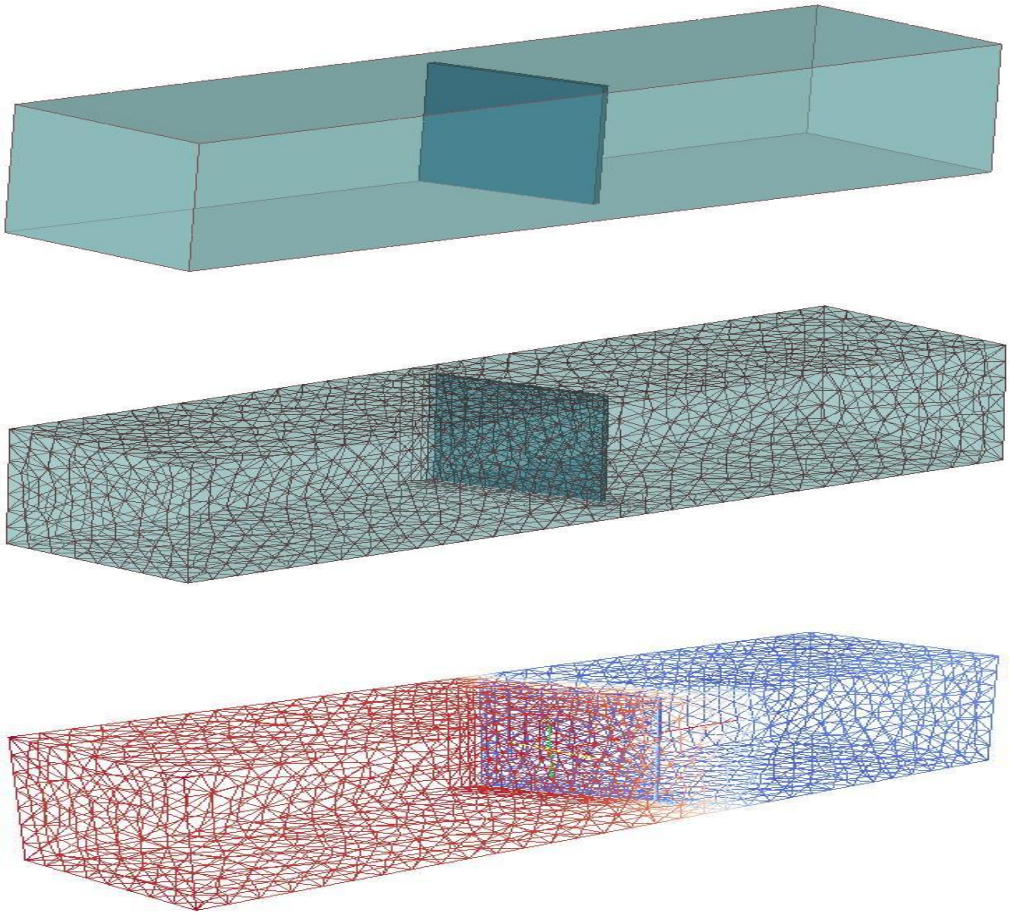
Technical report, IDE1135, June 2011

Fluid Structure Interaction

Evaluation of two coupling techniques

Master's Thesis in Computational Science and Engineering

Christoffer Andersson, Daniel Ahl



School of Information Science, Computer and Electrical Engineering
Halmstad University

Fluid Structure Interaction

Evaluation of two coupling techniques

Master's Thesis in Computational Science and Engineering

School of Information Science, Computer and Electrical Engineering
Halmstad University
Box 823, S-301 18 Halmstad, Sweden

June 2011

Acknowledgements

We would like to thank the following people:

- Our supervisors at FS Dynamics, Dr. Per Heintz and Dr. Christoffer Cromvik, for all knowledge and support given during the thesis.
- Our examinator, Professor Peter Hansbo, for supporting us throughout this master year and finally being a great mentor during the thesis.
- Dr.Bertil Nilsson for being a good inspiration and for making this master education possible.
- Marlow Springer from AcuSim, for responding quickly, answering all our questions about the softwares and theory in an informative way.
- Hans Corin for supporting us with accommodation during the spring of 2011.

Abstract

This thesis concerns one of the upcoming and well discussed subjects within calculations these days, namely how to perform an analysis of the interaction between fluid and structure, called FSI (Fluid Structure Interaction). In the report, evaluations of two different methods of simulating FSI are done. These are known as Practical FSI (P-FSI) and Direct Coupled FSI (DC-FSI). The methods are developed by Acusim in cooperation with Simulia and the softwares used are Abaqus and AcuSolve.

The first part of the thesis is dedicated to explain the general theory and the governing equations for FSI. After the general explanation a more delimitated explanation regarding P-FSI and DC-FSI are given. After this we show how to setup and perform the couplings regarding which parameters that need to be defined and how to perform the analyses using Abaqus and AcuSolve.

The last section of the thesis covers the evaluation process. We started with evaluating the methods against a benchmark problem where we compared the calculation time and accuracy regarding displacements and frequencies. The next thing we evaluated was how different numbers of modes used in the P-FSI coupling affects the result. The last thing we evaluated was the robustness of the methods using different mass densities of the structure and different time-step sizes.

The result of the evaluation regarding the criteria: accuracy, calculation time and robustness showed that the P-FSI method is the most efficient method compared to DC-FSI regarding FSI problems when the structural response is linear.

Table of contents

1 Thesis work in cooperation with FS Dynamics Sweden AB

1.1 Group members

Christoffer Andersson
0707-773642
christoffer.a@hotmail.se

Daniel Ahl
0736-716507
daniel.ahl@gmail.com

1.2 Supervisor Halmstad University

Peter Hansbo, Professor Scientific Computations

1.3 Examinator Halmstad University

Peter Hansbo, Professor Scientific Computations

1.4 Taskmaster

FS Dynamics Sweden AB
Mölndalsvägen 24
412 63 Göteborg

1.5 Supervisors at company

Per Heintz, Ph.D, Manager-Industrial Multiphysics
Christoffer Cromvik Ph.D, Senior Engineer-Industrial Multiphysics

2 Introduction

The following information about FS Dynamics is copied from their homepage [2].

"FS Dynamics (Fluid Structural Dynamics) is a consultancy for technical calculations. Since its start, in 2004, the company has continued to grow and, today, it is one of the largest providers of calculation services within CFD (fluid dynamics) and FEM (structural dynamics). More than half of our revenues come from calculation assignments and projects undertaken from one of our four offices located in Sweden, Denmark and Finland, but we also have a significant number of qualified consultants providing onsite client services. We work within a very broad spectrum of industries, including nuclear power, wind power, solar power, food, automotive, marine and aerospace. We have access to the vast majority of all the commonly used, commercial FEM and CFD software at our offices."

2.1 Problem statement

FS Dynamics has developed a unique competence within the area of coupled fluid- and structure dynamic simulations a.k.a FSI (Fluid Structure Interaction). The questions dealt with within FSI is partly general for each discipline structure- and fluid dynamics but also concerns how these two disciplines are linked together in the best way is of greatest interest.

The amount of data that needs to be communicated between the FEM and CFD codes is also of practical importance. Some codes use an API (Application Programming Interface), which handles the data management across the interface automatically. For many applications in FSI this is an excellent approach. However, when simulating physical phenomena that require short time steps and long analysis times, the computation time often becomes excessively long because of all the data transfer needed. Examples of such phenomena are Aero Acoustic (AA) and Fluid Induced Vibrations (FIV). High frequencies are often involved in these problems and in order to not lose any information and to find a stationary solution, these analyses require short time steps combined with long analysis time.

2.2 Problem description

FS Dynamics has identified an FSI methodology based on the assumption that the structure's response can be described as a combination of a subset of its eigenvectors. This results in a system of uncoupled ODE:s which are used in the CFD calculation. The fluid forces are projected onto the eigenvectors which gives a deformation of the structure and changes in the CFD mesh. The expected benefits are more efficient and stable FSI coupling when the codes are running without coupling and the problem is truncated at a selected frequency level.

The method is known as Practical-FSI (P-FSI) which has been developed in cooperation between the corporations Simulia and Acusim. Evaluation of this method is the main focus of this thesis.

2.3 Objectives

The mission aims to determine with certainty whether P-FSI is an appropriate methodology for calculating the AA and FIV problems.

2.4 Scope

Evaluate the P-FSI method and find out its abilities and limitations. The following criteria will be evaluated:

- Accuracy
- How different number of modes affect the result
- CPU time
- Robustness

Also a short tutorial of how to set up FSI analyses for each method will be worked out.

3 Theory

This section intends to give the reader some general information about fluid-structure interaction regarding:

- Different coupling approaches
- Stability issues
- The governing equations for each domain, fluid and structure
- The coupling between the domains
- The spatial discretization of each domain
- The temporal discretization of the coupled system

This is done in order to give the reader a basic understanding of the computational implementation of fluid- structure interaction.

3.1 Coupled systems

When two or more physical systems interact with each other and the solution of either one of the systems is dependent of the solution of the other one, they are said to be coupled.

One example of a coupled system is the dynamic interaction between a fluid and a structure (FSI). In this case neither the fluid nor the structural system can be solved independently due to the unknown forces in the interface region.

A more detailed definition of a coupled system is given by [3] as follows:

Coupled systems and formulations are those applicable to multiple domains and dependent variables which usually (but not always) describe different physical phenomena and in which

(a) *neither domain can be solved while separated from the other;*
(b) *neither set of dependent variables can be explicitly eliminated at the differential equation level.*

3.2 FSI

Fluid structure interaction is defined by [4] as the interaction between a fluid flow and a deformable body. The body can be exposed to either internal or external flow.

In many engineering applications the forces from a moving fluid applied onto the boundaries of a structure will only lead to very small deformations of the

structure. In these cases the structural response can be established independently after the characteristics of the fluid motion has been determined since the structural deformation does not affect the fluid flow. In those cases when the fluid forces lead to substantial deformations of the structure and the fluid flow pattern changes, the dynamics of the systems need to be coupled.

The FSI-problems are in general to complex to solve analytically so they have to be analyzed by numerical simulations. Two main approaches exist for the simulation of FSI-problems: monolithic and partitioned approach.

3.2.1 Monolithic approach

A monolithic or direct approach is when the equations governing the flow and the displacements of the structure are solved simultaneously. The discretization of the problem leads to a large matrix of equations which is solved with a single solver.

$$\begin{pmatrix} \boldsymbol{K}_S & \boldsymbol{K}_{\mathrm{FS}} \\ \boldsymbol{K}_{\mathrm{SF}} & \boldsymbol{K}_F \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_S \\ \boldsymbol{u}_F \end{pmatrix} = \begin{pmatrix} \boldsymbol{F}_S \\ \boldsymbol{F}_F \end{pmatrix}$$

Fig. 1. Example of a monolithic system containing both structural and fluid information

The monolithic approach has the advantages of stability since the mutual influence of the fluid and the structure can be taken into account directly. An example of commercial software that implements the monolithic approach is ADINA. ADINA is using finite elements to discretize both the structure and the fluid, whereas in computational fluid dynamics (CFD) the traditional discretization approach is to use finite volumes.

3.2.2 Partitioned approach

A partitioned approach is when the equations governing the flow and the displacement of the structure are solved separately with two distinct solvers. This will allow for codes specialized for flow equations and structural equations and hence possibly more efficient solution techniques for each of them. On the other hand; the partitioned approach requires a coupling algorithm to allow for the interaction and to determine the solution of the coupled problem.

Partitioned algorithms use subsequent solutions of the fluid and structure subproblems and the interaction is either loosely or strongly coupled. A loosely coupled algorithm is explicit and the codes will have only one bidirectional exchange of solved variables per time step, in a sequentially staggered manner. A strongly coupled algorithm (implicit) uses an iterative staggered scheme which performs subiterations over the single fields to converge at the end of the time-step.

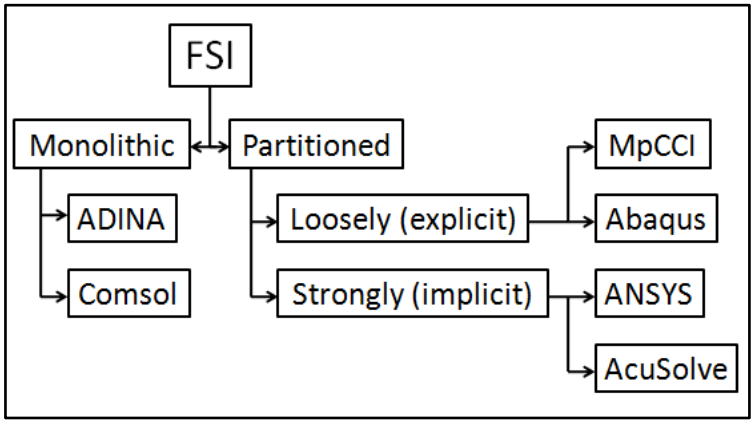


Fig. 2. Approaches and examples of softwares

3.2.3 Stability issues for partitioned algorithms

A wide variety of stability issues regarding FSI-problems and partitioned algorithms have been reported. Most of the problems are in regard to the loosely coupled algorithm.

When using a loosely coupled algorithm to solve FSI-problems with incompressible flow and slender structures, it has been observed that instabilities in the computations will occur. The instabilities depend on the mass density of the fluid versus the mass density of the structure, but also the geometry of the domain has importance [5]. It has further been observed that decreasing the time step size will result in even more instability, or that the instability occurs earlier. The instability is inherent in the scheme itself and has been named 'artificial added mass effect'. The name comes from that the fluid closest to the coupling interface will act as extra mass on the structure, increasing its inertia.

In sequentially staggered schemes the fluid forces depend on predictions of the interface displacement instead of the actual ones. Errors in the predictions together with the added mass effect will lead to incorrect coupling forces which yields the instability.

Since the density of the structure versus the density of the fluid affects the stability of the computations, a so called mass density ratio has been defined:

$$\mu = \frac{\rho_{solid}}{\rho_{fluid}}$$

It has been observed that loosely coupled algorithms are unstable for low density ratios and the instabilities increase as the ratio gets lower. Also, strong coupled algorithms are affected by the density ratio in such a way that an increased number of iterations are needed within each time-step in order to achieve convergence of forces along the interface at the end of the time-step.

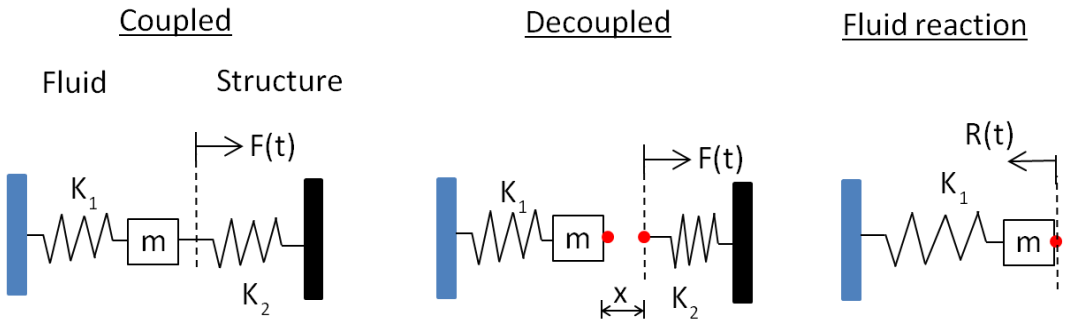


Fig. 3. The figure is showing a simplification of the upcoming instabilities due to the added mass effect in a loosely coupled algorithm. In the first time step in the algorithm, represented by the decoupled system, the displacement of the structure due to fluid forces is x. The resistance of the displacement consists only of the elasticity, K_2, of the structure.. In the next step in the algorithm the same displacement x is subjected to the fluid part where the resistance of displacement consist of K_1 and mass m. The same displacement on the fluid side will lead to a greater reaction force R(t) than the initial force F(t) if: $R(t) = (m\ddot{x} + K_1 x) > K_2 x = F(t)$, This will lead to lack of equilibrium at the end of the first time step.

