
    Calculate $u_B[n]$ using $F_A[n]$
    Residual = $u_A[n]$ - $u_B[n]$
    $u_B[n-1] = u_B[n]$
  end
end

Algorithm 3: Dirichlet Neumann

This is why in Figure 5, the results for all of the relaxation methods used and the Dirichlet Neumann are similar to the analytical solution, but he loosely coupled is diverging because since it is only iterating once for every time step, there is an error margin that is accumulating through time steps. This also implies that doing only Dirichlet Neumann with no relaxation gives the same results as with relaxation, but it will take more iterations to converge.

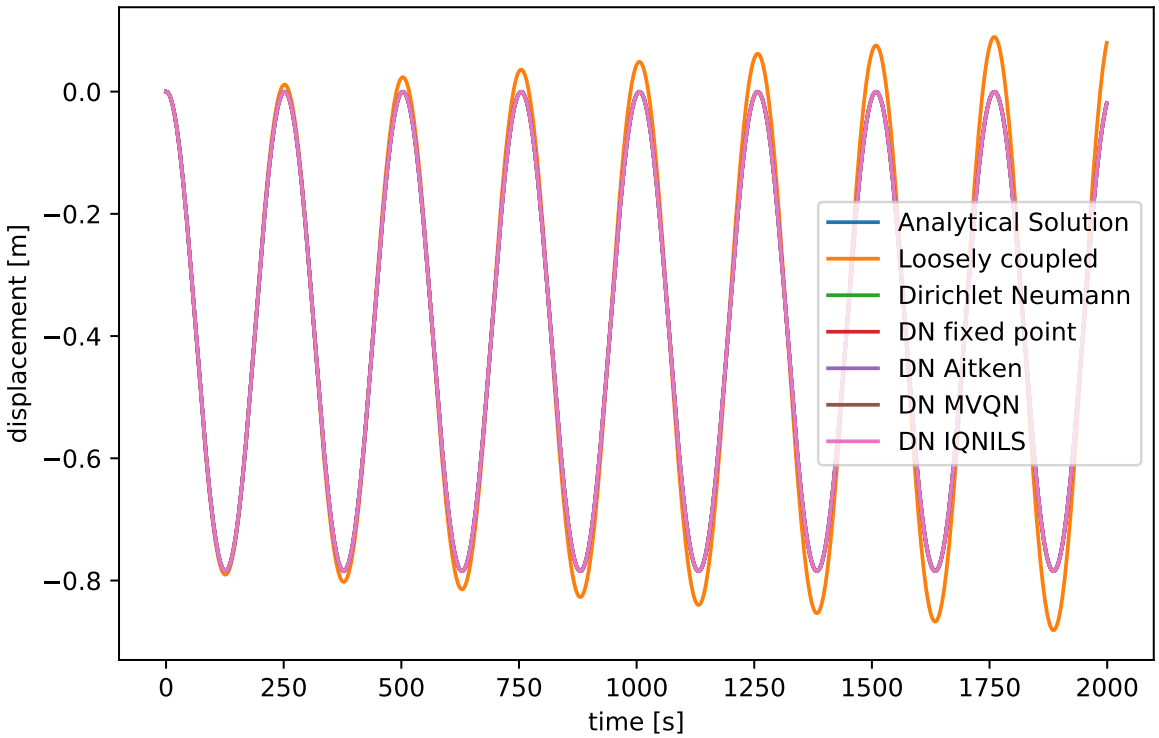


Figure 5: Resulting displacement for the two springs system using different relaxation methods

On the other hand, to conclude which relaxation method has the least number of iterations per time step and least total number of iterations, a comparison is listed in table 1, and it is concluded that the MVQN relaxation method for the case of two springs coupled system has the least number of iterations.

Table 1: Comparison of the number of iterations

Relaxation methods	Maximum iterations (per time step)	Total iterations (per 1000 time steps)
Without Relaxation	6	5586
Fixed Point (ω=0.9)	3	2368
Aitken	3	2999
IQNILS	6	5978
MVQN	3	2000

3 Three Coupled Linear Springs System

In order to validate the results obtained in Section 2, the coupling methods mentioned in Section 1 are implemented on a three springs - one mass dynamic linear system as depicted by the left system in Figure 6. The methodology followed to solve this problem is explained in Algorithm 4.

Numbers 1, 2 and 3 correspond to springs 1, 2 and 3 respectively, as depicted in Figure 6

for *time span* **do**
  $U_{pass} \leftarrow U_{2_{n-1}}$
  while $U_3 \neq U_{pass}$ **do**
    Compute F_3 using U_{pass}
    while $U_2 \neq U_{pass}$ **do**
      Compute F_1 using U_{pass}
      Re-compute U_2 using F_1 and F_3
      if $U_2 = U_{pass}$ *(or less than specified tolerance)* **then**
        break
      else
        Apply relaxation method to U_2
        $U_{pass} \leftarrow U_2$
      end
    end
    Compute U_3
    if $U_3 = U_{pass}$ *(or less than specified tolerance)* **then**
      break
    else
      Apply relaxation method to U_3
      $U_{pass} \leftarrow U_3$
    end
  end
end

REMARK: Unless it is explicitly specified, all variables are computed for the current time step n

Algorithm 4: Three springs - one mass coupling

The most remarkable difference between this algorithm and algorithm 1 relies on the presence of an inner loop. The nested loop here is important to take into account the contribution that every spring has on the displacement of the rest of the system.

To be able to determine whether the present algorithm works as expected, an equivalent system (shown in Figure 6) is implemented, where two of the springs can be assumed as one when their stiffnesses are added up.

