TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

Eetu Autio

DYNAMIC OVERSET CFD SIMULATION OF A PNEUMATIC IMPACT DEVICE

Master of Science Thesis

Examiners: Professor Reijo Kouhia,
MSc Antti Mikkonen
Examiners and topic approved by the
Faculty Council of the Faculty of
Engineering Sciences
on 3rd October 2018

ABSTRACT

Autio, Eetu: Dynamic overset CFD simulation of a pneumatic impact device
Tampere University of Technology
Master of Science Thesis, 55 pages, 11 appendix pages
September 2018
Master's Degree Programme in Mechanical Engineering
Major: Applied Mechanics and Thermal Sciences
Examiners: Professor Reijo Kouhia, MSc Antti Mikkonen

Keywords: numerical, CFD, pneumatics, overset grid, rigid body dynamics, transient, transonic, axisymmetry, DTH drilling, porosity, OpenFOAM

Down-the-hole drilling is a percussive drilling technique where typically pneumatic down-the-hole hammer is used to produce stress waves to crush overburden and rock. Down-the-hole drilling is especially applicable in drilling deep holes, such as geothermal wells and water wells, in hard and medium hard drilling conditions. Other common applications for down-the-hole drilling are blast hole drilling in mines and quarries and construction piling.

Design of down-the-hole hammers has been based on analytical and semi-analytical models and experimental testing. The purpose of this thesis is to investigate the possibilities of computational fluid dynamics in designing of geothermal well and water well hammers using OpenFOAM software and its overset grid functionality. With overset method, separate grids can be placed on top of each other and information between grids is interpolated. This simulation technique permits arbitrarily large determined or undetermined object movements inside the computational domain which is hard with other types moving mesh techniques. Flow inside the down-the-hole hammer is compressible and transient. Piston movement inside the hammer is pressure-induced.

In this thesis, a simplified axisymmetric grid was used for the down-the-hole hammer and with two separate simulation approaches. In the first simulation approach the clearances between the moving piston and the walls of the down-the-hole hammer were meshed with high-resolution grid. However, this approach was not successful because the overset grid could not be made robust in parallel computation when the walls of the piston and the hammer were near. In second simulation approach the hammer walls near the piston were replaced by porous media zones to remedy the problems caused by overset grid. With this technique the problem from the first simulation approach could be avoided and grid cell size could be increased in the clearances.

Piston movement was simulated with porous media technique by giving the piston a 5 m/s initial velocity. However, given velocity was too high, and piston collided to the upper pressure chamber. Pressure, temperature and flow velocity data for the down-the-hole hammer were acquired, which would have been hard without simulation. Several observations of flow behavior were made which will require further studies. For example, it was noted that outflow of the hammer develops to transonic velocity producing mass flux and temperature fluctuations.

TIIVISTELMÄ

Autio, Eetu: Pneumaattisen iskulaitteen dynaaminen CFD simulaatio limitetyllä hilalla
Tampereen teknillinen yliopisto
Diplomityö, 55 sivua, 11 liitesivua
Syyskuu 2018
Konetekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Sovellettu mekaniikka ja lämpötekniikka
Tarkastajat: professori Reijo Kouhia, DI Antti Mikkonen

Avainsanat: virtaussimulointi, CFD, pneumatiikka, limitetty hila, jäykän kappaleen dynamiikka, transientti, transooninen, pyörähdyssymmetria, uppovasaraporaus, huokoisuus, OpenFOAM

Uppovasaraporaus on iskuporaustekniikka, jossa tyypillisesti pneumaattisella uppovasaralla tuotetaan kallio- ja maaperää murskaavia iskuaaltoja. Uppovasaraporaus soveltuu erityisesti syvien reikien, kuten maalämpö- ja vesikaivojen, poraamiseen kovassa ja keskikovassa maa- ja kallioperässä. Muita tyypillisiä uppovasaraporauksen käyttökohteita ovat räjäytysreikien poraus kaivoksissa ja louhoksissa sekä porapaalujen asennus.

Uppovasaroiden suunnittelu on perustunut analyyttisiin tai puolianalyyttisiin malleihin ja kokeelliseen testaukseen. Tämän opinnäytetyön tarkoituksena on tutkia numeerisen virtauslaskennan mahdollisuuksia maalämpö- ja vesikaivoille tarkoitettujen uppovasaroiden suunnitteluprosessissa käyttäen OpenFOAM virtauslaskentaohjelmaa sekä sen limitetyn hilan ominaisuutta. Limitetyssä hilassa eri laskentaverkot voidaan sijoittaa päällekkäin ja informaatio verkkojen välillä interpoloidaan. Tämä simulointitapa mahdollistaa mielivaltaisen suuret määrätyt tai määräämättömät kappaleiden liikkeet laskenta-alueen sisällä, mikä on vaikeaa muilla liikkuvan laskentaverkon tekniikoilla. Virtaus uppovasaran sisällä on kokoonpuristuvaa ja aikariippuvaa. Männän liike upposaran sisällä on paineohjattua.

Tässä opinnäytetyössä uppovasaralle käytettiin yksinkertaistettua pyörähdyssymetristä laskentaverkkoa sekä kahta eri lähestymistapaa vasaran simulointiin. Ensimmäisessä lähestymistavassa uppovasaran liikkuvan männän ja vasaran seinämien välykset verkotettiin tiheällä laskentaverkolla. Tämä lähestymistapa ei kuitenkaan onnistunut, sillä limitetty hila ei toiminut luotettavasti rinnakkaisessa laskennassa männän ja vasaran seinämien ollessa hyvin lähellä toisiaan. Toisessa simulaatiotavassa mäntää lähellä olevat seinämät korvattiin huokoisilla alueilla limitetyn hilan toiminnan parantamiseksi. Tällä tekniikalla pystyttiin välttämään limitetyn hilan numeeriset ongelmat sekä kasvattamaan laskentaverkon elementtikokoa välyksissä.

Männän liike simuloitiin huokoisen aineen tekniikalla antamalla männälle 5 m/s alkunopeus. Annettu nopeus oli kuitenkin liian suuri, sillä mäntä ei ehtinyt hidastua tarpeeksi vaan osui yläpainekammioon. Paine, lämpötila ja virtausnopeusdataa saatiin kerättyä uppovasaralle, mikä olisi ollut vaikeaa ilman virtaussimulointia. Muutamia havaintoja virtauskäyttäytymisestä tehtiin, jotka vaativat jatkotutkimusta. Esimerkiksi havaittiin, että vasaran virtaus ulostulossa kehittyi transooniseksi muodostaen vaihtelua ulos menevään massavirtaan sekä lämpötilaan.

PREFACE

I would like to thank Professor Reijo Kouhia for his guidance and support and all the staff of Tampere University of Technology for exemplary teaching during the years. I would also like to thank my colleagues for their help and advices about drilling applications. Special thanks goes to Antti Mikkonen for his invaluable guidance about computational fluid dynamics and OpenFOAM. I am also grateful to CSC for providing me necessary computing resources. Lastly, I want to thank my family and friends and especially my girlfriend Kati for being there for me.

CONTENTS

ABBREVIATIONS AND SYMBOLS

Abbreviations

DTH	Down-The-Hole
ITH	In-The-Hole
CFD	Computational Fluid Dynamics
OpenFOAM	Open source Field Operation And Manipulation
DNS	Direct Numerical Simulation
LES	Large Eddy Simulation
RANS	Reynolds-Averaged Navier-Stokes
SST	Shear Stress Transport
STL	Standard Tessellation Language
CAD	Computer Assisted Design
FVM	Finite Volume Method
SIMPLE	Semi-Implicit-Method-Of-Pressure-Linked-Equations
PISO	Pressure-Implicit-of-Split-Operations
PIMPLE	Merged PISO-SIMPLE
OGA	Overset Grid Assembly
STEP	Standard for the Exchange of Product Data
CSC	IT Center for science Ltd.

Mathematical symbols

Greek

ρ	density
τ_{ij}	viscous stress tensor
μ	dynamic viscosity
δ_{ij}	Kronecker delta
λ	thermal conductivity
μ_{ref}	reference dynamic viscosity at reference temperature
γ	adiabatic index
ω	specific rate of dissipation
ε	rate of dissipation

μ_t	turbulent eddy viscosity
v	kinematic viscosity
κ	the von Karman constant
η	porosity
κ_{ij}	permeability tensor
κ_{1ij}	inertial permeability tensor
α	under-relaxation factor

Roman

t	time
x	spatial coordinate
u	velocity
p	pressure
f_i	external force
e	internal energy
T	temperature
S_E	energy source term
S_{ij}	strain rate tensor
V	volume
n	amount of moles
R_u	universal gas constant
h	enthalpy
c_p	specific heat in constant pressure
T_{ref}	reference temperature
T_s	the Sutherland's constant
Ma	the Mach number
c	local speed of sound
R	specific gas constant
q_h	heat flux
k	turbulent kinetic energy
S	strain invariant
F_1	first blending function
F_2	second blending function
P_k	production of turbulent kinetic energy
y^+	dimensionless wall distance
y	distance from the wall

u_t	shear velocity
u^+	dimensionless velocity
S_i	sink term
q_i	the Darcy flux
D_{ij}	inverse of the permeability tensor
F_{ij}	inverse of the inertial permeability tensor
Co	the Courant number
U	local cell velocity
Δt	time step
Δx	distance between cell centres
F	total force
M	total moment
$\mathrm{U_z}$	axisymmetric swirl component

Subscripts, superscripts and oversymbols

Q_o	Value of Q in stagnated flow
$\overline{\mathrm{Q}}$	mean value
Q'	fluctuating value
$\tilde{Q}$	Favre averaged mean value
Q''	Favre averaged fluctuating value

1. INTRODUCTION

Purpose of this thesis is to study down-the-hole (DTH), also known as in-the-hole (ITH), pneumatic hammer physics and operation with computational fluid dynamics (CFD) simulation to provide a new approach to DTH hammer design process. Rigid body movement with overset grid is used to analyse translation of the piston inside the hammer. The use of overset grid allows to simulate arbitrarily large undetermined piston movements induced by pressure forces. With other moving mesh techniques, like mesh morphing and remeshing, only small displacements are possible. Rigid body movement with overset grid is traditionally used to study ship physics, but in this thesis it proves to be applicable to pneumatic applications also. Opensource software OpenFOAM v1712 was chosen for its rigid body and overset implementations, free cost and possibility for massive parallelisation.

Even though first DTH hammers were developed in 1950's [1], little is known what actually happens inside the hammer during drilling. Hammer performance can be analysed by measuring certain drilling parameters and hammer outputs such as rate of penetration and impact frequency. However, hammer performance is dictated by drilling conditions, like rock hardness and amount of ground water in the drill hole, which makes the repeatability of measurements hard in different locations. Most importantly, data from inside the hammer is impossible to get with traditional measurement methods. This lack of information introduces substantial amount of trial and error to the design process. The possibility to analyse pressures, temperatures, flow velocities and other fluid flow phenomenon inside the hammer would be beneficial for optimising existing products and potentially developing entirely new ones. Well drilling DTH hammer studied in this thesis is illustrated in the figure 1.

Figure 1. *Well drilling down-the-hole hammer, from* [2]

There are analytical and semi-analytical design and calculation tools based of system of thermodynamic differential equations, but they cannot consider the effect complex geometries and fluid flow phenomena such as turbulence and transonic flow. Most importantly, they do not work as intended for well drilling hammers.

Using CFD to get information of the hammer operation is a relatively new approach in drilling equipment industry. No CFD simulations of pneumatic devices could be found using rigid body dynamics or overset grid. This is due to high computational requirements, immaturity of the overset technique and complexity of modelling high-speed, high-pressure air. However, CFD simulation using moving overset grid could offer time accurate transient data that is impossible to get without relative motion. Two distinct methods are used to simulate DTH hammer operation.

In chapter 2, a general oversight to the DTH drilling and DTH hammer is given. Two other drilling methods are also presented to understand DTH drilling properly. Also, studied well drilling hammer and its components are presented. Lastly, piston movement inside the hammer based on geometric timings is illustrated.

In chapter 3, mathematical background of this thesis is shown. Balance and thermodynamic equations are presented first followed by theory of turbulence and porous media.

In chapter 4, numerical methods of this thesis are investigated. First, pressure-velocity coupling algorithms are introduced followed with rigid body dynamics. Lastly, overset method and overset grid assembly algorithm are illustrated in detail.

In chapter 5, OpenFOAM is introduced and setup for the first simulation approach using high-resolution grid is performed. This includes pre-processing and specifying numerical settings for the simulation.

In chapter 6, several trial simulations have been performed using various techniques. However, high-resolution grid approach was not successful because the overset grid could not be made robust in parallel computation in the clearances between the hammer and the piston.

In chapter 7, porous zones were implemented to the simulation to bypass previous problems caused by the overset grid. First, grid is modified, and numerical settings are adjusted. Verification test for the porous media simulation approach is done.

In chapter 8, results from simulation using porous zones are presented and discussed. In chapter 9 conclusions are made and future studies are suggested.

2. DOWN-THE-HOLE DRILLING

There are several drilling methods available for rock and overburden which are suited for different kind of drilling conditions and applications. Three main applications for drilling in general are blast hole drilling, well drilling and construction applications like pile-driving. Most common drilling methods for these applications are down-the-hole, top hammer and rotary drilling. Quick insight to top hammer and rotary drilling are given to understand DTH drilling properly. Three most common drilling methods are shown in the figure 2.

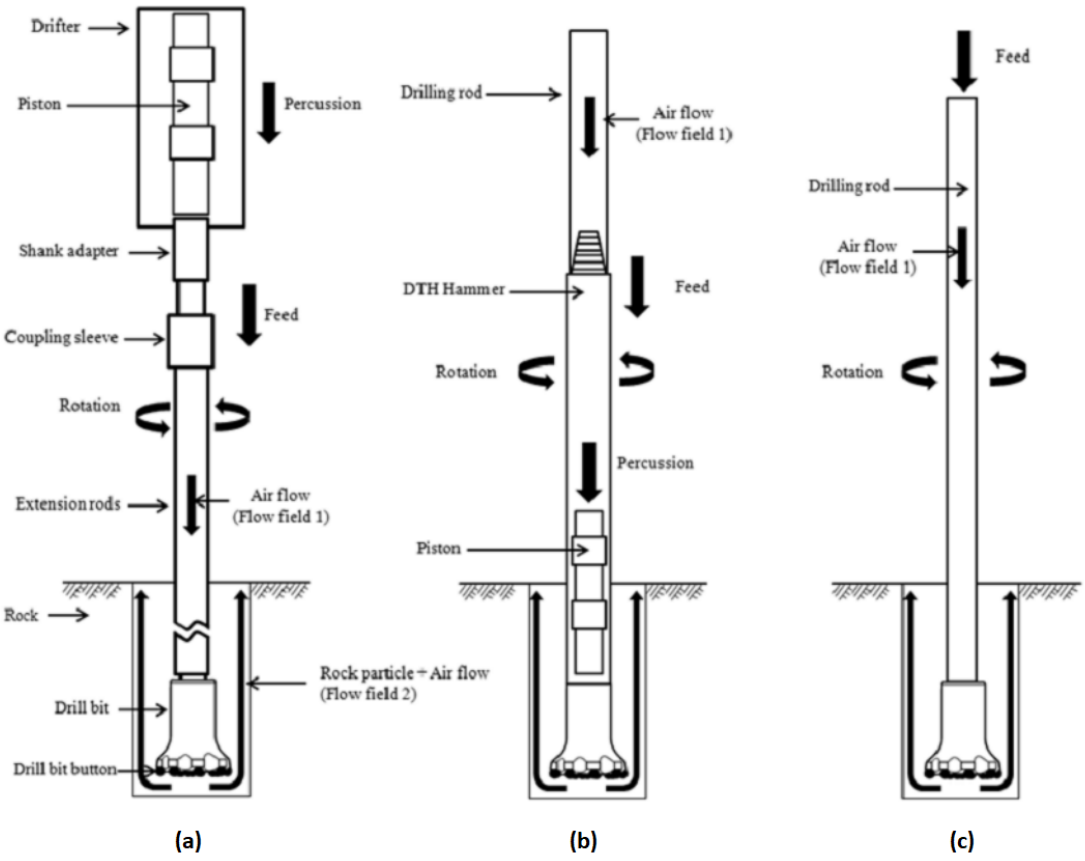

Figure 2. *Most common drilling methods (a) top hammer (b) down-the-hole (c) rotary, from* [3]

In percussive top hammer drilling hydraulic hammer attached to the drill rig is used to impact the drill string, which is assembled from multiple threaded rods. Impact, which is used to break rock in percussive manner, is transmitted as a stress wave travelling through the drill string to the drill bit. Because of the difference in cross section in threaded couplings, portion of energy is lost and dissipated as heat as the stress wave goes through the drill string. Energy lost at couplings varies from 3% to 10% depending on the coupling type [4][5]. Therefore, top hammer drilling can be used to drill holes effectively only up

to 30 m. Also, top hammer drilling is applicable only for small holes up to 115 mm when excluding reaming applications. Top hammer drilling is a good choice for drilling multiple small shallow holes in hard rock [6].

In rotary drilling a large feed force and torque is used to transmit energy trough the drill pipe. This drilling method crushes rock with shearing and not with percussive impact as in top hammer drilling. Rotary drilling requires heavy drill rig to provide sufficient thrust and does not perform well in hard drilling conditions [6]. It is most suitable for drilling large deep holes in soft ground.