It is well known that the strongly coupled algorithms are more stable than the loosely coupled ones, because the iterative scheme ensures balance of the forces at the end of each time-step.

Regarding choice of time-step length for loosely coupled algorithms, in [5] there has been defined an interval where the computations are conditionally stable. This interval has an upper boundary resulting from the Courant-Friedrichs-Lewy condition (CFL) and a lower boundary resulting from the highest eigenvalue of the added mass matrix.

$$A < \Delta t < CFL < 1$$

Where:

- A=Min size of time step size limited by the highest eigenvalue of the added mass matrix, see [5].
- CFL=The maximum size of the time step according to the general CFL condition, see [6].

The following list summarizes the instability issues prompted above:

- For loosely coupled algorithms:
  - Stability is dependent on mass density ratio and geometry of the domains.
  - Instability increases with decreasing mass density ratio.
  - For unstable computations, decreasing of time step size will lead to earlier occurrence of instabilities or increased instabilities.
  - For conditionally stable computations the time step size has an upper limit depending on the CFL number and a lower limit depending on the highest eigenmode of the added mass matrix.

- For strongly coupled algorithms:
  - More stable for low mass density ratio than the loosely coupled algorithm.
  - Decreasing of the ratio leads to more subiterations which in turn leads to increased computing time.

3.3 ALE (Arbitrary Lagrangian Eulerian)

In this section the ALE method is described briefly. We refer to [7] for the ALE-interested reader but also [8] and [9] treat the subject.

For applications with multiphysics problems, FSI in this thesis, one needs to choose an appropriate kinematic description of the continuum, managing the dynamics of both the fluid and the structure. An algorithm named ALE (Arbitrary Lagrangian Eulerian) has been developed. This is a combination of the classical two descriptions of motion, the *Lagrangian* motion and the *Eulerian* motion, with the purpose to summon their respective advantages and minimize their disadvantages. For a better understanding of the two respective motions a short description follows below:

Lagrangian motion:

Each individual node in the computational mesh is permanently connected to the associated material particle during motion. This can be described as if you are sitting in a boat following a current and you are having the same speed as the water. You are studying the same particle at different times at different places. This method is mainly used in structural mechanics.

The negative side of this approach is that when you get large deformations of the continuum, which leads to distortions of the mesh (for example vortices in water), it can contribute to loss in the accuracy of the solution and sometimes the FSI-problem cannot be solved.

A positive side is that it is possible to track back in history, by the inverse transformation, which can be helpful when working with materials with history dependent behaviour. Another positive side is that it gets a precise result in form of material displacement.
It also enables easy tracking of the free surfaces and interfaces between different materials.

Eulerian motion:

The computational mesh is fixed and the continuum moves and deforms with respect to the computational mesh. This can be described as if you are standing in the water and the current is passing you by. You are studying velocities and pressures at different times at the same place. This method works well for large deformations and is often used when working with fluid computations (CFD).

A negative aspect is that it is hard to get precise solution with respect to interfaces and of flow details.

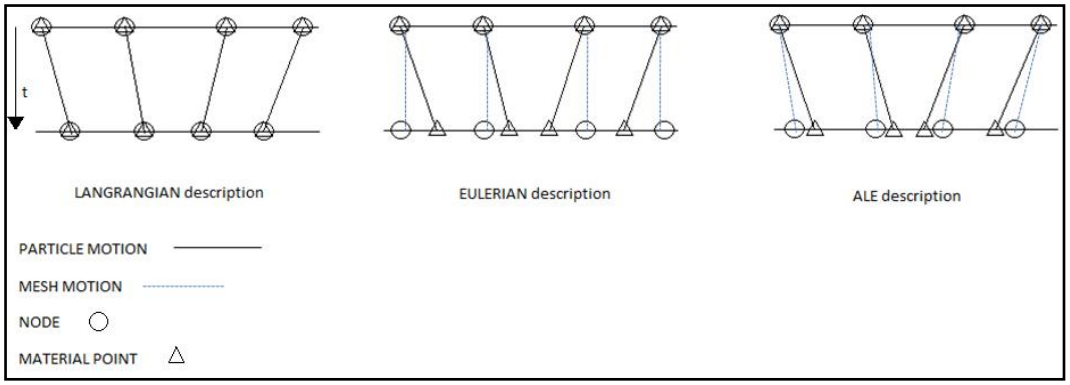


Fig. 4: Descripion of different motions

When talking about the Langrangian and Eulerian motions one refers to the material and the spatial domain. When using ALE one needs to introduce a third domain, called the referential domain (also often called the ALE domain). By mapping velocities and displacements between the domains one finally arrives at the ALE formulations of the conservation laws.

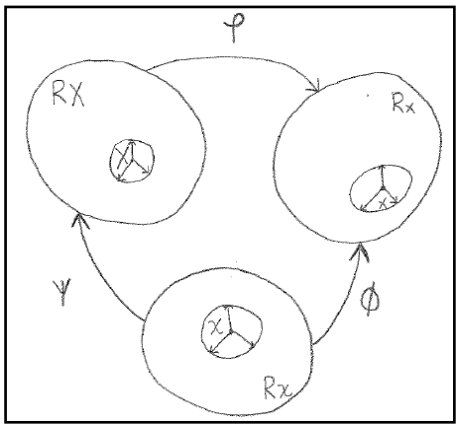


Fig. 5. Mapping between domains.

RX is the material domain, Rx is spatial domain and $R\chi$ is the referential domain and Fig. 5 shows the mapping between the domains. φ is the mapping between RX and Rx, and it describes the particle motion from RX to Rx. ϕ is the mapping between $R\chi$ and Rx, and it describes the motion of the mesh in the spatial domain. ψ is the mapping between $R\chi$ and RX, and it describes the particle motion in the referential domain. The $R\chi$ domain is mapped into the RX and Rx domains by ϕ and ψ respectively and the particle motion φ can be expressed as:
$\varphi = \phi \; o \; \psi^{-1}$ (o means function composition).

Regarding the velocity of the mesh and particles:

v = material velocity

$$v(X,\, t) = \frac{\partial x}{\partial t}\Big|_X \qquad (|_n \text{ means that the n-domain is fixed})$$

$\hat{v}$ = mesh velocity

$$\hat{v}(\chi, t) = \frac{\partial x}{\partial t}\Big|_{\chi}$$

Notice that both v and $\hat{v}$ are variations of the coordinate x.

w = particle velocity seen from the $R\chi$ domain $= \frac{\partial \chi}{\partial t}\Big|_{X}$

The relation between v, $\hat{v}$ and w can be obtained by differentiation of $\varphi = \phi\, o\, \psi^{-1}$ where one finally arrive at the convective velocity c, i.e., the relative velocity between the particle and the mesh seen from the Rx domain.

$$c := v - \hat{v} = \frac{\partial x}{\partial \chi} \cdot w$$

Notice that if v=$\hat{v}$ there is zero convection and it is seen from the Lagrangian point of view. And if $\hat{v}$=0 the mesh is fixed and it is seen from the Eulerian point of view.

To obtain the differential ALE formulations of the conservation laws, use the Eulerian forms and replace in the various convective terms the material velocity v with the convective velocity c:

Mass: $$\frac{\partial \rho}{\partial t}\Big|_{\chi} + c \cdot \nabla \rho = -\rho \nabla \cdot v$$

Momentum: $$\rho\left(\frac{\partial v}{\partial t}\Big|_{\chi} + (c \cdot \nabla)\, v\right) = \nabla \cdot \sigma + \rho b$$

Total energy: $$\rho\left(\frac{\partial \mathrm{E}}{\partial t}\Big|_{\chi} + c \cdot \nabla \mathrm{E}\right) = \nabla \cdot (\sigma \cdot v) + v \cdot \rho b$$

The boundary conditions concerning ALE are the same as for Langragian and Eulerian motion, regularly named as Neumann and Dirichlet BC:s.

Regarding the boundaries in FSI, along solid-wall boundaries, the particle velocity is coupled to the rigid or flexible structure. The kinematic requirement is that no particle can cross the interface. One also needs to define, due to the coupling between fluid and structure, some coupling conditions so the fluid and the structure do not overlap or detach during the motion. These conditions depend on the fluid and differ depending upon whether the fluid is viscous or inviscid.

3.4 Equations

The following sections will state and briefly explain the governing equations of the fluid flow, the structure and the coupling between them. The problems under consideration consist of a fluid that is occupying a given domain Ω^F and a structure that occupies another domain Ω^s which interact at the common boundary Γ. The information in this section has been gathered from [5].

3.4.1 Structural equation

The possibly large displacement of the structure is governed by:

$$\rho^s \frac{D^2 \boldsymbol{u}}{\mathrm{Dt}^2} - \nabla \cdot (\boldsymbol{F} \cdot \boldsymbol{S}(\boldsymbol{u})) = \rho^s \boldsymbol{b}^s \qquad \text{in } \Omega^s \times (0,\, T) \tag{1}$$

Where $\boldsymbol{u}$ represents the displacements of the structure, $\boldsymbol{b}^s$ represents the body forces applied on the structure. $\boldsymbol{S}$ represents the second Piola-Kirchhoff stress tensor, ρ^s represents the density of the structure and $\boldsymbol{F}$ represents the deformation gradient tensor.

3.4.2 Fluid flow equations

The fluid equations to be solved are the incompressible Navier-Stokes equations expressed in ALE formulation. The Navier-Stokes equations can be derived from the conservation laws for mass and linear momentum and takes in consideration that the fluid is viscous. For details of the derivation of the equations, see [10] (p. 43-45), together with [7]. The equations in ALE formulation take the form:

$$\rho^F \left.\frac{d\boldsymbol{v}}{\mathrm{dt}}\right|_{\chi} + \rho^F \cdot \boldsymbol{c} \cdot \nabla \boldsymbol{v} - 2\,\mu\,\nabla \cdot \boldsymbol{\varepsilon}(\boldsymbol{v}) + \nabla \overline{p} = \rho^F \boldsymbol{b}^F \qquad \text{in } \Omega^F \times (0,\, T) \tag{2}$$

$$\nabla \cdot \boldsymbol{v} = 0 \qquad \text{in } \Omega^F \times (0,\, T) \tag{3}$$

Here $\boldsymbol{v}$ denotes the fluid velocity and $\bar{p}$ denotes the physical pressure. The fluid density and viscosity is given by ρ^F respectively μ. The fluid body forces are represented by $\boldsymbol{b}^F$ and $\boldsymbol{\varepsilon}(\boldsymbol{v})$ represents the strain rate tensor. As can be seen in (2) the ALE formulation comes in to the equation in the fluid acceleration term and the convective term.

3.4.3 Coupling equations

At the interface Γ , kinematic and dynamic continuity is required. The governing kinematic coupling equations are:

$$\boldsymbol{u}_\Gamma(t) = \boldsymbol{d}_\Gamma^F(t), \quad \dot{\boldsymbol{u}}_\Gamma(t) = \boldsymbol{v}_\Gamma(t), \quad \ddot{\boldsymbol{u}}_\Gamma(t) = \dot{\boldsymbol{v}}_\Gamma(t) \tag{4}$$

Here $\boldsymbol{d}_\Gamma^F(t)$ represents the displacement of the fluid mesh nodes at the interface. The dynamic coupling equation takes the form:

$$\boldsymbol{h}^s(t) + \boldsymbol{h}^F(t) = \boldsymbol{0} \tag{5}$$

where $\boldsymbol{h} = \boldsymbol{\sigma} \cdot \boldsymbol{n}$ signifies the traction vector.

3.5 Time stepping process

It is important to point out that even though the solid and fluid equations for example are solved implicitly, the coupling information (4) and (5) is exchanged only once per time step in a sequential staggered manner. Since the sequential staggered algorithm is explicit this will add explicit features to the computations and thus the time step size needs to be considered.

3.5.1 Sequential staggered algorithm

An example of a sequential staggered algorithm is given by [5].

After both the structural domain and the fluid domain individually have been discretized in time with the same time step size Δt, the time step from time level n to $n+1$, when using a sequential staggered algorithm, will proceed in the following way:

1. Calculate an explicit predictor $\mathbf{u}_{\Gamma,\mathrm{P}}^{n+1}$ of the structural interface displacement at the new time level.
2. Compute the fluid velocity $\mathbf{v}_{\Gamma}^{n+1}(\mathbf{u}_{\Gamma,\mathrm{P}}^{n+1})$ at Γ to serve as Dirichlet boundary condition.
3. Update the mesh displacement $\mathbf{d}^{\mathrm{n+1}}\left(\mathbf{u}_{\Gamma,\mathrm{P}}^{n+1}\right)$.
4. Solve the fluid equations on the deforming domain to get $\mathbf{v}^{\mathrm{n+1}}$ and $\bar{\mathrm{p}}^{\mathrm{n+1}}$.
5. Obtain the fluid boundary traction $\mathbf{h}_{\Gamma}^{\mathrm{F,n+1}}$ along the coupling interface.
6. Solve the structural field to get $\mathbf{u}^{\mathrm{n+1}}$ under consideration of the fluid boundary traction $\mathbf{h}_{\Gamma}^{\mathrm{F,n+1}}$.
7. Proceed to the next time step.

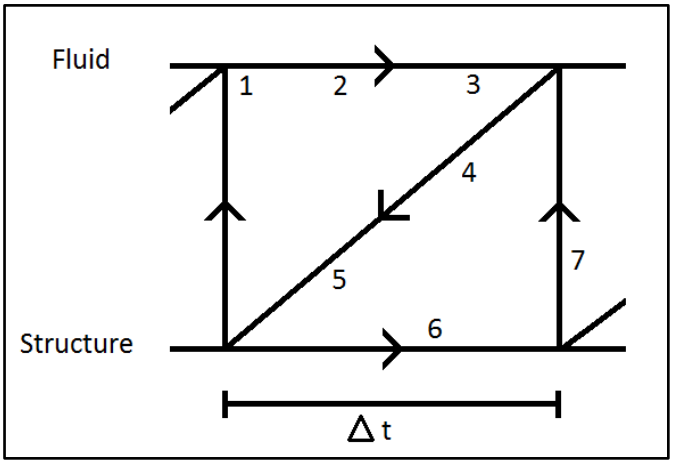


Fig. 6. Steps in the sequential staggered scheme.

In step 2 and 3 in the above algorithm the fluid velocity and the mesh displacement should be seen as functions of the predictor P.

