University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Mechanical and Structural Engineering and Materials Science – mechanical constructions	Spring semester, 2016 <u>Open</u> / Restricted access
Writer: Roger Usken Nevestveit	…… (Writer's signature)
Faculty supervisor: Bjørn H. Hjertager External supervisor(s): Stig Johansen	
Thesis title: CFD Modelling of Turbulence around Offshore Structure	
Credits (ECTS): 30	
Key words: CFD, OpenFOAM, methane, simpleFoam, turbMultiScalarTransportFoam, SST k-ω	Pages: 42 + enclosure: 28 Stavanger, 15/06-2015 Date/year

Front page for master thesis
Faculty of Science and Technology
Decision made by the Dean October 30th 2009

Abstract

Leakages of methane from the seabed is a common problem for oil companies, and they are held responsible for leakages surrounding the platforms at sea. These leakages has to be monitored in some way, and Stinger Technology has designed a construction for this purpose. The construction has different sensors mounted on it, two sonars detecting bubbles, one acoustic sensor monitoring flow direction and velocity, and concentration sensors, which measures the concentration of methane in the water.

The task here is to look at how the construction in itself affects the measurements done by the sensors. The first case is to look at the acoustic sensor, which monitors a user defined point in space above the construction. This point may very well be affected by the construction, causing wrong readings. The second task is to look at the concentration sensors located inside the construction. These readings may also be affected by the construction, and the amount of affection will probably vary depending on the flow direction, as the construction is not symmetric.

To know if the monitored regions are affected, equivalent monitoring points are setup in the simulations. The solver used in the simulations is the steady-state, incompressible solver simpleFoam, and the turbulence model is SST k-ω. For simulation of the methane in the water, a solver created by Knut Erik Giljarhus named turbMultiScalarTransportFoam has been used.

The results revealed that the point that the acoustic sensor now is monitoring, can be wrong with as much as up to 8% if the flow direction is directed so that the monitored point is downstream relative to the construction. This error can easily be reduced by moving the monitored point further away from the construction.

For the concentration measurements, most of the simulations did not show any affection from the construction at all, but if the flow comes in from the right angle, the concentration measurements showed a difference of up to 36,5% when comparing to the exact same simulation without the constructions presence.

The results seems to give logical answers, as the orientation of the flow was considered to give different results.

Table of contents

List of figures

List of tables

Acknowledgements

This thesis is the last stop on a long journey through the study program, a bachelor's degree in mechanical engineering, and a master's degree in Mechanical and Structural Engineering and Materials Science at the University of Stavanger, with a specialization in mechanical constructions.

The thesis has been hard work from the start, having to test and fail repeatedly, but in the end the knots came loose and a solution has taken form. It has been hard to stay motivated with this countercurrent, but luckily, I managed to complete.

I want to thank my supervisor Bjørn H. Hjertager for great mentoring. We have met almost weekly until the end of May, and the discussions and inputs has been of great help. I would also like to thank Knut Erik Giljarhus for being very helpful with any question I have had, and also for lending a self-made solver to me.

I would also like to thank my external supervisor, Stig Johansen, for always being available and helpful during these months. Thanks also to Bjarte Langeland for giving me the opportunity to write for Stinger Technology.

Introduction

Leakages of methane from the ground in the ocean is a problem that everyone working on the seabed encounters, but it is something that is kept in the shadows. Even so, it is a problem one must handle and control, and there are several solutions to how one can monitor leakages. Leakages can be natural, somewhere humans have not interfered, but around platforms built for the oil industry, there are bigger chances for leakages to be made. The companies responsible for the platforms are also responsible for leakages in this area, and thus they need to be able to monitor and control it.

Stinger has made a design for such a monitoring device, which is used and installed somewhere in the North Sea, by which company is anonymous. At this platform, there are two constructions installed at different locations, which is shown in Figure 1. The figure shows the constructions in red, with the blue part is pointing in the direction of where the acoustic sensor is monitoring. The blue arrows around shows the range of the incoming flow. The flow direction swaps between this range, and when the tide turns, one can mirror the flow directions to the other side. The construction is equipped with different sensors, and the task of this study is to find out if the construction in itself is bugging the measurements done by the sensors, and if so by how much.

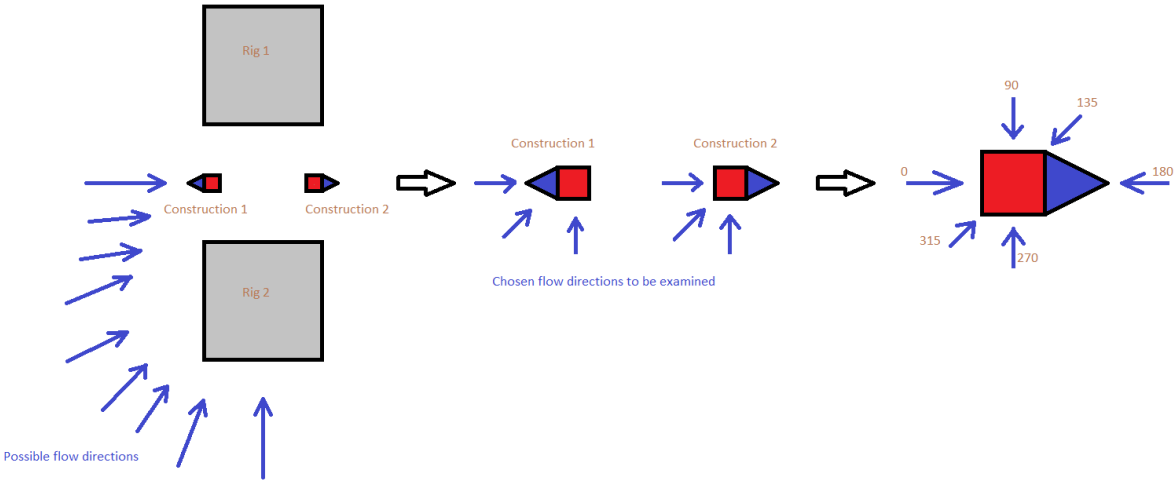

Figure 1

As the construction is not symmetric, it would implicate that measurements will be affected different depending on which direction the flow has. According to this, 6 different flow directions has been selected for examination, all of them given by blue arrows and numbers in the last step in Figure 1. The numbers are of course degrees, relative to the one that is given 0 degrees.

Previous analysis has been done by Stinger in 2-D, but a more thorough investigation of the problem in 3-D could be needed. The ocean currents is not predictable and something easily simulated, as many factors has an affection on the flow, such as the topology of the ground, installations and pipes around the constructions etc. As these things are not of any interest, they are neglected in this thesis, and the ground of the ocean is modelled as a flat plate.

Most parts in the course of a nychthemeron are constant flow, which of course varies depending on many variables, but it is fairly stable. There is however the time of the day when the tide turns, and this can cause some odd behavior of the flow, and maybe also odd readings from the sensor. The possibility to perform a transient study for this happening was discussed in the opening meeting, but as it turned out to be a relatively slow development in the project the first months, there was not enough time to do this, much because it would probably take a lot of time to do the transient simulations.

The simulations done were done with a RANS turbulence model, the simplest kind of turbulence model. It was discussed doing some simulations with an LES turbulence model as well, to compare the results. This was not done because of the same reasons as mentioned for the transient simulation, but my guess is that the simulations with LES would not differ a lot from the ones done with the RANS turbulence model.

Theory

In this short section, based on [1], basic fluid dynamics theory relevant for this thesis will be presented. Since there is no heat transfer relevant for the simulations, the energy equation is not incorporated, but the governing equations for mass conservation and the momentum is presented below.

Mass conservation:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{z} = 0$$

Can be written in a more compact form:

$$\frac{\partial \rho}{\partial t} + div(\rho \mathbf{u}) = 0$$

In this thesis, all simulations are done with a steady-state solver, which means that the transient term is not relevant, thus the equation is reduced to

$$div(\rho \mathbf{u}) = 0$$

As the fluid used in the simulations are not compressible, the density ρ is constant, and the resulting equation for mass conservation is

$$div(\mathbf{u}) = 0 \rightarrow \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$$

Momentum equations:

$$\rho \frac{Du}{Dt} = \frac{\partial(-p + \tau_{xx})}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + S_{Mx}$$

$$\rho \frac{Dv}{Dt} = \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial(-p + \tau_{yy})}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + S_{My}$$

$$\rho \frac{Dw}{Dt} = \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial(-p + \tau_{zz})}{\partial z} + S_{Mz}$$

Water, used in this simulation, is a Newtonian fluid. This means that the rate of deformation in the fluid is proportional with the applied shear stress. The momentum equations then reduces to the compressible Navier-Stokes equations:

$$\frac{\partial(\rho u)}{\partial t} + div(\rho u\mathbf{u}) = -\frac{\partial p}{\partial x} + div(\mu\ grad\ u) + S_{Mx}$$

$$\frac{\partial(\rho v)}{\partial t} + div(\rho v\mathbf{u}) = -\frac{\partial p}{\partial y} + div(\mu\ grad\ v) + S_{My}$$

$$\frac{\partial(\rho w)}{\partial t} + div(\rho w\mathbf{u}) = -\frac{\partial p}{\partial z} + div(\mu\ grad\ w) + S_{Mz}$$

By neglecting body forces, and include the fact that all simulations done in this thesis is both steady-state and incompressible, the momentum equations reduces to

$$div(u\mathbf{u}) = -\frac{\partial p}{\partial x} + div(\mu\ grad\ u)$$

$$div(v\mathbf{u}) = -\frac{\partial p}{\partial y} + div(\mu\ grad\ v)$$

$$div(w\mathbf{u}) = -\frac{\partial p}{\partial z} + div(\mu\ grad\ w)$$

Computational Fluid Dynamics

An introduction to CFD

This section is based on [1]. The codes used in CFD are structured around numerical algorithms. All codes are divided into three main elements: a pre-processor, a solver and a post-processor.

Pre-processor

This phase consist of the input of a flow problem to a CFD program, and transforming this input to a form that is possible to solve. The phase involves:

- Definition of geometry, the computational domain
- Grid generation, also called mesh
- Selection of the physical and chemical phenomena that is to be modelled
- Defining the fluid properties
- Selection of boundary conditions

Solver

This step uses the numerical algorithms defined to solve the case, and the process consists of the following steps:

- Integration of the governing equations in the whole control volume
- Discretization – the resulting integral equations are converted to a system of algebraic equations
- By iteration, the algebraic equations are solved

Post-processor

After defining and solving a flow field, a visualization of the results created by the program is very handy, and there are several programs developed for this specific task. They can display solutions in many forms, such as

- Domain geometry and grid display
- Vector plots
- Line and shaded contour plots
- 2D and 3D plots
- Particle tracking
- View manipulation

Even though these post-processor programs often are easy to use, and it looks fancy when you get a displayed solution, it can be easy to overlook faults done if one does not have enough knowledge of how one should expect the fluid to behave in the given situation. If one on the other hand has the knowledge and experience, this can be a great tool to see if the solution looks valid or not.

OpenFOAM

[2] OpenFOAM is a C++ library, used primarily to create executables, known as applications. These applications falls into two categories: solvers, that are made to simulate a specific case, and utilities, that are designed to perform tasks that involve data manipulation. OpenFOAM is very handy in the way that it is free for everyone to use, and it is also possible to create new solvers to work how you want them to work, if one has the skills and knowledge to do so. And that is also on of its drawbacks, that it is not that user friendly, and one has to have a certain amount of knowledge to be able to use the program.

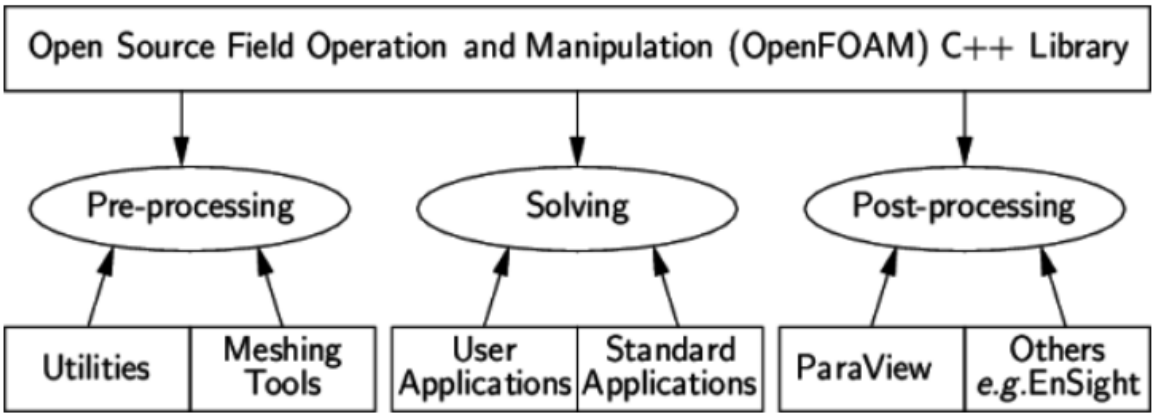

Figure 2

[3]To make a case, one needs a case folder that contains all files relevant for the case, and under this some sub dictionaries as shown in Figure 3. One needs a *system* folder, a *constant* folder, and a time dictionary, usually named *0*. Under *constant*, the stl-files are located, as well as the files created by the meshing process. In this dictionary the properties of the fluid is also specified, and also the turbulence model. Under the *0* folder, the flow field variables relevant for the given case is specified. This varies due to the solver chosen, and also depends on the turbulence model that is used. Under the *system* folder, all kinds of settings are kept, for the meshing process and the solving process. This includes at least *controlDict*, *fvSchemes* and *fvSolution*, but usually more files as well.

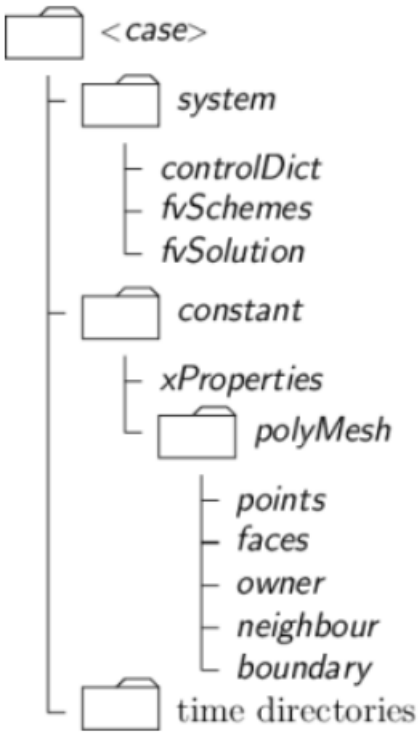

Figure 3

Numerical schemes

fvSchemes

[4]The fvSchemes file is located under the system folder. The file is user defined, and all the keywords in Table 1 has to be defined in terms of which scheme the user wants to use for the specific case. Not all schemes are good for all cases. An example is when the methane was simulated into the mix in this thesis, and the divScheme for the methane were set to "bounded Gauss linear". The solution went way overboard, as it gave a concentration in the flow of several billion percent, which is impossible. When switching to "bounded Gauss upwind", the solution process stabilized, and correct values for the concentration were given.