Down-the-hole drilling is also a percussive drilling method like top hammer drilling. But instead of using the impact device on surface, hammer is placed on top of the drill bit. In this way energy is not lost in couplings and holes can be drilled to great depths. Typically, compressed air is used to move the piston of the DTH hammer but also water operated hammers exist. Downside of using air as a working medium is the high compressibility. Only about 10% of the energy utilized to compress air can be used to create piston motion inside the hammer [1]. Because of this, compressors needed to operate DTH hammers must be powerful. Down-the-hole drilling is not as fast in drilling multiple shallow holes in hard rock as top hammer drilling and it is slower in soft overburden than rotary drilling [6]. However, it offers great all-around performance in various drilling conditions with various hole sizes. DTH drilling is excellent for deep and straight holes. It is mostly used in drilling blast holes in quarries and mines, drilling water wells and geothermal wells and in construction and applications. DTH drill rig used in blast hole drilling is presented in the figure 3.

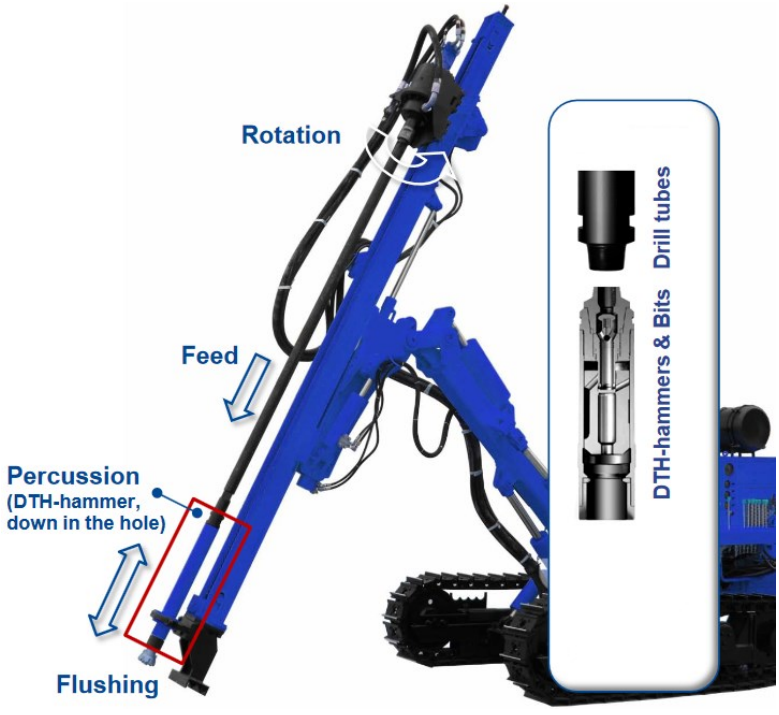

Figure 3. *DTH drill rig, from* [7]

DTH hammer is operated with drill rig and one or more air compressors. Sufficient feed force and rotation is applied from the drilling rig to drill tubes which are connected to DTH hammer. Compressed air is guided through the drill tubes to commence hammer piston cycle and to flush crushed cuttings out of the hole. For well drilling simpler rigs are typically used. Drill rig can be attached to a truck with a compressor or it can be a small movable rig illustrated in the figure 4.

Figure 4. *Well drilling rig* [8]

A well drilling DTH hammer inspected in this thesis is a high-pressure, high impact rate hammer designed to drill deep holes for water and geothermal wells in water filled ground conditions. Well drilling hammers are typically characterized by inner cylinders and relatively simple piston construction compared to conventional hammers like blast hole drilling and construction hammers. In conventional hammers air is typically guided through machined holes in the piston and inner cylinder is not needed. With these hammers, more energy can be transmitted with one blow as the surface area where the pressure accumulates under the piston is larger than in well hammers. Conventional hammers work well when drilling in dry conditions as rock is crushed more efficiently with powerful blows than with high impact rate. A well drilling hammer is compared to a conventional hammer in the figure 5..

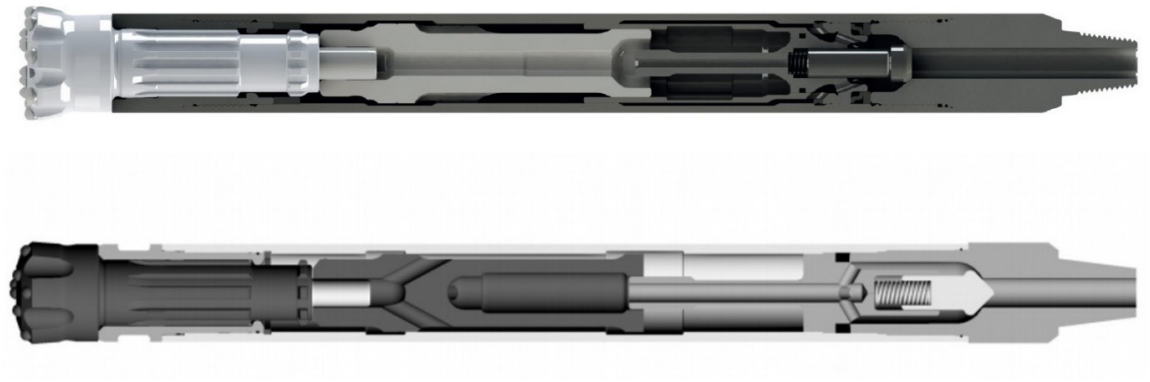

Figure 5. *Well hammer (top) and conventional hammer (bottom), from* [2][9]

Conventional hammers cannot evacuate water properly because the holes in the piston are constraining for water flow. Construction of well hammers is superior for water evacuation. Also, high pressure used in well hammers is desired when drilling deep holes since it compensates back pressure forming in the drill hole from air, water and cuttings. Well drilling hammer components are shown in table 1 and in the figure 6.

Table 1. *Well drilling hammer components* [2]

Part number	Part name	Part number	Part name
1	Air feed tube	9	Make-up ring
2	Aligner	10	Piston
3	Backhead	11	Seating ring
4	Bit retaining ring	12	Wear sleeve
5	Check valve	13	Air feed tube
6	Chuck	14	Backhead 'O'-ring
7	Inner cylinder	15	Seating 'O'-ring
8	Lock ring	16	Spring

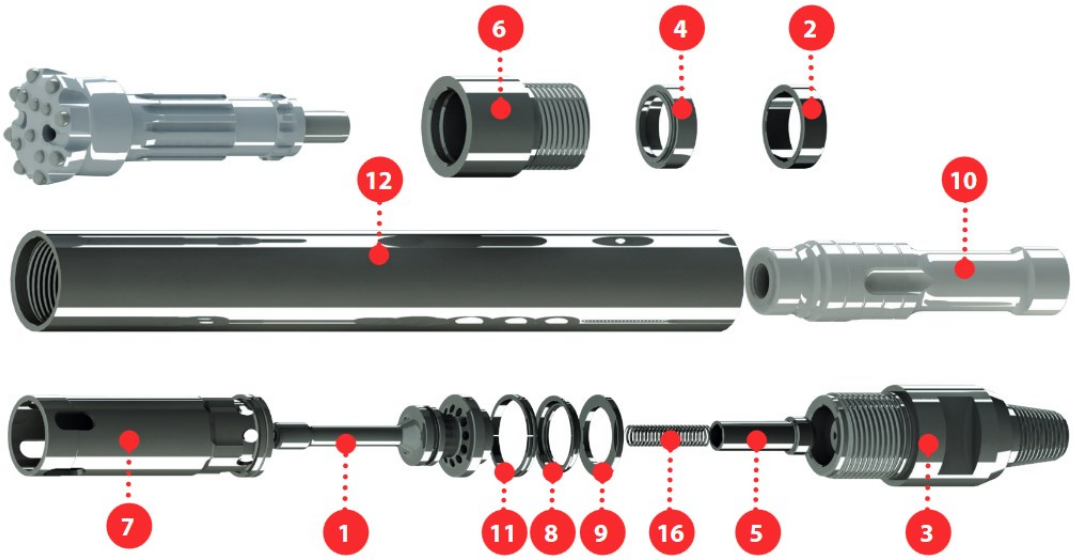

Figure 6. *Well drilling hammer components with drill bit* [2]

DTH hammer operation is characterized by piston cycle. To commence this cycle inside the hammer, compressed air is transmitted to the hammer from the backhead through the drill tubes. As air fills the hammer, pressure forces lift the piston due to differences in piston surface areas. Pressurized air has different functionality depending on the piston location inside the hammer. These different parts of the piston cycle are controlled with timings, which are geometric constraints of flow. Timings control where and how the flow is directed.

If the chuck of the hammer is not in contact with the drill bit, then all the air from the compressor goes straight through the piston and hammer is in flushing mode. Direct flushing is needed when there is accumulated water in the hole and it needs to be evacuated. When feed force is applied from a drill rig, chuck and bit make contact and cycle begins. First, air goes through the inner cylinder holes and from the sides of the piston to fill the chamber under the piston nose. Foot valve attached to the drill bit forms a seal by blocking the air from leaving trough the drill bit. As the chamber under the piston pressurizes, pressure forces accelerate piston upwards. This acceleration continues until the side channels are blocked by the piston. By this point piston has significant upward velocity. High-pressure air from the bottom chamber is released when the piston nose moves past the foot valve. Final deceleration to piston happens in the upper pressure chamber between the inner cylinder and air feed tube. As the piston moves upward, air is compressed in the top chamber and the high-pressure air acts like a spring giving piston downwards momentum. Lastly, piston impacts the drill bit. Most of the impact energy is transmitted to ground as a stress wave but some of it is transformed into piston rebound momentum. After impact, cycle starts again. Simplified 2D presentation showing the piston cycle with inner volume of the hammer is presented in the figure 7.

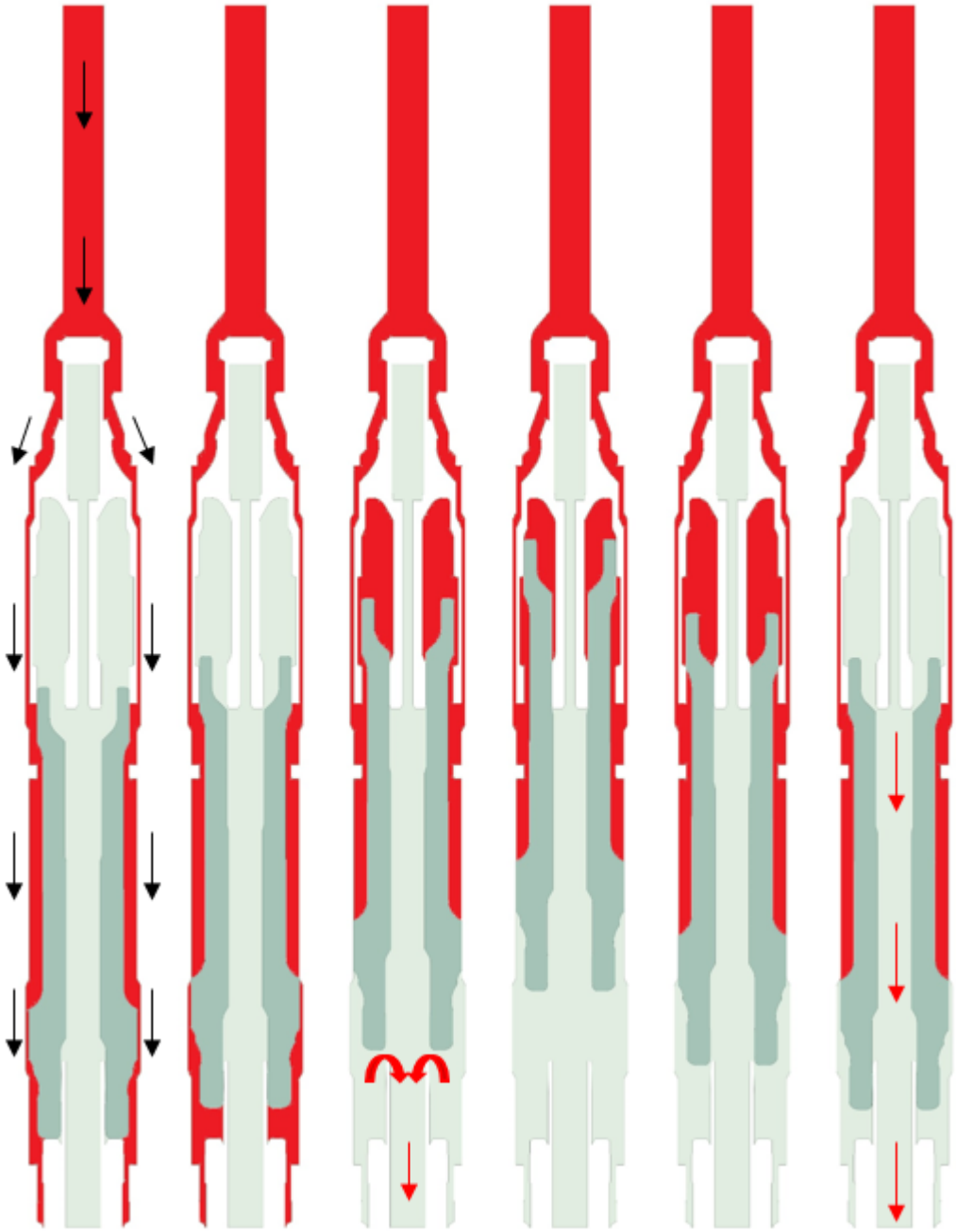

Figure 7. *Piston cycle excluding flushing. High-pressure air is marked with red, atmospheric air with light green and piston with dark green. Black arrows indicate the initial flow direction and red arrows the discharge of high-pressure air.*

3. MATHEMATICAL MODELS

3.1 Balance equations

Computational fluid dynamics is based on fundamental balance equations. Three principal conservation equations are continuity equation 1, momentum equation 2 and energy equation 3 [10]. Assumptions are that that fluid is homogenous, chemically non-reactive and Newtonian. Mass, momentum and energy balance equations can be expressed in the local form as

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho_i}{\partial x_i} = 0, \tag{1}$$

$$\left(\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_j u_i}{\partial x_j}\right) = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho f_i, \tag{2}$$

$$\left(\frac{\partial \rho e}{\partial t} + \frac{\partial \rho u_j e}{\partial x_j}\right) = -\frac{\partial p u_j}{\partial x_j} + \frac{\partial u_i \tau_{ji}}{\partial x_j} + \frac{\partial}{\partial x_j}\left(\lambda \frac{\partial T}{\partial x_j}\right) + S_E, \tag{3}$$

where ρ is the density, t is time, x is a spatial coordinate, u is the flow velocity, p is the pressure, f_i is a body force per unit mass applied on the fluid, e is the internal energy of the fluid, λ is the thermal conductivity, T is the temperature of the fluid, S_E is an energy source term and the viscous stress tensor is

$$\tau_{ij} = 2\mu S_{ij} - \frac{2}{3}\mu \frac{\partial u_k}{\partial x_k}\delta_{ij}, \tag{4}$$

where the strain rate tensor is

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right). \tag{5}$$

3.2 Thermodynamic equations

To relate pressure to density in compressible medium, state equation should be applied. In most cases in aerodynamic and pneumatic applications fluid can be considered as a calorically perfect gas where intermolecular effects are neglected, and heat capacity is assumed to be constant and it can be expressed with the ideal gas equation

$$pV = nR_u T, \tag{6}$$

where V is the volume of the fluid, n is the number of moles and R_u is the universal gas constant. These approximations are relatively accurate even in high-compression. Enthalpy is the total heat content of the system and it is

$$h = c_p T, \tag{7}$$

where c_p is the specific heat in constant pressure. Relation between dynamic viscosity and temperature can be expressed with Sutherland's law for air [11]

$$\mu = \mu_{ref} \left(\frac{T}{T_{ref}} \right)^{\frac{3}{2}} \frac{T_{ref} + T_s}{T + T_s}, \tag{8}$$

where the reference temperature is $\mathrm{T_{ref}} = 273.15$ K, reference dynamic viscosity in the reference temperature is $\mu_{\mathrm{ref}} = 1{,}716 \cdot 10^{-5} (\mathrm{kgm/ms})$ and the Sutherland's constant is T_s = 110.4 K.

In high-speed compressible flows, imposed pressure forces cause relative volume changes in fluid element and therefore significant density changes and gradients arise. It is useful to characterize compressible flows with the Mach number, which is the ratio of the local flow velocity to the local speed of sound

$$\mathrm{Ma} = \frac{u}{c}, \tag{9}$$

where the speed of sound is defined as a speed at which an infinitesimally small pressure wave travels through a medium. For an ideal gas it can be expressed as

$$c = \sqrt{\gamma R T}, \tag{10}$$

where γ is the adiabatic index and R is the specific gas constant. If Ma < 0.3, then flow can be assumed to be incompressible. Otherwise flow can be characterized as subsonic when Ma < 1, supersonic when Ma > 1, hypersonic when Ma >> 1 and transonic when Ma ≅ 1 [12].

When dealing with compressible flows it is convenient to combine the enthalpy and the kinetic energy of the fluid at a stagnation point into a single term called stagnation enthalpy

$$h_0 = h + \frac{u^2}{2}, \tag{11}$$

and to express stagnation temperature at the stagnation point as

$$T_0 = T + \frac{u^2}{2c_p}. \tag{12}$$

Stagnation temperature is the temperature of an ideal gas when it is brought to rest adiabatically. Stagnation properties for pressure and density are related to static properties by

$$\frac{P_0}{P} = \left(\frac{T_0}{T}\right)^{\frac{\gamma}{\gamma-1}}, \tag{13}$$

$$\frac{\rho_0}{\rho} = \left(\frac{T_0}{T}\right)^{\frac{\gamma}{\gamma-1}}. \tag{14}$$

3.3 Turbulence modelling

Turbulent flow is characterized by chaotic fluctuations and modelling of turbulence still presents a significant problem. It is possible to simulate turbulence directly with direct numerical simulation (DNS) or to simulate large scale fluctuations with large eddy simulation (LES). However, both methods are computationally expensive and rarely suitable for industrial applications. Practical approach to turbulence modelling is to use Reynolds-averaged Navier-Stokes (RANS) assumption, or shortly Reynolds averaging.