Temporal discretization schemes for respective domain will not be given in this thesis, but [11] suggests a Leap-Frog scheme for the structure and an implicit Euler scheme for the fluid

3.6 Spatial discretization

The fluid equations (2) and (3) are discretized in space using finite elements, based on the Galerkin form, where the time-dependent Dirichlet boundary conditions of the problem are satisfied. For information on how to arrive at the

spatially discretized equations in matrix form, which are presented below, we refer to [5].

Matrix notation of the discretized equations for the fluid:

$$\mathbf{M}^F \dot{\boldsymbol{v}} + \mathbf{N}^F(\boldsymbol{v}) + \mathbf{G}\mathbf{p} = \mathbf{f}_\Gamma \tag{6}$$

$$\mathbf{G}^T \boldsymbol{v} = \mathbf{0} \tag{7}$$

Here $\mathbf{M}^F$ and $\mathbf{N}^F(\boldsymbol{v})$ denotes the mass matrix and the internal force vector respectively. $\mathbf{G}$ represents the discrete gradient operator on the pressure field. The vectors $\boldsymbol{v}$ and $\mathbf{p}$ contain the nodal velocity and pressure values and these are to be solved. The right hand side in (6) only consists of forces present at the coupling interface.

Noticeable here is that no streamline diffusion stabilization method for the discretization, as in [10] (p. 97-99), is used. This means that the discretization will be only stable for low Reynold's numbers (measured in relative mesh-fluid velocity). A remark should be made that this is not the case for the fluid solver investigated in this thesis since it can handle both high and low Reynold's numbers.

Matrix notation of the discretized equation for the structure:

$$\mathbf{M}^S \ddot{\mathbf{u}} + \mathbf{N}^S(\mathbf{u}) = -\mathbf{f}_\Gamma \tag{8}$$

Here $\mathbf{M}^S$ and $\mathbf{N}^S(\mathbf{u})$ denote the mass matrix and the internal force vector respectively and $\mathbf{u}$ is the vector of nodal displacement. The right hand side only consists of forces present at the coupling interface.

3.7 Modal superposition

The method to be evaluated in this thesis work is based upon that the dynamic behaviour of a structure can be expressed as a linear combination of its mode shape vectors. Therefore a short description of modal superposition will follow.

3.7.1 Introduction

Consider equation (9) which describes the motions of an undamped multi-degree freedom system subjected to forced vibrations:

$$\boldsymbol{m}\,\ddot{\boldsymbol{u}}(t) + \boldsymbol{k}\boldsymbol{u}\,(t) = \boldsymbol{f}(t) \tag{9}$$

$$\boldsymbol{m}\,\ddot{\boldsymbol{u}}(t) + \boldsymbol{k}\boldsymbol{u}(t) = \mathbf{0} \tag{10}$$

The basic idea of modal superposition is then to solve the free vibration system, equation (10) as an eigenproblem to obtain the eigenvalues and the corresponding eigenvectors (eigenmodes). This information can then be used to uncouple the equations of motion for the forced vibration problem. The uncoupled equations are called modal coordinates; these can be solved independently from each other. By superposition of the solved modal coordinates; the original forced vibration system can be solved. By using a finite number of eigenmodes, an approximate solution to the original problem can be obtained.

3.7.2 In practice

Consider once more equation (9) with the initial conditions: $\boldsymbol{u}(0) = \boldsymbol{u}_0$; $\dot{\boldsymbol{u}}(0) = \boldsymbol{v}_0$. Then solve equation (10), assuming the solution has the general form:

$$\boldsymbol{u}(t) = \boldsymbol{\varphi} \sin(\omega t + \phi_0) \tag{11}$$

After inserting (11) in (10) we arrive at the eigenvalue problem:

$$(\boldsymbol{k} - \lambda_i \boldsymbol{m})\, \boldsymbol{\varphi}_i = \boldsymbol{0} \tag{12}$$

which yields i=1,.....,n eigenvalues λ_i and mode shape vectors $\boldsymbol{\varphi}_i$.

According to the basic idea of modal superposition, the displacement vector can be expressed as a linear combination of the mode shape vectors in the following way:

$$\boldsymbol{u}(t) = z_1(t) \cdot \boldsymbol{\varphi}_1 + z_2(t) \cdot \boldsymbol{\varphi}_2 + \ldots. + z_n(t) \cdot \boldsymbol{\varphi}_n \tag{13}$$

where: z_1(t), z_2(t),...., z_n(t) are the uncoupled equations or modal coordinates.

The linear combination can also be expressed in matrix notation in the following way:

$$\boldsymbol{u}(t) = \boldsymbol{\phi} \cdot \boldsymbol{z}(t) \tag{14}$$

where: $\boldsymbol{\phi} = (\boldsymbol{\varphi}_1, \boldsymbol{\varphi}_2, \ldots., \boldsymbol{\varphi}_n)$ and $\boldsymbol{z}(t) = (z_1(t), z_2(t), \ldots., z_n(t))^T$

Inserting (14) in (9) and pre-multiplying both sides with $\boldsymbol{\phi}^T$ will yield:

$$\boldsymbol{\phi}^T \boldsymbol{m} \cdot \boldsymbol{\phi} \cdot \ddot{\boldsymbol{z}}(t) + \boldsymbol{\phi}^T \boldsymbol{k} \cdot \boldsymbol{\phi} \cdot \boldsymbol{z}(t) = \boldsymbol{\phi}^T \boldsymbol{f}(t) \tag{15}$$

Defining mass- and stiffness orthogonality as

$$\boldsymbol{\varphi}_i^T \boldsymbol{m} \cdot \boldsymbol{\varphi}_j = \begin{cases} M_i & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad ; \quad \boldsymbol{\varphi}_i^T \boldsymbol{k} \cdot \boldsymbol{\varphi}_j = \begin{cases} K_i & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

and applying this to (15) will give the uncoupled equations:

$$\text{diag}(\boldsymbol{M}) \cdot \ddot{\boldsymbol{z}}(t) + \text{diag}(\boldsymbol{K}) \cdot \boldsymbol{z}(t) = \boldsymbol{F}(t) \tag{16}$$

Where the right hand side, the modal load vector, is defined as:

$\boldsymbol{F}(t) := \boldsymbol{\phi}^T \boldsymbol{f}(t)$

Pre-multiplying (16) with the inverted diag($\boldsymbol{M}$) on both sides will result in:

$$\ddot{z}_i(t) + \frac{K_i}{M_i} \cdot z_i(t) = \frac{F_i(t)}{M_i} \tag{17}$$

Returning to (12), pre-multiplying the equation with $\boldsymbol{\phi}^T$ and using mass- and stiffness orthogonality will give:

$$\lambda_i := \omega_i^2 = \frac{K_i}{M_i} \tag{18}$$

Inserting (18) in (17) will result in the modal equations:

$$\ddot{z}_i(t) + \omega_i^2 \cdot z_i(t) = \frac{F_i(t)}{M_i} \quad \text{for } i = 1,....,n \tag{19}$$

The modal equations need to be solved using the initial conditions rewritten for the modal equations. The solved modal equations will give the solution to the original equation (9) after inserted in (13). Thus the approximate solution to the original equation will be:

$$\boldsymbol{u}(t) \approx z_1(t) \cdot \boldsymbol{\varphi}_1 + z_2(t) \cdot \boldsymbol{\varphi}_2 + ...+ z_m(t) \cdot \boldsymbol{\varphi}_m \tag{20}$$

Where m is a finite number and represents the number of eigenmodes that are sufficient to well describe the dynamics of the original system.

4 The software packages

In this thesis, mainly two software packages have been used, Abaqus Standard and AcuSolve. The following section will give some brief information about the packages and the functions that have been used within each of them.

4.1 AcuSolve

General:

AcuSolve is developed by the company AcuSim. AcuSim is one of the leading providers of computational fluid dynamics (CFD) solver solutions. The company was founded in 1994 and today they have customers all over the world. AcuSolve is a FEM based CFD solver for incompressible and weakly compressible flows.

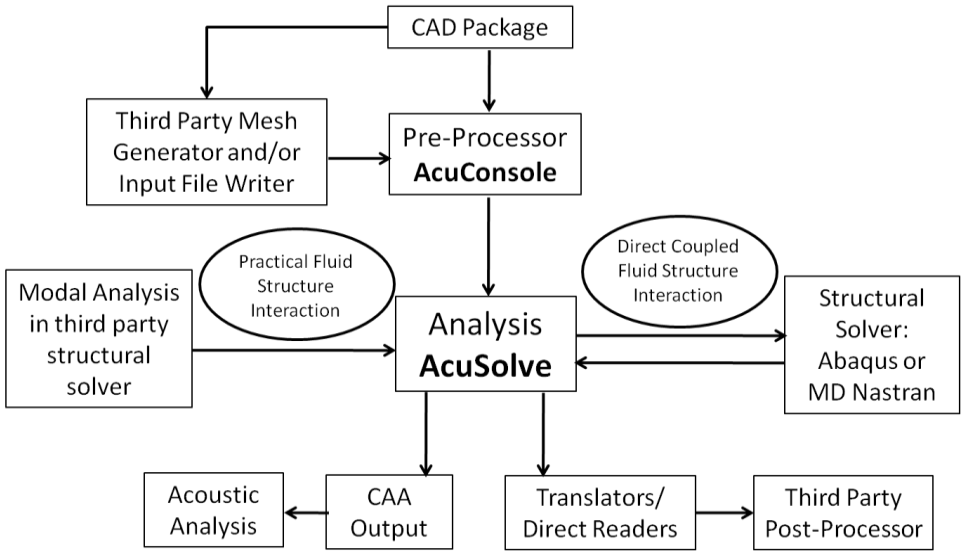


Fig. 7. AcuSolve's software structure

Flow equations:

AcuSolve implements the incompressible Stokes and incompressible/weakly compressible Navier-Stokes equation.

Features:

AcuSolve enables a number of different analyses, below are some of them.

- Fluid structure interaction (FSI) analysis
- Thermal analysis and conjugate heat transfer
- Computational Aero-Acoustic (CAA)

Furthermore AcuSolve supports several turbulence models and ALE mesh motion. For more information about the software, see [6].

Functions used:

In this thesis work, two approaches or methods have been used to solve FSI-problems, Direct Coupled FSI (DC-FSI) and practical FSI (P-FSI). In the subsections below, some brief information about the two methods will be given. Since P-FSI is the method to be evaluated, the emphasis will be on that section.

4.1.1 DC-FSI

DC-FSI is AcuSolve's own name for the partitioned approach of solving FSI-problems. Within DC-FSI, AcuSolve enables cooperation with the third-party structural solvers Abaqus and MD Nastran, supporting both the sequential staggered solution scheme (explicit) and the iterative staggered scheme (semi-implicit). The limitation is in the structural solver whether it supports the semi-implicit coupling or only the explicit one.

In this thesis work, Abaqus 6.91 has been used as the structural solver and from now on Abaqus 6.91 will be referred to as Abaqus. Since Abaqus does not support semi-implicit coupling, the direct coupled simulations have only been carried out as explicit.

4.1.2 P-FSI

General:

P-FSI is a unique method for AcuSolve. Within this method the FSI-problems are solved in AcuSolve without real-time coupling with the structural solver. P-FSI uses an iterative staggered algorithm called multi-iterative coupling (MIC). This coupling algorithm uses predictors and correctors in each iteration to stabilize the solution, see [12]. According to [12] the MIC scheme gives more stable computations for lower structure-fluid ratios than for example the sequential staggered scheme.

To carry out the analysis, AcuSolve needs information about the dynamic behaviour of the structure - this is imported to AcuSolve as eigenmodes, extracted from a previous frequency analysis in the structural solver. In this thesis work, the frequency analyses have been carried out in Abaqus.

Advantages:

According to [6] some of the advantages of the P-FSI method are:

- No run time coupling is required
  - The codes can be run with a different time increment and different duration
- More stable
  - Eliminates high wave number modes
- Efficiency
  - Shorter setup time
  - Shorter CPU time

Limitations:

According to [6] some of the limitations of the method are:

- Only linear structural analysis is supported
- Only a small number of modes is practical

The method:

P-FSI is only applicable for linear structural problems. The method will be described by a sequence of steps. The theory of the method is based on the theory of modal superposition given in (3.7).

1. Assume linear structural deformation. The dynamic behaviour of the structure can then, discretized with finite elements, be expressed as:

   $$\boldsymbol{Ma} + \boldsymbol{Cv} + \boldsymbol{Kd} = \boldsymbol{F} \tag{21}$$

   Where:

   $\boldsymbol{M}$: denotes the mass matrix
   $\boldsymbol{a}$: denotes the nodal acceleration
   $\boldsymbol{C}$: denotes the damping matrix
   $\boldsymbol{v}$: denotes the nodal velocity
   $\boldsymbol{K}$: denotes the stiffness matrix
   $\boldsymbol{d}$: denotes the nodal displacement
   $\boldsymbol{F}$: denotes the forces applied on the structure

2. Consider the simplified semi-discrete system:

   $$\boldsymbol{M}\frac{d^2\boldsymbol{u}}{dt^2} + \boldsymbol{Ku} = \boldsymbol{0} \tag{22}$$

   Where, in this case, $\boldsymbol{u}$ denotes the nodal displacement.
   Assume the solution has the general form:

   $$\boldsymbol{u}(t) = \boldsymbol{\varphi}\sin(\omega t + \phi_0) \tag{23}$$

When eq. (23) is inserted in eq. (22) it yields the generalized eigenvalue problem of finding eigenvalues λ_i and their corresponding eigenvectors $\boldsymbol{\varphi}_i$ satisfying the equation:

$$(\boldsymbol{K} - \lambda_i \boldsymbol{M})\,\boldsymbol{\varphi}_i = \mathbf{0} \tag{24}$$

where i=1,...,n_{modes} represents the number of eigenmodes. The method depends upon that the dynamic behaviour of the structure can be sufficiently described by n_{modes}. The modes are extracted from a frequency analysis carried out in a structural solver.

3. Projection of dynamic system into the eigenspace yields the modal equations:

$$\boldsymbol{m}\,\ddot{\boldsymbol{u}} + \boldsymbol{c}\,\dot{\boldsymbol{u}} + \boldsymbol{k}\boldsymbol{u} = \boldsymbol{f} \tag{25}$$

Where:

$$\boldsymbol{m} = \boldsymbol{\phi}^T \cdot \mathbf{M} \cdot \boldsymbol{\phi} = \mathbf{I}$$
$$\boldsymbol{c} = \boldsymbol{\phi}^T \cdot \mathbf{C} \cdot \boldsymbol{\phi}$$
$$\boldsymbol{k} = \boldsymbol{\phi}^T \cdot \mathbf{K} \cdot \boldsymbol{\phi} = \text{diag}(\lambda_i)$$
$$\boldsymbol{f} = \boldsymbol{\phi}^T \cdot \mathbf{F}$$

F is first computed from the flow field and is then projected to $\boldsymbol{f}$

4. After the modal equations have been solved, the solution to the original equation (21) can be obtained by the following linear combination:

$$\boldsymbol{d} = \mathbf{S}\mathbf{u}$$
$$\boldsymbol{v} = \mathbf{S}\,\dot{\boldsymbol{u}}$$
$$\boldsymbol{a} = \mathbf{S}\,\ddot{\boldsymbol{u}}$$

In practice:

- Step 1: Performed in structural modeller/solver
  - Model structural geometry
  - Mesh geometry
  - Perform eigenmode analysis
- Step 2: Model fluid domain
- Step 3: Projection, easiest performed in AcuConsole
  - An independent fluid mesh is created
  - The coupling interface is identified
  - Eigenvectors from the structural eigenmode analysis are projected to the fluid nodes at the interface

- Step 4: AcuSolve
    - ODE coefficients and projected eigenvectors are passed to the solver
    - The transient coupled flow and linear deformation is solved
        - Given the solution of ODE:s and eigenvectors
            - Construct and impose BC:s on mesh displacement or fluid velocity of fluid interface nodes
        - Given the flow solution
            - Compute the nodal forces
            - Multiply by eigenvectors to get the ODE-forces
            - Time advance the ODE-solution
- Step 5: Post processing

4.2 Abaqus

Abaqus is a software developed by Simula which is owned by Dassault Systems Cooperation. Catia, SolidWorks and Delmia are some other software packages owned by Dassault.