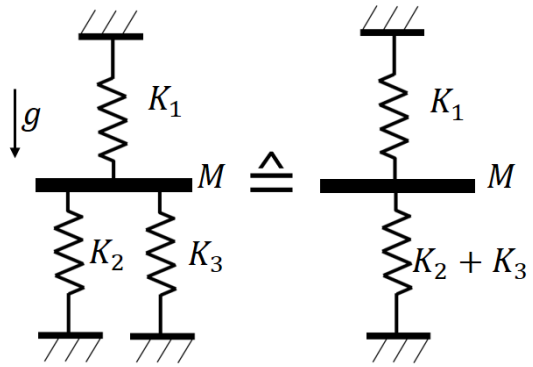


Figure 6: Equivalence between a dynamic two springs linear system and a three springs linear system

The resulting displacement of the mass, using the different relaxation methods[1] is plotted in Figure 7. It can be seen that the solution of the system when applying the three relaxation methods yields the same result as the analytical solution, which was also seen in Figure 5 for the two springs system.

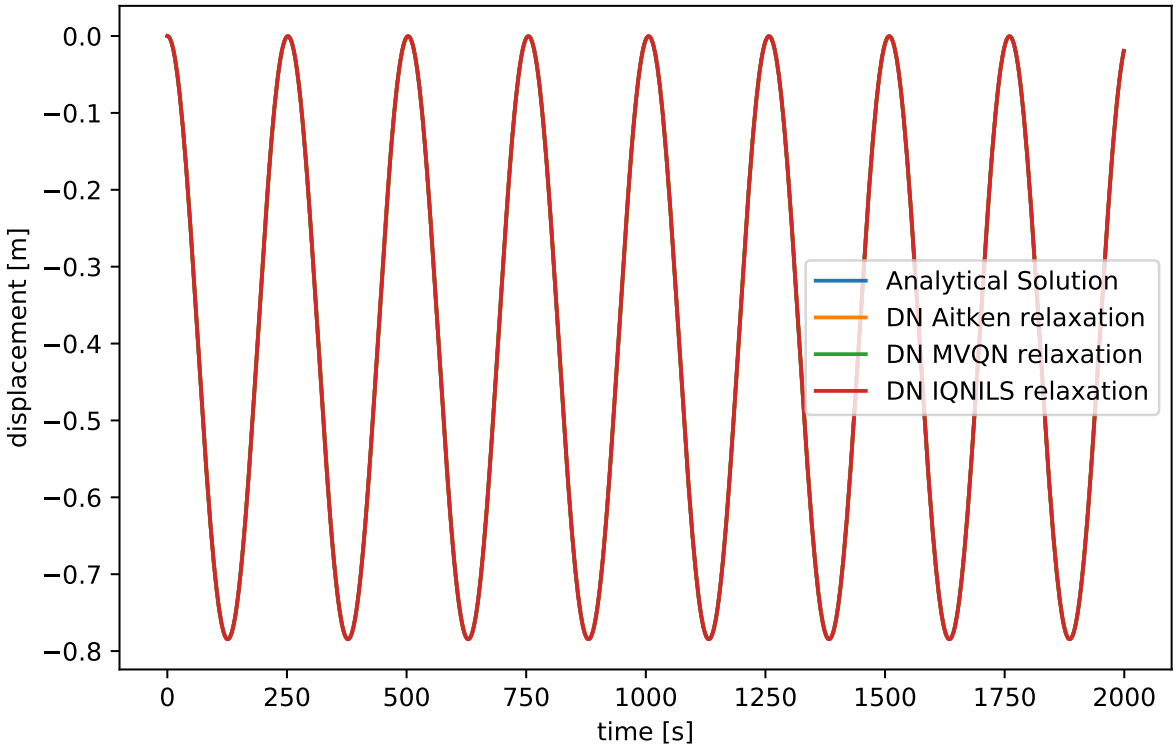


Figure 7: Resulting displacement for the three springs system using different relaxation methods

[1]This time only Aitken, IQNILS and MVQN were compared.

Table 2 shows the number of iterations each solver needs to converge to the solution. In order to guarantee an efficient solution procedure it becomes important to minimize the number of solver calls and hence the number of coupling iterations. It would further be ideal if the chosen coupling algorithm can be applied to a wide range of problems without the need for tuning a set of problem specific heuristics. It should be expected that MVQN has better performance when compared to Aitken or IQNILS, nevertheless this is only visible when the number of degrees of freedom differs from 1. This is due to the fact that MVQN makes use of a Singular Value Decomposition-like (SVD) approach, which cannot be exploited when having a single DOF [5].

Table 2: Comparison of the number of iterations for the three springs - one mass system

Relaxation methods	Maximum iterations per time step (inner loop / outer loop)	Total iterations (per 1000 time steps)
Aitken	3/3	8,951
IQNILS	6/9	32,874
MVQN	6/4	18,030

4 Three Coupled Beams System

Most of the times one is faced to problems where the interfaces to couple have more than one node (i.e. more than one DOF), thus the approach followed in sections 2 and 3 is no longer valid. In order to extend it to a multi-DOF solver it is necessary to vectorize the state variables involved (in the case of the springs system it is the displacement). This is done by simply changing the data type of the used variables so that the same algorithms proposed before can be used.

In order to test this and to serve as a guideline for the FSI problem in Section 5, a three bars static system is studied, where each coupling interface consists of 2 nodes per bar as shown in Figure 8. The procedure to solve this problem is the same as the one given by Algorithm 4: forces exerted by the upper and lower bars are used as Neumann boundary conditions in order to compute the nodal displacements on the middle bar. Once this is done, the nodal displacements belonging to the interface are given back to the other two bars to compute their forces and, subsequently, their displacements. This is done until the displacement of the interface nodes are the same for the three bars.