Keyword	**Category of mathematical terms**
interpolationSchemes	Point-to-point interpolations of values
snGradSchemes	Component of gradient normal to a cell face
gradSchemes	Gradient ∇
divSchemes	Divergence $\nabla\cdot$
laplacianSchemes	Laplacian ∇^2
timeScheme	First and second derivatives $\partial/\partial t, \partial^2/\partial^2 t$
fluxRequired	Fields which require the generation of flux

Table 1

fvSolution

[5]The fvSolution file is also located under the system folder. Within the file, several subdictionaries are defined. Some of these are *solvers, relaxationFactors* and *SIMPLE*. In the *SIMPLE* dictionary, residualcontrols are defined for all of the flow field parameters, so that if the residuals goes below the specified value, the solution is complete. Under solvers, each of the flow field parameters are given a solver to solve the equation for the given parameter. In this thesis, a linear solver control defined from Table 2 has been used for all flow fields.

Solver	Keyword
Preconditioned (bi-)conjugate gradient	PCG/PBiCG †
Solver using a smoother	smoothSolver
Generalised geometric-algebraic multi-grid	GAMG
Diagonal solver for explicit systems	diagonal

Table 2

SimpleFoam

This paragraph is based on [1] and [6].

This solver is a steady-state incomplessible solver. That means that the flow is assumed to not vary very much with time, so that is has a solution that is non-dependent with time. As water is the fluid in this analysis, without any high velocities or pressure, we can confidently assume that this is incompressible. The solver is based on the SIMPLE-algorithm, with is an abbreviation for Semi-Implicit Method for Pressure Linked Equations. The SIMPLE-algorithm is explained in [1] at page 186.

In this solver, the energy equation does not need to be solved, due to the incompressibility, as there is no link between the momentum and continuity equations. Instead, the momentum equations are solved and a pressure correction is performed, and a pressure difference is found. If the divergence is taken of the momentum equations and the continuity equation is used to expand the term $\frac{\partial}{\partial x_i}\left(\frac{\partial}{\partial t}\rho u_k\right)$, a Poisson equation for the pressure is found, and in case of constant density and viscosity the equation is

$$\frac{\partial}{\partial x_i}\left(\frac{\partial p}{\partial x_i}\right) = -\frac{\partial}{\partial x_i}\left[\frac{\partial}{\partial x_j}\left(\rho u_i u_j\right)\right]$$

The SIMPLE algorithm, that the simpleFoam solver is based upon, solves the momentum equations and the Poisson pressure equation.

TurbMultiScalarTransportFoam

TurbMultiScalarTransportFoam is a solver developed by Knut Erik Giljarhus, at the time working part-time at UiS as an associate professor, and teaching in the CFD course. The solver only solves for one or more scalars being added to the flow field. The flow field itself is not changed, so in order to make the solver do its job, the flow field has to be solved on beforehand,

and this flow field has to be moved into the 0-folder in a new case directory. The flow field is gathered from the cases made by simpleFoam, with a k-omega SST turbulence model, and then solves for the dissipation of the scalar afterwards.

The solver requires the scalars to be defined in transportProperties with the molecular mass diffusion coefficient relative to the water, thus we are able to separate between different gases and liquids that can be mixed. The molecular mass diffusion coefficient of methane in water is $1{,}5 * 10^{-9}\ {}^{m^2}/_{s}$ according to [7]. The kinematic viscosity also has to be stated in transportProperties, and as the simulations are of a construction placed on the seabed, a value is chosen for salt water at 4° Celsius. This value is $1{,}6094 * 10^{-5}\ {}^{m^2}/_{s}$, given by [8].

The source code for the solver has been asked to be kept secret, but the equation solved by the solver is given below.

$$\frac{\partial M}{\partial t} + \nabla \varphi M - \nabla \left(DM + turbulence \rightarrow nuEff( ) * {}^{1}/_{S_{c\_}} \right) \nabla \mathrm{M}$$

$nuEff = \nu_T + \nu$ [9]

$DM = molecular\ mass\ diffusion\ coefficient$

$M = mass\ fraction\ of\ the\ sewage\ water\ (scalar)$

$\varphi = \rho * U$ (flux)

$Sc = Schmidt\ number$

$\nu = kinematic\ viscosity$

$\nu_T = turbulent\ kinematic\ viscosity$

$\rho = density$

$U = velocity$

In addition to adding all the flow field variables from the last time step when solving with simpleFoam to the new 0-folder, a file for each of the scalars also has to be added to the folder, giving the initial conditions for the scalars concentration. The files has to be named starting with an M, and then naming the scalar. That leads to the file being named “MMethane”, since the scalar defined in transportProperties is named Methane. The bottom, e.g. the hole in the groundplate is then given a concentration of 1.0 (100%) as shown in Figure 4, so all the fluids coming out of the hole is methane. The other inlet has no methane, so this value is set to 0.0. A leakage of this size (1 meter diameter) is not something you will see out in the ocean under

normal circumstances, but whether it is realistic or not is really not that interesting, as our interest is to determine if the construction has any interference with the measured values detected.

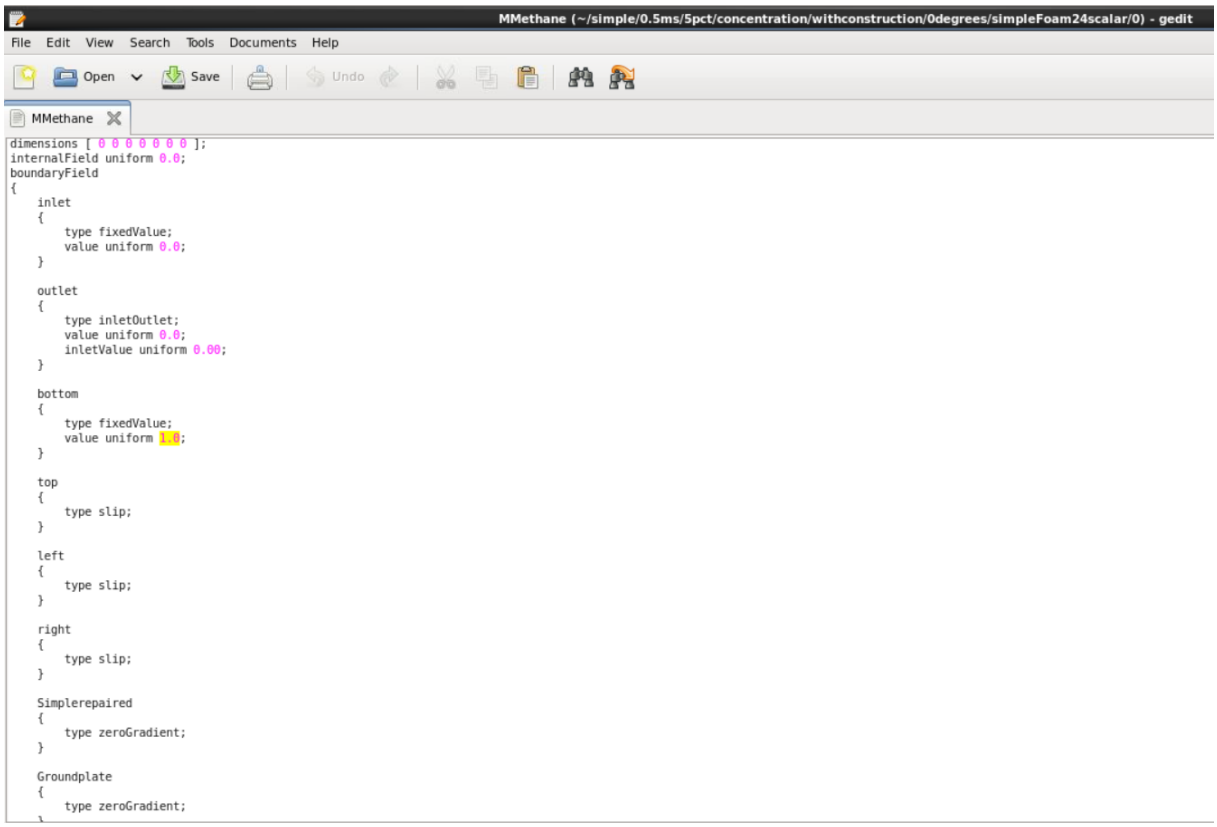

Figure 4

Turbulence modelling

This section is based on [10].

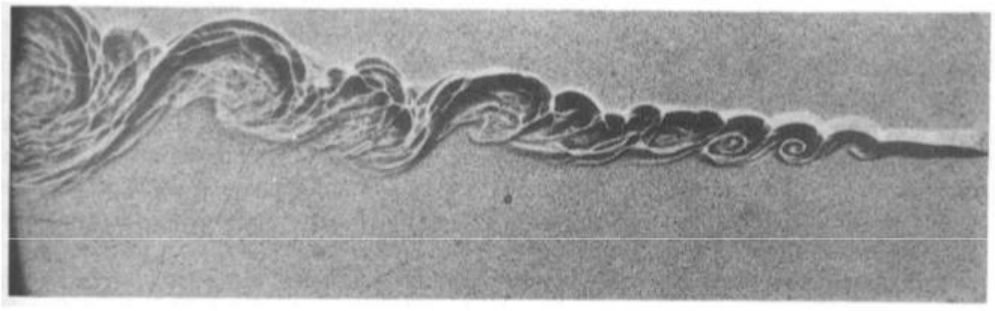

Figure 5

Turbulence modelling can be categorized into three different classes, which has different accuracy and computational power assumption. Within the three classes there are multiple different models, with different strengths and weaknesses. The three classes are

- RANS – Reynolds Averaged Navier-Stokes Equations
- LES – Large Eddy Simulation
- DNS – Direct Numerical Simulation

In Figure 6 one can see the different approaches for the different models. As seen, DNS will simulate the flow nearly exactly as it would behave in reality, given the right input values, and LES and RANS are approximations to the real values, where LES has a finer solution than RANS.

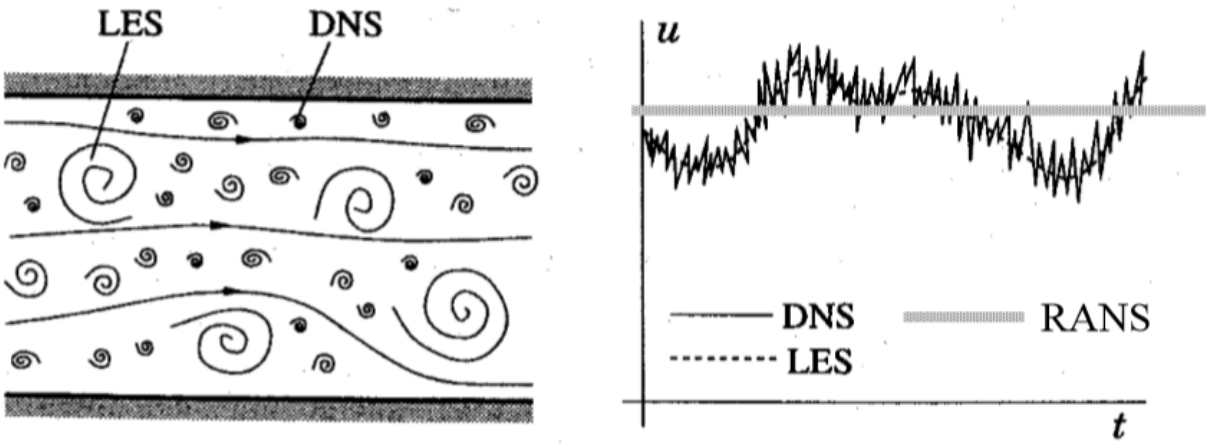

Figure 6

Most fluid flow processes is affected by turbulence in a certain degree, a 100% laminar flow is seldom encountered. A turbulent flow has several different properties, such as

- turbulent flows are highly unstable
- they are three-dimensional
- they contain a great deal of vorticity
- turbulence increases the rate at which conserved quantities are stirred
- by the processes mentioned above, turbulence brings fluids of differing momentum into contact
- turbulent flows fluctuate on a broad range of length and time scales

SST k-ω

[11] The SST k-ω turbulence model is a two-equation eddy-viscosity turbulence model, which goes under the RANS-classification. SST, the shear stress transport, makes the model more complete that only the k-ω model. The k-ω formulation simulates well on the inner parts of the boundary layer, so that the SST k-ω can be used as a Low-Reynolds turbulence model without having need for any extra damping functions. In the free-stream areas, the model turns to k-ε behavior, and thereby it is able to avoid the common problem with the k-ω being too sensitive to the inlet free-stream turbulence properties.

The equation for k, the turbulent kinetic energy is

$$\frac{\partial k}{\partial t} + U_j \frac{\partial k}{\partial x_j} = P_k - \beta^* k\omega + \frac{\partial}{\partial x_j}\left[(\nu + \sigma_k \nu_T)\frac{\partial k}{\partial x_j}\right]$$

and the equation for ω, the specific dissipation rate is

$$\frac{\partial \omega}{\partial t} + U_j \frac{\partial \omega}{\partial x_j} = \alpha S^2 - \beta \omega^2 + \frac{\partial}{\partial x_j}\left[(\nu + \sigma_\omega \nu_T)\frac{\partial \omega}{\partial x_j}\right] + 2(1 - F_1)\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i}$$

For more information on the formulas for the turbulence model, look at [11].

Modelling

The original model

To detect and control different parameters in leakages from the ground in the sea around offshore rigs, Stinger Technology AS has developed a solution that at this time is used some places in the north sea. The model has been drafted to become as similar as possible to the original model by an employee at Stinger, and this is the model shown in Figure 7.

Figure 7

Sensors

If one look closer at the construction in Figure 8, one can see that it is equipped with three sensors on the top, and some equipment inside. The top sensors are two sonars placed in directions normal to each other. These sensors pick up bubbles from the leakage, e.g. finds out where the leakages are located. The third sensor on the top is the acoustic sensor that monitors a user defined point in space, used to monitor flow speed. As known according to the no slip-condition, the fact that the construction is mounted at the bottom of the ocean results in relatively low flow velocities this close to the ground. The third type of monitoring equipment are the three pumps placed inside the module. These sensors measures the concentration of methane in the water, and if it is above what is legal or not.