Abaqus enables simulations of real-world problems in a virtual environment and provides the ability to solve complicated FEM-setups, multiphysics, process automation and design optimization just to mention a few of its capabilities.

In this thesis work, Abaqus CAE has been used for its modelling capabilities and for performing structural analyses. When performing analyses in Abaqus, one creates steps which describe the type of analysis to be carried out. In this thesis work, frequency analysis and dynamic implicit analysis have been performed, the frequency analysis to extract the eigenmodes of the structure to be used in P-FSI and the dynamic implicit analysis to carry out the direct coupled simulation together with AcuSolve (DC-FSI).

Furthermore the modelling capabilities have been used to model the domains of the fluid and the structure.

5 The approach to perform the couplings

The following section describes in a loose way how to set up the simulation for P-FSI and DC-FSI.

5.1 P-FSI

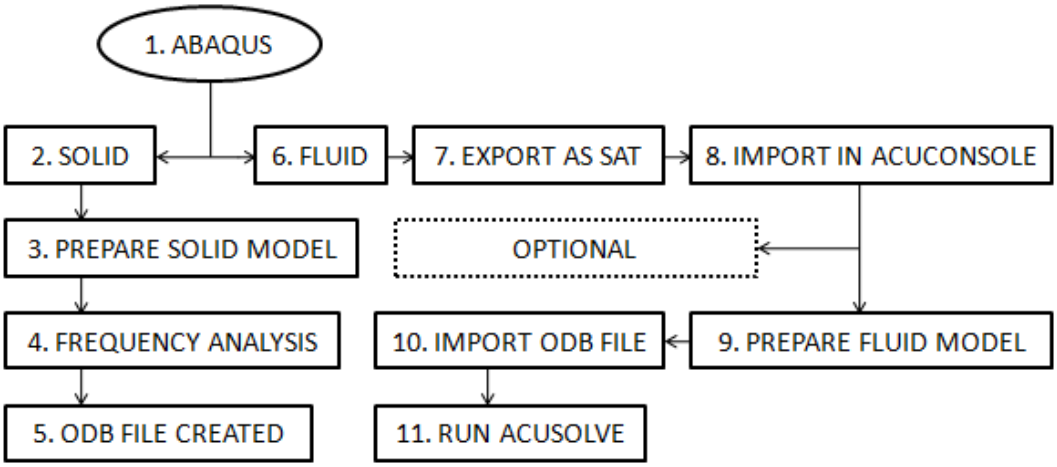


Fig. 8. Scheme for P-FSI

1. Start Abaqus version 6.91 in Windows or Linux environment. Later versions of Abaqus 6.91 do not support the P-FSI connection with AcuSolve.
2. Import the structure of the solid from an external program or simply use the CAE to make your own. Make sure to scale the dimensions in SI-units and remember to place the part so the coordinates will interact with the fluid part you will create later.
3. Prepare the solid model in the following steps:
    - Choose the material parameters of the solid.
    - Make a section and apply the material.
    - Create an instance in the assembly module.
    - Create a new step and choose to do a frequency analysis of the solid. Set the eigensolver to Lanczos, choose the amount of eigenvalues to extract (you can also choose to set maximum frequency) and choose to normalize the eigenvectors by mass.
    - Apply the proper BC:s
4. Create a new job and run the analysis (before running the analysis you can change the work directory to a chosen one).
5. The frequency analysis creates an odb file which contains information about the solids eigenmodes.
6. Import the geometry of the fluid into Abaqus from an external program or simply use the CAE to make your own. Make sure to scale the dimensions in SI-units and remember to place the part so the coordinates interacts with the solid part created earlier.
7. Export the fluid part as a sat file.

8. Open AcuConsole and create a new file. Import the sat file of the fluid (made in Abaqus) in the new part. Choose to import in meters.
9. When preparing the fluid for P-FSI several steps need to be taken. Delivered in short terms:
    - Define the fluid properties (if the fluid is other than water or air you have to create a new "material model" to use in this step)
    - Mesh the model (this have to be done in AcuConsole because its ability to import volumetric meshes are limited)
    - In the problem description put analysis type to transient, flow equation to Navier-Stokes, turbulence equation to laminar and set the mesh type as ALE.
    - In the auto solution menu you choose the appropriate values of max time steps, time increments, convergence tolerance etc.
    - Define the surfaces and the BC:s of the fluid
    - Create a new "flexible body". This will be used as reference for the movement of the structure.
    - Define the mesh displacement on the surfaces of the fluid. Make sure to set the surface/surfaces that interact with the solid to "flexible body".
    - Create output variables
    - Define a layer of nodes outside of the interface. These nodes will be defined to move in conjunction with the solid.
10. Use the eigenmode manager to import the odb file. Visualize the model (fluid and structur) to make sure that the dimensions and the placement are correct. Transfer flexible body, the BC and the layer of nodes.
11. Run AcuSolve. Use AcuProbe to visualize the output.

In the square named OPTIONAL you can choose to import the odb file and visualize the model confirming that the dimensions and the placement is correct.

5.2 DC-FSI

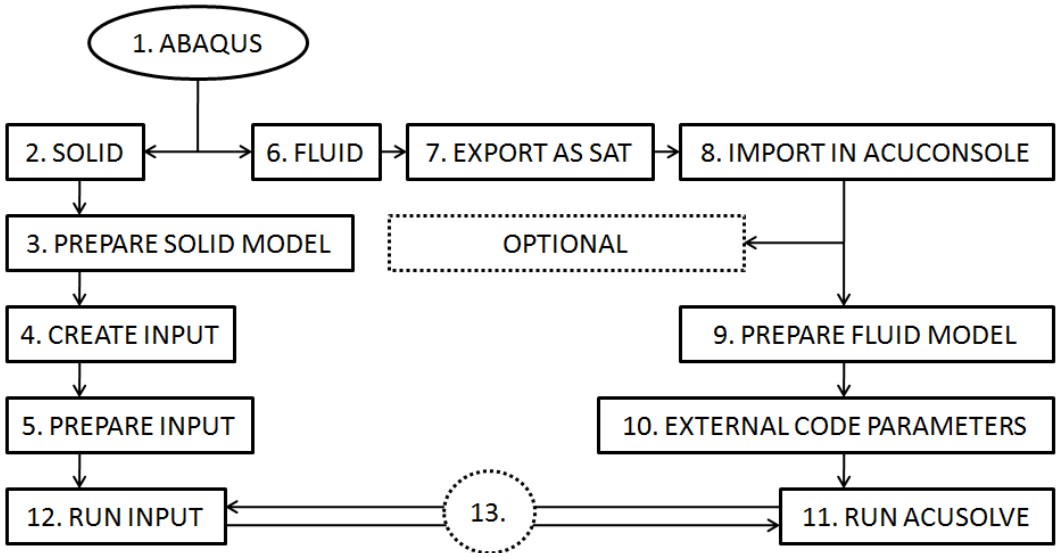


Fig. 9. Scheme for DC-FSI

1. Start Abaqus version 6.91 in Windows environment. Later versions of Abaqus 6.91and Linux do not support the DC-Coupling.
2. Import the structure of the solid from an external program or simply use the CAE to make your own. Make sure to scale the dimensions in SI-units and remember to place the part so the coordinates will interact with the fluid part you will create later.
3. Prepare the solid in the following steps:
    - Choose the material parameters of the solid.
    - Make a section and apply the material.
    - Create an instance in the assembly module.
    - Create a surface where the solid will interact with the fluid.
    - Create a dynamic implicit step. Choose time period and if you have large displacement set Nlgeom to on. Set proper time incrementations.
    - Apply the proper BC:s.
    - Create an interaction. Choose the interacting surface you created earlier.
4. Instead of running an analysis you create a new job and then you right click on the new job and choose write input (before writing the input you can change the work as desired). This input is the file we will use in the coupling (we still need to edit this a bit, see next step).
5. Go to the catalogue were the created input is. Right click on it and choose to edit with notepad. In this input you will have to edit a few things. Information of the proper input settings are available in (A.1).

6. Import the geometry of the fluid to Abaqus from an external program or simply use the CAE to make your own. Make sure to scale the dimensions in SI-units and remember to place the part so the coordinates interacts with the solid part created earlier.
7. Export the fluid part as a sat file.
8. Open AcuConsole and create a new file. Import the sat file of the fluid (made in Abaqus) in the new part. Choose to import in meters.
9. When preparing the fluid for DC-FSI several steps need to be taken. Delivered in short terms:
    - Define the fluid properties (if the fluid is other than water or air you have to create a new "material model" to use in this step)
    - Mesh the model (this have to be done in AcuConsole because its ability to import volumetric meshes are limited)
    - In the problem description put analysis type to transient, flow equation to Navier-Stokes, turbulence equation to laminar and set the mesh type as ALE. Set External Code to on.
    - In the auto solution menu you choose the appropriate values of max time steps, time increments, convergence tolerance etc.
    - Define the surfaces and the BC:s of the fluid
    - Define the mesh displacement on the surfaces of the fluid. Make sure to set the surface/surfaces that interact with the solid to "external code surface" and then set the gap-factor to zero.
    - Create output variables
    - Define a layer of nodes outside of the interface. These nodes will be defined to move in conjunction with the solid.
    - Create a "multiplier function" of which you can choose to ramp the forces at the beginning of the coupling.
10. Go to the "external code parameters" menu. Choose socket initiate to off and choose socket port. Choose the multiplier function you created earlier.
11. Run AcuSolve. AcuSolve is now waiting for Abaqus to connect.
12. Open dos-prompt. Go to the catalog where you created the input-file. Write the following command in the catalog:

    abq691 –job (name of your input-file) –port (the socket port nr you chosen in AcuConsole).

    This command connects Abaqus with the already waiting AcuSolve. Use AcuProbe to vizualize the output.
13. While running the programs exchange information at the start of every time-step (explicit).

In the square named OPTIONAL you can choose to import an odb file in the eigenmode manager and visualize the model confirming that the dimensions

and the placement is correct. You can create an odb-file performing a frequency analysis in Abaqus.

6 Evaluation method

6.1 Criteria

The accuracy, speed and robustness of the two methods will be investigated by comparing how well they perform solving a benchmark problem with changing data. The comparison criteria will be:

- For accuracy
  - The correlation in displacement between each method and the benchmark problem.
- For speed
  - The CPU-time
- For robustness
  - How well each method performs with changing time-step size and changing density ratios μ.

6.2 Steps

The evaluation will be conducted by performing the following sequence of steps:

1. The performance of each method solving the initial benchmark problem. Here *initial* benchmark problem refers to the data that is given in the original benchmark problem.
2. The difference in result when solving the initial benchmark problem with the P-FSI method using different number of modes.
3. The difference in result between the methods when changing the benchmark problem's initial data: density ratio and time-step size. The density ratio is varied by decreasing the density of the structure while keeping the density of the fluid constant.

7 Benchmark problem

In this section the benchmark problem is presented. The original benchmark problem *Elastic flap in a duct* is available in the tutorial section of [1]. We choose this set up because of the available results to use as reference values.

7.1 Problem description

The benchmark problem consists of an elastic flap in a duct, with inlet airflow of 8 m/s. The remaining boundary conditions, geometrical measurements and mesh settings can be found in the tutorial section of [1]. The displacement of a monitor point situated on the outer edge of the flap is to be evaluated. The time-step size is set to 0.25 ms in the benchmark problem.

7.2 Analysis preparation

The preparation of the coupled simulations follows the procedures stated in section 5.1 and 5.2. The following input files, regarding the initial benchmark problem, can be found in (A.1)-(A.4)

- DC-FSI simulation Abaqus (A. 1)
- DC-FSI simulation AcuSolve (A. 2)
- Frequency analysis Abaqus (A. 3)
- P-FSI simulation AcuSolve (A. 4)

7.2.1 Fluid mesh

The fluid domain is meshed with 26244 tetrahedral elements. At the surface of the flap the boundary layer meshing option is activated. This will allow for finer mesh close to the flap, see figure below.

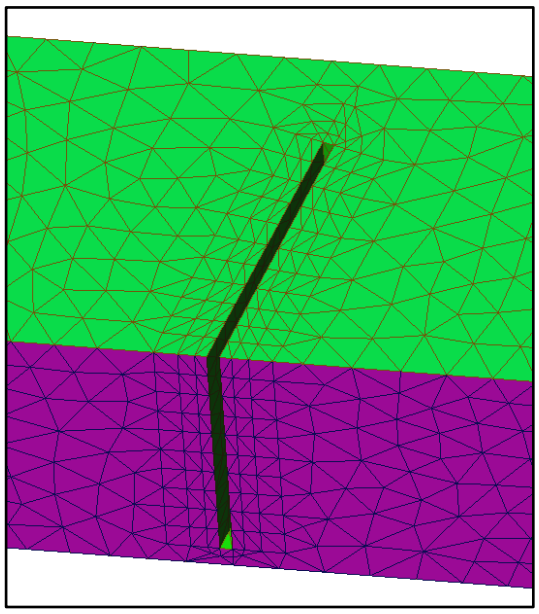


Fig. 10. Local refinements

7.2.2 Structural mesh

The structure is meshed with 800 elements of type 8-node linear brick element (C3D8R).