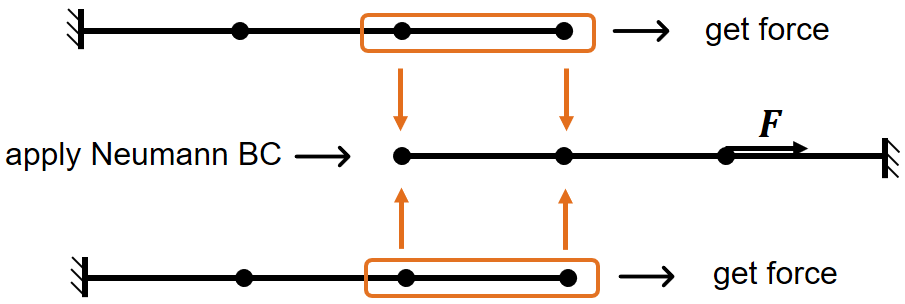


Figure 8: Three bars static system

Table 3 shows the number of iterations taken to solve the problem. It can be seen that the expected behaviour mentioned in the previous section is now visible since we have more than a single DOF at the coupling interface. MVQN should need less iterations than IQNILS and Aitken. Furthermore, the

iteration horizon of MVQN method is varied in order to determine the optimal value that yields the minimum number of iterations. In Table 4 is shown that the iteration horizon value which yields the minimum number of iterations for convergence is 12; this value is obtained by trial and error.

Table 3: Comparison of the number of iterations needed by each relaxation method

Relaxation methods	Number of iterations (only one time step)
Aitken	252
IQNILS	144
MVQN	127

Table 4: Influence of the iteration horizon on MVQN method

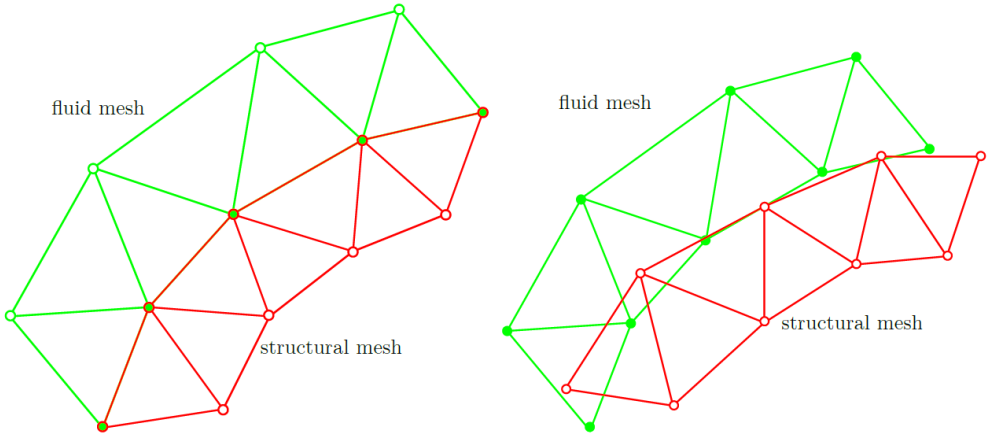
Horizon	Number of iterations (only one time step)
10	160
11	157
12	127
13	151
14	179

5 Fluid-Structure Interaction Simulation with Multiple Interfaces in Kratos Multiphysics

Kratos Multiphysics is an open-source finite element software developed at International Center for Numerical Methods in Engineering (CIMNE) of the Technical University of Catalonia (UPC). It provides both solvers for fluid dynamics applications and structural mechanics applications and enables the user to solve complex Fluid-Structure Interaction (FSI) problems. Furthermore, Kratos brings along a Meshing application and the EMPIRE application which is responsible for the mapping of the fluid mesh and the structural mesh.
This chapter will first present the basic structure of a simulation setup for Kratos and then the simulation results for different cases of FSI problems with multiple interfaces.

5.1 Data mapping

In general, the fluid mesh and the structural mesh are not matching to one another (Figure 9). Therefore, special mapping techniques have to be applied. The variable field can be either transferred in a *consistent* way or in a *conservative* way. By using the consistent approach the field is directly transferred from one mesh to the other while in the conservative approach the mapping is fulfilled in an integral sense. The coupling between the fluid and the structure is achieved by a Dirichlet-Neumann decomposition. The fluid exerts a force on the structure which leads to a displacement of the structure. At the same time, this displacement causes a movement of the fluid mesh. The interface variables are consequently the displacement, which will be mapped in a consistent way, and the force, which will be mapped in a conservative way.



(a) matching grid (b) non matching grid

Figure 9: Matching and non-matching grid [4]

5.2 Problem setup

After the implementation of simple structural problems with multiple interfaces in python (treated in Sections 3 and 4), a real Fluid-Structure Interaction problem with multiple interfaces is set up in Kratos. The problem setup is depicted in Figure 10. It consists of a 2D fluid traveling across a channel with two beam-like structures. The fluid enters the channel with a constant velocity and a no-slip boundary condition is applied at the outer walls and on the interfaces between the fluid and the structures. Using different coupling techniques it should be examined how the two structures influence each other.

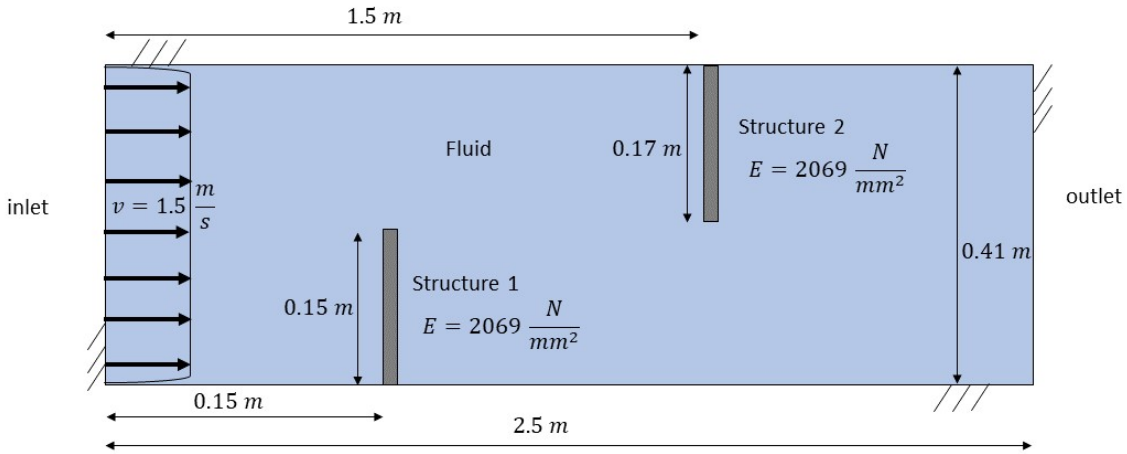


Figure 10: FSI problem setup

5.3 Structure of Kratos input files

A python interface provides the user with all the applications and functionalities of Kratos. The simulation settings, e.g. solver type and variable fields for the three parts, fluid, structure 1 and structure 2, are defined in different .json files (ProjectParameters.json). The model information, like boundary conditions and mesh are parsed from a .mdpa file. For both structures the material properties, e.g. Young's modulus