3.3.1 Reynolds and Favre averaging

The basic idea of Reynolds averaging is to decompose flow into mean and fluctuating parts [10]. The governing equations are then solved for mean values, which are the most important in engineering applications. Decomposed flow variables are

$$\phi = \bar{\phi} + \phi'. \tag{15}$$

Three different forms of the Reynolds averaging are time averaging, spatial averaging and ensemble averaging. Time averaging is appropriate for stationary turbulence and spatial averaging for homogenous turbulence. Ensemble averaging is the most general one and best suited for transient turbulence and it can be expressed as

$$\bar{\phi} = \lim_{N\to\infty} \frac{1}{N} \sum_{m=1}^{N} \phi\,. \tag{16}$$

For compressible flows density weighted Favre-averaging is used in conjunction with Reynolds averaging. Otherwise, the averaged governing equations would become considerably more complicated as the density fluctuations had to be considered. In Favre-averaging flow variables are decomposed to mean and fluctuating parts as

$$\theta = \tilde{\theta} + \theta'', \tag{17}$$

where $\tilde{\theta} = \overline{\rho\theta}/\bar{\rho}$. Therefore, θ'' includes turbulent and density fluctuations. Most convenient way is to use Reynolds averaging for density and pressure, and Favre averaging

for other flow variables such as velocity, enthalpy and temperature. Balance equations for turbulent flow with Reynolds and Favre averaging are

$$\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial \bar{\rho}\tilde{u}_i}{\partial x_i} = 0, \tag{18}$$

$$\frac{\partial \bar{\rho}\tilde{u}_i}{\partial t} + \frac{\partial \bar{\rho}\tilde{u}_j\tilde{u}_i}{\partial x_j} = -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\bar{\tau}_{ij} - \overline{\rho u_i'' u_j''}\right), \tag{19}$$

$$\frac{\partial \bar{\rho}\tilde{e}}{\partial t} + \frac{\partial \bar{\rho}\tilde{u}_j\tilde{e}}{\partial x_j} = -\frac{\partial \bar{p}\tilde{u}_j}{\partial x_j} + \frac{\partial \overline{u_\iota \tau_{j\iota}}}{\partial x_j} - \frac{\partial \bar{q}_h}{\partial x_j} - \frac{\partial \overline{u_j'' p}}{\partial x_j} - \frac{\partial \overline{\rho u_j'' e''}}{\partial x_j}, \tag{20}$$

where $\bar{q}_h$ is an averaged heat flux.

3.3.2 k-ω SST turbulence model

The k-ω SST turbulence model combines the positive features of the k-ω model with a high Reynolds number k-ε model [13]. The k-ω model is implemented in the viscous sublayer and in the logarithmic sublayer of the boundary layer. The k-ω model does not need damping functions and because of this, it is much more stable and as accurate in the viscous sublayer as the k-ε model [10]. In the logarithmic part of the boundary layer the k-ω model is superior to the k-ε model for adverse pressure and compressible flows. The k-ε model is used in the wake region of the boundary layer, as the k-ω model is sensitive to free stream values of ω, and in free shear layers as it well represents wakes, jets and mixing layers. The k-ω SST model also has a modified eddy-viscosity function. This function improves the accuracy of solving flows with strong adverse pressure gradients and pressure-induced boundary layer separations. Modified version of the k-ω SST model is often used where the turbulent viscosity is defined with strain invariant rather than with vorticity [14]. Transport equations for the modified k-ω SST model for k and ω are

$$\frac{\partial \rho k}{\partial t} + \frac{\partial \rho k u_i}{\partial x_i} = \check{P}_k - \beta^* \rho k \omega + \frac{\partial}{\partial x_i}\left[(\mu + \mu_t \sigma_k)\frac{\partial k}{\partial x_i}\right], \tag{21}$$

$$\frac{\partial \rho \omega}{\partial t} + \frac{\partial \rho \omega u_i}{\partial x_i} = \alpha_{SST}\frac{\rho \check{P}_k}{\mu_t} - \beta \rho \omega^2 + \frac{\partial}{\partial x_i}\left[(\mu + \mu_t \sigma_{\omega 1})\frac{\partial \omega}{\partial x_i}\right]$$

$$+2(1 - F_1)\frac{\rho \sigma_{\omega 2}}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i}, \tag{22}$$

where k is the turbulent kinetic energy, ω is the specific rate of dissipation, F_1 is a first blending function and $\alpha_{SST} = \alpha_1 F_1 + \alpha_2(1 - F_1)$. Turbulent eddy viscosity is

$$\mu_t = \frac{\rho a_1 k}{\max(a_1 \omega, S F_2)}, \tag{23}$$

where F_2 is a second blending function and $S = \sqrt{2S_{ij}S_{ij}}$ is the strain invariant. The definition of the turbulent viscosity guarantees that in adverse pressure gradient boundary layer the Bradshaw's assumption, which states that shear stress is proportional to turbulent kinetic energy, is satisfied [10]. A limiter to the production of the turbulent kinetic energy is used in the k-ω SST model to prevent the build-up of turbulence in stagnation regions

$$P_k = \mu_t \frac{\partial u_i}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{24}$$

$$\check{P}_k = \min(P_k, 10 \cdot \beta^* \rho k \omega)\,. \tag{25}$$

Coefficients for the k-ω SST model used in the equations (21-25) are presented in table 2.

Table 2. *k-ω SST coefficients* ***[15]***

α_1	α_2	$\sigma_{\omega 1}$	$\sigma_{\omega 2}$	β_1	β_2	β^*	a_1
0.85	1.0	0.5	0.856	0.075	0.0828	0.09	0.31

K-ω SST models has some drawbacks, but they are small compared to other turbulence models. First, low Reynolds number k-ω model requires first cell from the wall to be in the viscous sub-layer where viscous shear dominates turbulent behaviour and the shear stress of the fluid can be assumed to be equal to the wall shear stress. Dimensionless distance from the wall can be calculated with

$$y^+ = \frac{y u_t}{\upsilon}, \tag{26}$$

where y is the distance from wall, u_t is the shear velocity and υ is the kinematic viscosity. In viscous sub-layer velocity profile exhibits linear relation to the wall distance. To first cell to be in the viscous sub-layer, requirement for y^+ must satisfy at least $y^+ < 5$ and preferably $y^+ \approx 1$. This requirement can lead to excessive cell amount especially in large 3D simulations. However, an alternative to modelling viscous sub-layer is to place first cell in the logarithmic region where velocity profile exhibits a logarithmic variation to the wall distance. In logarithmic layer dimensionless distance is $30 < y^+ < 300$. In logarithmic layer, wall functions can be used to include the viscous effects of the viscous sub-layer.

A good choice in computational domains where y^+ cannot be kept relatively constant in the logarithmic layer is to use Launder-Spalding wall function [16]

$$y^+ = u^+ + \frac{1}{E}\left[e^{\kappa u^+} - 1 - \kappa u^+ - 0{,}5(\kappa u^+)^2 - \frac{1}{6}(\kappa u^+)^3\right], \quad (27)$$

where u^+ is the non-dimensional velocity and κ is the von Karman constant. It is continuous, and it works in viscous sublayer, in transition region where $5 < y^+ < 30$ and in logarithmic region. In transition region blending functions are implemented. Launder-Spalding wall function provides a velocity-based condition for turbulent kinematic viscosity.

3.4 Porous media

Flow in the porous media is modelled by attenuating the time derivative and adding a sink term to the momentum equation [17]. Modified momentum equation in porous media is

$$\frac{\partial \eta \rho u_i}{\partial t} + \frac{\partial \rho u_j u_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho f_i + S_i, \quad (28)$$

where η is the porosity and sink term is

$$S_i = -\left(\mu D_{ij} + \rho\sqrt{q_k q_k} F_{ij}\right) q_j, \quad (29)$$

where D_{ij} is the inverse of the permeability tensor κ_{ij}, F_{ij} is the inverse of the inertial permeability tensor κ_{1ij} and $q_i = \eta u_i$ is the Darcy flux. Equation 29 is called the Darcy-Forchheimer law. Porous media creates a pressure drop that is proportional to velocity for the viscous loss and velocity squared for the inertial loss. Consideration of non-linear inertial loss is significant only if the velocity of the flow in porous media is high [18].

4. NUMERICAL METHODS

4.1 Pressure-velocity coupling

Compressible flow problems require solving of momentum, pressure and energy equations. To this system of equations to be solvable, same amount of equations and unknown variables is necessary. However, there is a linear coupling between pressure and velocity because velocity divergence appears in the pressure equation and pressure gradient is present in the momentum equation. This coupling can be solved with algorithms like SIMPLE, PISO and PIMPLE [20]. While all the algorithms solve the same governing equations, algorithms differ how they iterate the equations.

SIMPLE (Semi-Implicit-Method-Of-Pressure-Linked-Equations) is the oldest pressure-momentum algorithm and it is used for steady state analyses. Time derivation is not implemented in SIMPLE and it is not consistent due to missing pressure term. Because of the lack of time derivative, and therefore physical time step, there is no limiter to the solution. This means that the equations should be under-relaxed to prevent solution instability. Under-relaxation can be presented as

$$\phi_{n+1} = \alpha\phi_{n+1} + (1-\alpha)\phi_n, \tag{30}$$

where ϕ_n is the value of previous approximation, ϕ_{n+1} is the value of new approximation and α is the under-relaxation factor. Pressure is under-relaxed explicitly and velocity implicitly. Under-relaxation slows down convergence speed and therefore, a good estimation for under-relaxation is important for optimal convergence and stability.

PISO (Pressure-Implicit-of-Split-Operations) algorithm is used in transient simulations where temporal accuracy is important. Time derivative is included in equations and pressure-momentum coupling is consistent. Under-relaxation is not needed in PISO coupling but stability must be fulfilled by small enough Courant number, which is a dimensionless number that contains information how fast the information travels through cells. The Courant number is

$$\mathrm{Co} = \frac{U\Delta t}{\Delta x}, \tag{31}$$

where U is the local cell velocity, Δt is time step and Δx is the distance between cell centres. Typically, the Courant number should not be larger than one. If Co is less than one, information from one cell can reach only the next neighbour cell within a single time step. Otherwise, information could reach second or even further neighbouring cells which is not allowed due to explicit velocity field update.

PIMPLE (Merged PISO-SIMPLE) is a combination of PISO and SIMPLE algorithms and it is best suited for transient simulations with complex geometries and non-optimal mesh quality. PIMPLE algorithm solves a steady state solution with under-relaxation in explicit PISO outer corrector loops. PIMPLE can be used with much larger Courant number (Co >> 1) than PISO. However, transient time scales in the simulation must be considered. Using too large Co causes essential information to be neglected which affects solution accuracy.

4.2 Temporal discretization

In transient flow simulations temporal derivative term determines how the solution is updated in time. For velocity, temporal derivative is

$$\frac{\partial \phi}{\partial t} = f(\phi, u). \tag{32}$$

Simplest discretization for temporal derivative is to use first order explicit forward Euler method

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = f(\phi^n). \tag{33}$$

However, explicit discretization is rarely practical in CFD since it does not perform well with stiff differential systems. Also, the stability of explicit methods is controlled with Co limitation. Because of this, implicit methods are typically used. First order accurate implicit temporal discretization is called backward Euler method and it is given by

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = f(\phi^{n+1}). \tag{34}$$

In backward Euler method, current and previous time steps are considered. It is robust, bounded, conservative in time and stable. However, it is only first order accurate and it will introduce noticeable discretization errors which will diffuse steep temporal gradients.

Second order discretization methods are more accurate than first order ones, but they can exhibit unwanted numerical oscillations. In general, higher-order discretization schemes for temporal derivatives can be used with good quality mesh and with well initialized boundary conditions. For second order accurate discretization the Crank Nicolson method

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} = \frac{1}{2}[f(\phi^{n+1}) + f(\phi^n)], \tag{35}$$

is a good choice [21]. It is a combination of forward Euler and backward Euler methods. However, Crank Nicolson method is not the average of these two first order methods as it has an implicit dependence on the solution. Crank Nicolson method can be off-centered

from 0 to 1 where 0 reduces equation to backward Euler method and 1 corresponds to pure Crank Nicolson method [15] . Using off-centering factor below 1, for example 0.5, reduces accuracy but increases stability and boundedness.

4.3 Rigid body motion

Rigid body motion describes how free or constrained rigid object will translate and rotate according to forces acting on the object. Number of degrees of freedom will refer how the movement of the body limited. If the body can move in all six degrees of freedom, then it is free to translate and rotate along all three orthogonal axes of the three-dimensional system. Movement can be limited by using constraints to remove degrees of freedom. Also, restraints like linear springs, dampers and angular springs can be used. All possible translations and rotations in the three-dimensional space are presented in the figure 8.

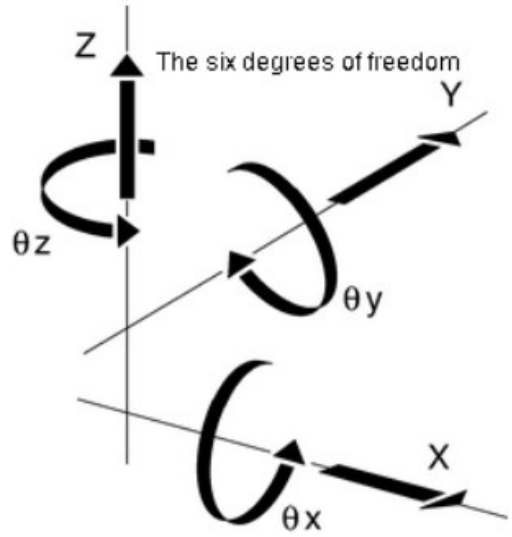

Figure 8. *Axes describing the 6 degrees of freedom, from* [22]

Total forces and moments acting on bodies interacting with the flow field can be calculated by integrating individual forces and moments acting on cells over the bodies with equations

$$F = \int_S \; F_{ext} + F_{flow}\,, \tag{36}$$

$$M = \int_S \; M_{ext} + M_{flow}\,, \tag{37}$$

where F is the total force and M is the total moment about the mass centre. A standard numerical approach for rigid body motion is:

1. Velocity and pressure are solved from balance equations in an initial position of the body. Forces and moments acting to the rigid body are obtained.
2. If the resulting moments and forces are balanced, iteration is stopped.
3. Else: The new position of the body is determined by solving the equations of motion based upon the forces and moments computed at stage 1.
4. Mesh is moved, morphed or remeshed and procedure is started again from 1.

4.4 Overset method

Typically, finite volume CFD solvers use a single continuous mesh to discretize the computational domain. However, this approach raises two problems. Generating a single structured grid for complex domain is challenging and it can be difficult for unstructured grids also. The second problem comes from solving large relative motions of one or multiple bodies in a single grid, which is only capable of handling small motions using mesh morphing or remeshing. Both issues can be alleviated by dividing computational domain into separate overlapping parts, performing Overset Grid Assembly (OGA) and interpolating the solution. This approach was first introduced in the early 1980's [23] and it is usually referred as overset method or Chimera technique due to its combination of grids of separate components. In overset method the information between separate grids is propagated by interpolating point or cell centre data between overlapping regions whereas in single continuous grid data is transferred trough matching face zones.

Overset method simplifies structured mesh generation and increases grid quality for computational domains because background and objects can be meshed separately. When considering rigid or elastic bodies in relative motion, each body can be represented by its own grid. With this approach bodies can move arbitrarily and independently of other bodies if physical boundaries do not collide. Overset method therefore provides a robust support of the full range of motion within the domain. Two overlapping structured grids and a Cartesian background mesh are illustrated in the figure 9.

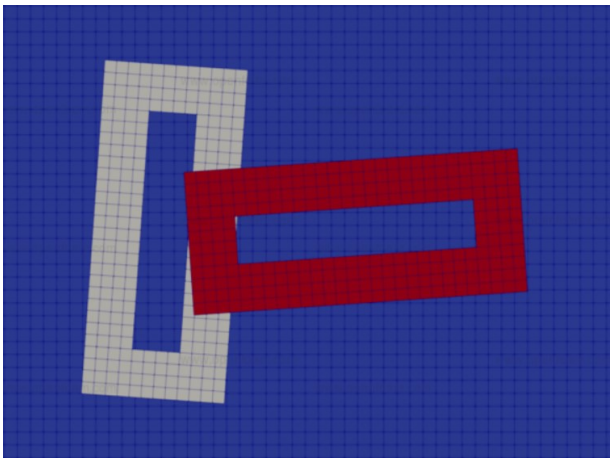

Figure 9. *Two overlapping grids and a background mesh* [24]. *Overlapping grids are represented with grey and red.*

Overset method has advantages particularly in simulating relative motions, but it comes with limitations. This process is computationally intensive and can represent a large portion of total computational cost in transient simulation where the domain must be reassembled after every time step. Also, simulations with overset grids are proven to be hard to parallelize as the computational load can be unevenly divided between processors causing load imbalance [25]. Overset method is also non-conservative which violates the basic principle of finite volume method (FVM). It means that in transient simulations mass is not perfectly conserved. Therefore, effort should be put on minimizing conservation error.

Overset method is based on algorithm called Overset Grid Assembly. The primary task of the OGA is to assign each point one of three following properties:

- Hole point: point where no solution is computed.
- Solve point: point where the solution is computed. Contains a subset called donor cell.
- Receptor point: point where the solution is interpolated from an overlapping cell called donor cell.