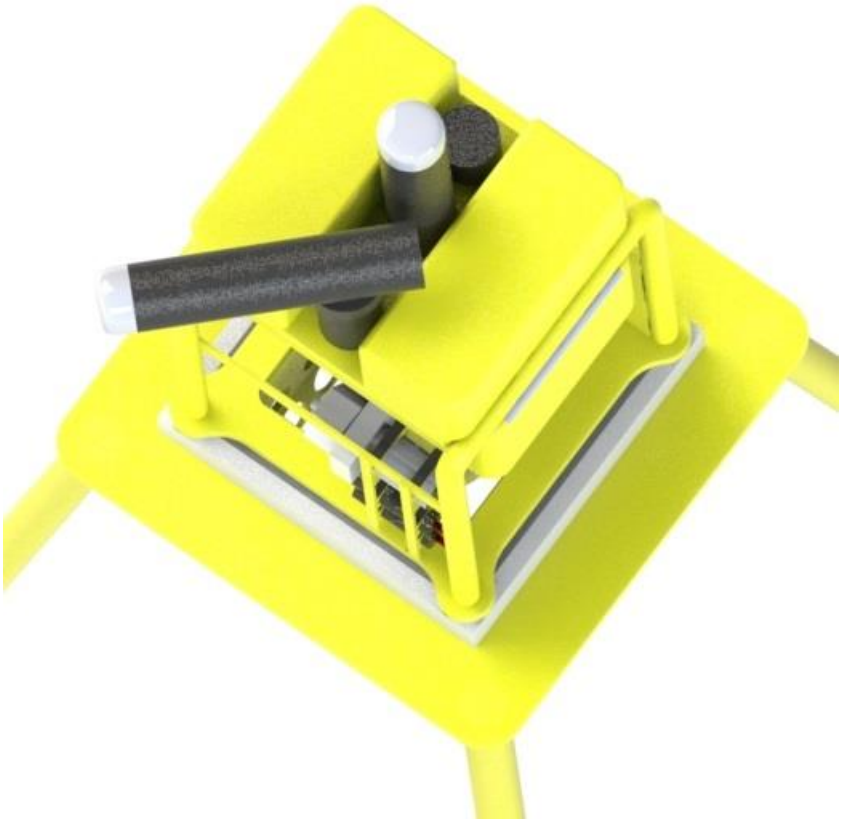

Figure 8

The revised model

The first step in CFD-simulation is a mesh discretization of the continuous domain. This though, turned out to be quite an intricate task. Using an automatic meshing tool, snappyHexMesh, this would sound like a simple step, but it turned out to be maybe the most time consuming part of the whole process. At first, meshing was performed on a simple laptop, but whenever it closed in on 1.5 million cells, the meshing process ended, and the mesh was nowhere near being good enough, due to several highly skewed cells. In need of more computational power, simulations from this point on where done on a cluster system at UiS, where one can access several different machines with relatively high performance. This solved the problem of the computational power, but when the mesh did not even get approved after refining it up to 44 million cells, it was time to make some changes to the design, and the result of this process is displayed in Figure 9.

Figure 9

To detect problem areas on the model, the skewed cells detected by the *checkMesh* function was transformed into VTK-files by the command *foamToVTK –faceSet skewFaces*. This made it possible to open both the mesh and the VTK-file in Paraview at the same time, and making a contrast between them, the skewed cells were possible to locate as shown in Figure 10. The hope was that the skewed cells occurred only in small concentrated areas and parts of the construction, but it turned out that the skewed cells were all over the place.

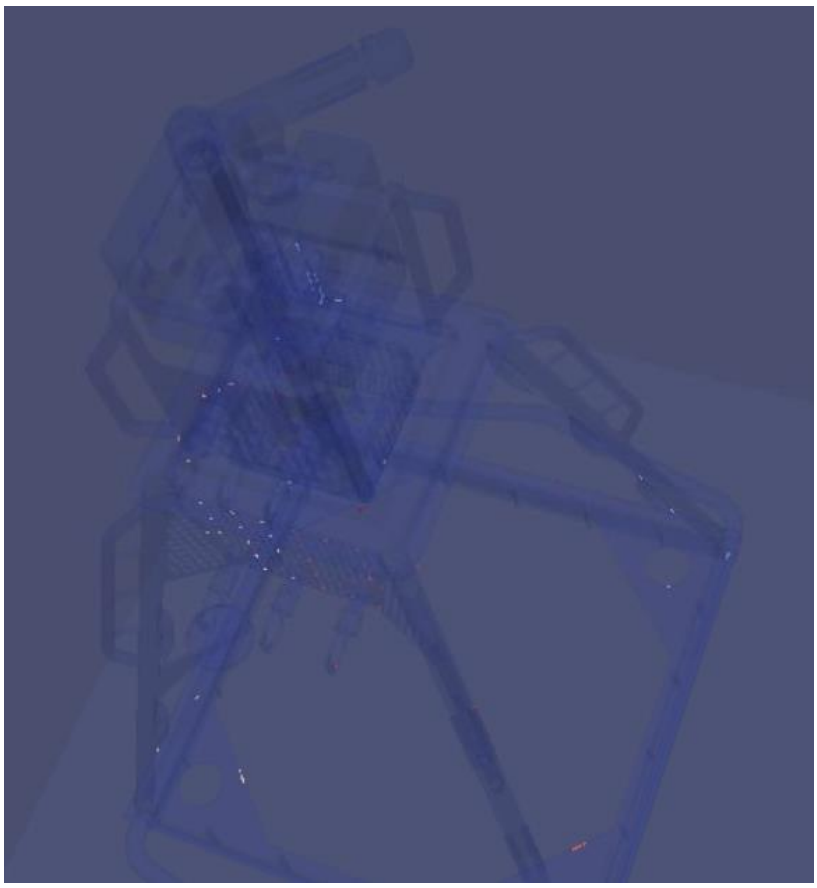

Figure 10

The model was drafted in SolidWorks, which is not used at school, so some consulting from Stinger was needed to be able to use the program to edit the model. Redesigning the model to make it work took closer to two weeks, as it often was hard to detect exactly where and why skewed cells appeared where they did. For each mesh test, the model was edited and the stl-file had to be fixed by a program named NetFabb Basic to avoid any holes in it, and then uploaded to the cluster. Meshing the model took 30-50 minutes, depending on how much of the CPU that was available. After meshing the model, one had to see if the changes worked in terms of fewer skewed cells, and then repeat the process. In the end though, it was actually a change in the blockmesh that was the turning point.

Meshing

BlockMesh

The blockmesh created by OpenFoam is defined by the the file blockMeshDict, which is located under the system folder in these cases. More complex geometries can be created using blockmesh, but in the simulations done in this thesis only rectangular/hexahedral blocks has been

made. Eight vertices in the Cartesian coordinate system is defined to be the corners of the rectangular domain, and the surfaces are defined by which vertexes they are made of. After choosing the length of the domain in each direction, one chooses how many parts it should be divided into, thus determining how many cells the blockmesh will consist of. If a change was made to the blockmesh, it was found that it was not certain that the mesh would be approved. This results in a small restriction when thinking about how large you want the computational domain to be, as the mesh, which has cells with a size of 15 cm in each direction, has to be expanded or reduced with a factor of 0.15 meters, e.g. 0.15, 0.30, 0.45, 0.60, 0.75… etc.

SnappyHexMesh

The snappyHexMesh [12] is an automatic meshing tool, and it is defined in the file snappyHexMeshDict, also located in the system folder. It uses the already existing mesh created by blockMeshDict, thus it does not mesh anything outside the domain defined by the blockmesh. SnappyHexMesh first of all needs one or more stl-files, and if the stl-file is a closed geometry, one can specify whether one wants the mesh to be generated on the inside or the outside of the geometry. The stl-files are located under the folders constant/triSurface. One of the pumps on the original construction was hollow and had an inlet and an outlet, so this had to be closed in order to not waste computation time on flow moving inside the pump.

Each of the geometries are defined in snappyHexMeshDict, where one determines how many levels they are to be refined to, and also refinement of the area around the geometries and layers close to the geometries.

To determine how fine the cells creating the geometry are, one has to determine how many levels of refinement one wants to have. This is done under *castellatedMeshControls* in *snappyHexMeshDict*. For each STL-file, a value ranging from a minimum to a maximum number of levels of refining in the form (<min> <max>) has to be defined. The program determines for itself where it needs to refine in the high levels and the low levels.

To create a finer mesh in the area around the construction, *refinementRegions* has been used, where there are specified 3 different distances from the construction: 0.1 meters, 0.5 meters and 2.0 meters away from the construction. These are given the levels of refinement 3, 2 and 1, respectively. This leads to a better simulation of the flow around the construction, and it also lowered the residuals after this was added to the mesh. In Figure 11 one can see the refinementRegion surrounding the construction.

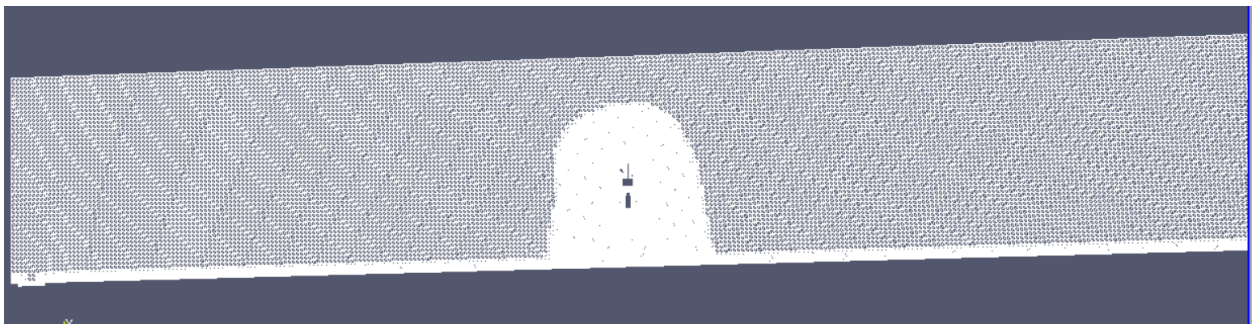

Figure 11

Layers close to the surface of the geometries are also added, to get a very fine simulation for the fluids impact on the surface. Two layers were added to both the groundplate and the construction.

Skewness

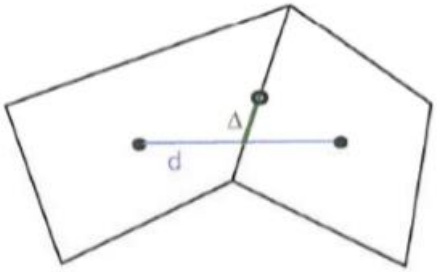

Figure 12

$$skewness = \frac{\Delta}{d}$$

[13] The main problem with the mesh was too high skewness, with values of around 10-15 on the original model. The skewness is calculated for faces, not for cells, and the formula is given above. The smaller this value is, the better, and in the mesh used in this thesis, the maximum skewness value is just below 4, which is the criteria to get an approved mesh. The simulation would probably have converged with a little higher values as well, but probably not with values as high as for the original model.

Case setup

Velocity

The first part of the simulations were to look at how valid the measurements of the acoustic sensor were, and see if the monitoring point should be changed. As explained earlier, the acoustic sensor monitors the flow speed and direction where the construction is located. The monitor point chosen at this time is a default setting, so this was not considered during the installation of the construction. It is located 75 cm from the sensor, in a 45° angle, as shown in Figure 13. The monitoring point can be moved closer or further away in the same direction as the original point.

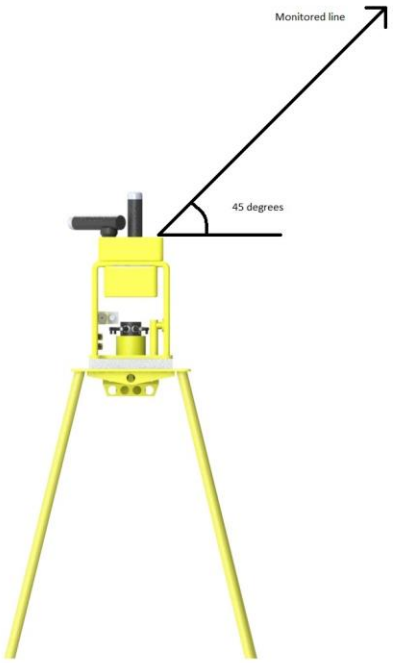

Figure 13

To monitor flow speeds, a set of points are chosen for printing of the flow velocity along a line having an angle of 45 degrees. The flow velocity at these points are compared to the inlet velocity, and by this one can see how large the differences are. The monitoring points are specified under system/controlDict, and in this case it monitors up to 5,65 meters from the sensor.

Only one flow direction were chosen to examine when looking at the flow velocity. It is assumed that this is the direction where the measurements are most affected, as the monitored points are downstream of the construction. Some other directions are probably also affected by the construction, but this is probably the most affected, thus being the relevant one to inspect.

As there is nothing happening upstream of the construction, there has not been given a large domain here when creating the blockMesh, but computational domain downstream is more than 20 meters, and even then, one can still see some unevenness in the flow.

Concentration

The second part of the simulations were to look at concentration measurements. There were some discussions with my supervisor, but eventually the best solution we could think of was to simulate the mixing of the methane with the construction present, and compare it to a simulation of the exact same case, but without the construction present.

To be able to have an inlet for the methane, a hole in the Groundplate was made, making it possible for fluids to pass through. The hole has a diameter of 1 meter. For the flow to become as normalized as possible at impact with the construction, the hole was placed approximately 20 meters upstream of the construction. This distance could ideally be even greater, but this would have caused even more cells, thus longer computational time, and this was not wanted as time was an issue.

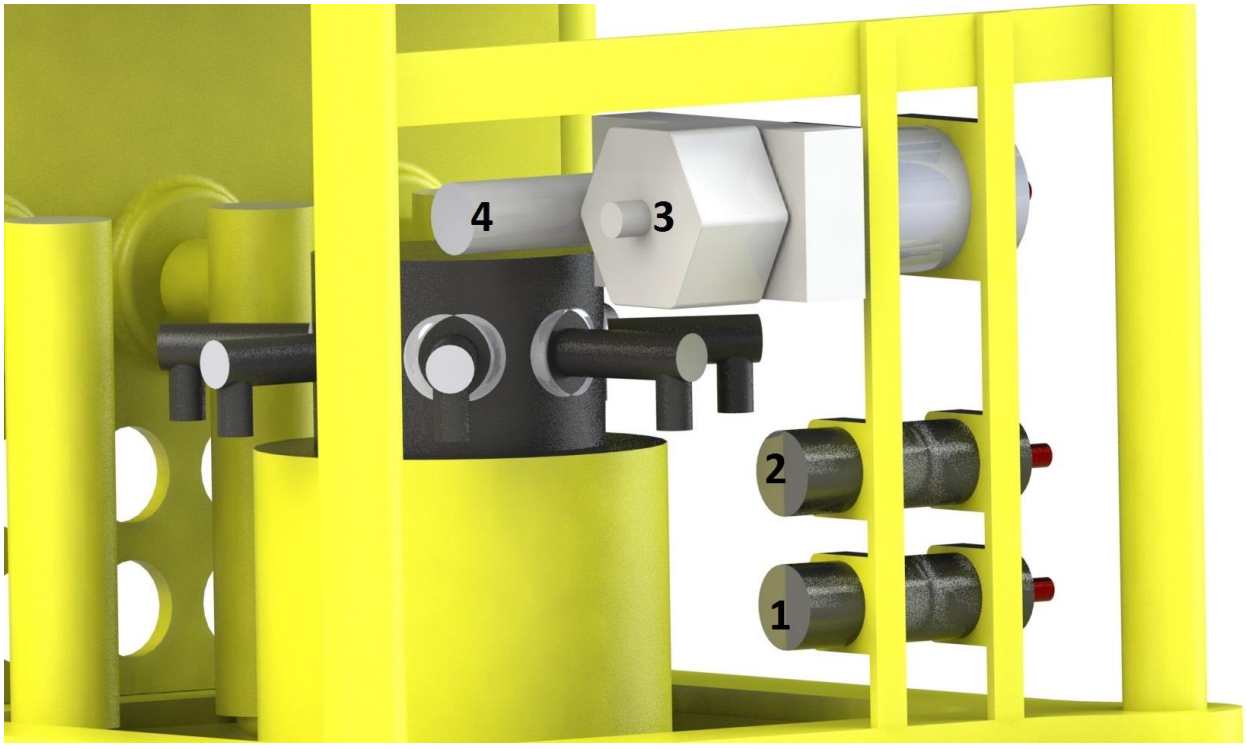

Figure 14

In this case, the monitoring points are located inside of the construction. There are four pumps in the model, but according to Stinger only two of the pumps are actually used. It is uncertain if it actually is the pumps in the model that are there physically, since different pumps has been

tested continuously, but either way there are not big differences in the design and size, as they all fit the same mounts. There are many points being monitored, and each one is modeled as a line, where each line is 6 cm long and has 24 points being monitored along this line. A monitor line is placed between pump number 1 and 2, later referred to as “methanebetween”, and the other four are placed around pump number 4 according to the numbers defined in Figure 14. Each line is parallel with the pump, placed 2 cm from the pump over, under, and on both sides. These lines are later referred to as “inside”, “outside”, “over” and “under”. This is considered close enough for the measurements validness.