8 Results

8.1 Displacements compared to benchmark problem

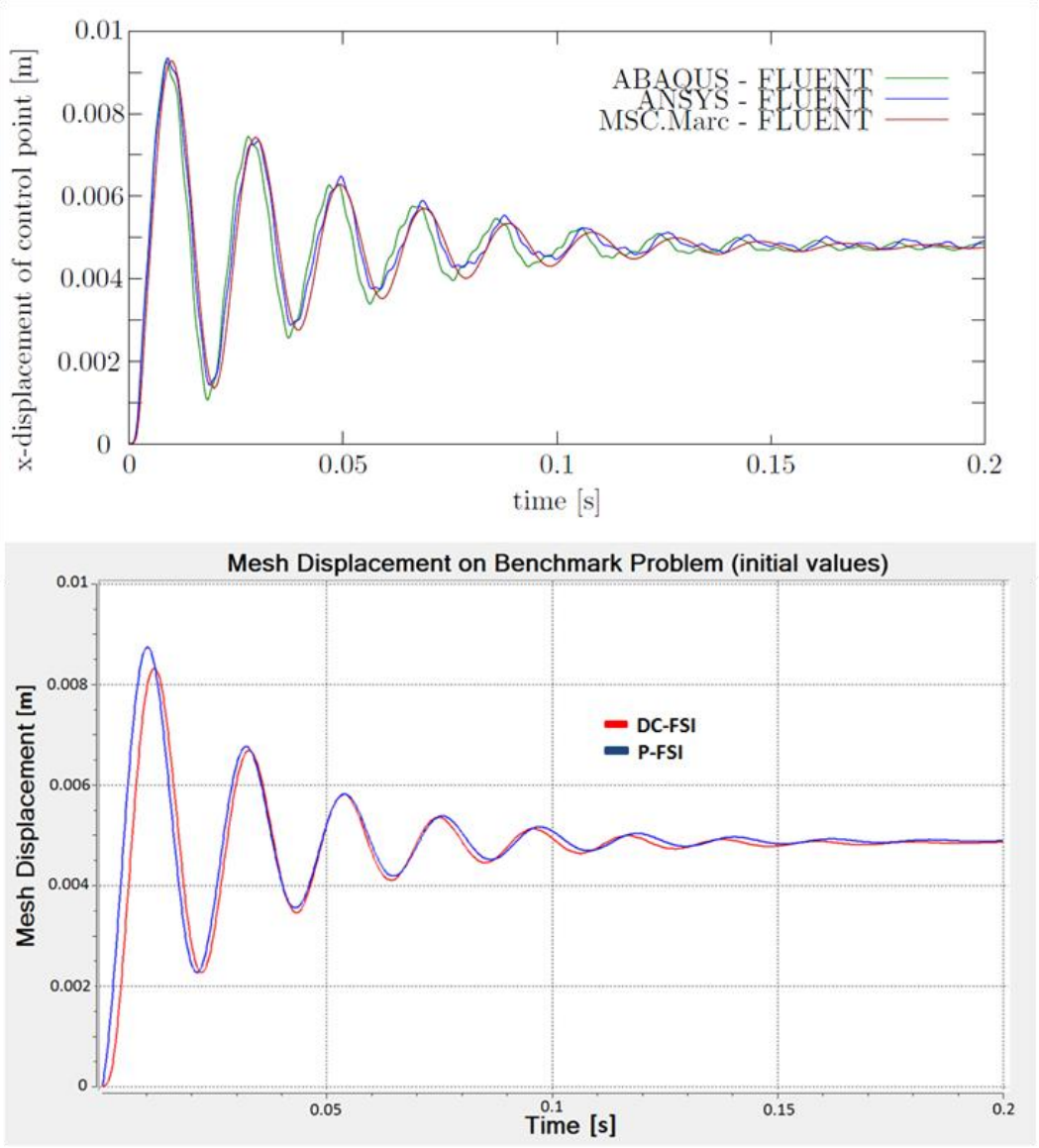


Fig. 11. Upper graph is copied from [1] and shows the displacement of the control point. The lower graph shows the result from the analysis of the same problem, but when using P-FSI and DC-FSI.

8.2 P-FSI, different number of modes

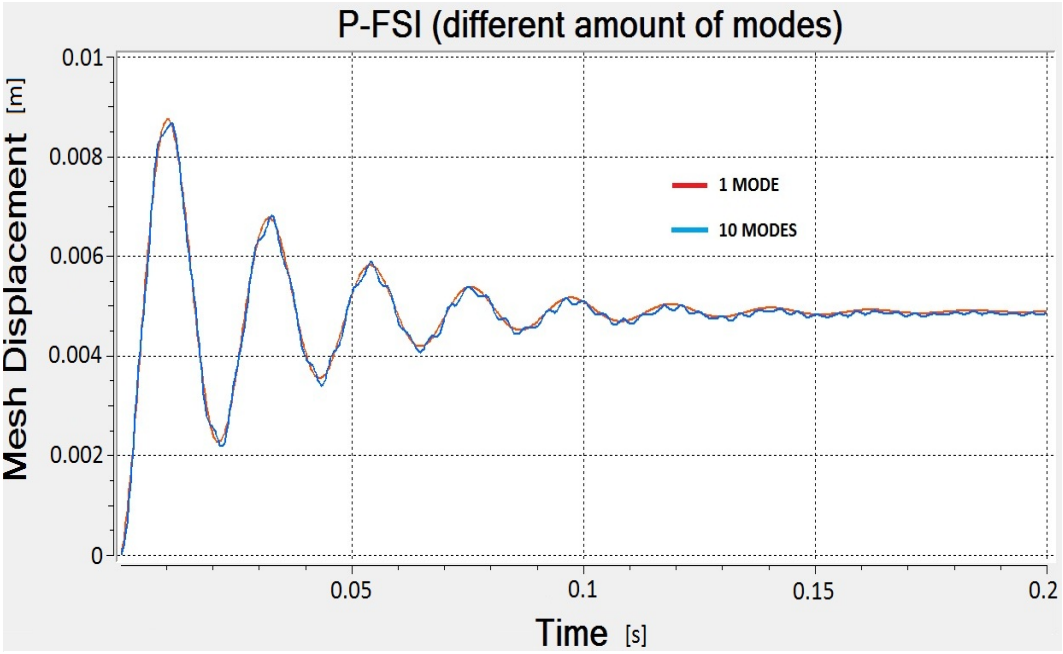


Fig. 12. The graph shows the difference in result analyzing the initial benchmark problem when using 1 mode respectively 10 modes.

8.3 Density ratios

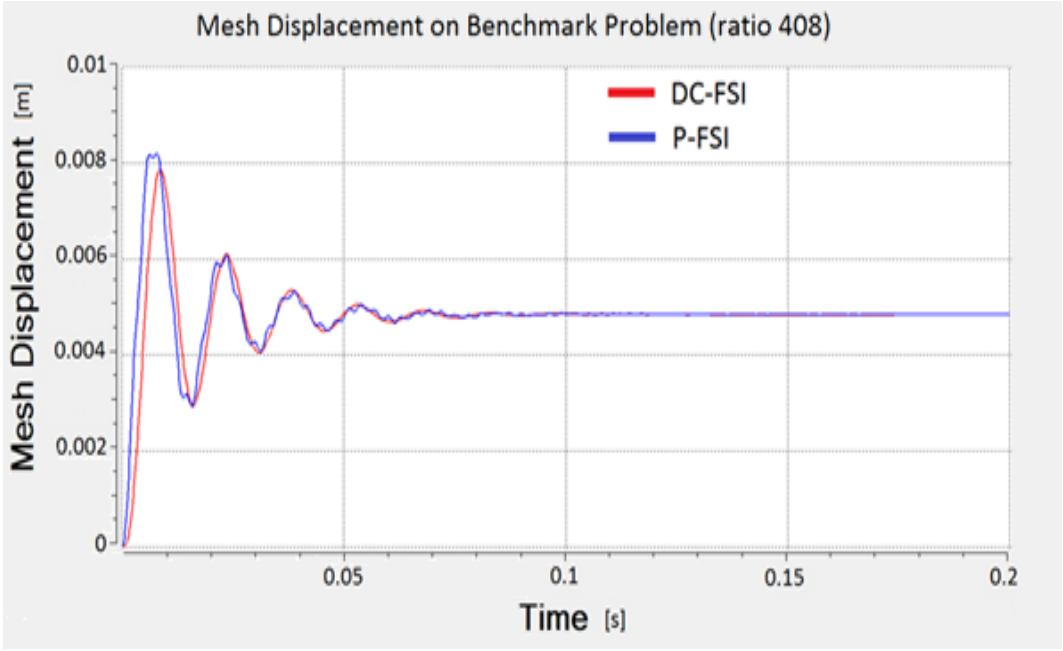


Fig. 13. The graph shows the difference in result between DC-FSI and P-FSI for ratio μ=408 and time-step size 0.25 ms.

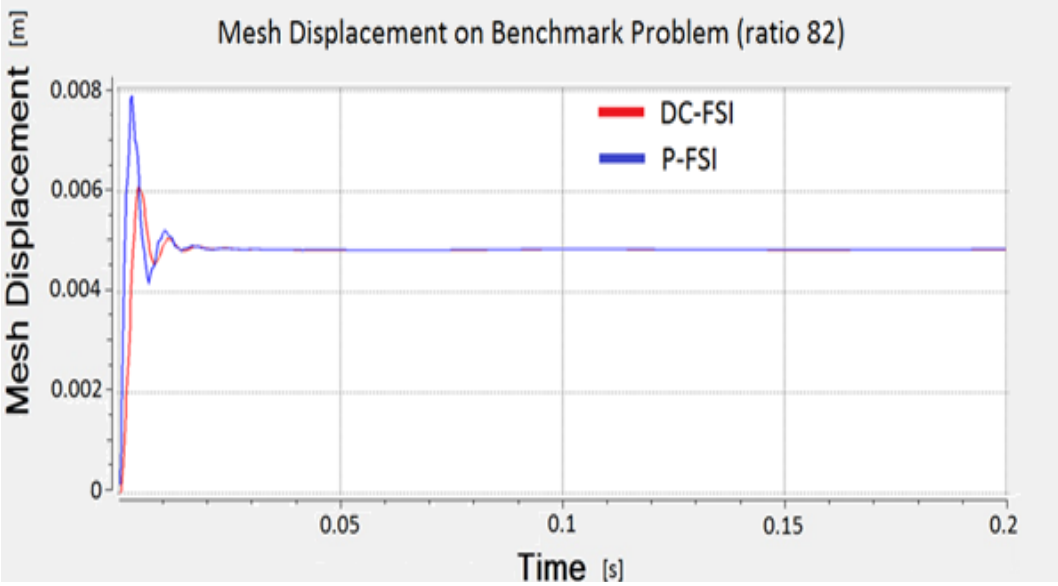


Fig. 14. The graph shows the difference in result between DC-FSI and P-FSI for μ = 82 and time-step size 0.25 ms.

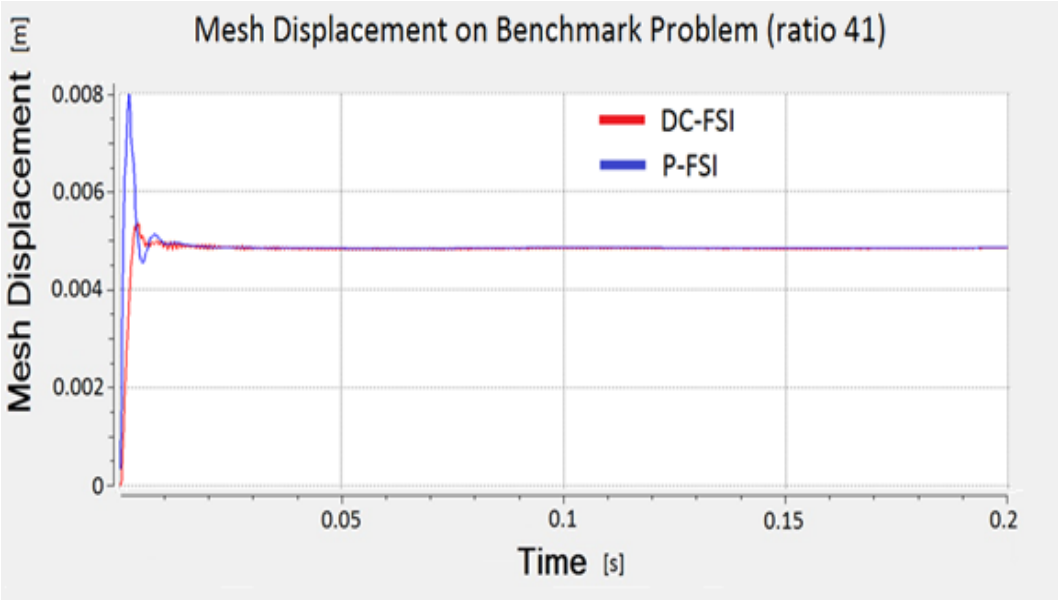


Fig. 15. The graph shows the difference in result between DC-FSI and P-FSI for μ = 41 and time-step size 0.25 ms.

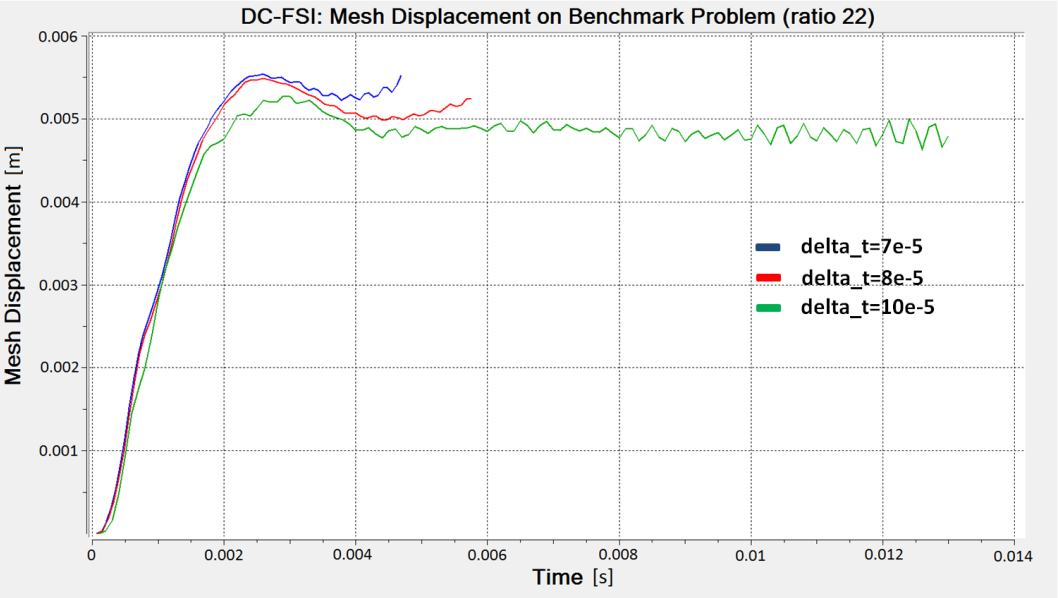


Fig. 16. The graph shows the difference in result for DC-FSI, analyzing the benchmark problem, with different time-step sizes.