and density, are defined in a separate .json file. The .json and .mdpa files are automatically generated by the pre- and post-processor GiD. Each part consists of a main model part and different submodel parts which contain only the nodes of the interfaces between the fluid an the structure. These parts can then be used in the mapper to transfer the variable field from the fluid to the structure and vice versa.

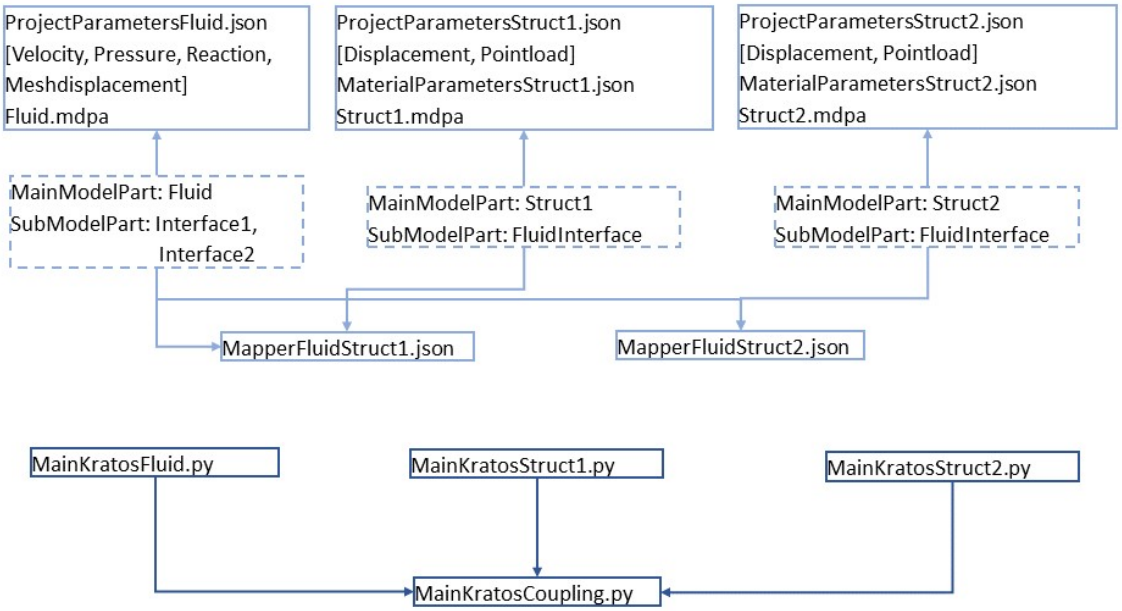


Figure 11: Overview of the necessary input files

For each part a python script was implemented, where the corresponding solvers provided by Kratos are initialized and called. Figure 11 shows how all the different files are brought together in the MainKratosCoupling.py file in which the two mappers, the different relaxation methods and the algorithm for coupling the two structures are defined. The coupling procedure will be explained in detail in the following susbsection. With the Kratos executable, the python script can be run and the result files will be generated.

5.4 Coupling methods

Two different ways of coupling the two structures in the fluid should implemented and their influence on the simulation results should be compared. Algorithm 5 explains the coupling of the structures through a nested loop, which leads to a strong coupling. In algorithm 6 the two interfaces (fluid-structure 1 and fluid-structure 2) are treated as one interface and combine the displacements of both interfaces in one big vector. By doing so, only this displacement vector will be relaxed and mapped to the fluid so that convergence is checked just once. The advantage of the second approach is that it is computationally more efficient but it has to be examined if this method also guarantees a strong coupling between the two structures.

for *time in range of starttime and endtime of simulation* **do**
  while $U_{interface1}$ *hasn't converged* **do**
    Get $Uold_{interface1}$
    while $U_{interface2}$ *hasn't converged* **do**
      Get $Uold_{interface2}$
      Interface 2: Map force from fluid to structure 2
      Get $Unew_{interface2}$
      Interface 2: Map relaxed $U_{interface2}$ from structure 2 to fluid
      Check residual
    end
    Interface 1: Map force from fluid to structure 1
    Get $Unew_{interface1}$
    Interface 1: Map relaxed $U_{interface1}$ from structure 2 to fluid
    Check residual
  end
end

Algorithm 5: Coupling with nested loop

for *time in range of starttime and endtime of simulation* **do**
  while $U_{interface1}$ *hasn't converged* **do**
    Get $Uold = Uold_{interface1} + Uold_{interface2}$
    Interface 1: Map force from fluid to structure 1
    Interface 2: Map force from fluid to structure 2
    Get $Unew = Unew_{interface1} + Unew_{interface2}$
    Map relaxed U to fluid
    Check residual
  end
end

Algorithm 6: Coupling with one vector

5.5 Simulation results

The FSI problem presented in chapter 5.2 is solved using the two different coupling techniques presented in the previous section and a decoupled approach. Additionally, two modified cases are simulated where on the one hand an oscillating inlet velocity is assumed and on the other hand the stiffness of the first structure is decreased. All other parameters are kept the same.