Boundary conditions

The boundary condition used in this the simulations for the concentration are given in Table 3 and Table 4:

Patch	**Boundary conditions**		
	U	p	nut
Inlet	*fixedValue*	*zeroGradient*	*zeroGradient*
Outlet	*pressureInletOutletVelocity*	*totalPressure*	*zeroGradient*
Bottom	*fixedValue*	*zeroGradient*	*zeroGradient*
Top	*slip*	*slip*	*slip*
Left	*slip*	*slip*	*slip*
Right	*slip*	*slip*	*slip*
Simplerepaired	*fixedValue*	*zeroGradient*	*nutUSpaldingWallFunction*
Groundplate	*fixedValue*	*zeroGradient*	*nutUSpaldingWallFunction*

Table 3

Patch	**Boundary conditions**	
	k	omega
Inlet	*turbulentIntensityKineticEnergyInlet*	*turbulentMixingLengthFrequencyInlet*
Outlet	*inletOutlet*	*inletOutlet*
Bottom	*turbulentIntensityKineticEnergyInlet*	*turbulentMixingLengthFrequencyInlet*
Top	*slip*	*slip*
Left	*slip*	*slip*
Right	*slip*	*slip*
Simplerepaired	*kqRWallFunction*	*omegaWallFunction*
Groundplate	*kqRWallFunction*	*omegaWallFunction*

Table 4

Patch	**Boundary conditions**
	MMethane
Inlet	*fixedValue*
Outlet	*inletOutlet*
Bottom	*fixedValue*
Top	*slip*
Left	*slip*
Right	*slip*
Simplerepaired	*zeroGradient*
Groundplate	*zeroGradient*

Table 5

For the inlet values in k and omega, there are som calculations to be done. According to [14] the formula for the mixing length to be specified in omega is

$$l = 0{,}07 * L$$

Using Helyx-OS, the height of the construction was found to have a height of approximately 3,3 meters, thus giving the mixing length;

$$l = 0{,}07 * 3{,}3 = 0{,}23\ m$$

For the case where the inlet velocity is 0,5 m/s, and the turbulence intensity is chosen to be 5%, the turbulent kinetic energy is calculated to be:

$$k = \frac{3}{2} * (U_{avg} * I)^2$$

$$k = \frac{3}{2} * \left(0{,}5\frac{m}{s} * 0{,}05\ \%\right)^2 = 0{,}0009375\ {}^{m^2}/_{s^2}$$

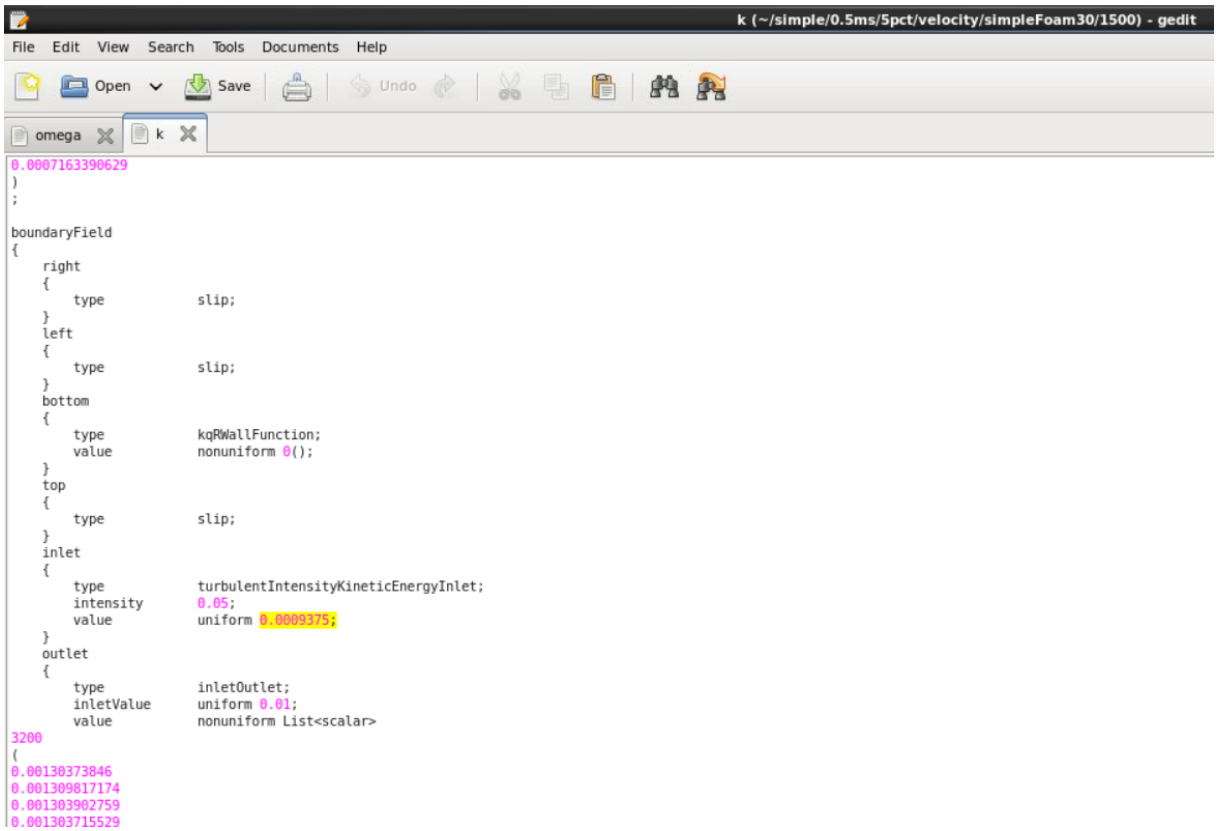

Figure 15

The value calculated for k above was not put into the files manually, but as the simulation starts, the program calculates this value itself based on the flow velocity and the turbulence intensity, and one can see the result of this in Figure 15, where one finds this value in the k-file created after 1500 iterations.

To calculate the value for omega, the value for k is used, and also the mixing length calculated earlier and a constant C_μ.

$$\omega = \frac{\sqrt{k}}{\sqrt[4]{C_\mu} * l}$$

$$\omega = \frac{\sqrt{0{,}0009375}}{\sqrt[4]{0{,}09} * 0.23\ m} = 0{,}2430508671\ s^{-1}$$

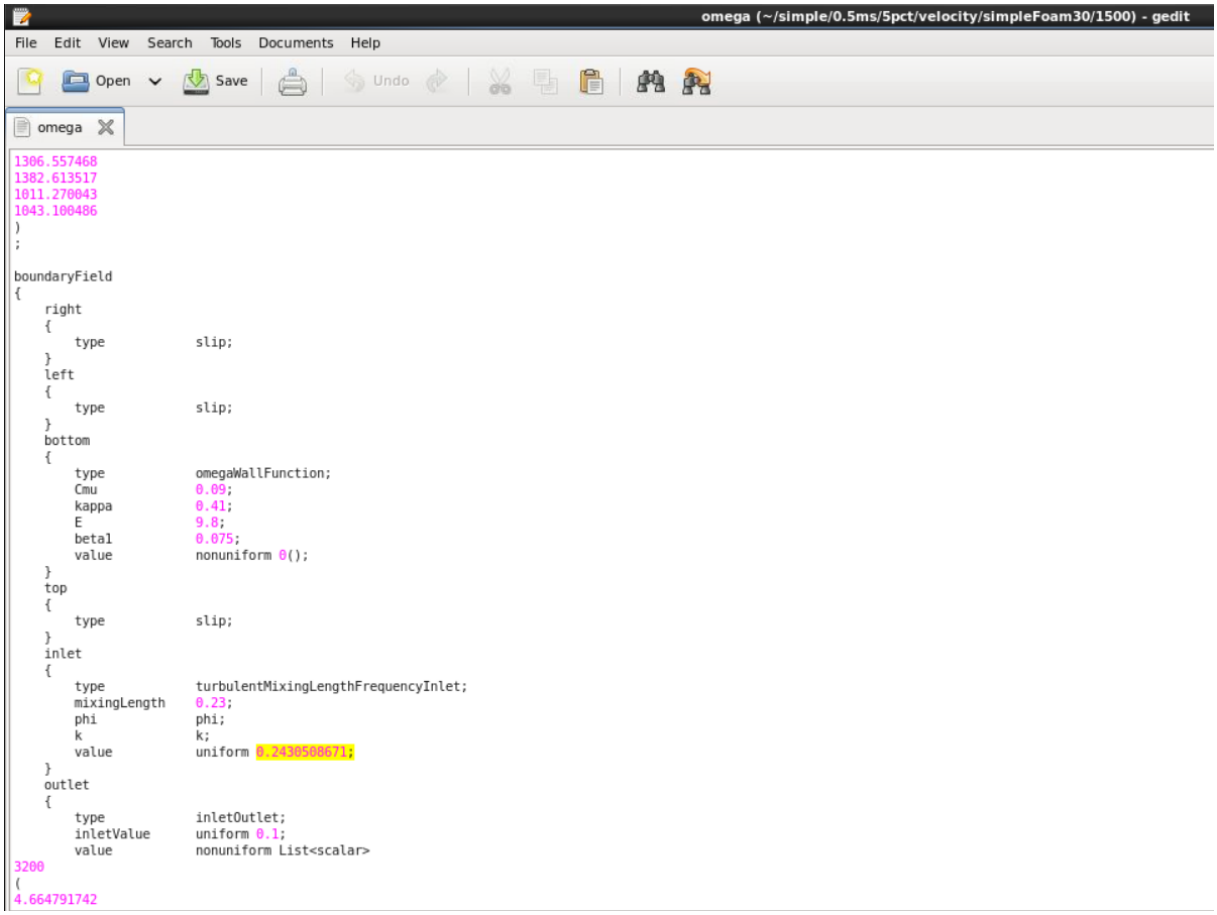

Figure 16

The same is true for the omega-file, and one can see the value calculated by the program in the omega file created after 1500 iterations in Figure 16.

$$k = turbulenc\ kinetic\ energy$$

$$I = turbulence\ intensity$$

$$U_{avg} = mean\ flow\ velocity$$

$$l = length\ scale, l = 0.07 * L$$

$$C_\mu = 0.09, empirical\ constant$$

$$\omega = specific\ rate\ of\ dissipation$$

More specifications on the boundary conditions can be found at [15]. In appendix number 2-7, one can also see the files.

setTest

To monitor the values of the different flow fields at certain locations in the domain, a function named setTest is used. The information for the setTest is plotted into the file *controlDict*, located under the *system* folder. By selecting two points in space, the line created between these two points are being monitored. One also specifies how many points on that line that will be monitored. From this input, an extra dictionary name *postProcessing* is created during simulation, and in this dictionary, the values for these monitored lines are gathered. To get a graphical view of how for instance the velocity U varies along the line called *inside*, the following command can be executed in gnuplot: *plot "inside_U.gplt" using 1:4 with lines* . This will show the magnitude of the velocity in the z-direction along this line, as a continuous line.

Simulation

To use OpenFoam, one needs to use a Linux operative system, and one also has to download the relevant OpenFoam version. The simulations mainly takes place in the terminal windows, but one can also use third-party programs like Helyx-OS, which are a bit more user friendly, but not that versatile. In this case, Helyx-OS was used to create the case files needed to simulate. After the files were created, they could easily be edited, since all of the files are plain text files.

Since the simulations were run on the cluster network, where one could access several different machine, the first thing to do when running a simulation was to log on to the chosen computer by using the *ssh*-command. After navigating to the case files, the commands executed in the terminal windows were as follows:

- *OF24x or OF30x*
- *blockMesh*
- *decomposePar*
- *mpirun –np 16 snappyHexMesh –overwrite –parallel*
- *reconstructParMesh –constant*

The next step were to delete all the processor folders created by *decomposePar*, and also edit the boundary-file located under constant/polyMesh, for the program to know if the boundaries are walls or patches.

- *decomposePar*
- *mpirun –n 16 renumberMesh –overwrite –parallel*
- *mpirun –np 16 simpleFoam –parallel >log*
- *reconstructPar*

After the last step, the simulations done with simpleFoam are completed, and the results are viewable in Paraview, by using the command *paraFoam*.

To precede with simulating the scalars added to the flow, only a couple of steps in the terminal has to be done. The solver used in this case, turbMultiScalarTransportFoam, is not a part of the library in OpenFoam. To be able to use it, one has to access the source code, execute the command *wmake*, which compiles the script. After doing this, one must navigate to the case one has created, and simply execute the command *turbMultiScalarTransportFoam*, and the solving will start. As this is a relatively simple solver which does not need very large amounts of computational power, it is not necessary to do it as a parallel simulation, like it is done when using simpleFoam.

Results and discussion

The results for the thesis is split into two parts, one where the velocity is measured, and one where the concentration of methane is measured.

Velocity

As explained in the modelling section, the acoustic sensor is monitoring the flow speed at a defined point in space above the sensor placed on top of the construction. This allows it to measure a point which is not hugely impacted by the construction, but the closer it is, the more affected the measured values will be. As a fabric setting, the monitoring point is located 75 cm in a 45 degree angle above the construction, and there has been set up a monitoring line in the simulations which reaches up to 5,65 meters away from the sensor, in a 45 degree angle. By using gnuplot, the text files created under *postProcessing/setTest/1500* can be visualized as a graph, showing the velocity of the flow versus the distance from the sensor. After navigating into the folder and executing the command *gnuplot*, one types in the following to command to get the graph which shows the velocity in the flow direction: *plot “linevelocity_U.gplt” using 1:4 with*

lines, and the resulting graph is in Figure 17.