8.4 Computing time with decreasing ratio

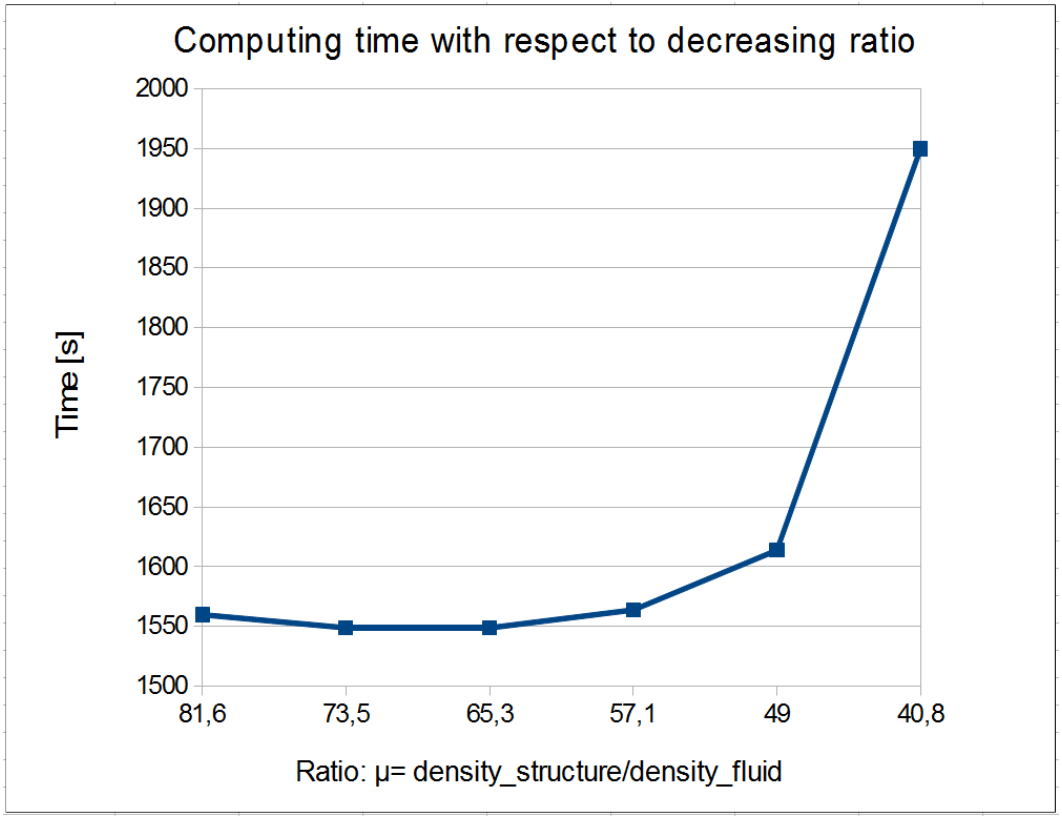


Fig. 17. The graph shows the computing time for DC-FSI, solving the benchmark problem, with decreasing ratio.

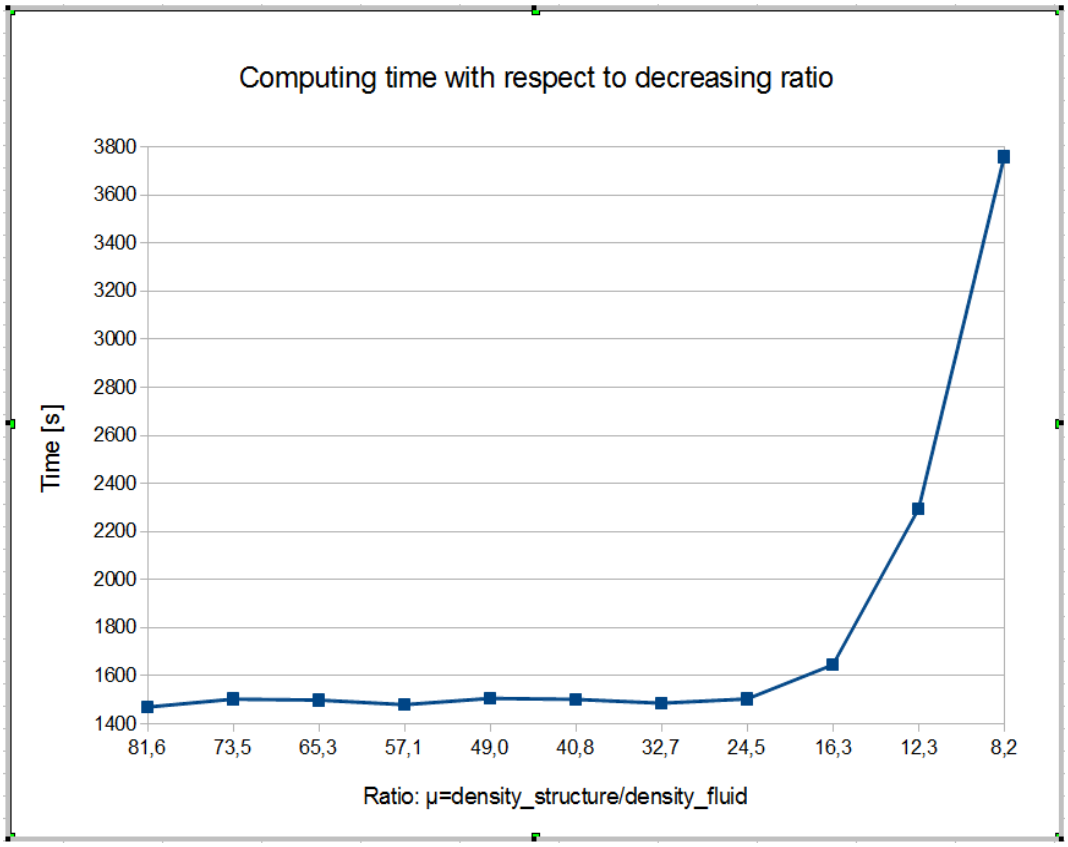


Fig. 18. The graph shows the computing time for P-FSI, solving the benchmark problem, with decreasing ratio.

9 Evaluation of results

9.1 Evaluation of displacements compared to benchmark problem

As can be seen in Fig. 11 there is a slight difference in displacement amplitude between the graph of the benchmark problem and the result from the DC- and P-FSI simulations. There is also a slight frequency difference, but they converge towards the same final displacement.

A slight difference in amplitude when comparing the result from DC-FSI to P-FSI can also be seen. This phenomenon is due to the different coupling techniques where the explicit scheme, because neglecting the fluid inertia in the first time step, shows a damping effect as discussed in 3.2.3.

9.2 Evaluation of different amount of modes

As can be seen in Fig. 12, increased number of modes will, in this case, result in more noise in the graph. This can be reduced by truncating the eigenmodes which can be seen in the figure when using only 1 mode. This is because the structural model looses degrees of freedom when using only one eigenmode to represent the dynamic behaviour.

The computing time will be affected by the choice of number of eigenmodes. This is because truncating the problem leads to less iterations and to a smaller matrix system. The computing time for 1 mode was 21.41 min. and for 10 modes was 24 min.

9.3 Evaluation of density ratio

When decreasing the density ratio, a big difference in amplitude when comparing the two methods can be seen, see Fig. 13, Fig. 14 and Fig. 15. The DC-FSI simulations lose a lot in amplitude compared to P-FSI which is a result of the explicit scheme together with the added mass term, see section 3.2.3. They both converge towards the same final displacement.

For the constant time-step size of 0.25 ms, DC-FSI was stable down to a ratio of 41, where only small disturbances could be seen. P-FSI showed stability down to a density ratio of 8.2. This is also consistent with the theory discussed in section 3.2.3.

Fig. 16. shows that the instabilities will occur earlier with decreased time-step size. This is also consistent with what is mentioned in section 3.2.3.

9.4 Evaluation of computing time

When comparing the computing time between the two methods, Fig. 17. and Fig. 18. show similar computing time down to a ratio of 57. Below this ratio, DC-FSI starts to be unstable and more stagger iterations are needed to reach the tolerance. Below ratio 40.8 and a time-step size of 0.25 ms, DC-FSI cannot reach a solution. For P-FSI, computing time down to a ratio of 8.2 was registered. The method shows almost a constant computing time down to a ratio of 24.5 and below this ratio the computing time increases exponentially.

10 Conclusions

When recalling the comparison criteria: accuracy, speed and robustness, the P-FSI method shows the best result in each category when compared to DC-FSI. A remark concerning that the P-FSI method only works for linear problems should be made. The method also requires that the user is familiar with the analysis to be carried out and can decide how many eigenmodes that is sufficient to well describe the dynamic behaviour of the system. For problems where small deformation theory can be used, the result shows that this is an accurate, robust and efficient method.

11 Discussion

The two coupling techniques between Abaqus and AcuSolve were at the outset of this project barely used within FS dynamics. Furthermore the authors (thesis workers) had no experience of the software packages in question. As a result, it took quite some time to get the programs to interact. This was maybe due to inexperience but we also found some bugs in the packages during this time. A few of them are as follows:

- The DC-coupling did not work using Linux because of a bug in AcuSolve. This led to the use of Windows interface. Sadly Windows had some other limitations.
    - In Windows you cannot use the Abaqus FSI module to define the interaction surfaces to use in the coupling. This led to correction of the input-file in notepad.
    - The ability of postprocessing the output from AcuSolve and convert it to an odb-file did not work in Windows.
- When creating an odb-file in Abaqus this had to be done in Abaqus 691 or older versions. Else the file cannot be imported in AcuSolve.
- AcuSolve cannot import volumetric meshes. This is very limiting because FS Dynamics often uses Ansa as a pre-processor.
- You must import the fluid model in SI units in AcuSolve otherwise it does not match the dimensions when the coupling is carried out. This, in turn, gives the need to use SI units when modelling the structures in Abaqus.

Further investigations should be done in order to really find out the benefits and limitations of P-FSI and whether it is an appropriate method for simulating (AA) and (FIV). In this thesis work a large number of analyses were carried out, but most of them on problems with similar geometry. To further evaluate the method analyses on problems with more complex geometry, where real-world data for comparison exists, should be carried out.

Furthermore the writers think that the method should be compared to a coupling which supports an iterative staggered algorithm.

12 References

1. *MpCCI 4.1.0*. 2011-04-20 [cited 2011 -05-17]; Available from: http://www.mpcci.de/fileadmin/mpcci/download/MpCCI-4.1.0/doc/pdf/MpCCIdoc.pdf.
2. *FS-Dynamics Web Page*. [cited 2011- 05-06]; Available from: www.fsdynamics.se.
3. O.C. Zienkiewicz, R.L.T.J.Z.Z., *The finite element method - its basis & fundamentals*. 2008: Butterworth-Heinemann.
4. *Adina FSI*. [cited 2011- 05-03]; Available from: http://www.adina.com/fluid-structure-interaction.shtml.
5. Förster, C., W.A. Wall, and E. Ramm, *Artificial added mass instabilities in sequential staggered coupling of nonlinear structures and incompressible viscous flows.* Computer Methods in Applied Mechanics and Engineering, 2007. **196**(7): p. 1278-1293.
6. *AcuSolve documentation*. 2010 [cited 2011- 05-03]; Available from: http://www.acusim.com/doc/.
7. J. Donea, A.H., J.-Ph. Ponthot and A. Rodriguez-Ferran, *Arbitrary Lagrangian-Eulerian methods*, in *Encyclopedia of Computational Mechanics*. 2004. p. 1-38.
8. H, M., *ALE*. 1999: Gløshaugen. p. 13.
9. Redaelli, L., *Numerical Approximation of Fluid-Structure Interaction Problems*, in *Department of Mechanical Engineering Faculty of Engineering*, Fukui University: Fukui. p. 8.
10. Hansbo, P., *Applied partial differential equations for computational science and engineering, part 2: nonlinear models in continuum mechanics*. 2011.
11. P. Causin, J.F.G., F. Nobile, *Added-mass effect in the design of partitioned algorithms for fluid-structure problems.* Science Direct, 2005 p. 4506–4527.
12. Jaiman, S., Oakley, Constantinides, *Fully Coupled Fluid-Structure Interaction for Offshore Applications*, in *Offshore Mechanics and Arctic Engineering* 2009: Honolulu, Hawaii. p. 9.