Case 1: Constant velocity ($1.5\ \frac{m}{s}$), same Young's modulus for both structure ($E = 2069\ \frac{N}{mm^2}$)

Case 2: Oscillating velocity ($0.5 * ((1 + cos(t * 5))1.5)\ \frac{m}{s}$), same Young's modulus for both structure ($E = 2069\ \frac{N}{mm^2}$)

Case 3: Constant velocity ($1.5\ \frac{m}{s}$), lower Young's modulus for first structure ($E_1 = 20.69\ \frac{N}{mm^2}$)

5.5.1 Case 1

Both structures are deformed in direction of the fluid motion whereas the second structure shows a higher deformation due to the increased fluid velocity resulting from the decreased width of the channel due to the first structure. At some time instances the second structure is even pushed back by vertices occurring behind the structure which leads to negative displacements of this structure.

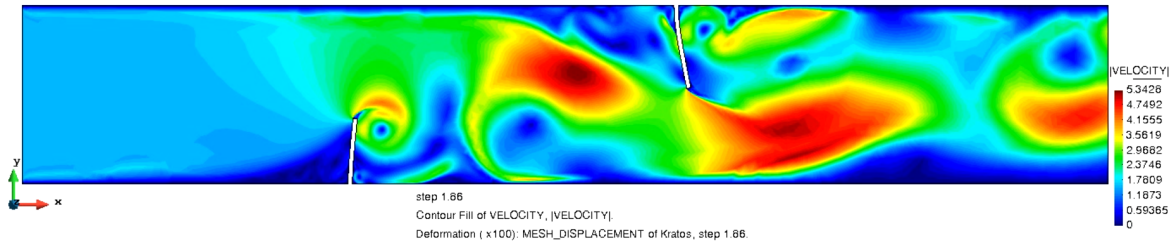


Figure 12: FSI simulation of case 1

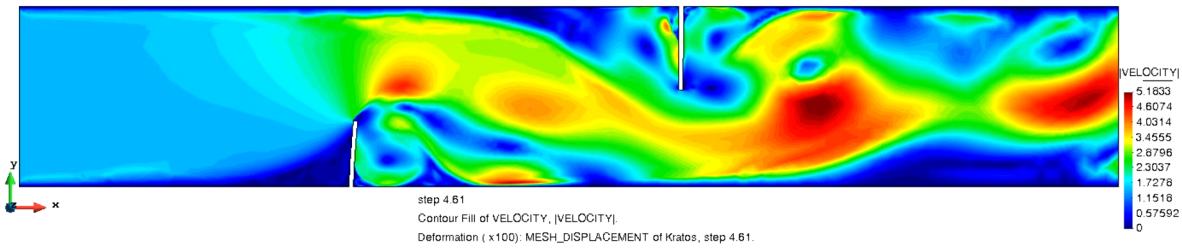


Figure 13: FSI simulation of case 1: structure 2 is pushed back by the vertices occurring behind the structure

For comparison of the three different coupling techniques -coupling with nested loop, coupling with one array, decoupled- the tip displacement for each structure is plotted. The legend shown in Figure 14 is valid for all following plots of the tip displacements. In the plots 15 and 16 one can see that all three methods show quite similar simulation results which leads to the conclusion that the coupling between the two structures is not very strong.

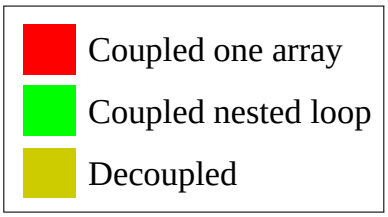


Figure 14: Legend

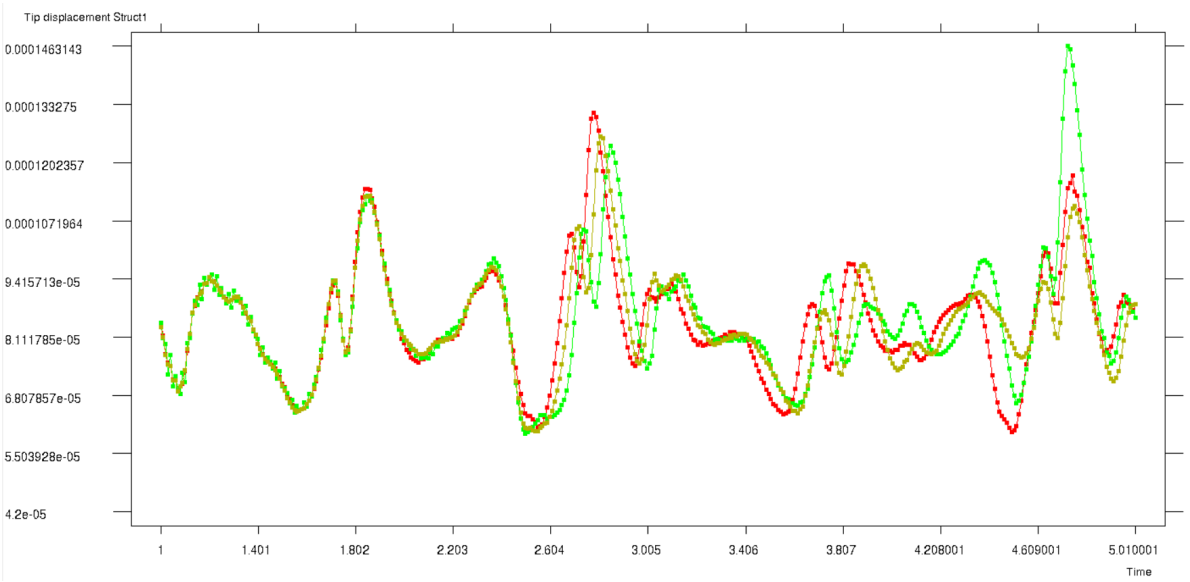


Figure 15: Comparison of the tip displacement of structure 1 for the different coupling methods