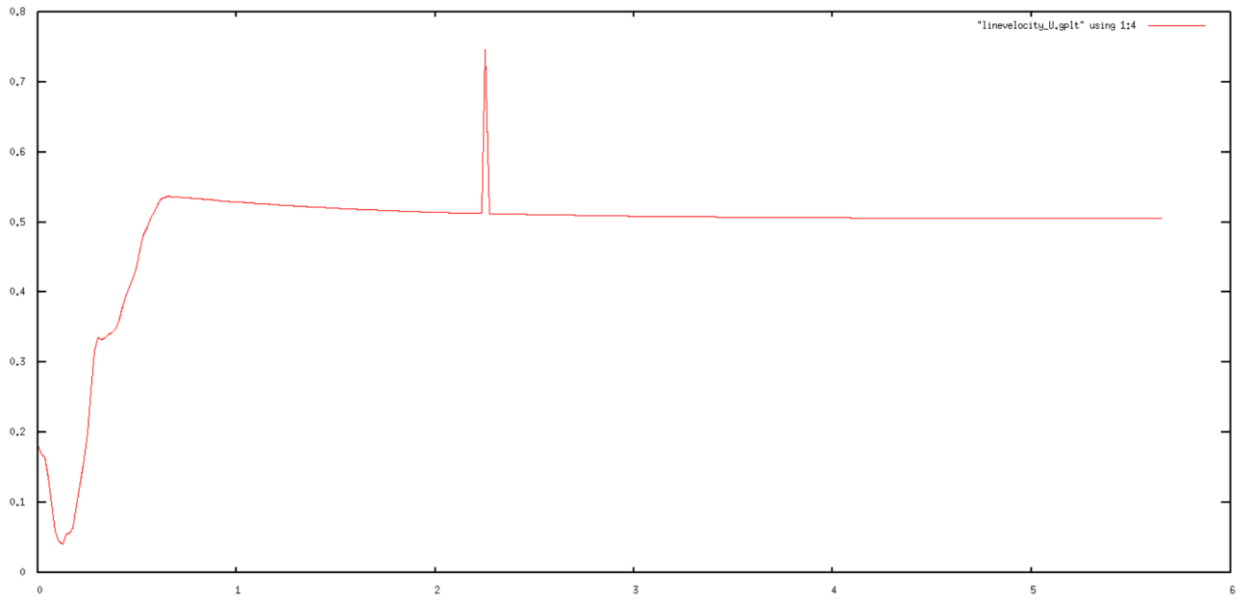

Figure 17

The graph in Figure 17 is collected from the case where the flow velocity is 0.5 m/s and the turbulence intensity is 5%. The x-axis represents the distance from the sensor, and the y-axis represents the velocity in the z-direction, e.g. the original flow direction. After reaching 0.65 meters from the sensor, the velocity starts decreasing evenly downwards to 0.5 m/s, but at a point 2.25 meters away from the sensor, the velocity does a big leap upwards. This was first considered to be a fault in the solution, but testing with several different velocities and turbulence intensities has shown that this is a repeated solution. What causes the fluctuation is not known at the present time being, but in this thesis it will be neglected.

The residuals for the simulation is also plotted, and visualized using gnuplot as shown in Figure 18. The pressure residual could ideally have been lower, but the residuals are stable and the solution looks good, so this is not very critical.

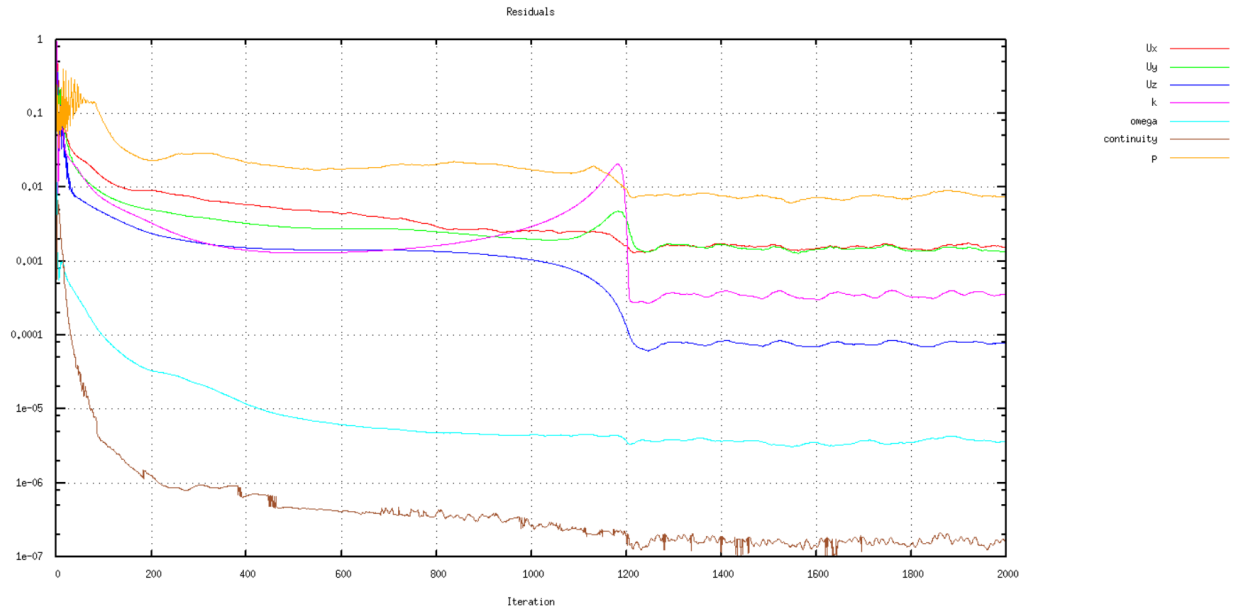

Figure 18

Using Paraview to visualize the results, a slice has been done through the center of the domain, which is also the center of the construction, and the slice is done parallel to the flow direction. Below there will be shown pictures of how the different flow variables k, omega, U, p and nut are in this slice, in Figure 19, Figure 20, Figure 21, Figure 22 and Figure 23, respectively.

k:

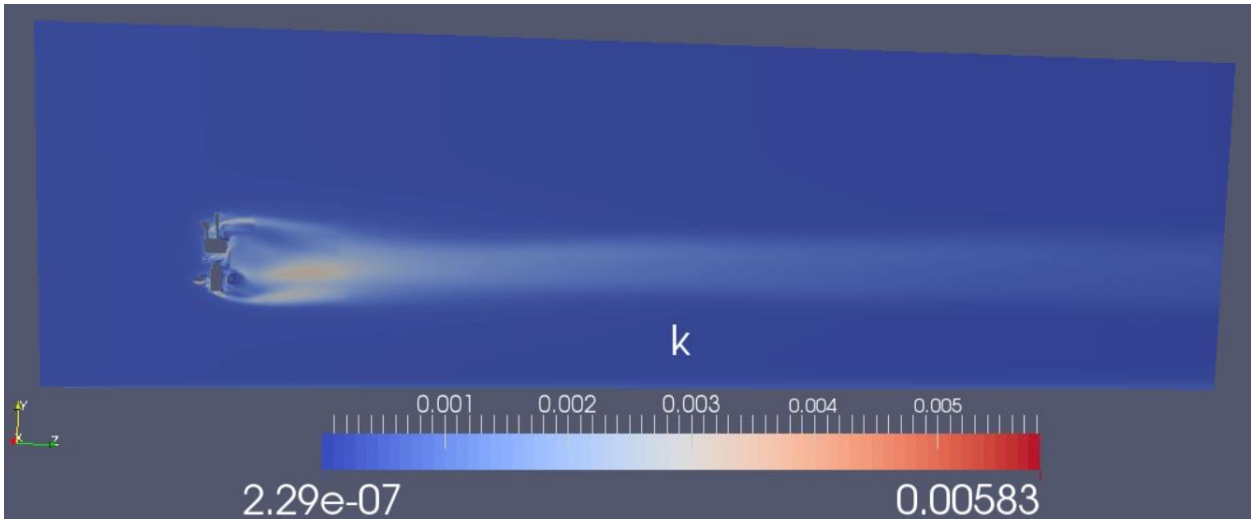

Figure 19

omega:

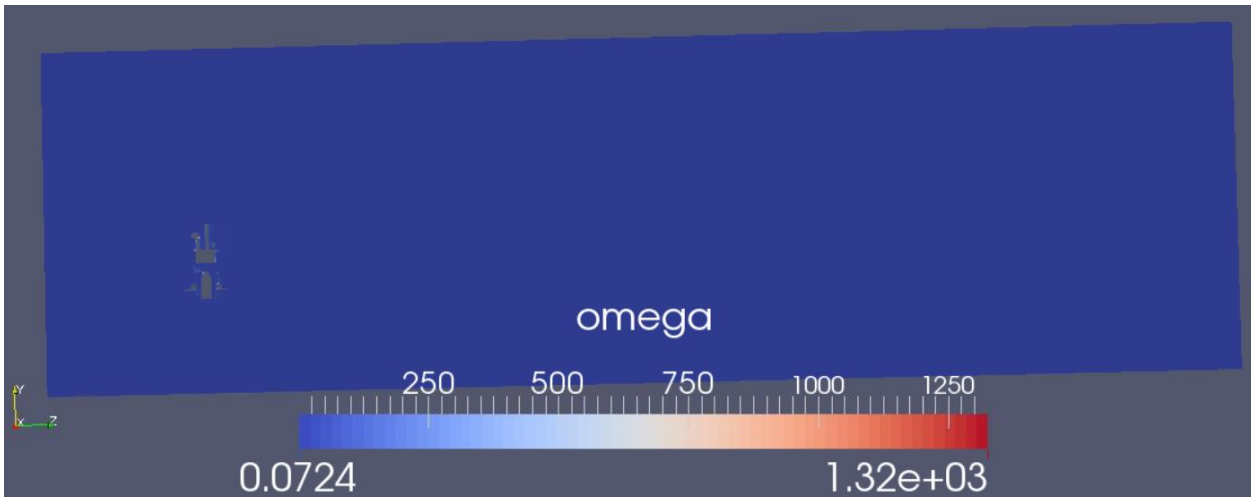

Figure 20

U:

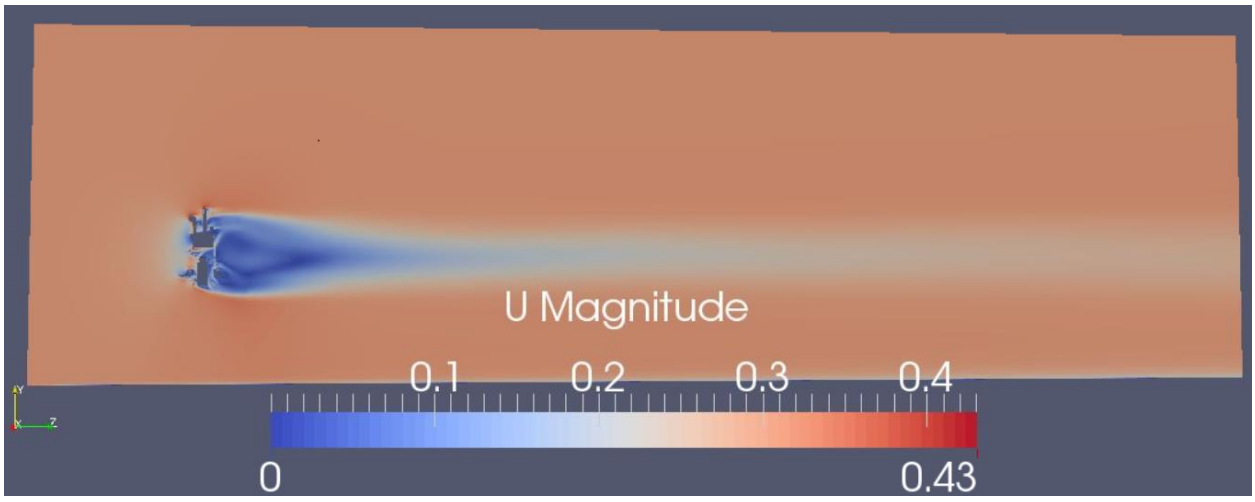

Figure 21

p:

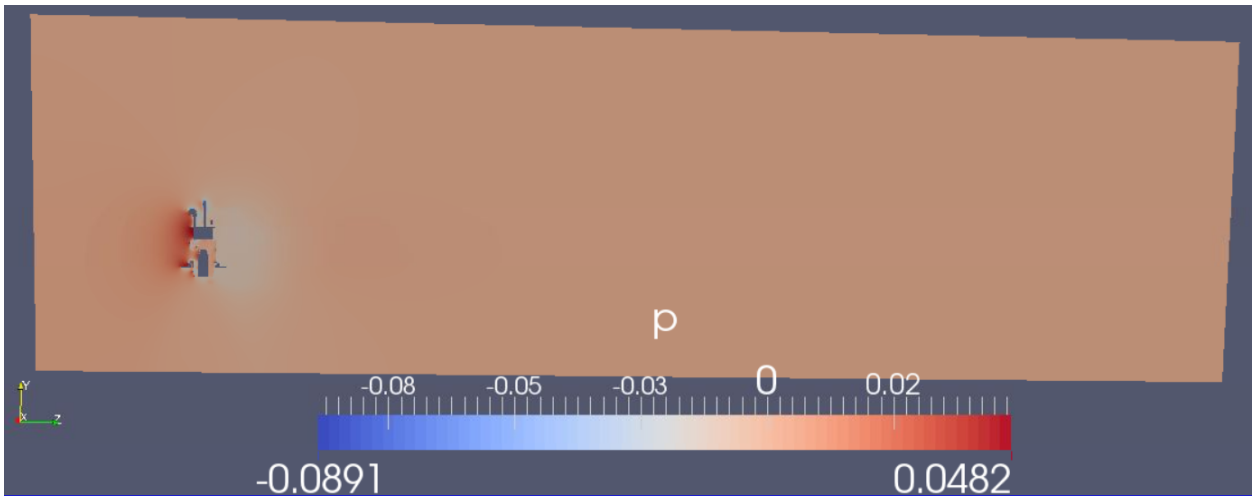

Figure 22

nut:

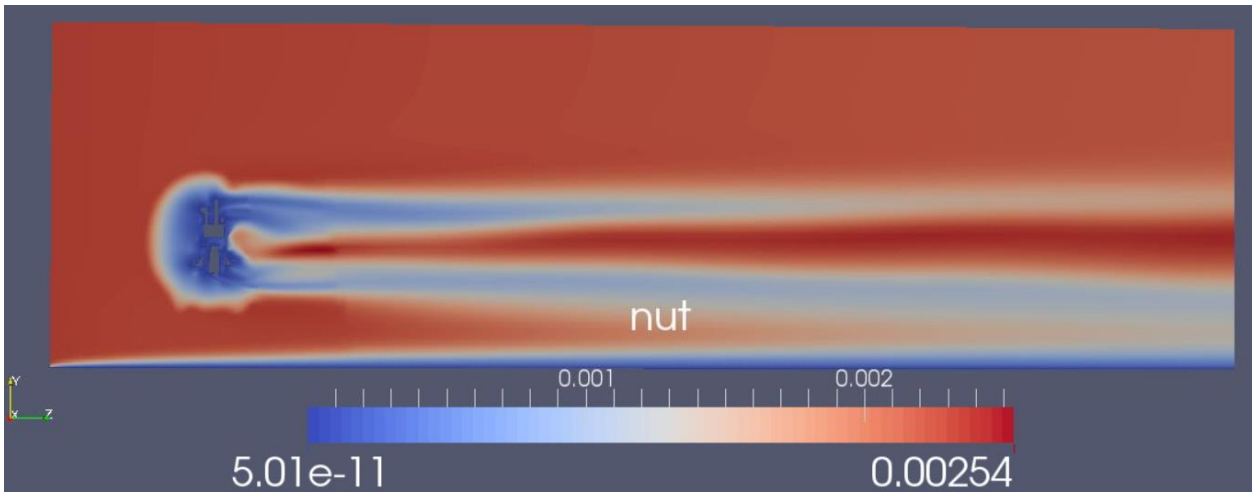

Figure 23

75 cm from the sensor is where the measurements takes place now. The values obtained by the sensor against the actual flow velocity is listed in the table below

Flow velocity / ***Turb. intensity***	**0,5 m/s**	**0,3 m/s**
5%	0,5349 m/s	0,3239 m/s
3,5%	0,5386 m/s	0,3225 m/s
2%	0,5399 m/s	0,3242 m/s

Table 6

The results show that the errors differ in the range between 7% and 8%, so the results are not largely affected by the turbulence intensity, as they produce the merely the same errors. An error on 7-8% is probably not catastrophic, but it can easily be reduced drastically only by simple adjustments, so as this is not expensive in any way, one should consider moving the monitoring point further away. Let us look at the same results as in the table above, only now moving 2,5 meters away from the sensor.