13 Appendix

```
A1
***************The inputs for the DC-coupling in ABAQUS********************
**in a regular input file, the structural models nodes and their coordinates is
written before this section
*Surface, type=ELEMENT, name=combine
_combine_S6, S6
_combine_S2, S2
_combine_S4, S4
_combine_S3, S3
*End Assembly
**
** MATERIALS
**
*Material, name=Material-1
*Density
50.,
*Elastic
 1e+08, 0.49
**
** BOUNDARY CONDITIONS
**
** Name: BC-1 Type: Symmetry/Antisymmetry/Encastre
*Boundary
_PickedSet10, PINNED
** ----------------------------------------------------------------
**
** STEP: DC-FSI
**
*Step, name=DC-FSI, nlgeom=YES, inc=10000
*Dynamic,alpha=-0.05,direct
4e-5,2.,1e-10
**1e-10,2e-4
**
** OUTPUT REQUESTS
**
*Restart, write, frequency=10
**
** FIELD OUTPUT: F-Output-1
**
*Output, field, frequency=5
*Node Output
RF, U
**
** FIELD OUTPUT: F-Output-2
**
*Output, field
*Element Output, directions=YES
S,
**
** HISTORY OUTPUT: H-Output-1
**
*Output, history, variable=PRESELECT
**
*CO-SIMULATION, PROGRAM=ACUSOLVE, CONTROLS=CTR
*CO-SIMULATION REGION, TYPE=SURFACE, EXPORT
combine,U
*CO-SIMULATION REGION, TYPE=SURFACE, IMPORT
combine, CF
*CO-SIMULATION CONTROLS,NAME=CTR, TIME INCREMENTATION=SUBCYCLE, STEP
SIZE=0.00004,TIME MARKS=NO
**
*End Step
*
*
```

```
A2
***************The inputs for the DC-coupling in ACUSOLVE*******************
**in a regular input file, the structural models nodes and their coordinates is
written before this section
# +-----------------------------------------------------------------------+
# | Problem Description                                                   |
# +-----------------------------------------------------------------------+

ANALYSIS {
    title                               = "AcuSolve Problem"
    sub_title                           = "AcuSolve Problem"
    input_version                       = "1.8"
    type                                = transient
}

EQUATION {
    flow                                = navier_stokes
    absolute_pressure_offset            = 0.0           # N/m2
    temperature                         = none
    species_transport                   = none
    turbulence                          = none
    mesh                                = arbitrary_lagrangian_eulerian
    external_code                       = on
    running_average                     = off
}

# +-----------------------------------------------------------------------+
# | External Code Parameters                                              |
# +-----------------------------------------------------------------------+

EXTERNAL_CODE {
    communication                       = socket
    socket_initiate                     = off
    socket_host                         = "tirian50"
    socket_port                         = 10000
    multiplier_function                 = "Multiplier Function 1"
    filter                              = none
}

# +-----------------------------------------------------------------------+
# | Auto Solution Strategy                                                |
# +-----------------------------------------------------------------------+

AUTO_SOLUTION_STRATEGY {
    max_time_steps                      = 10000
    final_time                          = 2.0           # sec
    initial_time_increment              = 0.00025       # sec
    auto_time_increment                 = off
    convergence_tolerance               = 0.001
    min_stagger_iterations              = 0
    max_stagger_iterations              = 10
    num_krylov_vectors                  = 10
    relaxation_factor                   = 0.0
    flow                                = on
    mesh                                = on
    external_code                       = on
}
MULTIPLIER_FUNCTION( "Multiplier Function 1" ) {
    type                                = piecewise_linear
    curve_fit_values                    = { 0.0, 0.0;
                                            10.0, 1.0; }
    curve_fit_variable                  = time_step
    evaluation                          = once_per_time_step
    filter                              = none
}

# +-----------------------------------------------------------------------+
# | Material Model: Air                                                   |
```

```
# +--------------------------------------------------------------------+

MATERIAL_MODEL( "Air" ) {
    Type                                = "fluid"
    density_model                       = "Air"
    viscosity_model                     = "Air"
    porosity_model                      = "Air"
}

DENSITY_MODEL( "Air" ) {
    type                                = constant
    density                             = 1.225         # kg/m3
    isothermal_compressibility          = 0.0           # m2/N
}

VISCOSITY_MODEL( "Air" ) {
    type                                = constant
    viscosity                           = 1.781e-005
}
# +--------------------------------------------------------------------+
# | Flexible Body                                                      |
# +--------------------------------------------------------------------+

FLEXIBLE_BODY( "elastic flap" ) {
    equation                            = mesh_displacement
    num_modes                           = 5
    type                                = trapezoidal
    num_sub_steps                       = 1
    mass                                = { 1.0, 1.0, 1.0, 1.0, 1.0; }
    stiffness                           = { 859.801335005, 9821.73796462,
31996.624274, 61892.842884, 104442.491882; }
    damping                             = { 0.0, 0.0, 0.0, 0.0, 0.0; }
    contact_constraints                 = { 0, 0, 0, 0, 0, 0, 0; }
    external_force                      = { 0, 0, 0, 0, 0; }
    initial_displacement                = { 0, 0, 0, 0, 0; }
    initial_velocity                    = { 0, 0, 0, 0, 0; }
    initial_force                       = { 0, 0, 0, 0, 0; }
    surface_outputs                     = { "flap" }
    evaluation                          = once_per_solution_update
    filter                              = none
}

# +--------------------------------------------------------------------+
# | Nodal Output                                                       |
# +--------------------------------------------------------------------+

NODAL_OUTPUT {
    output_frequency                    = 2
    output_time_interval                = 0.0           # sec
    output_initial_condition            = on
    continuous_output                   = off
    num_saved_states                    = 0
        }
        #
+--------------------------------------------------------------------+
# | Nodal Initial Condition                                          |
# +--------------------------------------------------------------------+

NODAL_INITIAL_CONDITION( "pressure" ) {
    default_value                       = 0.0           # N/m2
}

NODAL_INITIAL_CONDITION( "velocity" ) {
    default_values                      = { 8.0, 0.0, 0.0; }
}

NODAL_INITIAL_CONDITION( "mesh_displacement" ) {
    default_values                      = { 0.0, 0.0, 0.0; }
```

```
}
ELEMENT_SET( "fluid" ) {
    elements                              = Read(
"MESH.DIR/DC_FSI_initial.fluid.tet4.cnn" )
    shape                                 = four_node_tet
    medium                                = fluid
    quadrature                            = full
    material_model                        = "Air"
    body_force                            = "none"
    mesh_motion                           = "none"
    reference_frame                       = "none"
    residual_control                      = on
    oscillation_control                   = on
    mesh_distortion_correction_factor     = 0.0
    mesh_distortion_tolerance             = 0.0
}

# +--------------------------------------------------------------------+
# | Simple Boundary Condition                                          |
# +--------------------------------------------------------------------+

SIMPLE_BOUNDARY_CONDITION( "bottom" ) {
    surfaces                              = Read(
"MESH.DIR/DC_FSI_initial.fluid.tet4.bottom.tri3.ebc" )
    shape                                 = "three_node_triangle"
    element_set                           = "fluid"
    type                                  = wall
    active_type                           = all
    precedence                            = 1
    reference_frame                       = "none"
    wall_velocity_type                    = match_mesh_velocity
    temperature_type                      = flux
    heat_flux                             = 0.0
    convective_heat_coefficient           = 0.0
    convective_heat_reference_temperature = 0.0
    turbulence_wall_type                  = wall_function
    roughness_height                      = 0.0
    mesh_displacement_type                = slip
}

# +--------------------------------------------------------------------+
# | Surface Output                                                     |
# +--------------------------------------------------------------------+

SURFACE_OUTPUT( "bottom" ) {
    surfaces                              = Read(
"MESH.DIR/DC_FSI_initial.fluid.tet4.bottom.tri3.ebc" )
    shape                                 = "three_node_triangle"
    element_set                           = "fluid"
    integrated_output_frequency           = 1
    integrated_output_time_interval       = 0.0
    nodal_output_frequency                = 0
    nodal_output_time_interval            = 0.0
    num_saved_states                      = 0
}

# +--------------------------------------------------------------------+
# | External Code Surface                                              |
# +--------------------------------------------------------------------+

EXTERNAL_CODE_SURFACE( "flap" ) {
    surfaces                              = Read(
"MESH.DIR/DC_FSI_initial.fluid.tet4.flap.tri3.ebc" )
    shape                                 = "three_node_triangle"
    element_set                           = "fluid"
    velocity_type                         = wall
    temperature_type                      = tied
    mesh_displacement_type                = tied
```

```
    turbulence_wall_type                = wall_function
    roughness_height                    = 0.0           # m
    gap_factor                          = 0.0
    gap                                 = 0.0           # m
}

# +--------------------------------------------------------------------+
# | Surface Output                                                     |
# +--------------------------------------------------------------------+

SURFACE_OUTPUT( "flap" ) {
    surfaces                            = Read(
"MESH.DIR/DC_FSI_initial.fluid.tet4.flap.tri3.ebc" )
    shape                               = "three_node_triangle"
    element_set                         = "fluid"
    integrated_output_frequency         = 1
    integrated_output_time_interval     = 0.0
    nodal_output_frequency              = 0
    nodal_output_time_interval          = 0.0
    num_saved_states                    = 0
}

# +--------------------------------------------------------------------+
# | Simple Boundary Condition                                          |
# +--------------------------------------------------------------------+

SIMPLE_BOUNDARY_CONDITION( "inflow" ) {
    surfaces                            = Read(
"MESH.DIR/DC_FSI_initial.fluid.tet4.inflow.tri3.ebc" )
    shape                               = "three_node_triangle"
    element_set                         = "fluid"
    type                                = inflow
    inflow_type                         = velocity
    active_type                         = all
    precedence                          = 1
    reference_frame                     = "none"
    inflow_velocity_type                = cartesian
    x_velocity                          = 8.0           # m/sec
    y_velocity                          = 0.0           # m/sec
    z_velocity                          = 0.0           # m/sec
    non_reflecting_factor               = 0.0
    mesh_displacement_type              = fixed
    mesh_motion                         = "none"
}

# +--------------------------------------------------------------------+
# | Surface Output                                                     |
# +--------------------------------------------------------------------+

SURFACE_OUTPUT( "inflow" ) {
    surfaces                            = Read(
"MESH.DIR/DC_FSI_initial.fluid.tet4.inflow.tri3.ebc" )
    shape                               = "three_node_triangle"
    element_set                         = "fluid"
    integrated_output_frequency         = 1
    integrated_output_time_interval     = 0.0
    nodal_output_frequency              = 0
    nodal_output_time_interval          = 0.0
    num_saved_states                    = 0
}

# +--------------------------------------------------------------------+
# | Simple Boundary Condition                                          |
# +--------------------------------------------------------------------+

SIMPLE_BOUNDARY_CONDITION( "left" ) {
    surfaces                            = Read(
"MESH.DIR/DC_FSI_initial.fluid.tet4.left.tri3.ebc" )
```

```
    shape                                = "three_node_triangle"
    element_set                          = "fluid"
    type                                 = wall
    active_type                          = all
    precedence                           = 1
    reference_frame                      = "none"
    wall_velocity_type                   = match_mesh_velocity
    temperature_type                     = flux
    heat_flux                            = 0.0
    convective_heat_coefficient          = 0.0
    convective_heat_reference_temperature = 0.0
    turbulence_wall_type                 = wall_function
    roughness_height                     = 0.0
    mesh_displacement_type               = slip
}

# +--------------------------------------------------------------------+
# | Surface Output                                                     |
# +--------------------------------------------------------------------+

SURFACE_OUTPUT( "left" ) {
    surfaces                             = Read(
"MESH.DIR/DC_FSI_initial.fluid.tet4.left.tri3.ebc" )
    shape                                = "three_node_triangle"
    element_set                          = "fluid"
    integrated_output_frequency          = 1
    integrated_output_time_interval      = 0.0
    nodal_output_frequency               = 0
    nodal_output_time_interval           = 0.0
    num_saved_states                     = 0
}

# +--------------------------------------------------------------------+
# | Simple Boundary Condition                                          |
# +--------------------------------------------------------------------+

SIMPLE_BOUNDARY_CONDITION( "outflow" ) {
    surfaces                             = Read(
"MESH.DIR/DC_FSI_initial.fluid.tet4.outflow.tri3.ebc" )
    shape                                = "three_node_triangle"
    element_set                          = "fluid"
    type                                 = outflow
    back_flow_conditions                 = off
    active_type                          = all
    precedence                           = 1
    pressure                             = 0.0           # N/m2
    pressure_loss_factor                 = 0.0
    hydrostatic_pressure                 = off
    non_reflecting_factor                = 0.0
    mesh_displacement_type               = fixed
    mesh_motion                          = "none"
}

# +--------------------------------------------------------------------+
# | Surface Output                                                     |
# +--------------------------------------------------------------------+

SURFACE_OUTPUT( "outflow" ) {
    surfaces                             = Read(
"MESH.DIR/DC_FSI_initial.fluid.tet4.outflow.tri3.ebc" )
    shape                                = "three_node_triangle"
    element_set                          = "fluid"
    integrated_output_frequency          = 1
    integrated_output_time_interval      = 0.0
    nodal_output_frequency               = 0
    nodal_output_time_interval           = 0.0
    num_saved_states                     = 0
}
```

```
# +--------------------------------------------------------------------+
# | Simple Boundary Condition                                          |
# +--------------------------------------------------------------------+

SIMPLE_BOUNDARY_CONDITION( "right" ) {
    surfaces                                = Read(
"MESH.DIR/DC_FSI_initial.fluid.tet4.right.tri3.ebc" )
    shape                                   = "three_node_triangle"
    element_set                             = "fluid"
    type                                    = wall
    active_type                             = all
    precedence                              = 1
    reference_frame                         = "none"
    wall_velocity_type                      = match_mesh_velocity
    temperature_type                        = flux
    heat_flux                               = 0.0
    convective_heat_coefficient             = 0.0
    convective_heat_reference_temperature = 0.0
    turbulence_wall_type                    = wall_function
    roughness_height                        = 0.0
    mesh_displacement_type                  = slip
}

# +--------------------------------------------------------------------+
# | Surface Output                                                     |
# +--------------------------------------------------------------------+

SURFACE_OUTPUT( "right" ) {
    surfaces                                = Read(
"MESH.DIR/DC_FSI_initial.fluid.tet4.right.tri3.ebc" )
    shape                                   = "three_node_triangle"
    element_set                             = "fluid"
    integrated_output_frequency             = 1
    integrated_output_time_interval         = 0.0
    nodal_output_frequency                  = 0
    nodal_output_time_interval              = 0.0
    num_saved_states                        = 0
}

# +--------------------------------------------------------------------+
# | Simple Boundary Condition                                          |
# +--------------------------------------------------------------------+

SIMPLE_BOUNDARY_CONDITION( "top" ) {
    surfaces                                = Read(
"MESH.DIR/DC_FSI_initial.fluid.tet4.top.tri3.ebc" )
    shape                                   = "three_node_triangle"
    element_set                             = "fluid"
    type                                    = wall
    active_type                             = all
    precedence                              = 1
    reference_frame                         = "none"
    wall_velocity_type                      = match_mesh_velocity
    temperature_type                        = flux
    heat_flux                               = 0.0
    convective_heat_coefficient             = 0.0
    convective_heat_reference_temperature = 0.0
    turbulence_wall_type                    = wall_function
    roughness_height                        = 0.0
    mesh_displacement_type                  = slip
}

# +--------------------------------------------------------------------+
# | Surface Output                                                     |
# +--------------------------------------------------------------------+

SURFACE_OUTPUT( "top" ) {
```

```
    surfaces                                = Read(
"MESH.DIR/DC_FSI_initial.fluid.tet4.top.tri3.ebc" )
    shape                                   = "three_node_triangle"
    element_set                             = "fluid"
    integrated_output_frequency             = 1
    integrated_output_time_interval         = 0.0
    nodal_output_frequency                  = 0
    nodal_output_time_interval              = 0.0
    num_saved_states                        = 0
}

# +-------------------------------------------------------------------+
# | Nodal Boundary Condition                                          |
# +-------------------------------------------------------------------+

NODAL_BOUNDARY_CONDITION( "layers mesh_x_displacement" ) {
    nodes                                   = Read(
"MESH.DIR/DC_FSI_initial.layers.nbc" )
    variable                                = "mesh_x_displacement"
    type                                    = external_code
    precedence                              = 1
    mesh_motion                             = "none"
    active_type                             = all
}

NODAL_BOUNDARY_CONDITION( "layers mesh_y_displacement" ) {
    nodes                                   = Read(
"MESH.DIR/DC_FSI_initial.layers.nbc" )
    variable                                = "mesh_y_displacement"
    type                                    = external_code
    precedence                              = 1
    mesh_motion                             = "none"
    active_type                             = all
}

NODAL_BOUNDARY_CONDITION( "layers mesh_z_displacement" ) {
    nodes                                   = Read(
"MESH.DIR/DC_FSI_initial.layers.nbc" )
    variable                                = "mesh_z_displacement"
    type                                    = external_code
    precedence                              = 1
    mesh_motion                             = "none"
    active_type                             = all
}
```

```
A3
***************The inputs for frequency analysis in ABAQUS*******************
**in a regular input file, the structural models nodes and their coordinates is
written before this section