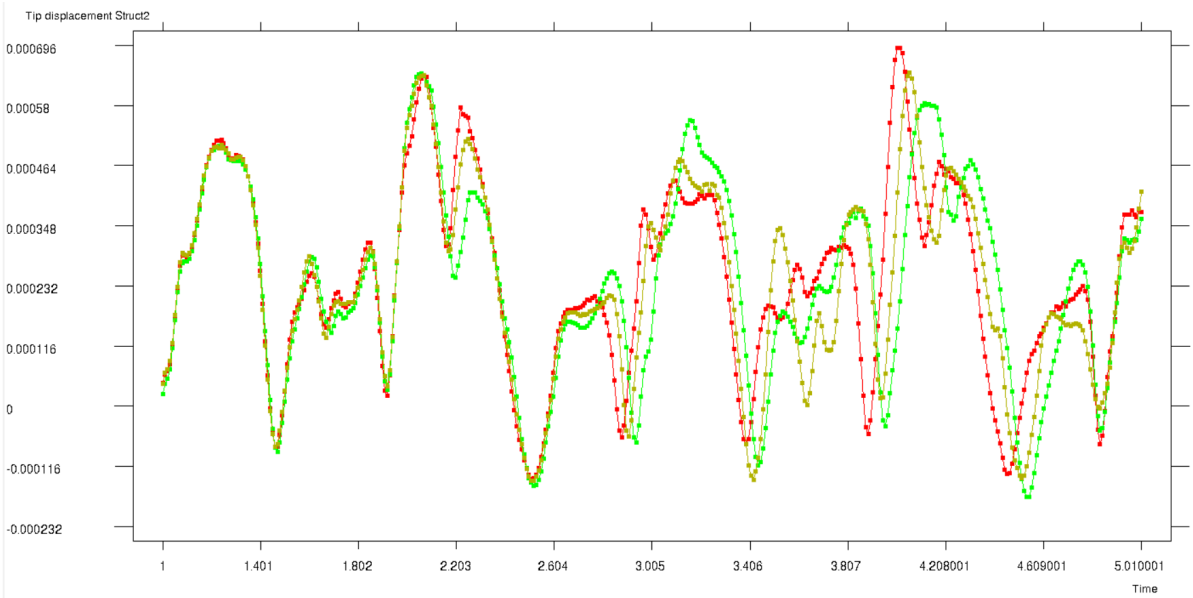


Figure 16: Comparison of the tip displacement of structure 2 for the different coupling methods

5.5.2 Case 2

In the simulation of the second case the velocity oscillates between 0 $\frac{m}{s}$ and 1.5 $\frac{m}{s}$. This leads to the emergence of toroidal-shaped vortexes when the velocity reaches 0 $\frac{m}{s}$ (see figure 18).

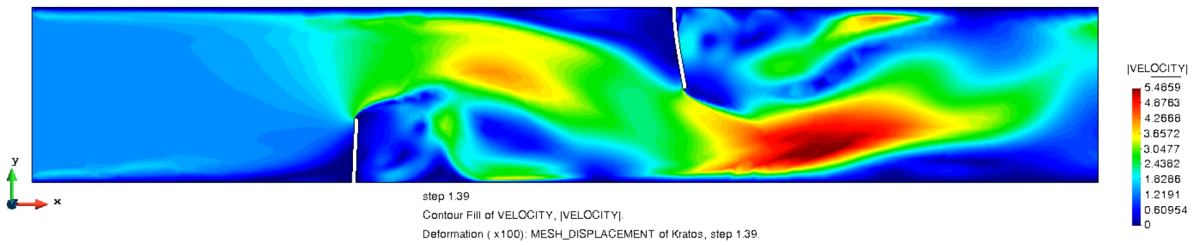


Figure 17: FSI simulation of case 2

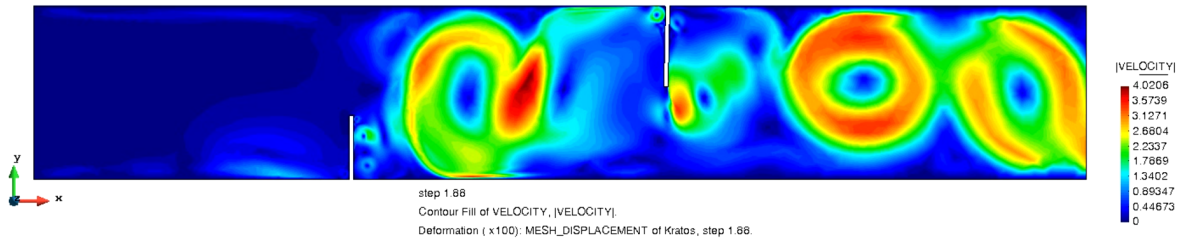


Figure 18: FSI simulation of case 2: emergence of toroidal-shaped structures behind structure 2

Also in the simulation of the second case all three coupling methods show quite similar behaviour. The tip displacements of the first structure oscillates with a cosine function which corresponds to the applied velocity.