Point data refers to cell centre values or vertex data depending on the CFD solver used. In the figure 10 OGA algorithm is performed for the overset grid presented in the figure 9 and each point has been assigned a point property.

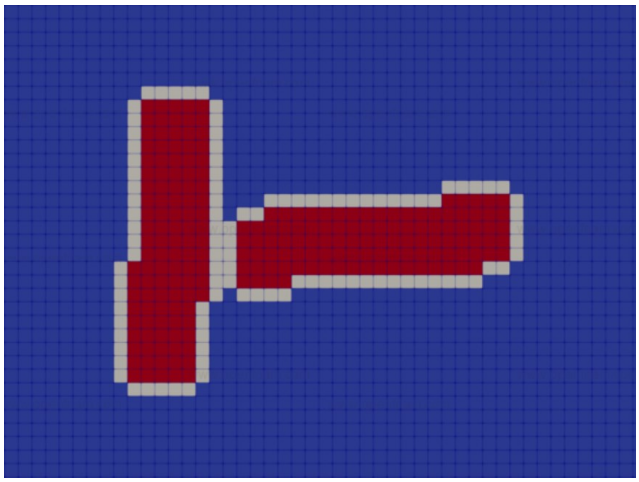

Figure 10. *Overset grid after OGA* [24]. *Blue indicates solve points, grey interpolated receptor points and red hole points.*

OGA algorithm contains three main steps followed by solution interpolation. Emphasis is in parallel algorithms.

4.4.1 Hole-cutting

The basic idea of hole-cutting algorithms is to identify the approximate representation of the overset wall boundary by utilizing cutting surface and to mark all points inside the boundary and outside the computational domain as hole points. In parallel grids, all the cells intersecting the cutting surface generated from the overset wall surface are marked by employing rapid point search algorithms like virtual grid searching algorithm [25] or auxiliary grid algorithm [26]. As in parallel grid the information of the wall boundary is spread among several processors, communication between processors is necessary which is not a trivial problem [26].

After wall boundaries have been marked, a flood-fill algorithm is applied to points inside the wall boundaries. For flood-filling to work, seed cells need to be identified. This can be done with internal or outer algorithms. Outer algorithms like scan-line flood-filling [26] have advantage over internal ones. For complicated geometries, identifying seed cells outside the overset wall boundary is more robust than identifying seed cells inside the object. Also, some frequently used flood-filling algorithms like ray-tracing [27] work only for structured grids.

Hole-cutting can fail for several reasons. If internal flood-fill algorithm is used, complexities in the topology or the geometry of the overset wall boundary or insufficient mesh resolution might cause identification of bad seed cells. These cells are identified in the computational domain instead of outside the domain. This results in flood-filling the domain or complete cell zones with hole points. If flood-filling failure is encountered while using internal flood-fill algorithms, the only way to remedy the problem is to make changes to the model [28]. Hole-cutting can also fail by hole point leakage. This occurs when no bad seed cells are found but when overlapping meshes do not match sufficiently. This causes flood-filling to leak out from inside of the boundary to outside causing holes in the domain. Especially unstructured curved surfaces might create problems.

4.4.2 Donor-search

The principle of donor-search is to determine the donor-receiver pairs in the domain. First, candidate receptor points and candidate donor cells are extracted from the grid. If the overlapping grids do not belong to same processor, the candidate receptor points are sent to process that owns the donor grid. Candidate points and cells are defined as: [25]

- Candidate receptor point: any point that is within a user or program specified distance from the outer boundary or from the hole point boundary defined in the hole cutting process.
- Candidate donor cell: any cell that is within the overlap region that does not have any hole points.

To search every candidate receptor point for a candidate donor cell, a line-walk search algorithm, also known as stencil jumping or stencil walking, is used. After line-walk each candidate receptor point has established whether it has a donor cell or not and then the information is communicated to the processor that owns the candidate receptor point.

Complications in donor-search occur when acceptable donor points cannot be found for interpolation, producing orphan points. These points might occur when overset grids have insufficient overlap or when significant differences in mesh spacing between grids exist [29]. A point is orphan when one or more of its donors is also a mandatory receptor point requiring interpolated solution. Existence of orphan points leads to solution inaccuracy as mandatory receptor points cannot be resolved. With moving mesh, the effect of orphan points is emphasized as relative motion can increase the number of orphans or change their locations over time. Orphan points can be prevented by increasing mesh resolution and improving mesh overlap.

4.4.3 Point-type assignment

Once the donor-search has been performed and processors have exchanged their candidate donor cell information, unidentified points are marked as solve points or receptor points. Hole points have been identified in the hole-cutting process earlier.

First, points in the neighbourhood of hole points are marked as mandatory receptor points, which should always interpolate from another mesh point. This ensures that the solver algorithm will have no invalid points in the discretization stencil of the governing partial differential equations [26].

After identifying mandatory receptor points, solve and receptor points are distinguished from remaining cells. The resolution of a point is compared to the resolution of all its candidate donor cells. If the point resolution is the highest, then the point is marked as solve point. If not, then it is a receptor point, and its donor cell is chosen as the one with the best resolution. As for the last step, processors share the information about accepted and rejected candidate donor cells.

4.4.4 Interpolation

After each point has been assigned a property, interpolation weights for each receptor point associated with a donor cell are computed. After that, solution can be interpolated to the receiver point with a simple linear combination. Interpolated solution is the weighted average of the data points participating in solution.

5. HIGH-RESOLUTION GRID SIMULATION SETUP

For the numerical simulation the OpenFOAM v1712 by ESI Group is used in this thesis. It is a free and open-source C++ toolbox for developing executables, solvers and utilities, that use packaged functionalities contained in libraries. These applications and libraries can be extended and modified by experienced users. OpenFOAM contains numerical solvers and pre-/post-processing utilities for solving CFD problems, but it can also be used for other applications such as electromagnetics and solid mechanics.

OpenFOAM is designed to run in Linux based operating systems and it uses Unix style commands and text file dictionaries. Simulation case is run in main folder containing *time*, *constant* and *system* directories. Directories for time contain initial conditions of used variables and results corresponding to specific time steps. Constant directory has mesh data and physical properties like turbulence and thermodynamic settings. System directory contains settings associated with solution procedure in *controlDict*, *fvSchemes* and in *fvSolution*. Run time parameters are specified in *controlDict*, numerical schemes in *fvSchemes* and linear solver settings in *fvSolution.* Minimum OpenFOAM case structure is shown in the figure 11.

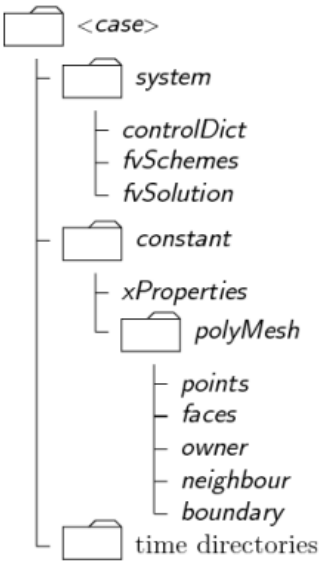

Figure 11. *OpenFOAM case structure, from* [19]

Two separate simulation approaches are used with different grids and numerical settings. First simulation approach uses high-resolution grid in the clearances. As a solver, *overRhoPimpleDyMFoam* is used, which is a transient solver for compressible fluids with overset and dynamic grid implementation. Overset and dynamic setting used in the high-resolution grid simulation are presented in the appendix A and additional solver settings in the appendix B [15].

5.1 Pre-processing

Simulation process is started with creating or modifying an existing CAD geometry to suit the simulation needs. In this case a 3D geometry is simplified to 2D model to greatly reduce computing resource requirements. Spaceclaim is used as a CAD tool because of its flexibility and history-free direct modelling approach.

Meshing is done with opensource meshing tool cfMesh that has been implemented to the OpenFOAM v1712. It has advantages over native OpenFOAM unconstructed meshing tool snappyHexMesh in boundary layer generation and in user friendliness. Grid and geometry are first generated using high-resolution mesh in the clearances between piston and hammer walls.

5.1.1 Geometry modification

As a geometry a STEP file imported from Solidworks is used and modified in Spaceclaim. Model is first simplified to 2D to greatly reduce cell count. Some simplifications are required since the original geometry is not axisymmetric but has cyclic symmetry in sectors. Simplified 2D geometry is presented in the figure 12.

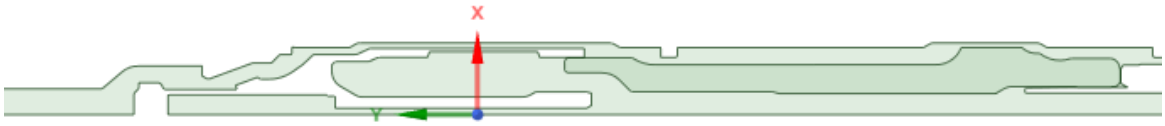

Figure 12. *Axisymmetric simplifications*

Inner cylinder has been modified in two ways. Small air feed holes on top of the inner cylinder have been removed and replaced with a simple channel. Same simplification has been done to the air feed holes on bottom of the inner cylinder. Piston has been changed by removing oil grooves and piston material between air channels. These simplifications will change the flow characteristics of the hammer, but they can be justified as they do not affect cycle timings.

Axisymmetric geometry has also been modified by relaxing the original CAD model to the largest allowed tolerance limit to make clearances as loose as possible. This is beneficial when using overset grid in near wall regions as the cell size in the clearance could be increased. Small local cells impose limitation to the time step size and increase cell count. Also, by using maximum clearance tolerances the geometry represents better the actual hammer as the inner parts of the hammer will wear during operation.

Some additional changes to the piston geometry have been done to make overset grid more robust. Rounded piston corners have been notched to produce more uniform Cartesian mesh in the near wall regions. Next, the overset boundary is created around the piston. Overset boundary should be far enough from the piston that minimum of four

cells can be placed between to ensure robustness of OGA algorithm. As overset grid assembly is time consuming, goal is to minimize overset cell count. Overset boundary created around the piston is presented in the figure 13.

Figure 13. *Overset boundary*

2D meshing in cfMesh uses STL file in ribbon format so the surface geometry file created in Spaceclaim must be modified and patches named accordingly. Also, the moving piston with overset boundary must be meshed separately.

5.1.2 Meshing

Meshing is started with overset piston geometry. Mesh must be refined in the areas that will have close proximity to hammer walls during the piston cycle. Refined mesh from the upper part of the piston is illustrated in the figure 14.

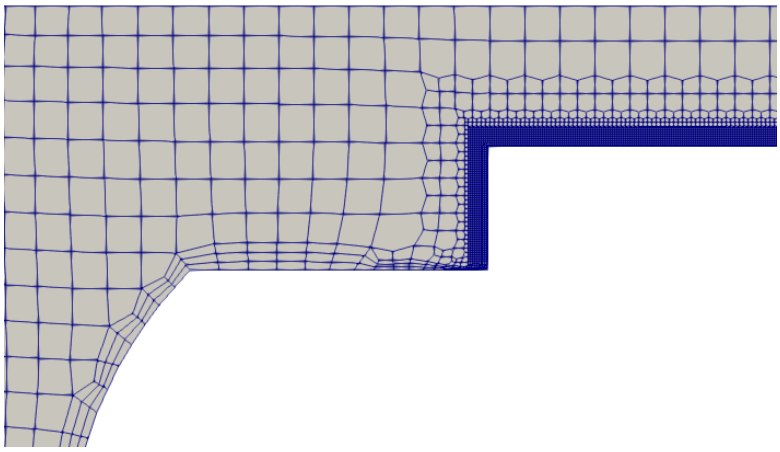

Figure 14. *Refined piston mesh*

Figure 14 shows the notch created on top of the piston as well as high-resolution mesh refinement. Accuracy of the solution will most likely suffer in the regions where transition from large cells to small cells occurs. However, this approach is necessary to create refined enough grid for the clearances and keeping cell count in reasonable limits.

Before the background mesh is generated the alignment of the geometry to the y-symmetry axis must be checked as misalignment creates numerical problems in axisymmetric simulations. Misalignment can be present because of rounding of errors in CAD software or inaccuracies in surface triangulation. It can be treated by manually modifying cfMesh fms geometry file and editing small negligible values to zero.

Background mesh is generated separately, and mesh refinements are placed to regions which have or will have proximity to piston during simulation. Also, additional box refinements are placed in these areas to make dynamic mesh movement more robust. Background mesh with patch and box refinements in the bottom of the inner cylinder is presented in the figure 15.

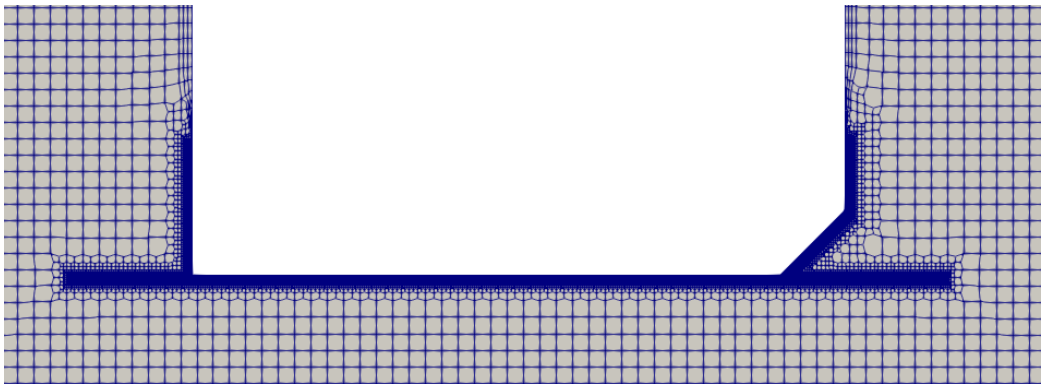

Figure 15. *Mesh refinements in the bottom of the inner cylinder*

Two grids can be combined and then transformed into axisymmetric form by using *wedgePlease* utility [30], which is based on native OpenFOAM utility *flattenMesh*. Best way to simulate axisymmetric geometry is to create a wedge with sector angle of 5 degrees [31]. Because OpenFOAM treats 2D meshes in 3D, wedges smaller than 5 degrees would result in neglectable cell volumes near the symmetry axis creating numerical problems. Angles larger than 5 degrees will cause solution inaccuracy. Axisymmetric mesh compared to original 2D mesh is shown in the figure 16.

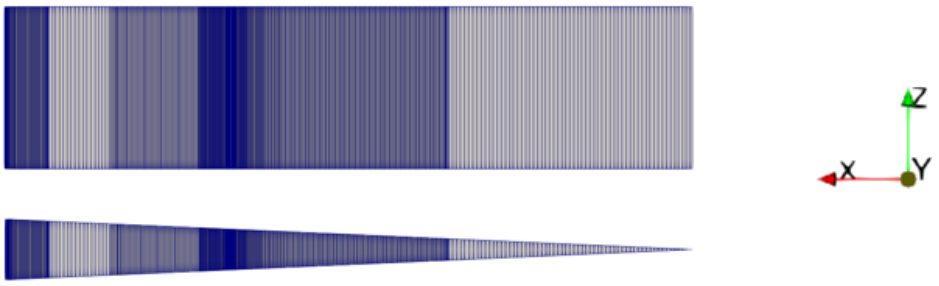

Figure 16. *Original 2D mesh (top) and axisymmetric wedge mesh (bottom)*

Axisymmetric mesh in OpenFOAM does not neglect velocities the in surface normal direction and therefore wedge geometries can produce different results than cyclic simulations, which have cyclic symmetry planes.

Creating an axisymmetric mesh requires collapsing edges on the symmetry axis with *collapseEdges* utility and it is controlled by *collapseDict* dictionary in the *system* folder. This utility merges edges that have been translated from the original 2D mesh and are overlapping. Modifying *minimumEdgeLength* might be necessary if cell size in the domain is small.

```
collapseEdgesCoeffs
{
      minimumEdgeLength 1e-5;
      maximumMergeAngle 179;
}
```

Final step in creating an axisymmetric mesh is to change the front and back boundaries to wedges manually or by using *changeDictionary* utility controlled by *changeDictionaryDict*.

```
boundary
{
      "bottomEmptyFaces.*"
      {
            type            wedge;
            inGroups        (wedge);
      }

      "topEmptyFaces"
      {
            type            wedge;
            inGroups        (wedge);
      }
}
```

Mesh quality is checked for both piston and background grids. Most important mesh metrics for hexahedral dominant meshes are non-orthogonality and aspect ratio. Non-orthogonality is defined as the angular deviation of the surface normal vector from the vector connecting two cell centres. Aspect ratio is the ratio between the longest and the shortest face of the cell. Skewness is also checked but it is typically not a limiting factor in hexahedral dominant meshes. For detailed mesh quality metrics definitions, see [32]. Mesh quality results for both piston and background mesh are presented in table 3.

Table 3. *Mesh quality metrics for piston and background mesh*

	Piston mesh	Background mesh
Cell count	98699	576961
Maximum aspect ratio	8.37	8.57
Maximum non-orthogonality	45.18	53.00
Average non-orthogonality	2.61	1.60
Maximum skewness	1.36	2.33

Maximum aspect ratio and maximum non-orthogonality are in allowed limits. As cell size in the clearances must be small, cell count will be large for 2D simulation and time step must be small. Minimum cell size is 0.015 mm.