Flow velocity / ***Turb. Intensity***	**0,5 m/s**	**0,3 m/s**
5%	0,5108 m/s	0,3067 m/s
3,5%	0,5109 m/s	0,3066 m/s
2%	0,5113 m/s	0,3068 m/s

Table 7

The results for the flow velocity at 2,5 meters away from the sensor tells us that the error here lies slightly above 2%, and is not varying much between the different velocity and turbulence combinations. If we move even further away, the error will be even smaller, so depending on the importance of having a measurement close or not, one would ideally move the measuring point even further away.

Concentration

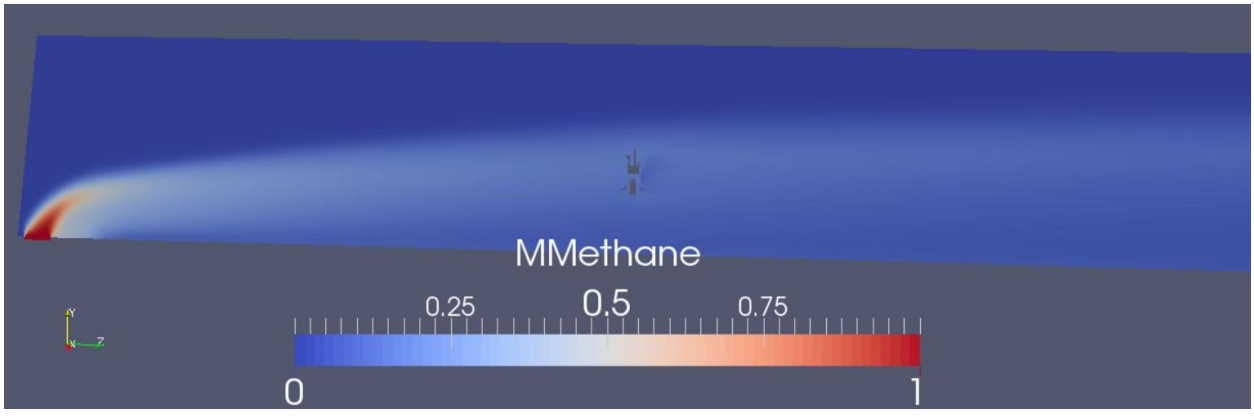

Figure 24

The simulations done for the sensors measuring the concentration of methane is done in six different flow direction, and also a sensitivity test is done for turbulence intensity, so one of the flow directions is simulated with 3 different turbulence intensities. One of the simulations is

shown in Figure 24. All simulations are done both with and without the presence of the construction, and all simulations has the same monitoring lines. Tables are used to present the data from the graphs, where the value of concentrations are put in to the tables for all the monitoring lines. The value along the line, especially for the simulations done with the construction present, varies, as shown in Figure 25. Therefore a qualified mean value approximation for the lines are done, so that the values does not need to be overanalyzed. 3 digits after the commas are used, after that the differences are so small that they are neglectable. The solution is probably not that accurate anyway, that it can predict the concentration precisely down to 0,01% variation.

The four monitoring points are called the same as they are called in the controlDict file: inside, outside, over, under and methanebetween. The first four lines are around pump number 4 as explained in the modelling section, and the last line is between pump number 1 and 2. Numbers are shown in Figure 14.

For 0°, 0.5 m/s and 5% turbulence intensity:

	Inside	**Outside**	**Over**	**Under**	**Methanebetween**
With construction	0,152	0,154	0,156	0,151	0,147
No construction	0,152	0,152	0,155	0,151	0,147

Table 8

For 90°, 0.5 m/s and 5% turbulence intensity:

	Inside	**Outside**	**Over**	**Under**	**Methanebetween**
With construction	0,153	0,154	0,156	0,151	0,147
No construction	0,153	0,153	0,155	0,152	0,147

Table 9

For 135°, 0.5 m/s and 5% turbulence intensity:

	Inside	Outside	Over	Under	Methanebetween
With construction	0,052	0,052	0,053	0,052	0,052
No construction	0,051	0,052	0,052	0,051	0,050

Table 10

For 180°, 0.5 m/s and 5% turbulence intensity:

	Inside	Outside	Over	Under	Methanebetween
With construction	0,121	0,130	0,115	0,139	0,147
No construction	0,155	0,154	0,157	0,153	0,148

Table 11

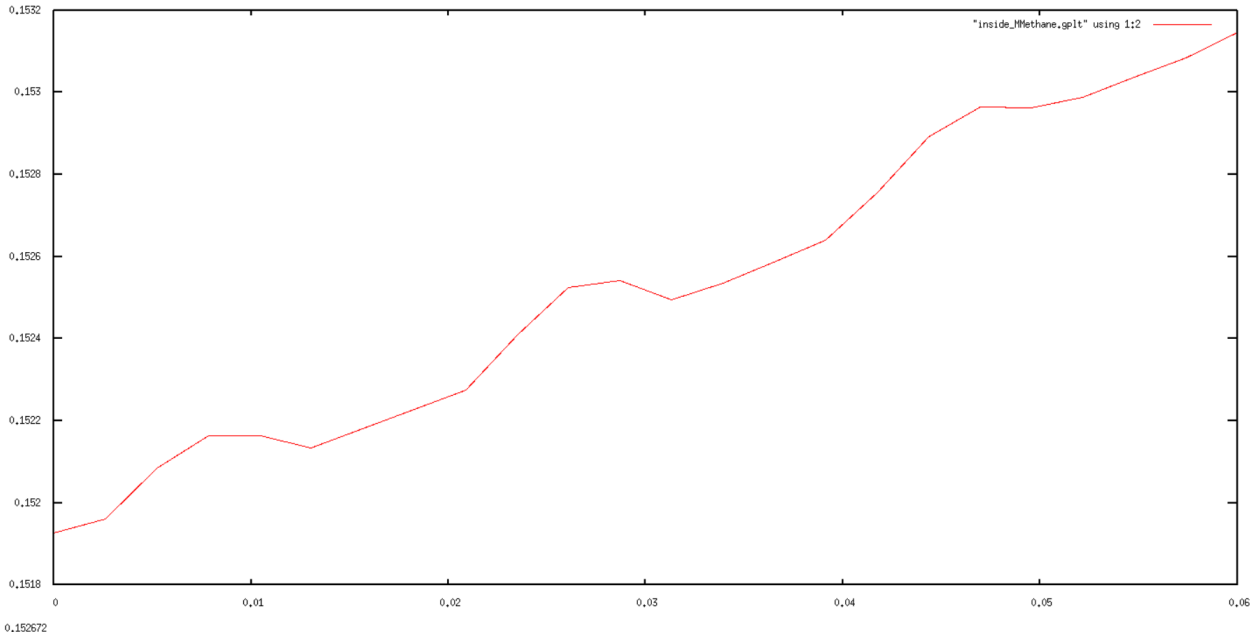

Figure 25

For 270°, 0.5 m/s and 5% turbulence intensity:

	Inside	Outside	Over	Under	Methanebetween
With construction	0,152	0,155	0,154	0,151	0,148
No construction	0,153	0,153	0,155	0,151	0,148

Table 12

For 315°, 0.5 m/s and 5% turbulence intensity:

	Inside	Outside	Over	Under	Methanebetween
With construction	0,053	0,054	0,055	0,054	0,053
No construction	0,053	0,054	0,054	0,053	0,052

Table 13

For 0°, 0.5 m/s and 3.5% turbulence intensity:

	Inside	Outside	Over	Under	Methanebetween
With construction	0,156	0,158	0,160	0,155	0,150
No construction	0,156	0,157	0,159	0,155	0,151

Table 14

For 0°, 0.5 m/s and 2% turbulence intensity:

	Inside	Outside	Over	Under	Methanebetween
With construction	0,161	0,163	0,165	0,160	0,155
No construction	0,161	0,162	0,164	0,160	0,156

Table 15

As one can see from the results collected and displayed in the tables above, most parts of the flow directions does not seem to have much effect on the concentration measurements. It is only in one direction one can see a distinct differences in the measurements, and that is for the 180°-case. When looking at the construction, this is as expected, because there is a plate located upstream relative to the pumps, and it is probably this part that is reducing the concentration in the volume behind it. For the monitoring line placed over pump number 4, the concentration is actually 36,5% higher for the no construction case, than the measured value with the construction, so one might say there is a noticeable difference.

Conclusion and discussion

The results has shown that the construction has an affection on the sensors, due to the results gathered. It has also shown that for most cases, there will not be much different, but given the right conditions, we can see that it will. For the acoustic sensor, the simulations done are assumed to be the ones having most error, and therefore the results shows that even the worst error possible is not so bad.

For the concentration measurements, the results were pretty good for most flow directions, but the flow direction where the plate on the construction is located upstream relative to the pumps, is affected to some degree. The difference though, is not so critical. If one should be so unlucky to have a leakage coming from this flow direction, one should take this fact into account when looking at the measurements.

The results looks valid, but as mentioned in the introduction, there could also have been done more simulations on the project. It could be interesting to look at an LES turbulence model, to see if it generates the same results as the RANS model, or if the construction in reality has more or less affection than in these simulations.

The transient simulation, also mentioned in the introduction, could also have been exciting to look at. With such a big model, it was known from the start that this would demand much time, but as there was a lack of exactly that, and also limited computer power due to many users in the cluster system, this idea had to be discarded.

The simulations can very well be something to build on for Stinger Technology if they are interested in finding out more about their construction, and I hope they do!

Bibliography

[1] H. K. V. a. W. Malalasekera, An Introduction to Computational Fluid Dynamics: The Finite Volume Method, 2 ed., Harlow: Pearson Education Limited, 2007.

[2] CFD Direct, "CFD Direct," [Online]. Available: http://cfd.direct/openfoam/user-guide/introduction/#x3-20001. [Accessed 14 June 2016].

[3] CFD Direct, "CFD Direct," [Online]. Available: http://cfd.direct/openfoam/user-guide/case-file-structure/#x17-930004.1. [Accessed 14 June 2016].

[4] CFD Direct, "CFD Direct," [Online]. Available: http://cfd.direct/openfoam/user-guide/fvSchemes/. [Accessed 14 June 2016].

[5] CFD Direct, "CFD Direct," [Online]. Available: http://cfd.direct/openfoam/user-guide/fvsolution/. [Accessed 14 June 2016].

[6] M. Winter, "Benchmark and validation of Open Source CFD codes, with focus on compressible and rotating capabilities, for integration on the SimScale platform.," Chalmers University of Technology, 2013. [Online]. Available: http://publications.lib.chalmers.se/records/fulltext/198965/198965.pdf. [Accessed 14 June 2016].

[7] "The Engineering Toolbox," [Online]. Available: http://www.engineeringtoolbox.com/diffusion-coefficients-d_1404.html. [Accessed 14 June 2016].

[8] International Towing Tank Conference, "Testing and Extrapolation Methods, General Density and Viscosity of Water," 1999. [Online]. Available: http://ittc.info/downloads/Archive%20of%20recommended%20procedures/2002%20Recommended%20Procedures/7.5-02-01-03.pdf. [Accessed 14 June 2016].

[9] M. Payandeh, "Implementation of a Temperature Dependent Viscosity Model in OpenFOAM," 2012. [Online]. Available: http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2012/MostafaPayandeh/ViscosityModel%20in%20openFoam%20v2.pdf. [Accessed 14 June 2016].

[10] P. B. H. Hjertager, Turbulence Theory and Modelling, 2 ed., Stavanger: Dept. of Mechanical and Structural Engineering and Material Science, 2014.

[11] "CFD Online," [Online]. Available: http://www.cfd-online.com/Wiki/SST_k-omega_model. [Accessed 14 June 2016].

[12] CFD Direct, "CFD Direct," [Online]. Available: http://cfd.direct/openfoam/user-guide/snappyhexmesh/. [Accessed 14 June 2016].

[13] B. H. Hjertager, "Lecture notes in OpenFOAM*," University of Stavanger, Stavanger, 2009.

[14] FLUENT INCORPORATED, "FLUENT INCORPORATED," [Online]. Available: http://jullio.pe.kr/fluent6.1/help/html/ug/node178.htm. [Accessed 14 June 2016].

[15] CFD Direct, "CFD Direct," [Online]. Available: http://cfd.direct/openfoam/user-guide/boundaries/. [Accessed 14 June 2016].