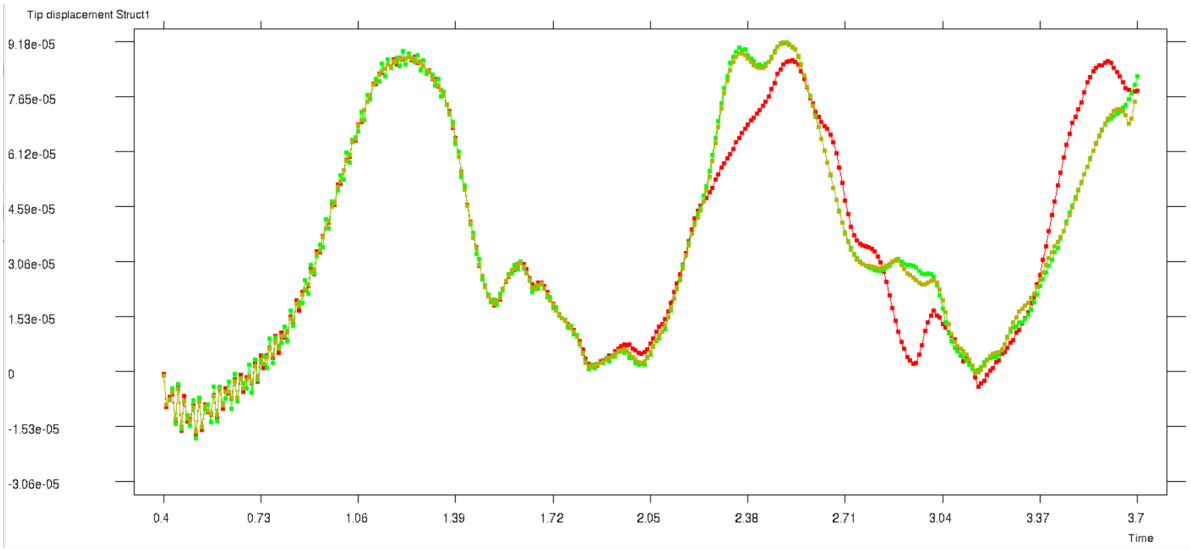


Figure 19: Comparison of the tip displacement of structure 1 for the different coupling methods

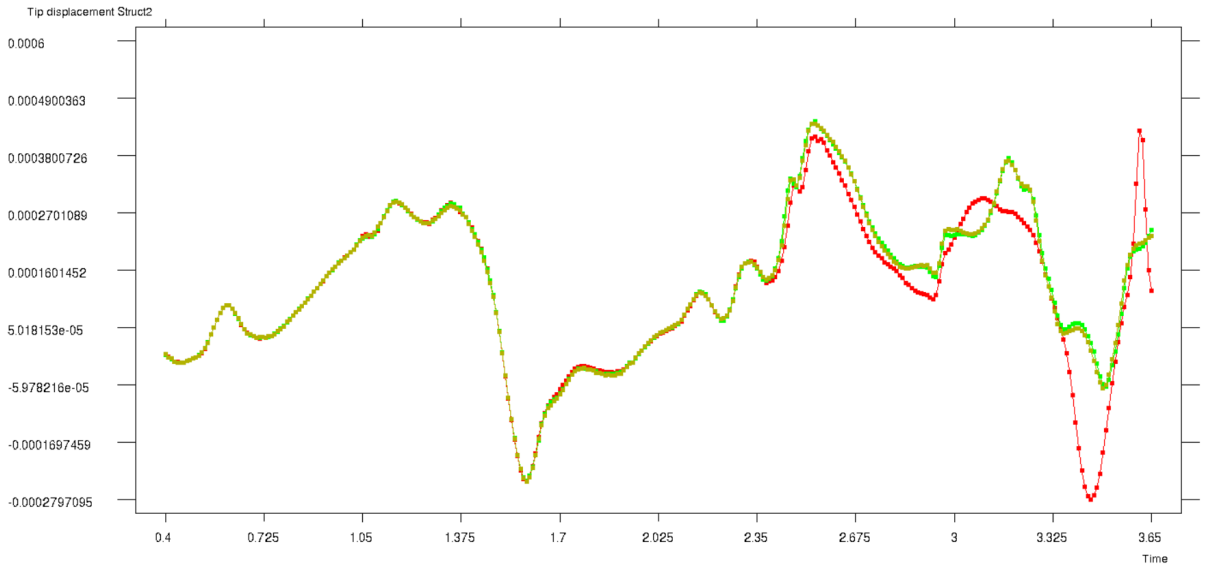


Figure 20: Comparison of the tip displacement of structure 2 for the different coupling methods

5.5.3 Case 3

Due to the decreased stiffness of the first structure (by factor 100) the first structure deforms much more than the second structure. Besides that, the simulation of the fluid flow in the channel resembles the simulation results of the first case.

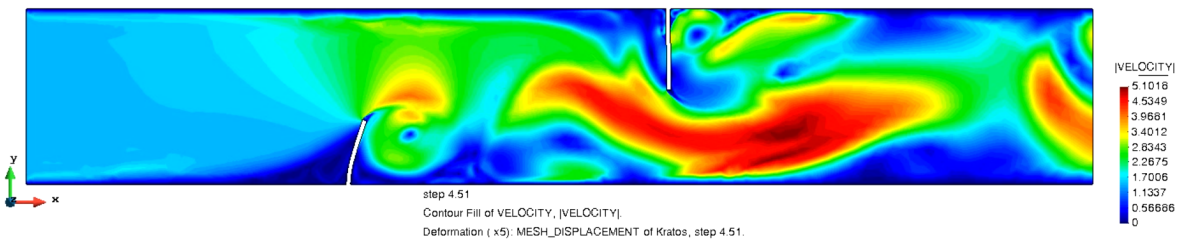


Figure 21: FSI simulation of case 3

Compared to the two previous cases, the results of the tip displacements for the three different coupling methods differ much more from one another. This draws the conclusion that a lower stiffness of the first structure leads to a stronger coupling of both structures. As already mentioned in subsection 5.4 one advantage of the coupling with one array is its computational efficiency. This is proven by comparing the average number of iterations per time step. The coupling with nested loop needs in average 5 iterations per time step whereas the coupling in one array only needs 2 iterations. This will lead to a reduction of the simulation time by factor 2.5.