** ASSEMBLY
**
*Assembly, name=Assembly
**
*Instance, name=scaled-1, part=scaled
*End Instance
**
*Nset, nset=_PickedSet10, internal, instance=scaled-1, generate
 1261,  1323,     1
*Elset, elset=_PickedSet10, internal, instance=scaled-1, generate
 761,  800,    1
*End Assembly
**
** MATERIALS
**
*Material, name=Material-1
*Density
1000.,
*Elastic
 1e+08, 0.49
**
** BOUNDARY CONDITIONS
**
** Name: BC-1 Type: Symmetry/Antisymmetry/Encastre
*Boundary
_PickedSet10, PINNED
** ----------------------------------------------------------------
**
** STEP: freq
**
*Step, name=freq, perturbation
*Frequency, eigensolver=Lanczos, acoustic coupling=on, normalization=mass
5, , , , ,
**
** OUTPUT REQUESTS
**
*Restart, write, frequency=0
**
** FIELD OUTPUT: F-Output-1
**
*Output, field, variable=PRESELECT
*End Step
```

```
A4
***************The inputs for the P-FSI coupling in ACUSOLVE*******************
**in a regular input file, the structural models nodes and their coordinates is
written before this section
# +--------------------------------------------------------------------+
# | Problem Description                                                |
# +--------------------------------------------------------------------+

ANALYSIS {
    title                               = "AcuSolve Problem"
    sub_title                           = "AcuSolve Problem"
    input_version                       = "1.8"
    type                                = transient
}

EQUATION {
    flow                                = navier_stokes
    absolute_pressure_offset            = 0.0           # N/m2
    temperature                         = none
    species_transport                   = none
    turbulence                          = none
    mesh                                = arbitrary_lagrangian_eulerian
    external_code                       = off
    running_average                     = off
}

# +--------------------------------------------------------------------+
# | Auto Solution Strategy                                             |
# +--------------------------------------------------------------------+

AUTO_SOLUTION_STRATEGY {
    max_time_steps                      = 10000
    final_time                          = 0.2           # sec
    initial_time_increment              = 0.00025       # sec
    auto_time_increment                 = off
    convergence_tolerance               = 0.001
    min_stagger_iterations              = 0
    max_stagger_iterations              = 10
    num_krylov_vectors                  = 10
    relaxation_factor                   = 0.0
    flow                                = on
    mesh                                = on
}

# +--------------------------------------------------------------------+
# | Multiplier Function                                                |
# +--------------------------------------------------------------------+

MULTIPLIER_FUNCTION( "Multiplier Function 1" ) {
    type                                = piecewise_linear
    curve_fit_values                    = { 0.0, 0.0;
                                            10.0, 1.0; }
    curve_fit_variable                  = time_step
    evaluation                          = once_per_time_step
    filter                              = none
}

# +--------------------------------------------------------------------+
# | Material Model: Air                                                |
# +--------------------------------------------------------------------+

MATERIAL_MODEL( "Air" ) {
    Type                                = "fluid"
    density_model                       = "Air"
    viscosity_model                     = "Air"
    porosity_model                      = "Air"
}
```

```
DENSITY_MODEL( "Air" ) {
    type                                = constant
    density                             = 1.225         # kg/m3
    isothermal_compressibility          = 0.0           # m2/N
}

VISCOSITY_MODEL( "Air" ) {
    type                                = constant
    viscosity                           = 1.781e-005    # kg/m-sec
}

POROSITY_MODEL( "Air" ) {
    type                                = none
}

# +--------------------------------------------------------------------+
# | Flexible Body                                                      |
# +--------------------------------------------------------------------+

FLEXIBLE_BODY( "elastic flap" ) {
    equation                            = mesh_displacement
    num_modes                           = 5
    type                                = trapezoidal
    num_sub_steps                       = 1
    mass                                = { 1.0, 1.0, 1.0, 1.0, 1.0; }
    stiffness                           = { 1383031.96392, 7112344.96178,
46432155.9508, 83003373.0132, 95153102.771; }
    damping                             = { 0.0, 0.0, 0.0, 0.0, 0.0; }
    contact_constraints                 = { 0, 0, 0, 0, 0, 0, 0; }
    external_force                      = { 0, 0, 0, 0, 0; }
    initial_displacement                = { 0, 0, 0, 0, 0; }
    initial_velocity                    = { 0, 0, 0, 0, 0; }
    initial_force                       = { 0, 0, 0, 0, 0; }
    surface_outputs                     = { "flap" }
    evaluation                          = once_per_solution_update
    filter                              = none
}

# +--------------------------------------------------------------------+
# | Nodal Output                                                       |
# +--------------------------------------------------------------------+

NODAL_OUTPUT {
    output_frequency                    = 2
    output_time_interval                = 0.0           # sec
    output_initial_condition            = on
    continuous_output                   = off
    num_saved_states                    = 0
}


# +--------------------------------------------------------------------+
# | Nodal Initial Condition                                            |
# +--------------------------------------------------------------------+

NODAL_INITIAL_CONDITION( "pressure" ) {
    default_value                       = 0.0           # N/m2
}

NODAL_INITIAL_CONDITION( "velocity" ) {
    default_values                      = { 8.0, 0.0, 0.0; }
}

NODAL_INITIAL_CONDITION( "mesh_displacement" ) {
    default_values                      = { 0.0, 0.0, 0.0; }
}

# +--------------------------------------------------------------------+
```

```
# | Element Sets                                                          |
# +-----------------------------------------------------------------------+

ELEMENT_SET( "fluid" ) {
    elements                              = Read( "MESH.DIR/own_P.fluid.tet4.cnn"
)
    shape                                 = four_node_tet
    medium                                = fluid
    quadrature                            = full
    material_model                        = "Air"
    body_force                            = "none"
    mesh_motion                           = "none"
    reference_frame                       = "none"
    residual_control                      = on
    oscillation_control                   = on
    mesh_distortion_correction_factor     = 0.0
    mesh_distortion_tolerance             = 0.0
}

# +-----------------------------------------------------------------------+
# | Simple Boundary Condition                                             |
# +-----------------------------------------------------------------------+

SIMPLE_BOUNDARY_CONDITION( "bottom" ) {
    surfaces                              = Read(
"MESH.DIR/own_P.fluid.tet4.bottom.tri3.ebc" )
    shape                                 = "three_node_triangle"
    element_set                           = "fluid"
    type                                  = wall
    active_type                           = all
    precedence                            = 1
    reference_frame                       = "none"
    wall_velocity_type                    = match_mesh_velocity
    temperature_type                      = flux
    heat_flux                             = 0.0
    convective_heat_coefficient           = 0.0
    convective_heat_reference_temperature = 0.0
    turbulence_wall_type                  = wall_function
    roughness_height                      = 0.0
    mesh_displacement_type                = slip
}

# +-----------------------------------------------------------------------+
# | Surface Output                                                        |
# +-----------------------------------------------------------------------+

SURFACE_OUTPUT( "bottom" ) {
    surfaces                              = Read(
"MESH.DIR/own_P.fluid.tet4.bottom.tri3.ebc" )
    shape                                 = "three_node_triangle"
    element_set                           = "fluid"
    integrated_output_frequency           = 1
    integrated_output_time_interval       = 0.0
    nodal_output_frequency                = 0
    nodal_output_time_interval            = 0.0
    num_saved_states                      = 0
}

# +-----------------------------------------------------------------------+
# | Simple Boundary Condition                                             |
# +-----------------------------------------------------------------------+

SIMPLE_BOUNDARY_CONDITION( "flap" ) {
    surfaces                              = Read(
"MESH.DIR/own_P.fluid.tet4.flap.tri3.ebc" )
    shape                                 = "three_node_triangle"
    element_set                           = "fluid"
    type                                  = wall
```

```
    active_type                         = all
    precedence                          = 1
    reference_frame                     = "none"
    wall_velocity_type                  = match_mesh_velocity
    temperature_type                    = flux
    heat_flux                           = 0.0
    convective_heat_coefficient         = 0.0
    convective_heat_reference_temperature = 0.0
    turbulence_wall_type                = wall_function
    roughness_height                    = 0.0
    mesh_displacement_type              = flexible_body
    flexible_body                       = "elastic flap"
    nodal_x_modes                       = Read(
"MESH.DIR/own_P.fluid.tet4.flap.tri3.xmd" )
    nodal_y_modes                       = Read(
"MESH.DIR/own_P.fluid.tet4.flap.tri3.ymd" )
    nodal_z_modes                       = Read(
"MESH.DIR/own_P.fluid.tet4.flap.tri3.zmd" )
}

# +-----------------------------------------------------------------------+
# | Surface Output                                                        |
# +-----------------------------------------------------------------------+

SURFACE_OUTPUT( "flap" ) {
    surfaces                            = Read(
"MESH.DIR/own_P.fluid.tet4.flap.tri3.ebc" )
    shape                               = "three_node_triangle"
    element_set                         = "fluid"
    integrated_output_frequency         = 1
    integrated_output_time_interval     = 0.0
    nodal_output_frequency              = 0
    nodal_output_time_interval          = 0.0
    num_saved_states                    = 0
}

# +-----------------------------------------------------------------------+
# | Simple Boundary Condition                                             |
# +-----------------------------------------------------------------------+

SIMPLE_BOUNDARY_CONDITION( "inflow" ) {
    surfaces                            = Read(
"MESH.DIR/own_P.fluid.tet4.inflow.tri3.ebc" )
    shape                               = "three_node_triangle"
    element_set                         = "fluid"
    type                                = inflow
    inflow_type                         = velocity
    active_type                         = all
    precedence                          = 1
    reference_frame                     = "none"
    inflow_velocity_type                = cartesian
    x_velocity                          = 8.0           # m/sec
    y_velocity                          = 0.0           # m/sec
    z_velocity                          = 0.0           # m/sec
    non_reflecting_factor               = 0.0
    mesh_displacement_type              = fixed
    mesh_motion                         = "none"
}

# +-----------------------------------------------------------------------+
# | Surface Output                                                        |
# +-----------------------------------------------------------------------+

SURFACE_OUTPUT( "inflow" ) {
    surfaces                            = Read(
"MESH.DIR/own_P.fluid.tet4.inflow.tri3.ebc" )
    shape                               = "three_node_triangle"
    element_set                         = "fluid"
```

```
    integrated_output_frequency         = 1
    integrated_output_time_interval     = 0.0
    nodal_output_frequency              = 0
    nodal_output_time_interval          = 0.0
    num_saved_states                    = 0
}

# +--------------------------------------------------------------------+
# | Simple Boundary Condition                                          |
# +--------------------------------------------------------------------+

SIMPLE_BOUNDARY_CONDITION( "left" ) {
    surfaces                            = Read(
"MESH.DIR/own_P.fluid.tet4.left.tri3.ebc" )
    shape                               = "three_node_triangle"
    element_set                         = "fluid"
    type                                = wall
    active_type                         = all
    precedence                          = 1
    reference_frame                     = "none"
    wall_velocity_type                  = match_mesh_velocity
    temperature_type                    = flux
    heat_flux                           = 0.0
    convective_heat_coefficient         = 0.0
    convective_heat_reference_temperature = 0.0
    turbulence_wall_type                = wall_function
    roughness_height                    = 0.0
    mesh_displacement_type              = slip
}

# +--------------------------------------------------------------------+
# | Surface Output                                                     |
# +--------------------------------------------------------------------+

SURFACE_OUTPUT( "left" ) {
    surfaces                            = Read(
"MESH.DIR/own_P.fluid.tet4.left.tri3.ebc" )
    shape                               = "three_node_triangle"
    element_set                         = "fluid"
    integrated_output_frequency         = 1
    integrated_output_time_interval     = 0.0
    nodal_output_frequency              = 0
    nodal_output_time_interval          = 0.0
    num_saved_states                    = 0
}

# +--------------------------------------------------------------------+
# | Simple Boundary Condition                                          |
# +--------------------------------------------------------------------+

SIMPLE_BOUNDARY_CONDITION( "outflow" ) {
    surfaces                            = Read(
"MESH.DIR/own_P.fluid.tet4.outflow.tri3.ebc" )
    shape                               = "three_node_triangle"
    element_set                         = "fluid"
    type                                = outflow
    back_flow_conditions                = off
    active_type                         = all
    precedence                          = 1
    pressure                            = 0.0           # N/m2
    pressure_loss_factor                = 0.0
    hydrostatic_pressure                = off
    non_reflecting_factor               = 0.0
    mesh_displacement_type              = fixed
    mesh_motion                         = "none"
}

# +--------------------------------------------------------------------+
```

```
# | Surface Output                                                       |
# +----------------------------------------------------------------------+

SURFACE_OUTPUT( "outflow" ) {
    surfaces                                = Read(
"MESH.DIR/own_P.fluid.tet4.outflow.tri3.ebc" )
    shape                                   = "three_node_triangle"
    element_set                             = "fluid"
    integrated_output_frequency             = 1
    integrated_output_time_interval         = 0.0
    nodal_output_frequency                  = 0
    nodal_output_time_interval              = 0.0
    num_saved_states                        = 0
}

# +----------------------------------------------------------------------+
# | Simple Boundary Condition                                            |
# +----------------------------------------------------------------------+

SIMPLE_BOUNDARY_CONDITION( "right" ) {
    surfaces                                = Read(
"MESH.DIR/own_P.fluid.tet4.right.tri3.ebc" )
    shape                                   = "three_node_triangle"
    element_set                             = "fluid"
    type                                    = wall
    active_type                             = all
    precedence                              = 1
    reference_frame                         = "none"
    wall_velocity_type                      = match_mesh_velocity
    temperature_type                        = flux
    heat_flux                               = 0.0
    convective_heat_coefficient             = 0.0
    convective_heat_reference_temperature = 0.0
    turbulence_wall_type                    = wall_function
    roughness_height                        = 0.0
    mesh_displacement_type                  = slip
}

# +----------------------------------------------------------------------+
# | Surface Output                                                       |
# +----------------------------------------------------------------------+

SURFACE_OUTPUT( "right" ) {
    surfaces                                = Read(
"MESH.DIR/own_P.fluid.tet4.right.tri3.ebc" )
    shape                                   = "three_node_triangle"
    element_set                             = "fluid"
    integrated_output_frequency             = 1
    integrated_output_time_interval         = 0.0
    nodal_output_frequency                  = 0
    nodal_output_time_interval              = 0.0
    num_saved_states                        = 0
}

# +----------------------------------------------------------------------+
# | Simple Boundary Condition                                            |
# +----------------------------------------------------------------------+

SIMPLE_BOUNDARY_CONDITION( "top" ) {
    surfaces                                = Read(
"MESH.DIR/own_P.fluid.tet4.top.tri3.ebc" )
    shape                                   = "three_node_triangle"
    element_set                             = "fluid"
    type                                    = wall
    active_type                             = all
    precedence                              = 1
    reference_frame                         = "none"
    wall_velocity_type                      = match_mesh_velocity
```

```
    temperature_type                    = flux
    heat_flux                           = 0.0
    convective_heat_coefficient         = 0.0
    convective_heat_reference_temperature = 0.0
    turbulence_wall_type                = wall_function
    roughness_height                    = 0.0
    mesh_displacement_type              = slip
}

# +--------------------------------------------------------------------+
# | Surface Output                                                     |
# +--------------------------------------------------------------------+

SURFACE_OUTPUT( "top" ) {
    surfaces                            = Read(
"MESH.DIR/own_P.fluid.tet4.top.tri3.ebc" )
    shape                               = "three_node_triangle"
    element_set                         = "fluid"
    integrated_output_frequency         = 1
    integrated_output_time_interval     = 0.0
    nodal_output_frequency              = 0
    nodal_output_time_interval          = 0.0
    num_saved_states                    = 0
}

# +--------------------------------------------------------------------+
# | Nodal Boundary Condition                                           |
# +--------------------------------------------------------------------+

NODAL_BOUNDARY_CONDITION( "layers mesh_x_displacement" ) {
    nodes                               = Read( "MESH.DIR/own_P.layers.nbc" )
    variable                            = "mesh_x_displacement"
    type                                = flexible_body
    precedence                          = 1
    mesh_motion                         = "none"
    flexible_body                       = "elastic flap"
    nodal_modes                         = Read(
"MESH.DIR/own_P.layers.meshX.eig" )
    active_type                         = all
}

NODAL_BOUNDARY_CONDITION( "layers mesh_y_displacement" ) {
    nodes                               = Read( "MESH.DIR/own_P.layers.nbc" )
    variable                            = "mesh_y_displacement"
    type                                = flexible_body
    precedence                          = 1
    mesh_motion                         = "none"
    flexible_body                       = "elastic flap"
    nodal_modes                         = Read(
"MESH.DIR/own_P.layers.meshY.eig" )
    active_type                         = all
}

NODAL_BOUNDARY_CONDITION( "layers mesh_z_displacement" ) {
    nodes                               = Read( "MESH.DIR/own_P.layers.nbc" )
    variable                            = "mesh_z_displacement"
    type                                = flexible_body
    precedence                          = 1
    mesh_motion                         = "none"
    flexible_body                       = "elastic flap"
    nodal_modes                         = Read(
"MESH.DIR/own_P.layers.meshZ.eig" )
    active_type                         = all
}
```