Appendixes

Appendix 1

Plotscript, used to plot the residuals:

```
set logscale y
set title "Residuals"
set ylabel 'Residual'
set xlabel 'Iteration'
set key outside
set grid

nCorrectors=1
nNonOrthogonalCorrectors=3

nCont = nCorrectors
nP = (nNonOrthogonalCorrectors+1)*nCont
nPSkip = nP-1
nContSkip = nCont-1



plot "< cat log | grep 'Solving for Ux' | cut -d' ' -f9 | tr -d ',\'" title
'Ux' with lines,\
"< cat log | grep 'Solving for Uy' | cut -d' ' -f9 | tr -d ',\'" title 'Uy'
with lines,\
"< cat log | grep 'Solving for Uz' | cut -d' ' -f9 | tr -d ',\'" title 'Uz'
with lines,\
"< cat log | grep 'Solving for k' | cut -d' ' -f9 | tr -d ',\'" title 'k'
with lines,\
"< cat log | grep 'Solving for omega' | cut -d' ' -f9 | tr -d ',\'" title
'omega' with lines,\
"< cat log | grep 'time step continuity errors :' | cut -d' ' -f9 | tr -d
',\'" every nCont::nContSkip title 'continuity' with lines,\
"< cat log | grep 'Solving for p' | cut -d' ' -f9 | tr -d ',\'" title 'p'
with lines
pause 1
#xmax = GPVAL_DATA_X_MAX+2
#xmin = xmax-40
#set xrange [xmin:xmax]
reread
```

Appendix 2

k:

```
/*--------------------------------*- C++ -*----------------------------------*\
|       o          |
|
|    o     o       | HELYX-OS
|
|   o   O   o      | Version: v2.1.1
|
|    o     o       | Web:     http://www.engys.com
|
|       o          |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version 2.0;
    format ascii;
    class volScalarField;
    location "0";
    object k;
}
dimensions [ 0 2 -2 0 0 0 0 ];
internalField uniform 0.1;
boundaryField
{
    inlet
    {
        type turbulentIntensityKineticEnergyInlet;
        value uniform 0.01;
        intensity 0.05;
    }

    outlet
    {
        type inletOutlet;
        value uniform 0.1;
        inletValue uniform 0.01;
    }

    bottom
    {
        type turbulentIntensityKineticEnergyInlet;
        value uniform 0.01;
        intensity 0.05;
    }

    top
    {
        type slip;
    }

    left
    {
        type slip;
    }
```

```
    right
    {
        type slip;
    }

    Simplerepaired
    {
        type kqRWallFunction;
        value uniform 1e-20;
    }

    Groundplate
    {
        type kqRWallFunction;
        value uniform 1e-20;
    }

}
```

Appendix 3

nut:

```
/*--------------------------------*- C++ -*----------------------------------*\
|       o          |
|
|    o     o       | HELYX-OS
|
|   o   O   o      | Version: v2.1.1
|
|    o     o       | Web:     http://www.engys.com
|
|       o          |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version 2.0;
    format ascii;
    class volScalarField;
    location "0";
    object nut;
}
dimensions [ 0 2 -1 0 0 0 0 ];
internalField uniform 0.001;
boundaryField
{
    inlet
    {
        type zeroGradient;
    }

    outlet
    {
        type zeroGradient;
    }

    bottom
    {
```

```
        type zeroGradient;
    }

    top
    {
        type slip;
    }

    left
    {
        type slip;
    }

    right
    {
        type slip;
    }

    Simplerepaired
    {
        type nutUSpaldingWallFunction;
        value uniform 0.001;
    }

    Groundplate
    {
        type nutUSpaldingWallFunction;
        value uniform 0.001;
    }

}
```

Appendix 4

omega:

```
/*--------------------------------*- C++ -*----------------------------------*\
|       o          |
|
|    o     o       | HELYX-OS
|
|   o   O   o      | Version: v2.1.1
|
|    o     o       | Web:     http://www.engys.com
|
|       o          |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version 2.0;
    format ascii;
    class volScalarField;
    location "0";
    object omega;
}
dimensions [ 0 0 -1 0 0 0 0 ];
```

```
internalField uniform 0.1;
boundaryField
{
    inlet
    {
        type turbulentMixingLengthFrequencyInlet;
        value uniform 0.01;
        mixingLength 0.23;
    }

    outlet
    {
        type inletOutlet;
        value uniform 0.1;
        inletValue uniform 0.1;
    }

    bottom
    {
        type turbulentMixingLengthFrequencyInlet;
        value uniform 0.01;
        mixingLength 0.07;
    }

    top
    {
        type slip;
    }

    left
    {
        type slip;
    }

    right
    {
        type slip;
    }

    Simplerepaired
    {
        type omegaWallFunction;
        value uniform 1;
    }

    Groundplate
    {
        type omegaWallFunction;
        value uniform 1;
    }

}
```

Appendix 5

p:

```
/*--------------------------------*- C++ -*----------------------------------*\
|       o          |
|
|    o     o       | HELYX-OS
|
|   o   O   o      | Version: v2.1.1
|
|    o     o       | Web:     http://www.engys.com
|
|       o          |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version 2.0;
    format ascii;
    class volScalarField;
    location "0";
    object p;
}
dimensions [ 0 2 -2 0 0 0 0 ];
internalField uniform 0.0;
boundaryField
{
    inlet
    {
        type zeroGradient;
    }

    outlet
    {
        type totalPressure;
        p0 uniform 0;
        value uniform 0;
        rho none;
        psi none;
        U U;
        phi phi;
        gamma 1;
    }

    bottom
    {
        type zeroGradient;
    }

    top
    {
        type slip;
    }

    left
    {
        type slip;
    }
```

```
    right
    {
        type slip;
    }

    Simplerepaired
    {
        type zeroGradient;
    }

    Groundplate
    {
        type zeroGradient;
    }

}
```

Appendix 6

U:

```
/*--------------------------------*- C++ -*----------------------------------*\
|       o          |
|
|    o     o       | HELYX-OS
|
|   o   O   o      | Version: v2.1.1
|
|    o     o       | Web:     http://www.engys.com
|
|       o          |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version 2.0;
    format ascii;
    class volVectorField;
    location "0";
    object U;
}
dimensions [ 0 1 -1 0 0 0 0 ];
internalField uniform (0.0 0.0 0.0);
boundaryField
{
    inlet
    {
        type fixedValue;
        value uniform (0.0 0.0 0.5 );
    }

    outlet
    {
        type pressureInletOutletVelocity;
        value uniform ( 0 0 0);
    }
```

```
    bottom
    {
        type fixedValue;
      value uniform ( 0.0 1.0 0.0);
    }

    top
    {
        type slip;
    }

    left
    {
        type slip;
    }

    right
    {
        type slip;
    }

    Simplerepaired
    {
        type fixedValue;
        value uniform ( 0 0 0);
    }

    Groundplate
    {
        type fixedValue;
        value uniform ( 0 0 0);
    }

}
```

Appendix 7

MMethane:

```
/*--------------------------------*- C++ -*----------------------------------
---*\
|       o          |
|
|    o     o       | HELYX-OS
|
|   o   O   o      | Version: v2.1.1
|
|    o     o       | Web:     http://www.engys.com
|
|       o          |
|
\*---------------------------------------------------------------------------
---*/
FoamFile
{
    version 2.0;
    format ascii;
    class volScalarField;
```

```
    location "0";
    object MMethane;
}
dimensions [ 0 0 0 0 0 0 0 ];
internalField uniform 0.0;
boundaryField
{
    inlet
    {
        type fixedValue;
        value uniform 0.0;
    }

    outlet
    {
        type inletOutlet;
        value uniform 0.0;
        inletValue uniform 0.00;
    }

    bottom
    {
        type fixedValue;
      value uniform 1.0;
    }

    top
    {
        type slip;
    }

    left
    {
        type slip;
    }

    right
    {
        type slip;
    }

    Simplerepaired
    {
        type zeroGradient;
    }

    Groundplate
    {
        type zeroGradient;
    }
}
```

Appendix 8

RASProperties:

```
/*---------------------------------*- C++ -*----------------------------------
---*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.4.0
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------
---*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      RASProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* //

RASModel            kOmegaSST;

turbulence          on;

printCoeffs         on;


//
************************************************************************
//
```

Appendix 9

transportProperties – concentration

```
/*---------------------------------*- C++ -*----------------------------------
---*\
|       o          |
|
|    o     o       | HELYX-OS
|
|   o   O   o      | Version: v2.3.1
|
|    o     o       | Web:     http://www.engys.com
|
|       o          |
|
\*---------------------------------------------------------------------------
---*/
```

```
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location constant;
    object transportProperties;
}

transportModel  Newtonian;
nu              nu [0 2 -1 0 0 0 0] 1.6094e-06; // Kinematic viscosity of
fluid in m2/s, sea water at 4 degrees celsius.
Sc              Sc [0 0 0 0 0 0 0] 0.7;      // Turbulent Schmidt number
scalars
(
Methane
{
  DM              DM [0 2 -1 0 0 0 0] 1.5e-09;  // Methane in water
}
);
```

Appendix 10

blockMesh:

```
/*--------------------------------*- C++ -*----------------------------
---*\
|       o          |
|
|    o     o       | HELYX-OS
|
|   o   O   o      | Version: v2.1.1
|
|    o     o       | Web:     http://www.engys.com
|
|       o          |
|
\*---------------------------------------------------------------------
---*/
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location system;
    object blockMeshDict;
}

    convertToMeters 1;
    vertices
    (

        ( -3.75 -2.3 -22.25)
        ( 3.75 -2.3 -22.25)
        ( 3.75 5.2 -22.25)
        ( -3.75 5.2 -22.25)
        ( -3.75 -2.3 22.0)
        ( 3.75 -2.3 22.0)
        ( 3.75 5.2 22.0)
```

```
        ( -3.75 5.2 22.0)
    );
    blocks
    (
    hex (0 1 2 3 4 5 6 7) (50 50 295 ) simpleGrading (1 1 1)
    );
    edges ( );
    patches
    (

 wall right ( (0 4 7 3) )
 wall left ( (1 2 6 5) )
 wall bottom ( (0 1 5 4) )
 wall top ( (3 7 6 2) )
 wall inlet ( (0 3 2 1) )
 wall outlet ( (4 5 6 7) )

    );
    mergePatchPairs ( );
```

Appendix 11

controlDict:

```
/*--------------------------------*- C++ -*----------------------------------*\
|       o          |
|
|    o     o       | HELYX-OS
|
|   o   O   o      | Version: v2.3.1
|
|    o     o       | Web:     http://www.engys.com
|
|       o          |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location system;
    object controlDict;
}

    startFrom startTime;
    startTime 0;
    stopAt endTime;
    endTime 2500;
    deltaT 1.0;
    writeControl timeStep;
    writeInterval 100.0;
    purgeWrite 0;
    writeFormat ascii;
    writePrecision 10;
```

```
    writeCompression uncompressed;
    timeFormat general;
    timePrecision 6;
    graphFormat raw;
    runTimeModifiable true;
    adjustTimeStep false;
    maxCo 0.0;
    maxDeltaT 0.0;

functions
{

      setTest
      {
            type sets;
            functionObjectLibs ("libsampling.so");

            setFormat gnuplot;

            interpolationScheme cellPointFace;

            fields
            (
                  p U MMethane
            );

            sets
            (
                  linevelocity
                  {
                        type       uniform;
                        axis       distance;


                        start      (-0.025 1.04 0.16);
                        end        (-0.025 5.04 4.16);
                        nPoints          320;
                  }

                  methanebetween
                  {
                        type       uniform;
                        axis       distance;


                        start      (-0.065 0.26 -0.27);
                        end        (-0.1 0.26 -0.27);
                        nPoints          140;
                  }

                  over
                  {
                        type       uniform;
                        axis       distance;


                        start      (0.03 0.505 -0.17);
                        end        (0.09 0.505 -0.17);
                        nPoints          24;
                  }
```

```
                under
                {
                      type       uniform;
                      axis       distance;


                      start      (0.03 0.381 -0.17);
                      end        (0.09 0.381 -0.17);
                      nPoints          24;
                }

                inside
                {
                      type       uniform;
                      axis       distance;


                      start      (0.03 0.443 -0.108);
                      end        (0.09 0.443 -0.108);
                      nPoints          24;
                }

                outside
                {
                      type       uniform;
                      axis       distance;


                      start      (0.03 0.443 -0.232);
                      end        (0.09 0.443 -0.232);
                      nPoints          24;
                }
          );

          outputControl outputTime;
     }

}
```

Appendix 12

decomposeParDict:

```
/*--------------------------------*- C++ -*----------------------------------*\
|       o          |
|
|    o     o       | HELYX-OS
|
|   o   O   o      | Version: v2.3.1
|
|    o     o       | Web:     http://www.engys.com
|
|       o          |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
```

```
    version 2.0;
    format ascii;
    class dictionary;
    location system;
    object decomposeParDict;
}

    numberOfSubdomains 16;
    method hierarchical;
    hierarchicalCoeffs
    {
        n ( 2 4 2);
        delta 0.001;
        order yxz;
    }

    distributed false;
    roots
    (
    );
```

Appendix 13

fvSchemes:

```
/*--------------------------------*- C++ -*----------------------------------
---*\
|       o          |
|
|    o     o       | HELYX-OS
|
|   o   O   o      | Version: v2.3.1
|
|    o     o       | Web:     http://www.engys.com
|
|       o          |
|
\*---------------------------------------------------------------------------
---*/
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location system;
    object fvSchemes;
}

    ddtSchemes
    {
        default steadyState;
    }

    gradSchemes
    {
        default cellLimited Gauss linear 1;
        grad(nuTilda) cellLimited Gauss linear 1;
        grad(k) cellLimited Gauss linear 1;
        grad(kl) cellLimited Gauss linear 1;
```

```
    grad(omega) cellLimited Gauss linear 1;
    grad(epsilon) cellLimited Gauss linear 1;
    grad(q) cellLimited Gauss linear 1;
    grad(zeta) cellLimited Gauss linear 1;
    grad(v2) cellLimited Gauss linear 1;
    grad(f) cellLimited Gauss linear 1;
    grad(sqrt(kt)) cellLimited Gauss linear 1;
    grad(kt) cellLimited Gauss linear 1;
    grad(sqrt(kl)) cellLimited Gauss linear 1;
}

divSchemes
{
    default bounded Gauss linear;
    div(phi,U) bounded Gauss linearUpwindV grad(U);
    div(phi,k) bounded Gauss linearUpwind grad(k);
    div(phi,epsilon) bounded Gauss linearUpwind grad(epsilon);
    div(phi,zeta) bounded Gauss linearUpwind grad(zeta);
    div(phi,q) bounded Gauss linearUpwind grad(q);
    div(phi,omega) bounded Gauss linearUpwind grad(omega);
    div(phi,nuTilda) bounded Gauss linearUpwind grad(nuTilda);
    div(phi,T) bounded Gauss limitedLinear 1;
    div(phi,kl) Gauss limitedLinear 1;
    div(phi,kt) Gauss limitedLinear 1;
    div(phi,R) Gauss upwind;
    div(phi,MMethane) bounded Gauss upwind;
    div(R) Gauss linear;
    div((nuEff*dev(grad(U).T()))) Gauss linear;
    div(phi,v2) bounded Gauss linearUpwind grad(v2);
    div(phi,f) bounded Gauss linearUpwind grad(f);
}

interpolationSchemes
{
    default linear;
    interpolate(HbyA) linear;
}

laplacianSchemes
{
    default Gauss linear limited 0.333;
}

snGradSchemes
{
    default limited 0.333;
}

fluxRequired
{
    default no;
    p ;
}

wallDist
{
  method meshWave;
}
```