5.2 Boundary conditions

Specifying accurate initial values for turbulence is complicated but if the area of interest is not close to inlet, then turbulence will level into physically right values as the flow develops. For initial values, good practice is to use rather large amount of rate of turbulent dissipation at the inlet [33]. This method will introduce some additional turbulent viscos-

ity near the inlet, but it will stabilize computation by filtering out small time scale phenomena. Also, using small placeholder values for turbulence properties other than omega is recommended at the walls. This will reduce the risk of solver crash caused by floating point exception by division with zero.

The k-ω SST turbulence model with Launder-Spalding wall function is used with standard coefficients. Gravity is defined in negative y-direction and Sutherland's law for viscosity is used with assumptions of calorically perfect gas and constant enthalpy. Targeted mass flow rate was chosen to be 0.475 kg/s. For simplicity, physical boundary conditions used in the simulations are presented in tabular form when needed in the following chapters.

5.3 Numerical schemes

In OpenFOAM, numerical settings must be specified by the user and new types of simulations will most likely require adjusting and testing. Choice of numerical schemes has major impact on solution accuracy and stability.

Different numerical schemes were tested. As the mesh has relatively good quality, second order linear upwind was used for most of the divergence terms. It has good accuracy and stability and it is not prone to oscillations. For turbulence properties, linear upwind scheme was first used but it led to instability due to unboundedness and first order bounded upwind scheme was chosen instead. For overset interpolation, generally recommended inverse distance method is chosen [34]. For wall distance calculation, Poisson equation is used since the mesh wave method typically used in dynamic simulations is not supported with overset grid. As for time discretization, off-centered Crank Nicolson 0.5 scheme is chosen for the first part of the simulation where the piston motion is restricted as it is less diffusive than first order backward Euler scheme. However, when pressure has accumulated under the piston and translational movement is allowed, backward Euler scheme is recommended since Crank Nicolson scheme may give non-physical hotspots with moving mesh [34]. Numerical schemes used are presented in the appendix C.

5.4 CSC Taito supercluster

Taito supercluster from Finnish IT center for science is used in this thesis as overset simulations even in 2D require substantial amount of computational power. In Taito, maximum of 28 nodes can be used for single simulation containing total of 672 cores. In this thesis a single node is used with 16 to 24 cores.

6. TRIAL SIMULATIONS

Trial solutions are presented to illustrate different methods used for the simulation with high-resolution mesh in the clearances. The second simulation approach is based on these results.

6.1 Case 1

In the case 1, simulation was tried with specified inlet velocity as inlet mass flow rate was unstable. Inlet velocity was stable only up to 150 m/s which corresponds to mass flow rate of 0.132 kg/s with air in atmospheric density. Boundary conditions for the case 1 are presented in table 4.

Table 4. *Case 1 boundary conditions*

	U (m/s)	T (K)	p (Pa)
internalField	(0 0 0)	300	10^5
inlet	fixedValue (0 -150 0)	fixedValue 300	zeroGradient
outlet	zeroGradient	zeroGradient	fixedValue 10^5
allPistonWalls	movingWallVelocity (0 0 0)	zeroGradient	zeroGradient
allHammerWalls	fixedValue (0 0 0)	zeroGradient	zeroGradient

	nut (m^2/s)	k (m^2/s^2)
internalField	10^-9	0.1
inlet	calculated 0	fixedValue 0.1
outlet	calculated 0	zeroGradient
allPistonWalls	nutUSpaldingWallFunction 10^-9	kqRWallFunction 0.1
allHammerWalls	nutUSpaldingWallFunction 10^-9	kqRWallFunction 0.1

	alphat (kg/(m·s)	omega (1/s)
internalField	0	10^5
inlet	calculated 0	fixedValue 10^5
outlet	zeroGradient	zeroGradient
allPistonWalls	compressible:alphatWallFunction (Prt 0.85 fixedValue 0)	omegaWallFunction 10^5
allHammerWalls	compressible:alphatWallFunction (Prt 0.85 fixedValue 0)	omegaWallFunction 10^5

Simulation is run with specified settings.

```
PIMPLE
{
      momentumPredictor               true;
      transonic                       false;
      nOuterCorrectors                30;
      nCorrectors                     1;
      nNonOrthogonalCorrectors        0;
      oversetAdjustPhi                true;
      turbOnFinalIterOnly             false;
}
```

Using this approach had serious drawbacks. Convergence could only be achieved when using momentum predictor. However, for some reason the axisymmetric swirl component Uz did not converge during the simulation. Coordinate system is illustrated in the figure 12. Unconverged swirl component did not affect the accuracy or the stability of the solution but it prevented the use of residual control. This means that specified amount of outer corrections had to be performed for every time step despite the convergence of the solution. An effort was made to remove solving of Uz by modifying the source code, as it could be assumed that no notable swirl would develop in the DTH hammer cycle. However, removing Uz from incompressible solver was relatively easy but to do so for compressible solver was not successful as the relation of velocity to energy equation was too complex. Also, transonic option could not be used because of convergence issues.

6.2 Case 2

In the case 2 the simulation was initialized by using converged axisymmetric pipe flow solution to obtain correct velocity profile and initial values at the start. By this way the previously unstable mass flow rate inlet condition could be used with correct inlet value. Settings for the case 2 are presented below.

```
PIMPLE
{
      momentumPredictor               false;
      transonic                       true;
      nOuterCorrectors                150;
      nCorrectors                     1;
      nNonOrthogonalCorrectors        0;
      oversetAdjustPhi                true;
      turbOnFinalIterOnly             false;
}
```

By achieving convergence without momentum predictor, the swirl component was not a problem anymore as it was solved in the pressure equation rather than in the momentum predictor. Careful adjustment of turbulent inlet conditions was required to obtain stable solution with transonic option turned on. Boundary conditions are presented in table 5.

Table 5. *Case 2 boundary conditions*

	U (m/s)	T (K)	p (Pa)
internalField	(0 0 0)	300	10^5
inlet	flowRateInletVelocity massFlowRate 0.0066 rhoInlet 2.2	fixedValue 300	zeroGradient
outlet	fluxCorrectedVelocity	zeroGradient	fixedValue 10^5
allPistonWalls	movingWallVelocity (0 0 0)	zeroGradient	zeroGradient
allHammerWalls	fixedValue (0 0 0)	zeroGradient	zeroGradient

	nut (m^2/s)	k (m^2/s^2)
internalField	10^-9	50
inlet	calculated 0	turbulentIntensityKineticEnergyInlet 50, intensity 0.03
outlet	calculated 0	zeroGradient
allPistonWalls	nutUSpaldingWallFunction 10^-9	kqRWallFunction 50
allHammerWalls	nutUSpaldingWallFunction 10^-9	kqRWallFunction 50

	alphat (kg/(m·s)	omega (1/s)
internalField	0	3·10^6
inlet	calculated 0	fixedValue 10^4
outlet	zeroGradient	zeroGradient
allPistonWalls	compressible:alphatWallFunction (Prt 0.85 fixedValue 0)	omegaWallFunction 3·10^6
allHammerWalls	compressible:alphatWallFunction (Prt 0.85 fixedValue 0)	omegaWallFunction 3·10^6

The easiest way to initialize the flow near the inlet would have been to use steady state solver *rhoSimpleFoam*. However, *rhoSimpleFoam* was proven to be unstable with high velocity and transient solver *rhoPimpleFoam* was used instead. Velocity magnitude results for pipe flow are shown in the figure 17.

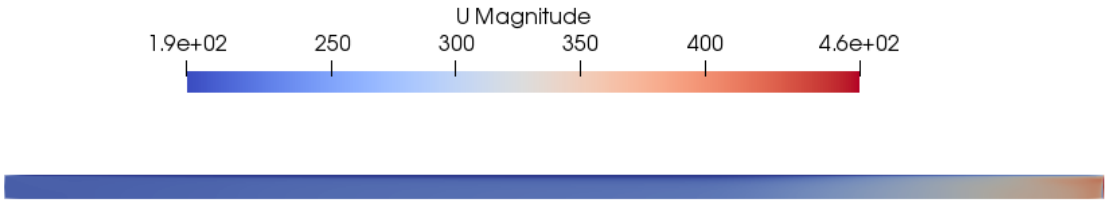

Figure 17. *Pipe flow with 0.475 kg/s mass flow rate (m/s)*

However, the velocity in the pipe was unphysical since the flow is supersonic at the last cells next to outlet. This unphysical behaviour can be avoided by mapping the solution to a slightly shorter pipe illustrated in the figure 18 and using that in initialization.

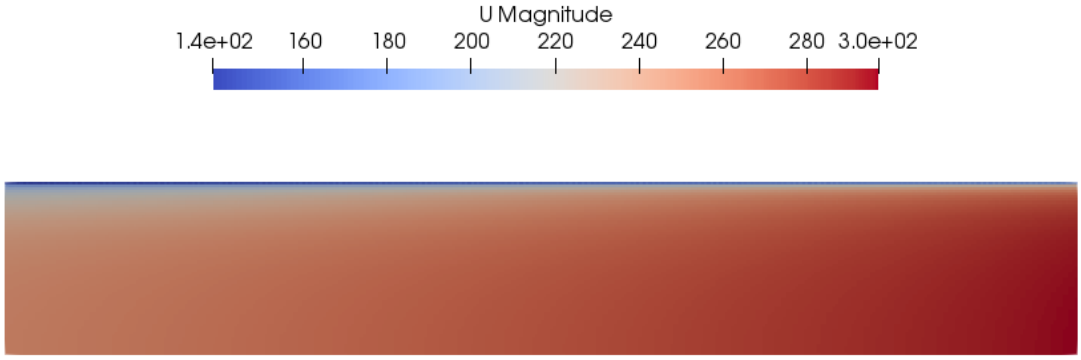

Figure 18. *Pipe flow result used in initialization (m/s)*

In the case 2 the simulation was run with stationary piston until the pressure chamber below the piston nose was filled. Simulation was stable with stationary mesh but temperature in the hammer was unphysically high. Temperatures in different part of the hammer at time t = 2 ms are presented in the figures 19 and 20.

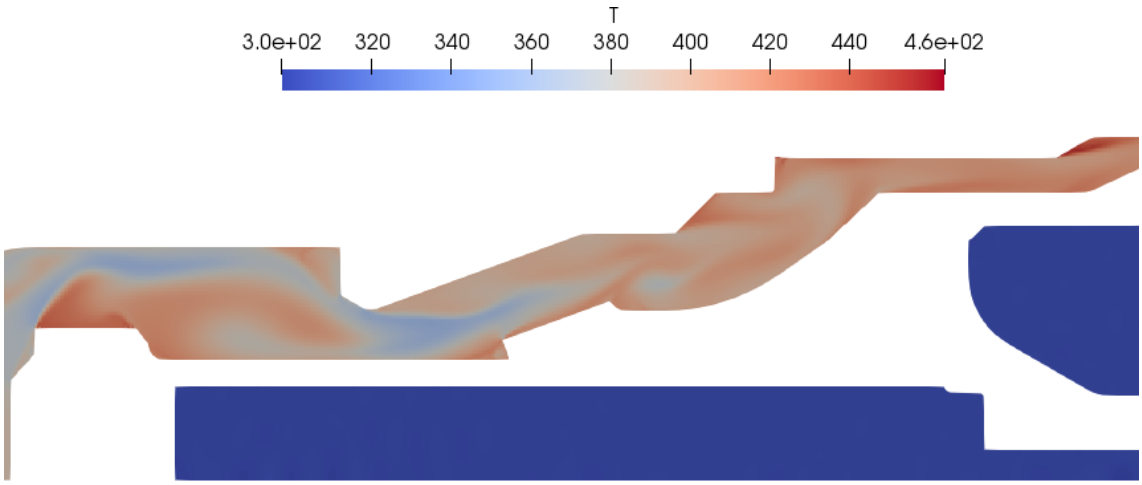

Figure 19. *Temperature (K) near the inner cylinder in the case 2*

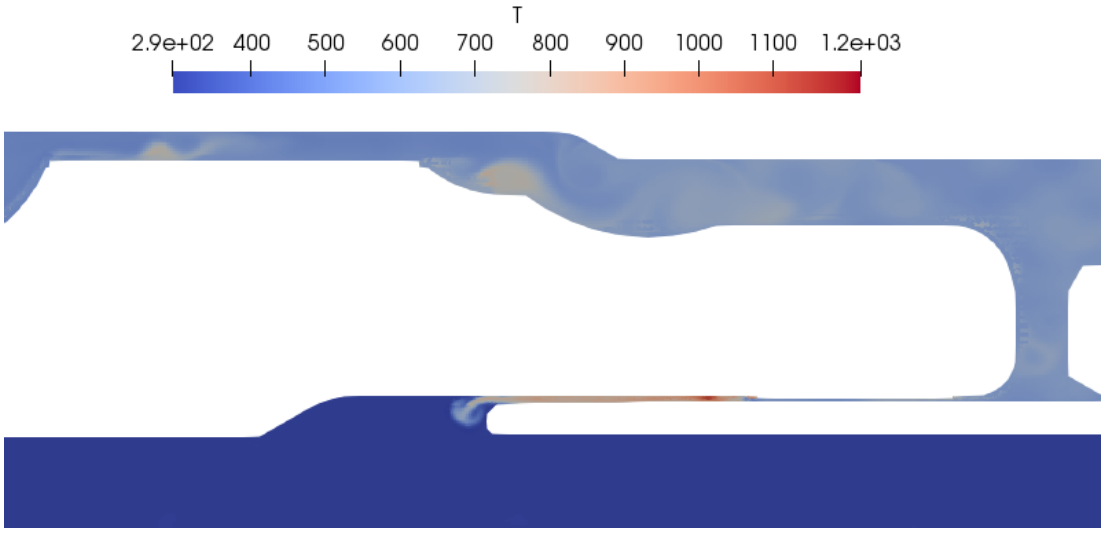

Figure 20. *Temperature (K) near the piston nose in the case 2*

From the figure 19, the average temperature is approximately 150 °C. Average temperature in the figure 20 near the piston nose is 430 °C and the maximum value is over 900 °C in the gap between piston and foot valve.

First, thermophysical properties were investigated but no apparent reason for the high temperatures was found. Temperature for the stagnated flow at the top of the check valve was investigated with equation (12) and with constant values of U = 190 m/s and Cp = 1000 J/(kgK). Theoretical temperature rise for stagnated flow was 18 °C. Stagnation temperature at the top of the check valve is shown in the figure 21.

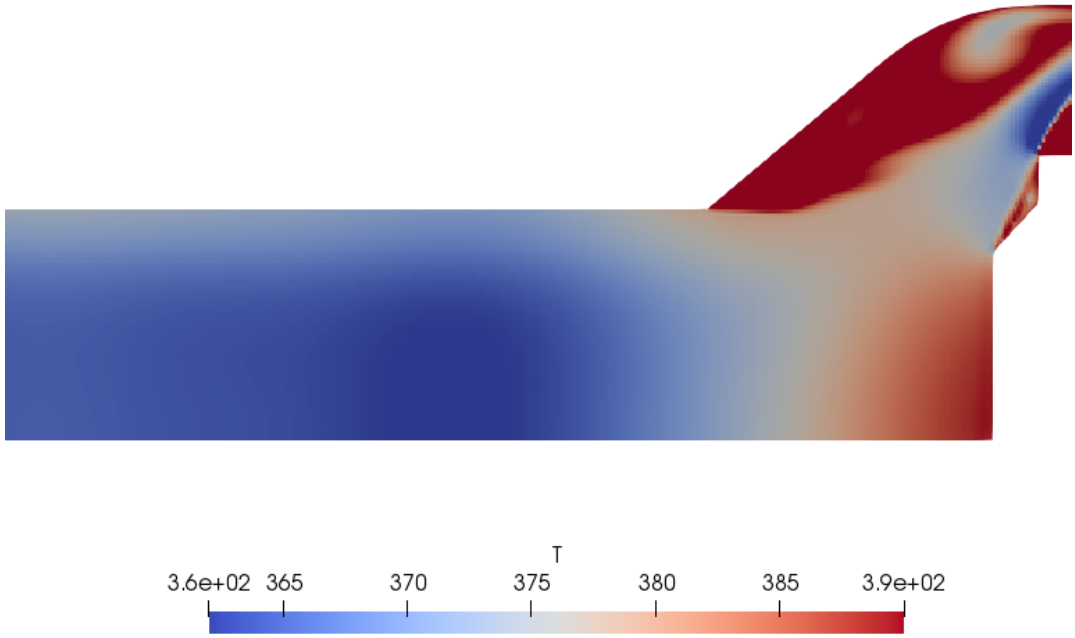

Figure 21. *Stagnation temperature (K) at the top of the check valve*

Stagnation temperature corresponds to theoretical value. Therefore, using ideal gas approximation with enthalpy as an energy variable should not be an incorrect approach for high-speed compressible flows.

After further testing the reason for high temperature values was found to be the use of relaxation in the SIMPLE loop.

```
relaxationFactors
{
    fields
    {
        "p|rho"              0.3;
        "(p|rho)Final"       1;
    }
    equations
    {
        "k|omega"            0.7;
        "(k|omega)Final"     1;
    }
}
```

It is a standard procedure to use relaxation as it helps to stabilize the simulation. Using relaxation at the intermediate instantaneous steady state solutions should be justified when the last iterations are performed without relaxation to obtain time accurate solution [20]. However, this seems not to be the case for pure transient flow and works in pseudo-transient cases only. Using relaxation seems to cause the simulation to converge in to a wrong solution, and in this case, in to a wrong temperature.

6.3 Case 3

Simulation in the case 3 is performed without relaxation. Also, the piston is moved closer to the bit to better represent the actual piston cycle where the piston is either in contact with the bit or close to the bit due to impact rebound. Temperature near the inner cylinder is presented in the figure 22.