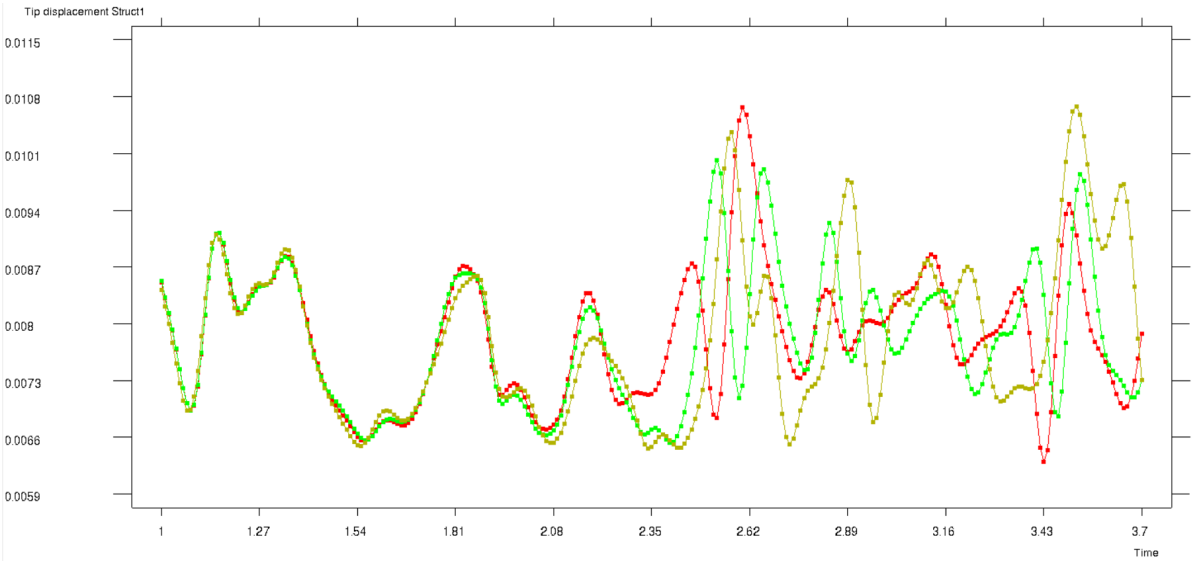


Figure 22: Comparison of the tip displacement of structure 1 for the different coupling methods

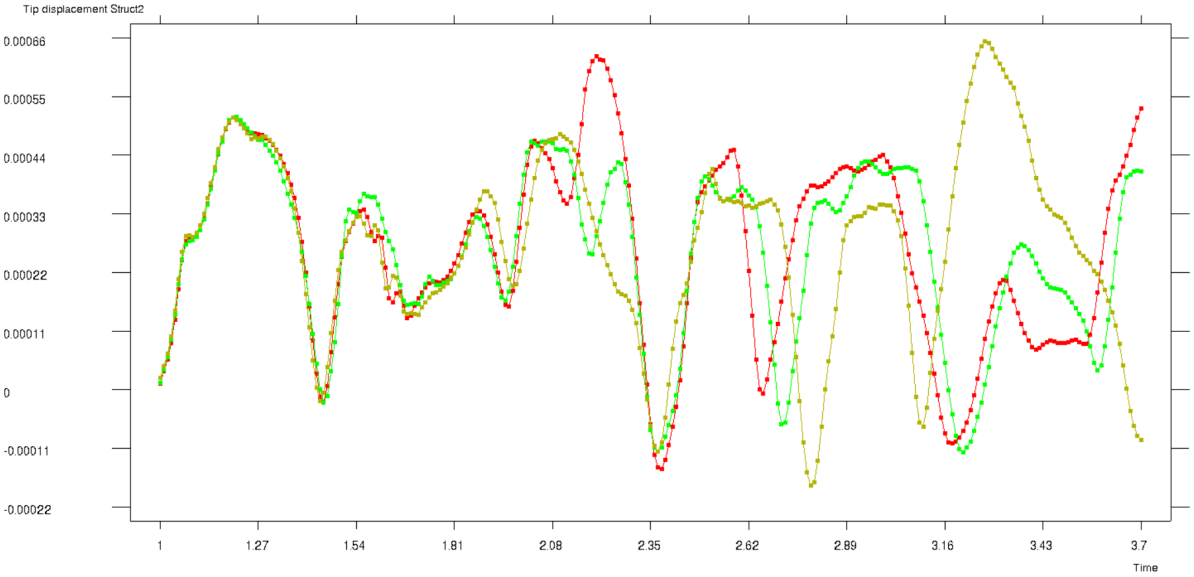


Figure 23: Comparison of the tip displacement of structure 2 for the different coupling methods

Conclusion

In this paper different relaxation methods for the implicit coupling of partitioned solvers, using these across a small selection of structural benchmark problems and extrapolating them to a strongly coupled FSI problem, were analyzed. It was shown that the MVQN method outperformed the other relaxation methods on interfaces where the number of grid points was greater than one, such as in the three bars static system and the FSI problem. It was also seen that the one-array coupling outperformed the well-known nested coupling as long as there is no presence of a strong coupling between interfaces. Furthermore, when working with MVQN one has to be aware of the influence the iteration horizon has on the performance of the method. In this work it was obtained by trial and error, although an optimal way for determining it has to be further investigated in order to avoid unnecessary computational time.

References

[1] Bernhard Gatzhammer.
"A Partitioned Approach for Fluid-Structure Interaction on Cartesian Grids".
MA thesis. Technische Universität Münchenen, 2008.

[2] Emmanuel Lefrançois and Jean-Paul Boufflet.
"An Introduction to Fluid-Structure Interaction: Application to the Piston Problem".
In: *SIAM Review* 52.4 (2010), pp. 747–767.
DOI: 10.1137/090758313.
eprint: https://doi.org/10.1137/090758313.
URL: https://doi.org/10.1137/090758313.

[3] Joris Degroote et al.
"An interface quasi-Newton algorithm for partitioned simulation of fluid-structure interaction".
In: *Proc. of the International Workshop on Fluid-Structure Interaction: Theory, Numerics and Applications* (Sept. 2008), pp. 55–64.

[4] Wei He.
"Partitioned Fluid-Structure Interaction in Kratos Multiphysics".
MA thesis. Technische Universität München, 2017.

[5] Alfred EJ Bogaers et al.
"Quasi-Newton methods for implicit black-box FSI coupling".
In: *Computer Methods in Applied Mechanics and Engineering* 279 (2014), pp. 113–132.