Appendix 14

fvSolution:

```
/*--------------------------------*- C++ -*----------------------------------*\
|       o          |
|
|    o     o       | HELYX-OS
|
|   o   O   o      | Version: v2.3.1
|
|    o     o       | Web:     http://www.engys.com
|
|       o          |
|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location system;
    object fvSolution;
}

    SIMPLE
    {
        nNonOrthogonalCorrectors 0;
        pressureImplicitPorousity false;
        pRefCell 0;
        pRefValue 0;
        residualControl
        {
            U 1.0E-5;
            k 1.0E-5;
            epsilon 1.0E-5;
            omega 1e-5;
            nuTilda 1e-5;
            T 1e-5;
            p_rgh 1e-5;
            p 1.0E-5;
          "M.*" 1e-5;
        }

    }

    solvers
    {
        p
        {
            solver GAMG;
            agglomerator faceAreaPair;
            mergeLevels 1;
            cacheAgglomeration true;
            nCellsInCoarsestLevel 200;
            tolerance 1e-7;
            relTol 0.01;
            smoother GaussSeidel;
            nPreSweeps 0;
```

```
    nPostSweeps 2;
    nFinestSweeps 2;
    minIter 1;
}

p_rgh
{
    solver GAMG;
    agglomerator faceAreaPair;
    mergeLevels 1;
    cacheAgglomeration true;
    nCellsInCoarsestLevel 200;
    tolerance 1e-7;
    relTol 0.01;
    smoother GaussSeidel;
    nPreSweeps 0;
    nPostSweeps 2;
    nFinestSweeps 2;
    minIter 1;
}

f
{
    solver GAMG;
    agglomerator faceAreaPair;
    mergeLevels 1;
    cacheAgglomeration true;
    nCellsInCoarsestLevel 200;
    tolerance 1e-7;
    relTol 0.01;
    smoother GaussSeidel;
    nPreSweeps 0;
    nPostSweeps 2;
    nFinestSweeps 2;
    minIter 1;
}

U
{
    solver smoothSolver;
    smoother GaussSeidel;
    tolerance 1e-6;
    relTol 0.1;
    minIter 1;
}

k
{
    solver smoothSolver;
    smoother GaussSeidel;
    tolerance 1e-6;
    relTol 0.1;
    minIter 1;
}

kl
{
    solver smoothSolver;
    smoother GaussSeidel;
    tolerance 1e-6;
    relTol 0.1;
```

```
    minIter 1;
}

kt
{
    solver smoothSolver;
    smoother GaussSeidel;
    tolerance 1e-6;
    relTol 0.1;
    minIter 1;
}

q
{
    solver smoothSolver;
    smoother GaussSeidel;
    tolerance 1e-6;
    relTol 0.1;
    minIter 1;
}

zeta
{
    solver smoothSolver;
    smoother GaussSeidel;
    tolerance 1e-6;
    relTol 0.1;
    minIter 1;
}

epsilon
{
    solver smoothSolver;
    smoother GaussSeidel;
    tolerance 1e-6;
    relTol 0.1;
    minIter 1;
}

R
{
    solver smoothSolver;
    smoother GaussSeidel;
    tolerance 1e-6;
    relTol 0.1;
    minIter 1;
}

nuTilda
{
    solver smoothSolver;
    smoother GaussSeidel;
    tolerance 1e-6;
    relTol 0.1;
    minIter 1;
}

omega
{
    solver smoothSolver;
    smoother GaussSeidel;
```

```
        tolerance 1e-6;
        relTol 0.1;
        minIter 1;
    }

    h
    {
        solver smoothSolver;
        smoother GaussSeidel;
        tolerance 1e-6;
        relTol 0.1;
        minIter 1;
    }

    T
    {
        solver smoothSolver;
        smoother GaussSeidel;
        tolerance 1e-6;
        relTol 0.1;
        minIter 1;
    }

    v2
    {
        solver smoothSolver;
        smoother GaussSeidel;
        tolerance 1e-6;
        relTol 0.1;
        minIter 1;
    }

    rho
    {
        solver PCG;
        preconditioner DIC;
        tolerance 0;
        relTol 0;
        minIter 1;
    }

    rhoFinal
    {
        solver PCG;
        preconditioner DIC;
        tolerance 0;
        relTol 0;
        minIter 1;
    }

    e
    {
        solver PBiCG;
        preconditioner DILU;
        tolerance 1e-06;
        relTol 0.1;
        minIter 1;
    }

"M.*"
{
```

```
        solver    smoothSolver;
        smoother  GaussSeidel;
        tolerance 1e-8;
        relTol    0.1;
        minIter 1;
       // nSweeps   1;

    }

}

relaxationFactors
{
    fields
    {
        p_rgh 0.3;
        p 0.3;
        rho 0.05;
    }

    equations
    {
        U 0.7;
        h 0.5;
        k 0.7;
        kl 0.7;
        kt 0.7;
        q 0.7;
        zeta 0.7;
        epsilon 0.7;
        R 0.7;
        nuTilda 0.7;
        omega 0.7;
        T 0.7;
        v2 0.7;
        f 0.7;
      "M.*" 0.7;
    }

}
```

Appendix 15

snappyHexMeshDict:

```
/*--------------------------------*- C++ -*----------------------------------
---*\
|       o          |
|
|    o     o       | HELYX-OS
|
|   o   O   o      | Version: v2.1.1
|
|    o     o       | Web:     http://www.engys.com
|
|       o          |
|
\*---------------------------------------------------------------------------
---*/
FoamFile
```

```
{
    version 2.0;
    format ascii;
    class dictionary;
    location system;
    object snappyHexMeshDict;
}

    castellatedMesh true;
    snap true;
    addLayers true;
    geometry
    {
        Simplerepaired.stl
        {
            type triSurfaceMesh;
            name Simplerepaired;
        }

      Box0.stl
        {
            type triSurfaceMesh;
            name Groundplate;
        }

    }

    castellatedMeshControls
    {
        features
        (
        );
        refinementSurfaces
        {
            Simplerepaired
            {
                level (1 5 );
            }

          Groundplate
            {
                level (1 4 );
            }

        }

        refinementRegions
        {

          Simplerepaired
          {
            mode distance;
            levels ( ( 0.1 3 ) ( 0.5 2 ) ( 2.0 1 ) );
          }

        }

        locationInMesh (1.0 0.0 0.0 );
        maxLocalCells 10000000;
        maxGlobalCells 200000000;
        minRefinementCells 0;
```

```
    nCellsBetweenLevels 1;
    resolveFeatureAngle 30.0;
    allowFreeStandingZoneFaces true;
    planarAngle 30;
    maxLoadUnbalance 0.1;
}

snapControls
{
    nSolveIter 30;
    nSmoothPatch 3;
    tolerance 2.0;
    nRelaxIter 5;
    nFeatureSnapIter 10;
    implicitFeatureSnap true;
    explicitFeatureSnap false;
    multiRegionFeatureSnap false;
}

addLayersControls
{
    layers
    {
        Simplerepaired
        {
            nSurfaceLayers 2;
            expansionRatio 1.2;
        }

      Groundplate
        {
            nSurfaceLayers 2;
            expansionRatio 1.2;
        }

    }

    relativeSizes true;
    expansionRatio 1.0;
    finalLayerThickness 0.3;
    minThickness 0.25;
    nGrow 0;
    featureAngle 130;
    slipFeatureAngle 30;
    nRelaxIter 5;
    nSmoothSurfaceNormals 1;
    nSmoothNormals 3;
    nSmoothThickness 10;
    maxFaceThicknessRatio 0.5;
    maxThicknessToMedialRatio 0.3;
    minMedianAxisAngle 90;
    nBufferCellsNoExtrude 0;
    nLayerIter 50;
    nRelaxedIter 20;
  writeVTK false;
    noErrors false;
    layerRecovery 1;
    growZoneLayers false;
    projectGrownUp 0.0;
}
```

```
    meshQualityControls
    {
        maxNonOrtho 65;
        maxBoundarySkewness 20;
        maxInternalSkewness 4;
        maxConcave 80;
        minFlatness 0.5;
        minVol 1.0E-13;
        minTetQuality 1.0E-15;
        minArea -1.0;
        minTwist 0.02;
        minDeterminant 0.001;
        minFaceWeight 0.05;
        minVolRatio 0.01;
        minTriangleTwist -1.0;
        nSmoothScale 4;
        errorReduction 0.75;
      relaxed
      {
      }
    }

    debug 0;
    mergeTolerance 1.0E-6;
    autoBlockMesh false;
```

Appendix 16

checkMesh-output for the used model, and full domain:

```
/*---------------------------------------------------------------------------
---*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  2.4.x
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------
---*/
Build  : 2.4.x-2b147f41daf9
Exec   : checkMesh
Date   : Jun 14 2016
Time   : 11:36:56
Host   : "gorina2.ux.uis.no"
PID    : 28786
Case   :
/nfs/student/rogunev/simple/0.5ms/5pct/concentration/withconstruction/0deg
rees/simpleFoam24
nProcs : 1
sigFpe : Enabling floating point exception trapping (FOAM_SIGFPE).
fileModificationChecking : Monitoring run-time modified files using
timeStampMaster
allowSystemOperations : Allowing user-supplied system call operations
```

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* //
Create time

Create polyMesh for time = 0

Time = 0

Mesh stats
    points:           3276336
    faces:            8754634
    internal faces:   8337232
    cells:            2751927
    faces per cell:   6.210871873
    boundary patches: 8
    point zones:      0
    face zones:       0
    cell zones:       0

Overall number of cells of each type:
    hexahedra:     2417732
    prisms:        34269
    wedges:        0
    pyramids:      0
    tet wedges:    1315
    tetrahedra:    34
    polyhedra:     298577
    Breakdown of polyhedra by number of faces:
        faces   number of cells
            4   15012
            5   13718
            6   56916
            7   44773
            8   25247
            9   104355
           10   2394
           11   969
           12   26565
           13   55
           14   244
           15   7782
           16   9
           17   51
           18   487

Checking topology...
    Boundary definition OK.
    Cell to face addressing OK.
    Point usage OK.
    Upper triangular ordering OK.
    Face vertices OK.
    Number of regions: 1 (OK).

Checking patch topology for multiply connected surfaces...
                  Patch    Faces   Points                  Surface
topology
                  right    17110    17754  ok (non-closed singly
connected)
                   left    17110    17754  ok (non-closed singly
connected)
```

```
                  bottom      608      710  ok (non-closed singly
connected)
                     top    14750    15096  ok (non-closed singly
connected)
                   inlet     2900     3054  ok (non-closed singly
connected)
                  outlet     2900     3054  ok (non-closed singly
connected)
          Simplerepaired   296042   322804  ok (non-closed singly
connected)
             Groundplate    65982    67997  ok (non-closed singly
connected)

Checking geometry...
    Overall domain bounding box (-3.75 -2.3 -22.25) (3.75 5.2 22)
    Mesh (non-empty, non-wedge) directions (1 1 1)
    Mesh (non-empty) directions (1 1 1)
    Boundary openness (1.100255593e-14 -1.376079186e-13 -5.387453876e-17)
OK.
    Max cell openness = 6.514119042e-16 OK.
    Max aspect ratio = 21.986107 OK.
    Minimum face area = 6.542611615e-08. Maximum face area =
0.02366219514.  Face area magnitudes OK.
    Min volume = 9.329218644e-10. Max volume = 0.00348101122.  Total
volume = 2455.701983.  Cell volumes OK.
    Mesh non-orthogonality Max: 64.85235508 average: 8.862176946
    Non-orthogonality check OK.
    Face pyramids OK.
    Max skewness = 3.949652129 OK.
    Coupled point location match (average 0) OK.

Mesh OK.

End
```

Appendix 17

checkMesh of original construction, without groundplate, and with a small domain

```
/*---------------------------------------------------------------------------
---*\
| =========                 |
|
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
|
|  \\    /   O peration     | Version:  3.0.x
|
|   \\  /    A nd           | Web:      www.OpenFOAM.org
|
|    \\/     M anipulation  |
|
\*---------------------------------------------------------------------------
---*/
Build  : 3.0.x-4709bde9acf0
Exec   : checkMesh
Date   : Jun 14 2016
Time   : 13:27:27
Host   : "gorina1.ux.uis.no"
PID    : 112544
```

```
Case   : /nfs/student/rogunev/originalmesh
nProcs : 1
sigFpe : Enabling floating point exception trapping (FOAM_SIGFPE).
fileModificationChecking : Monitoring run-time modified files using
timeStampMaster
allowSystemOperations : Allowing user-supplied system call operations

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* //
Create time

Create polyMesh for time = 0

Time = 0

Mesh stats
    points:           4662287
    faces:            11889925
    internal faces:   11133499
    cells:            3643329
    faces per cell:   6.319337068
    boundary patches: 7
    point zones:      0
    face zones:       0
    cell zones:       0

Overall number of cells of each type:
    hexahedra:     2942198
    prisms:        72082
    wedges:        0
    pyramids:      0
    tet wedges:    2703
    tetrahedra:    75
    polyhedra:     626271
    Breakdown of polyhedra by number of faces:
        faces   number of cells
            4   44286
            5   40019
            6   112867
            7   84591
            8   51766
            9   206882
           10   5197
           11   2681
           12   59604
           13   338
           14   884
           15   15319
           16   55
           17   150
           18   1632

Checking topology...
    Boundary definition OK.
    Cell to face addressing OK.
    Point usage OK.
    Upper triangular ordering OK.
    Face vertices OK.
    Number of regions: 1 (OK).

Checking patch topology for multiply connected surfaces...
```

```
                    Patch    Faces   Points                  Surface
topology
                  ffminx     4113     4252  ok (non-closed singly
connected)
                  ffmaxx     4185     4325  ok (non-closed singly
connected)
                  ffminy     3240     3355  ok (non-closed singly
connected)
                  ffmaxy     2271     2379  ok (non-closed singly
connected)
                  ffminz     3432     3565  ok (non-closed singly
connected)
                  ffmaxz     3702     3833  ok (non-closed singly
connected)
          Simplerepaired   735483   801589  multiply connected (shared
edge)
  <<Writing 12 conflicting points to set nonManifoldPoints

Checking geometry...
    Overall domain bounding box (-1 -1 -1) (3.05 5 3.5)
    Mesh has 3 geometric (non-empty/wedge) directions (1 1 1)
    Mesh has 3 solution (non-empty) directions (1 1 1)
    Boundary openness (4.872284122e-15 2.604850829e-16 6.513982159e-15)
OK.
    Max cell openness = 7.602741223e-16 OK.
    Max aspect ratio = 25.34287972 OK.
    Minimum face area = 1.306010727e-08. Maximum face area =
0.02253910136.  Face area magnitudes OK.
    Min volume = 6.440021277e-10. Max volume = 0.003381828027.  Total
volume = 108.9748972.  Cell volumes OK.
    Mesh non-orthogonality Max: 72.25340478 average: 11.43011681
   *Number of severely non-orthogonal (> 70 degrees) faces: 9.
    Non-orthogonality check OK.
  <<Writing 9 non-orthogonal faces to set nonOrthoFaces
    Face pyramids OK.
 ***Max skewness = 11.30954387, 107 highly skew faces detected which may
impair the quality of the results
  <<Writing 107 skew faces to set skewFaces
    Coupled point location match (average 0) OK.

Failed 1 mesh checks.

End
```