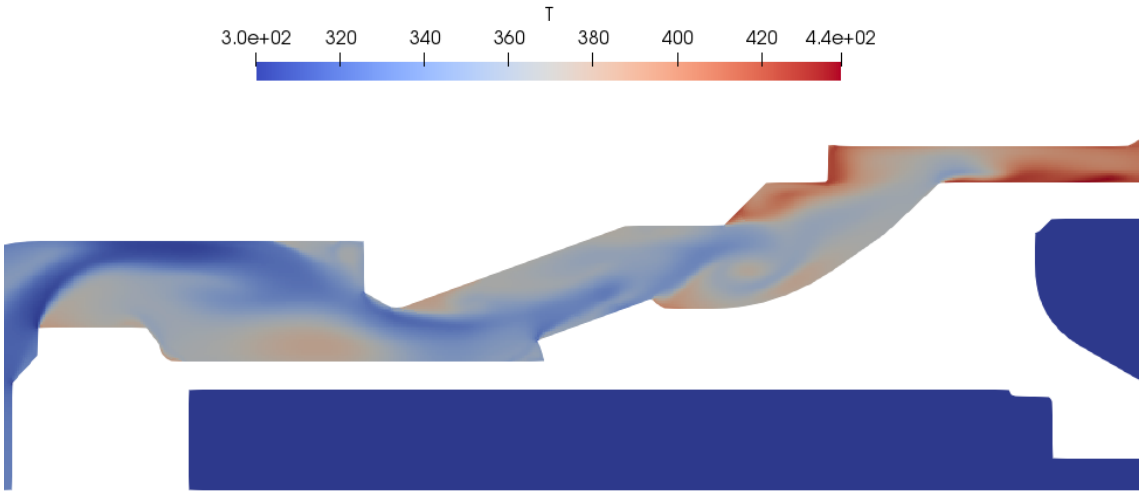

Figure 22. *Temperature (K) near the inner cylinder in the case 3*

Temperature is lower and thus more physical than in the case 2. However, it is still impossible to say whether the values are correct or not without validation of the results.

Even though temperature is more reasonable, lack or relaxation lead to other problems. Transition to small cell size in the gap surfaces causes unphysical numerical swirling where high local velocities and neglectable densities occur. If pressure is kept constant, decrease in density causes rise in temperature to satisfy the ideal gas equation (6). Example of solver crash from log file due to negative density is presented below.

```
fieldMinMax fieldMinMax1 write:
    min(mag(U)) = 0 in cell 0 at location (0.0154367 -0.386597 1.07189e-20)
on processor 0
    max(mag(U)) = 363.062 in cell 21681 at location (0.0339755 -0.0641925 -0.0014834)
on processor 13
    min(p) = 82115.9 in cell 11578 at location (0.0339983 -0.0641694 -5.53407e-21)
on processor 13
    max(p) = 372592 in cell 339 at location (0.000128816 0.206499 1.79211e-22)
on processor 18
    min(T) = 114.793 in cell 361 at location (0.0166415 -0.38259 -9.19888e-21)
on processor 0
    max(T) = 2126.87 in cell 31167 at location (0.0164337 -0.381905 2.13662e-20)
on processor 0

smoothSolver:  Solving for rho
Initial residual = 0.000383023, Final residual = 1.65308e-13, No Iterations 1
rhoEqn max/min: 6.96807 -11.0826
PIMPLE: iteration 1
smoothSolver:  Solving for h
Initial residual = 0.000743294, Final residual = 1.96852e-06, No Iterations 2
DILUPBiCGStab:  Solving for p
Initial residual = 0.000399309, Final residual = 4.52171e-08, No Iterations 10
smoothSolver:  Solving for rho
Initial residual = 0.00023313, Final residual = 1.03658e-13, No Iterations 1
time step continuity errors:
sum local = 2.37742e-06, global = 2.29868e-06, cumulative = 0.00115805
rho max/min: 7.42103 -7.44887
PIMPLE: iteration 2
smoothSolver:  Solving for h
Initial residual = 0.000524758, Final residual = 5.99462e-07, No Iterations 4
--------------------------------------------------------------------------
A process has executed an operation involving a call to the
"fork()" system call to create a child process.
```

From the log file, temperature and minimum density are extremely unphysical. Minimum and maximum values for pressure are reasonable. Because of this solver instability, limiters to temperature, velocity and density must be applied. However, limiters must be specified in a way that they do not affect actual physical results of the simulation. In the case 3, temperature is limited between 0-500 K, velocity to 0-500 m/s and density to 0.2-10 kg/m^3. More common approach for stabilizing simulations in general is to use pressure limiter rather than density limiter. However, in test cases the use of pressure limiter instead of density limiter caused simulation to crash soon after start. It seems that *overRhoPimpleDyMFoam* and *rhoPimpleFoam* in general is more sensitive to negative values of density than negative values of pressure. Limiting both pressure and density is not possible as it would violate mass conservation. Temperature at the nose of the piston is presented in the figure 23.

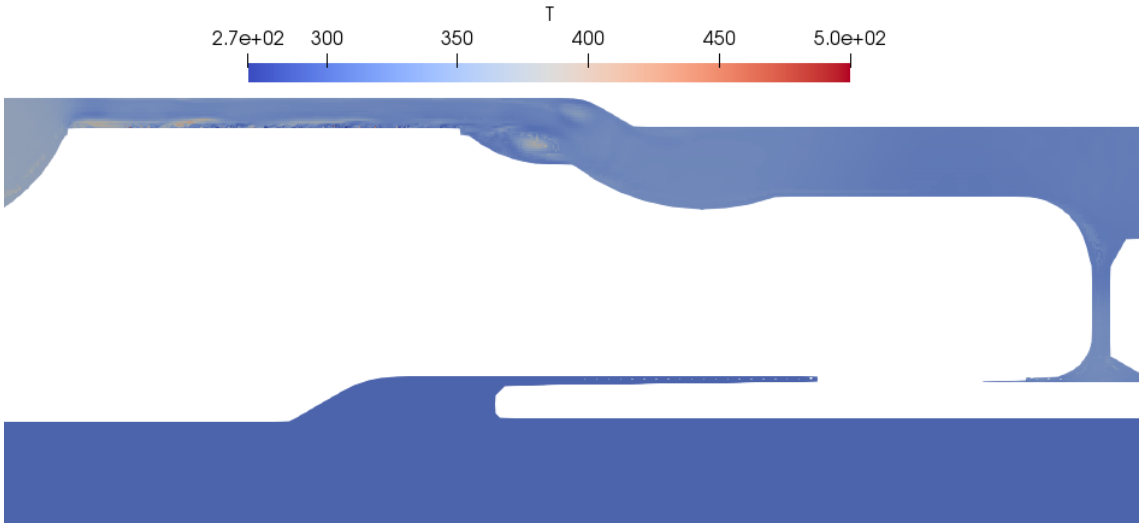

Figure 23. *Temperature (K) near the piston nose in the case 3*

Even though the average temperature in the case 3 is more physical than in the case 2, numerical swirling is notable indicated by the high maximum temperature value limited to 500 K. More detailed illustration of swirling is presented in the figures 24-26.

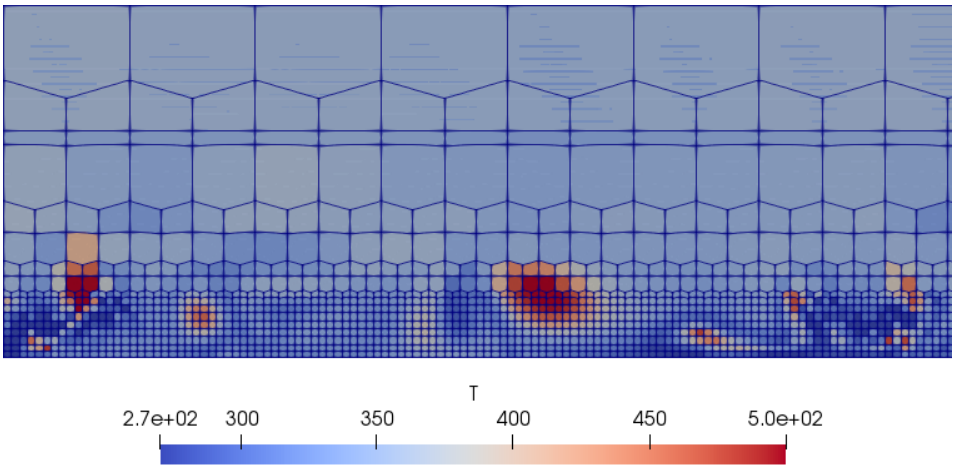

Figure 24. *Temperature (K) at small cells*

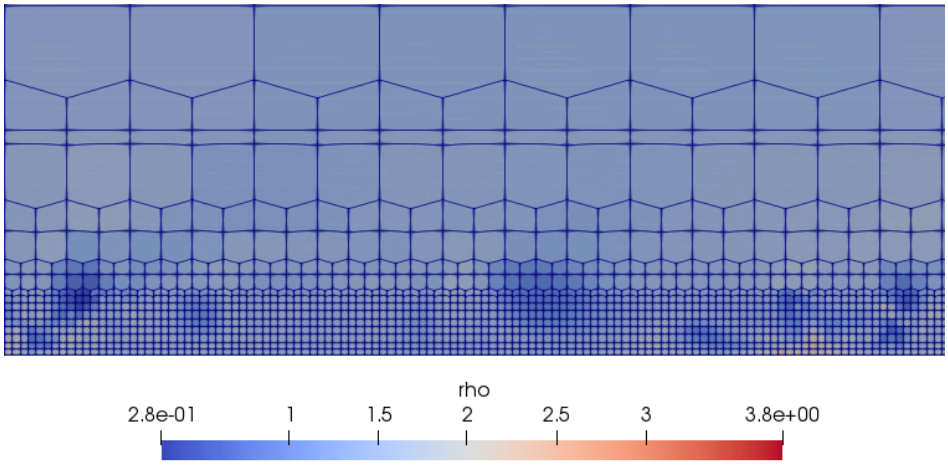

Figure 25. *Density (kg/m^3) at small cells*

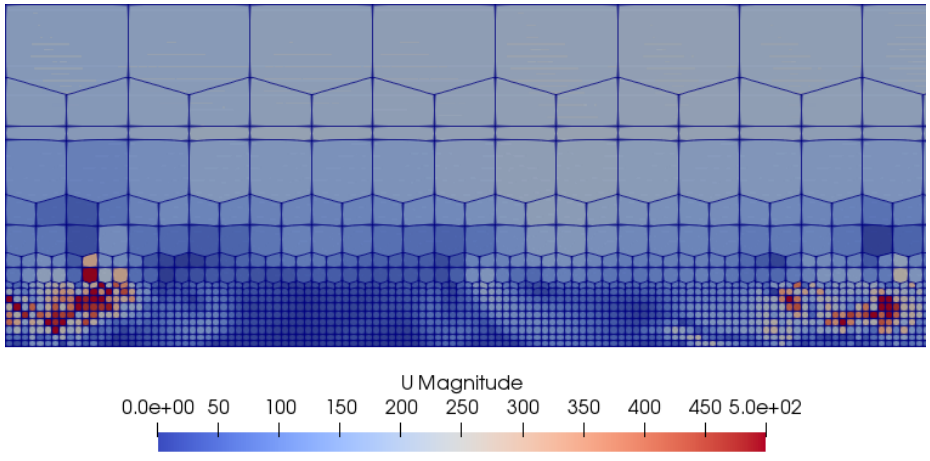

Figure 26. *Velocity (m/s) at small cells*

Swirling occurs in a larger scale starting from the sharp corner of the inner cylinder air feed hole combined with large cell size differences. Large scale swirling is illustrated in the figure 27.

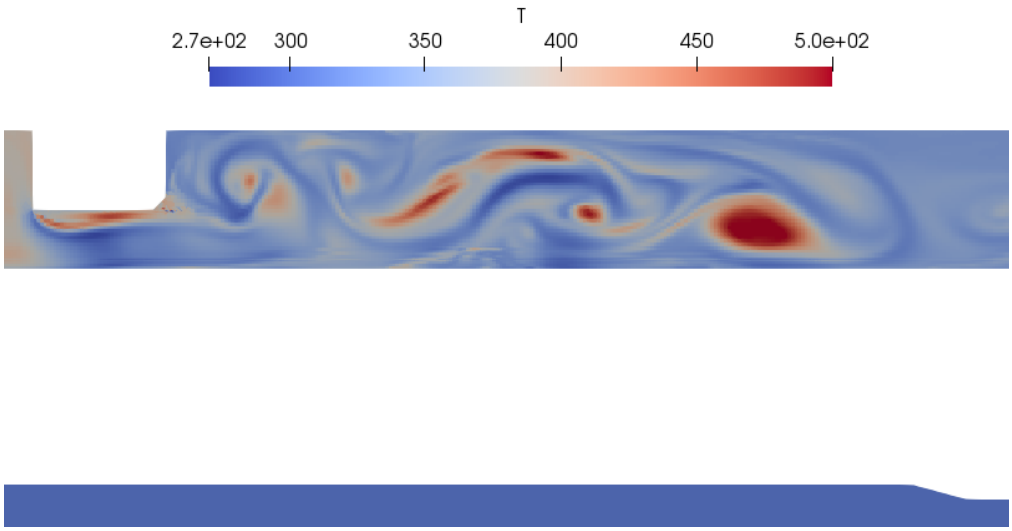

Figure 27. *Temperature (K) at the event of large scale numerical swirling*

As the overset grid requires use of backward Euler temporal discretisation with moving mesh, swirling will be diffused when movement is not restrained. Diffusion helps but does not eliminate the problem.

Besides the numerical instability of the solution, problem from the OGA is encountered. Even though high-resolution grid in the clearances is used to create sufficient number of cells between walls, OGA cannot be made robust. Holes are formed into the grid quite arbitrarily based on tests with different piston positions and mesh adjustments. Holes violate mass conservation as air can leak out of the domain and they make the simulation unstable with moving mesh. Holes forming between piston and foot valve are presented in the figure 28.

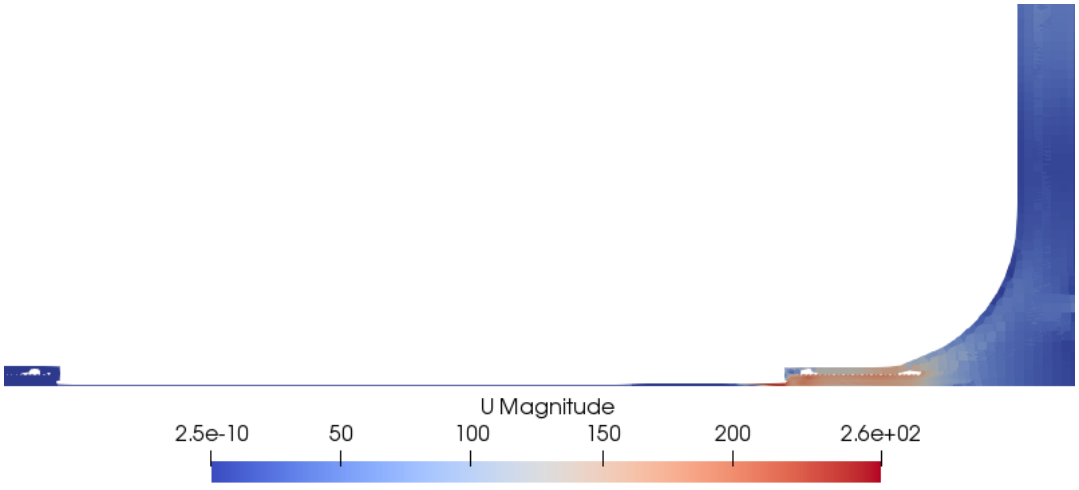

Figure 28. *Overset holes between piston and foot valve*

Holes in the figure 28 completely block the gap between piston and foot valve and no air can get through. In the case 2 this was not a problem and it is likely explained by different piston position. Holes are also formed between piston and inner cylinder shown in the figure 29.

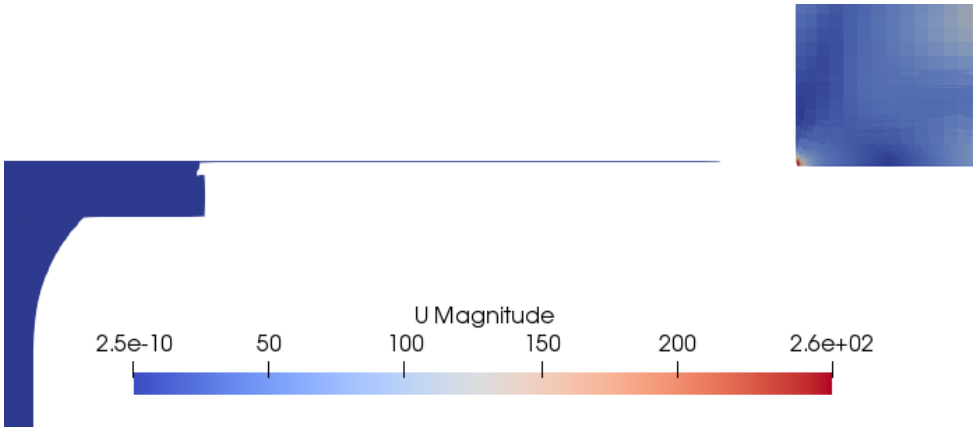

Figure 29. *Overset holes between piston and inner cylinder*

From both figures 28 and 29, high velocities occur near the holes which indicates flow exiting the domain. Case 2 also has OGA failures between inner cylinder and piston, but it is not as notable.

Based on testing, OGA seems to be sensitive for domain decomposition in parallel simulations. In serial run while using only one processor, OGA was robust at least with non-moving grid. However, when the domain is decomposed into multiple parts, holes start to occur. Different number of processors used cause different OGA results. For example, test simulation done with a desktop pc with 6 processor cores would result in continuous domain with no holes while the same simulation done in a cluster with 18 cores would result in domain with multiple holes. Therefore, with unconstructed mesh the overset simulation guideline for using at least 4 cells in the gaps is insufficient. Even when using similar sized cell layers with 8 cells in the clearances and adjusting cell position carefully, problems still occur. Even more refined mesh in the clearances could be tried but that would make simulation unpractical.

7. SIMULATION WITH POROUS ZONES

As modelling clearances using high-resolution grid was not successful, another approach had to be found. Luckily, using porous zones to restrict flow instead of walls proved to a be a successful technique. The use of porous zones allowed to bypass the numerical problems of the OGA in near wall regions while not significantly reducing physicality of the solution.

Porous zones imitate wall boundaries when permeability is set to a high value. By this way the flow velocity can be restricted to negligible values in the medium while removing actual wall boundaries. The removal of wall boundaries eliminates near wall overset limitations as porous regions are part of the continuous fluid domain. Pressure is limited with porous pressure jump in high-permeability porous zone which acts like a zero-gradient boundary condition. High-permeability porous zones implemented to simulation are illustrated in the figure 30.

Figure 30. *Implemented porous zones with blue colour*

From the figure 30, parts of the wall boundaries have been removed and substituted with porous zones. As the cell count restrictions in the clearances are removed, cell size can be increased. Clearances are still modelled but with fewer cells in between boundaries. Modified overset grid is presented in the figure 31.

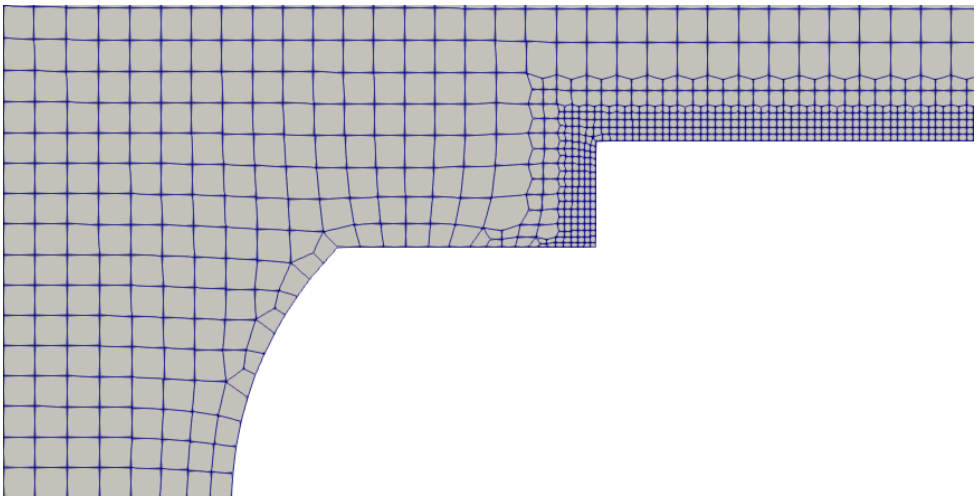

Figure 31. *Modified piston mesh*

The smallest cell size is doubled when comparing the new piston grid to the old one in the figure 14. Also, boundary layers have been removed from both background and piston geometries to produce higher quality mesh by improving cell transitions and reducing cell count. Removal of boundary layers is justified by analysing y^+ values from previous simulations. With modified mesh, local y^+ values can reach even 200 in areas where high-speed flow deflects from wall but in general y^+ values are below 100. Mesh metrics for modified mesh are shown in table 6.

Table 6. *Mesh metrics for modified mesh*

	Piston mesh	Background mesh
Cell count	45237	462221
Maximum aspect ratio	2.65	3.06
Maximum non-orthogonality	39.04	50.00
Average non-orthogonality	2.24	0.95
Maximum skewness	1.14	1.59

Mesh quality has been improved compared to the original mesh. Most notably, cell count in the piston mesh has been halved, which greatly reduces time used for the OGA. Parallel scalability test is performed to modified mesh to optimize simulation performance compared to computational cost. Results are presented in the figure 32.

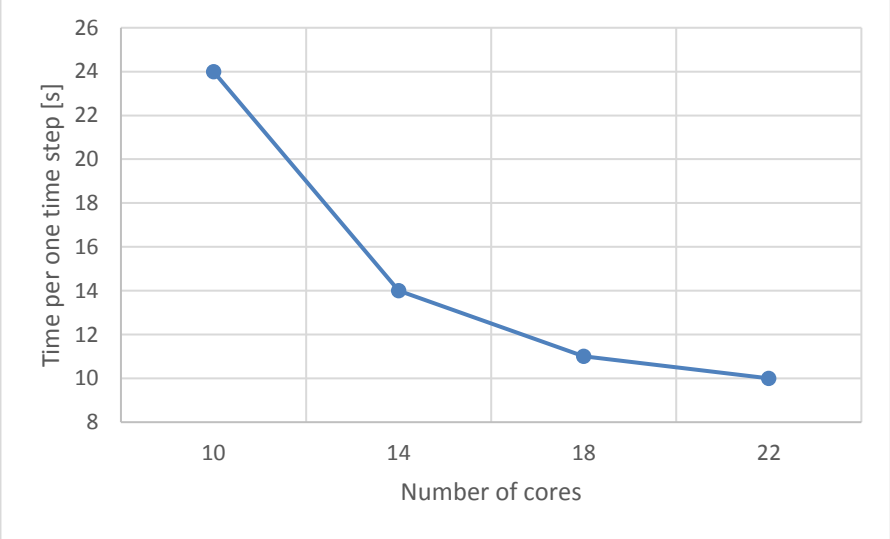

Figure 32. *Parallel scalability test*

Test was done by running a simulation using five PIMPLE outer correctors and wall time for each time step was measured. It is notable that overset grid seems to be inefficient when using too few cores. By increasing core count 40% from the 10 cores, wall time decreased 10 seconds which is 56% less than with 10 cores. Based on scalability test, 18 cores for the simulation was chosen as with more cores performance gain starts to decrease.

Using porous zones requires care when used with overset grid. Simulation was tried with different permeability settings and it was noticed that using too much permeability causes problems in the OGA. Therefore, arbitrarily high amount of porosity cannot be used. Porous zones with too high permeability combined with moving grid cause local high pressures with negative pressures which will eventually crash simulation. However, using too low permeability value causes flow to go through porous zone. Also, as porous zones are not physical wall boundaries, no-slip boundary condition for velocity and turbulence wall functions cannot be used for these regions.

Based on previous simulation results, limiters for temperature, velocity and density have been adjusted. Temperature is limited to 0 - 400 K, velocity to 0 - 400 m/s and density to 0.7 - 11 kg/m^3. Based on testing, permeability in the porous zones is set to $5 \cdot 10^3$ d which can be considered as a good compromise between high permeability and numerical robustness. Forchheimer's terms are neglected. Backward Euler discretization is used for moving mesh as recommended. Acceleration relaxation and acceleration damping parameters have been tightened. Relaxation factor for h, k and ω are set to 0.9 to stabilize the simulation. Other settings are used from the trial case 3.

7.1 Porous media verification

Simulation using a porous zone to restrict fluid flow was tested and compared to similar geometry using only physical walls to verify porous media approach. Flow was studied in 2D with 0.2 mm x 40 mm gap. Total pressure inlet with 200 kPa pressure was used with 100 kPa fixed value outlet. Permeability in the porous zone was set to $5 \cdot 10^{13}$ d. Turbulence settings are same as in the trial case 2 and temperature was limited to 400 K. Test case geometries are illustrated in the figure 33.

Figure 33. *Simulation with physical wall (top) and simulation with porous zone (bottom) marked with blue*

Uniform Cartesian grid was constructed for the test case and two cell layers were placed in the gap to present the mesh between piston and hammer walls. Velocity magnitude results for the test cases are shown in the figure 34.

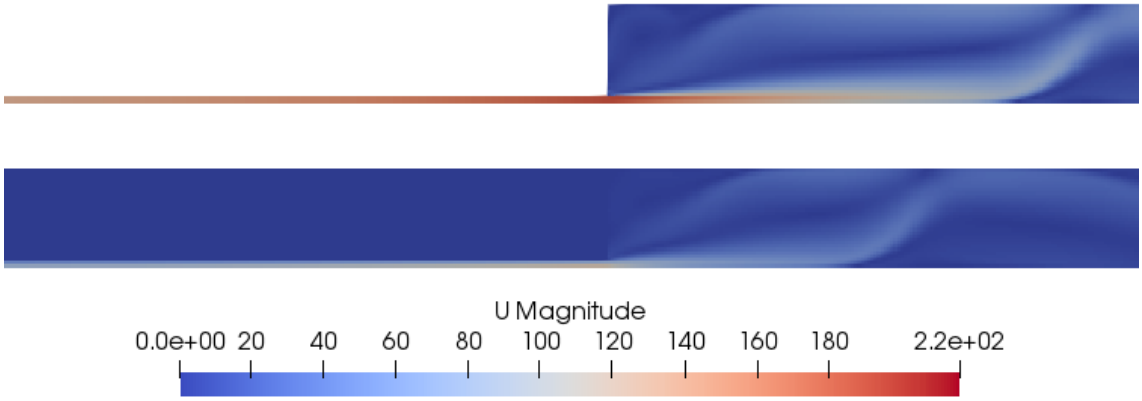

Figure 34. *Velocity magnitudes (m/s) in the gaps with physical walls (top) and with porous zone (bottom)*

Assumption was that the flow velocity would be slower in the gap using solid walls as wall functions and no-slip boundary condition cannot be used for porous zones. Surprisingly, flow in the gap with solid walls had significantly higher velocity. To study this behavior, fully developed flow profiles were investigated at the end of the gap for both test cases using 2 and 20 cell layers in the gap. For both test cases, dimensionless wall distance for 2 cell layers were $y^+ > 25$ and for 20 layers $y^+ < 5$. Velocity profiles are presented in the figure 35.

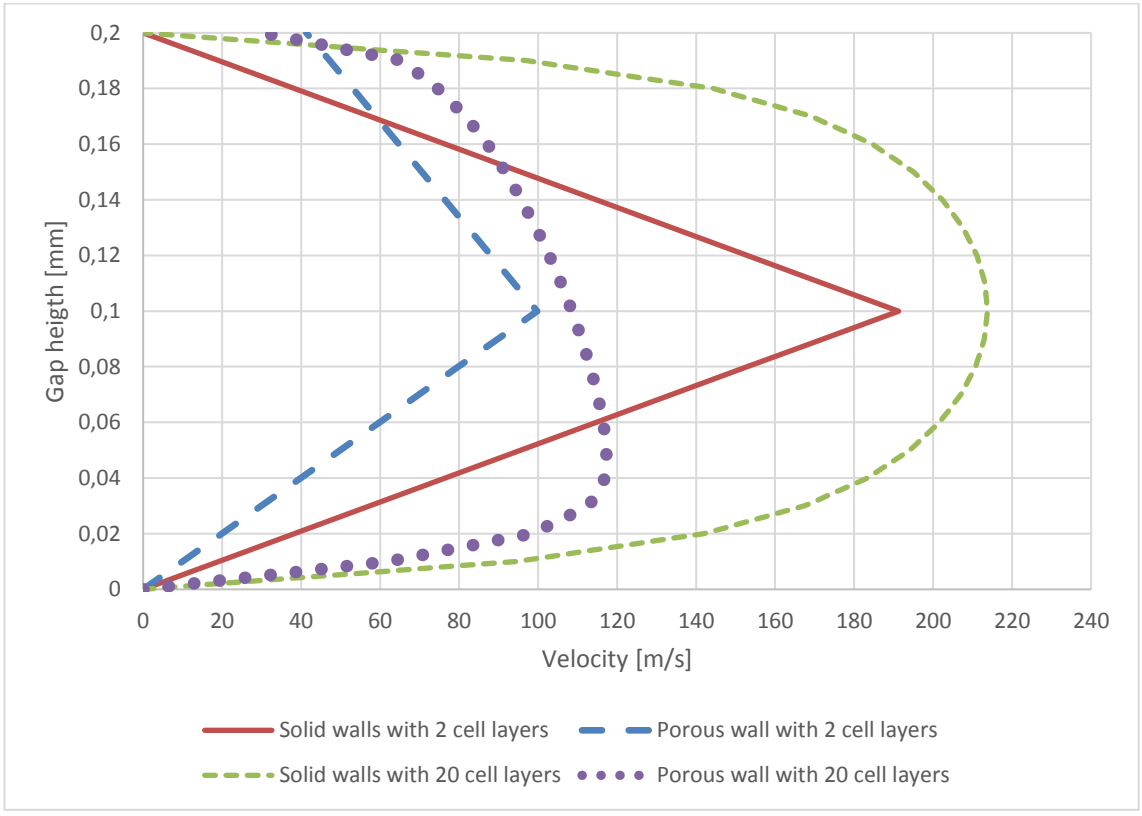

Figure 35. *Fully developed gap flow velocity profiles for 4 verification test cases*

From the figure 35, using porous zone causes flow velocity to be non-zero at the porous boundary which is expected. Also, using only two cell layers causes triangular flow profiles with lower velocity compared to flow profiles in refined mesh. Because of this, mass flow rate in the gap with two cell layers is significantly less than in the gap using refined mesh. Reason for lower flow rate in the gap using porous wall and a physical wall compared to two physical walls could not be found.

Turbulence quantities in the gap are not comparable as flow velocities differ. Maximum flow velocity in the porous zone was approximately 0.05 m/s at the left corner of the gap, so the porous zone is practically impermeable. Some smearing of temperature and pressure occurred inside the porous zone near the zone boundaries, but it did not affect the physicality of the porous test case.

Because of lower flow rate in the gap, porous zones cannot be used to accurately simulate gap flow with low-resolution grid. High-resolution grid slightly improves solution. However, the use of porous zones with low-resolution grid can still be justified to make the mesh movement in the near wall regions possible and to reduce computational cost. Also, lower flow rate in the porous zone gaps is somewhat mitigated by larger clearances used in the simulation. A study should be made if the clearances could be made larger to compensate lower mass flow and to increase minimum cell size when using low-resolution grid.

7.2 Trial simulation with off-centered Crank Nicolson scheme

Simulation was tried with off-centered Crank Nicolson 0.5 temporal scheme. With off-centered Crank Nicolson scheme simulation was stable up to Co = 10, but negative pressure accumulation caused a problem where the average Courant number was reduced from approximately 0.12 to 0.0004. Average Co can be considered as an indicator of computational efficiency and ideally it should be in the same order as maximum Co. Average Co is shown in the figure 36.

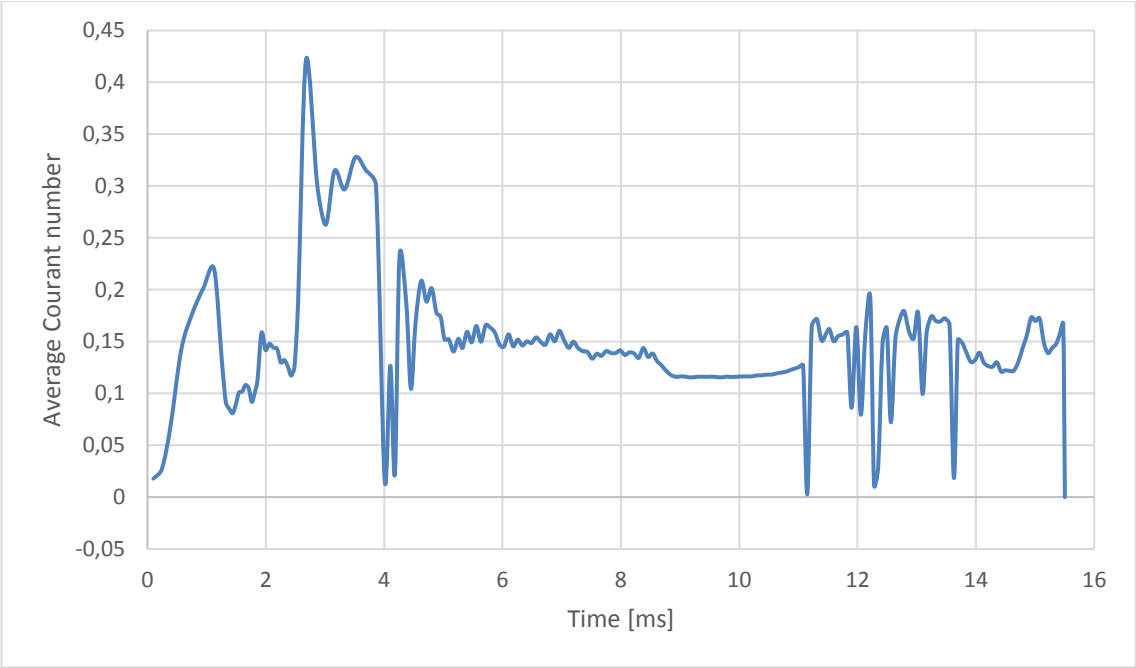

Figure 36. *Average Co for off-centered Crank Nicolson simulation*

Reduction of average Co made the simulation impractical to continue. Negative pressure accumulation in the overset region causing the problem at t = 15.5 ms is presented in the figure 37.

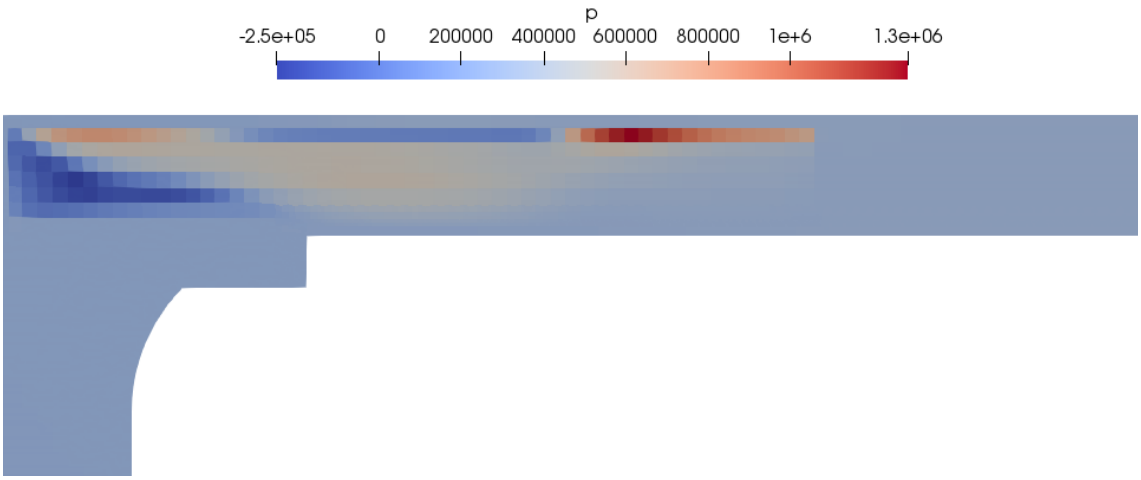

Figure 37. *Local negative and large positive pressures (Pa)*

Simulation with backward Euler scheme was stable only up to Co = 5 but it was more robust than with off-centered Crank Nicolson scheme and could be run without any problems.

8. RESULTS AND DISCUSSION

Simulation was run with 18 cores for approximately 10 days with off-centered Crank Nicolson temporal scheme from time zero up to t = 2.5 ms and with backward Euler scheme from t = 2.5 ms to t = 25 ms. Most of the computational time was used for over-set grid assembly. Maximum Co = 5 was used and average time step was $\Delta t = 5 \cdot 10^{-4}$ ms. Piston translation was allowed at t = 2.5 ms and initial velocity of 5 m/s was given. Validation of the results is done afterwards with DTH test rig when it is operational. Settings used are presented in the appendices C – F.

Simulation was run up to the collision of piston to the upper pressure chamber at t = 25 ms. Because of the collision, results up to t = 24.5 ms are used. Collision was most likely caused by too high initial velocity given for the piston. Initial value of 5 m/s was based on semi-analytical model designed for conventional hammers. With these hammers the volume of air in the upper pressure chamber approaches zero as the piston moves upwards causing downwards acceleration of the piston to approach infinity. Because of this, the piston always stops before collision. However, this is not the case for well hammers with inner cylinders. As the piston moves upwards, the volume of air in the upper pressure chamber is limited to a finite minimum value limiting possible deceleration. Piston velocity, acceleration and displacement are shown in the figures 38-40.

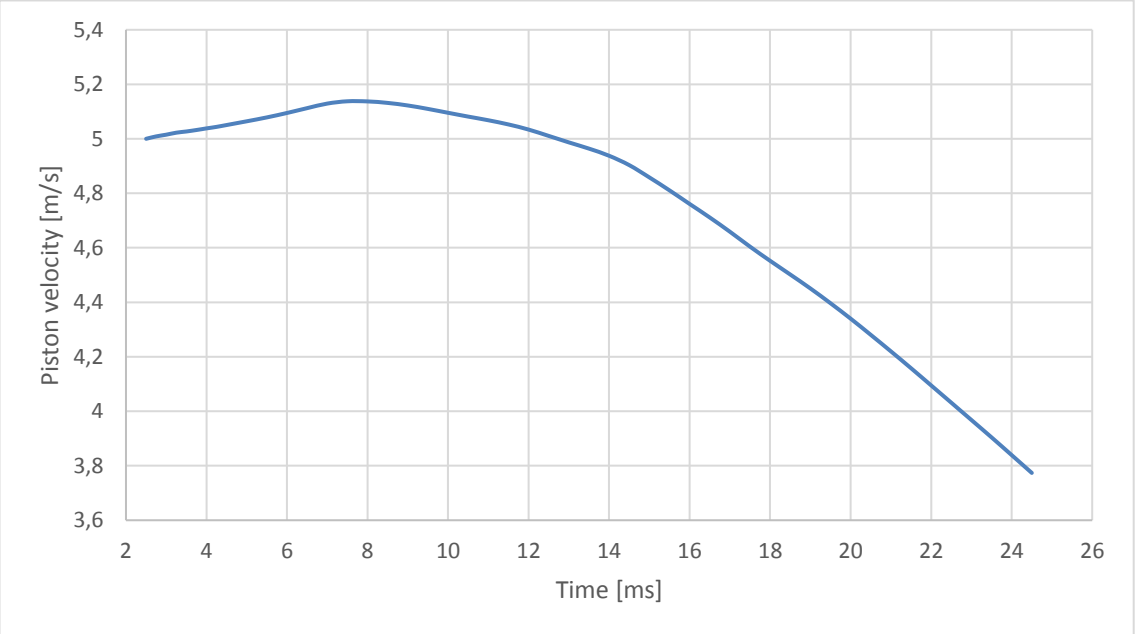

Figure 38. *Piston velocity*

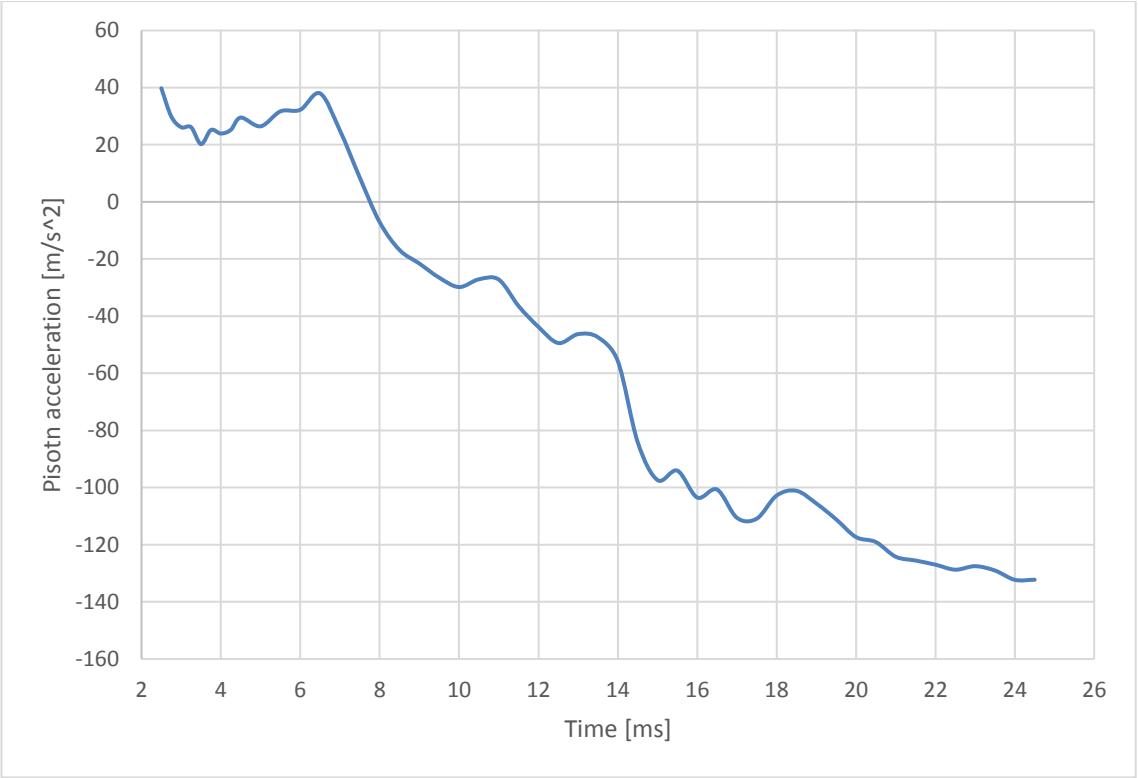

Figure 39. *Piston acceleration*

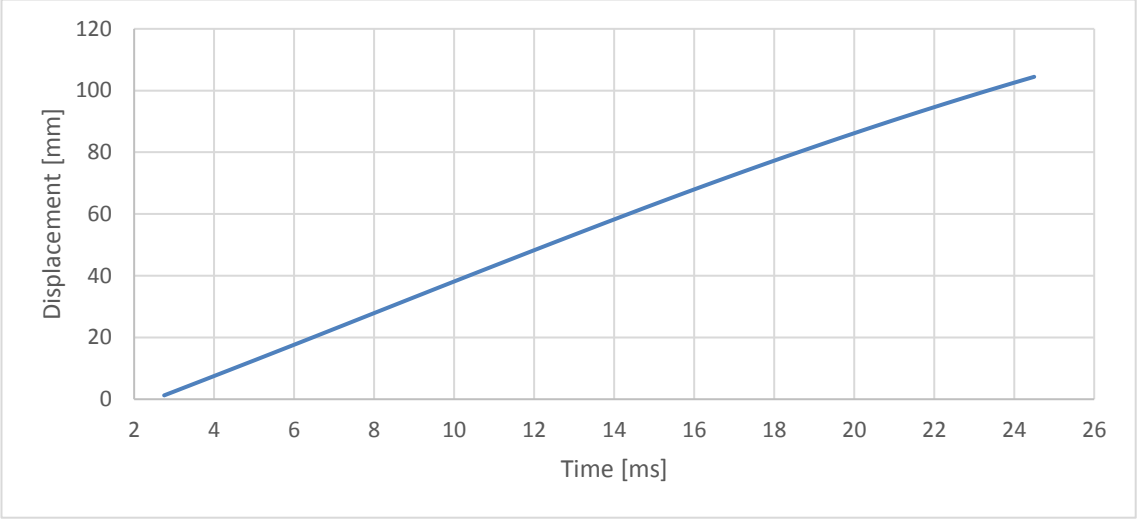

Figure 40. *Piston displacement*

From the figure 38, piston acceleration decreases when translational movement is allowed. Next, piston starts to decelerate at t = 7.5 ms when the side channels are blocked. At t = 14 ms high-pressure air from the lower pressure chamber is discharged and at t = 18 ms upper pressure chamber is sealed which further increases deceleration.

Important metric in transient overset simulation is the cumulative mass continuity error and it is defined for compressible simulation as

```
{
      dimensionedScalar totalMass = fvc::domainIntegrate(rho);

      scalar sumLocalContErr =
      (fvc::domainIntegrate(mag(rho - thermo.rho()))/total-
      Mass).value();

      scalar globalContErr =
      (fvc::domainIntegrate(rho - thermo.rho())/totalMass).value();

      cumulativeContErr += globalContErr;
}
```

a deviation of mass inflow and mass outflow. However, in this simulation air is trapped inside the hammer to produce piston motion and because of this, it is hard to say how much of the cumulative continuity error is caused by inaccuracy of the overset interpolation. Cumulative continuity error is shown in the figure 41.

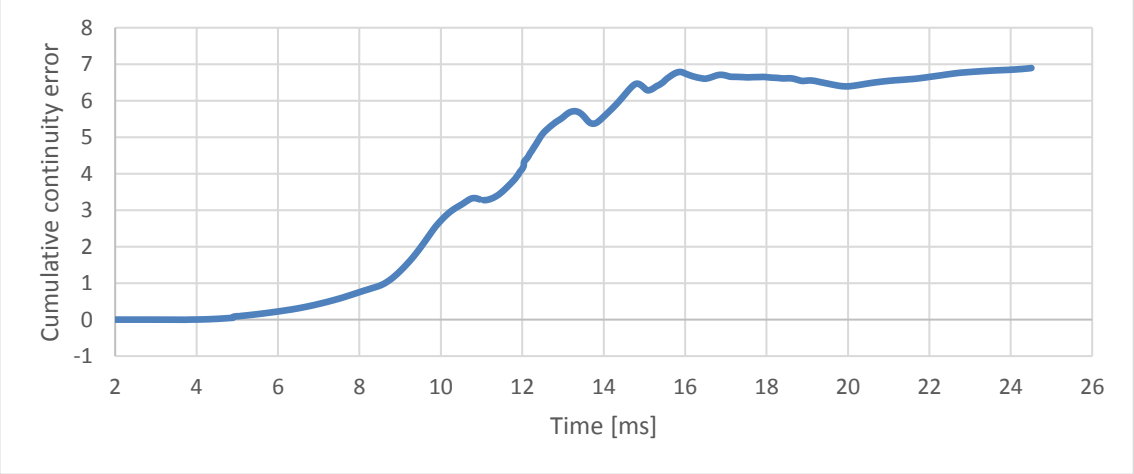

Figure 41. *Cumulative continuity error*

As the initialization of the simulation is done using second order off-centered Crank Nicolson temporal discretization with stationary mesh, conservation error is negligible until t = 2.5 ms. Surprisingly, decline of acceleration shown in the figure 39 at t = 2.5 ms is not related to error in mass conservation as error starts to increase only after t = 4 ms, even while using backward Euler discretization. Cumulative continuity error increases from t = 4 ms to t = 16 ms and stays relatively constant afterwards. Further studies should be made to extract the cumulative continuity error caused by the overset interpolation.

Only most important scalar results are shown for simplicity. Vector and turbulent results are not presented as it would require large pictures for multiple time steps. Velocities and the Mach numbers at six different time steps ranging from t = 2.5 ms to t = 24.5 ms are given in the figures 42-43.

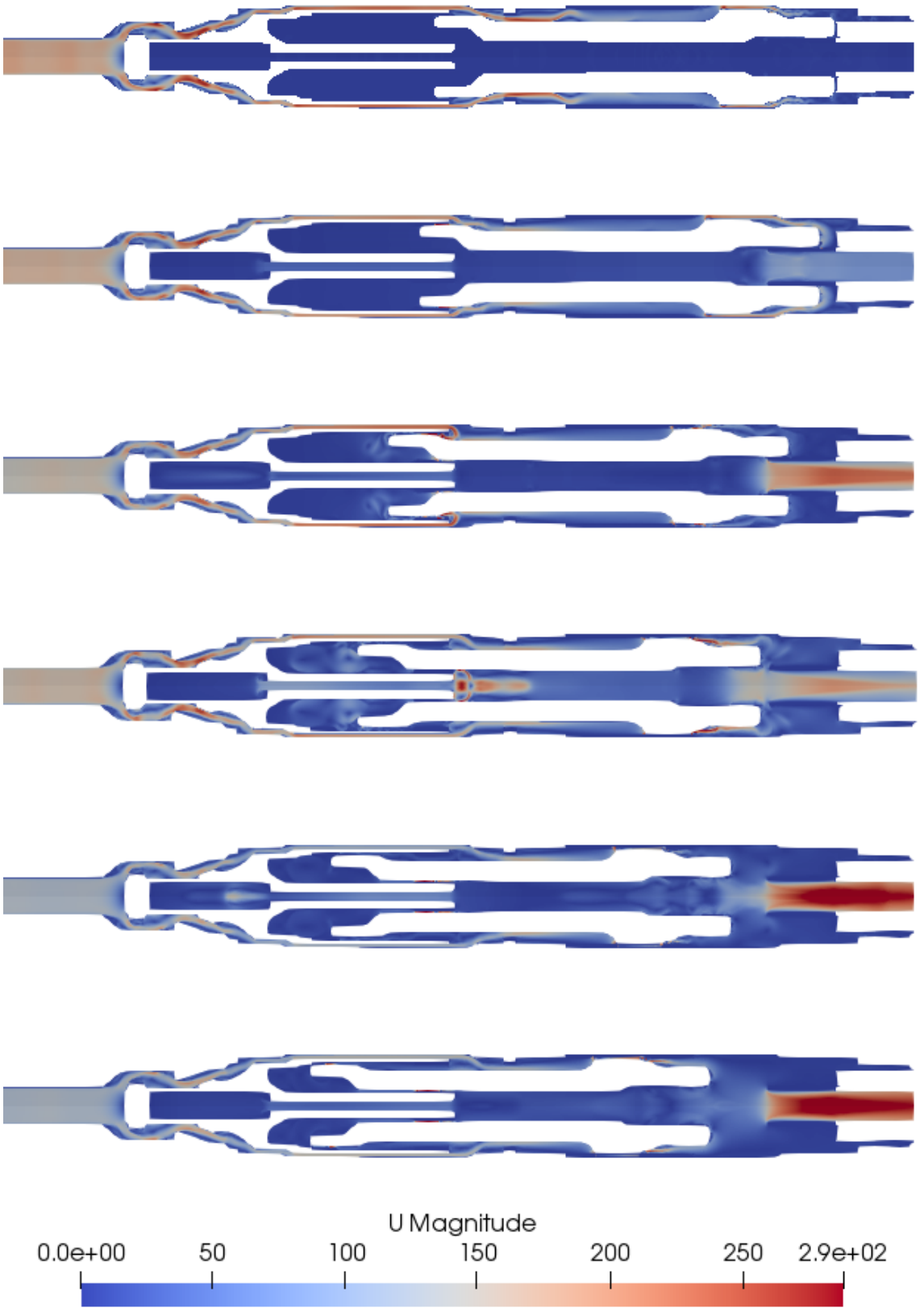

Figure 42. *Velocities (m/s) in the hammer at time steps 2.5 ms, 5.0 ms, 10 ms, 15 ms, 20 ms and 24.5 ms ranging up to down*

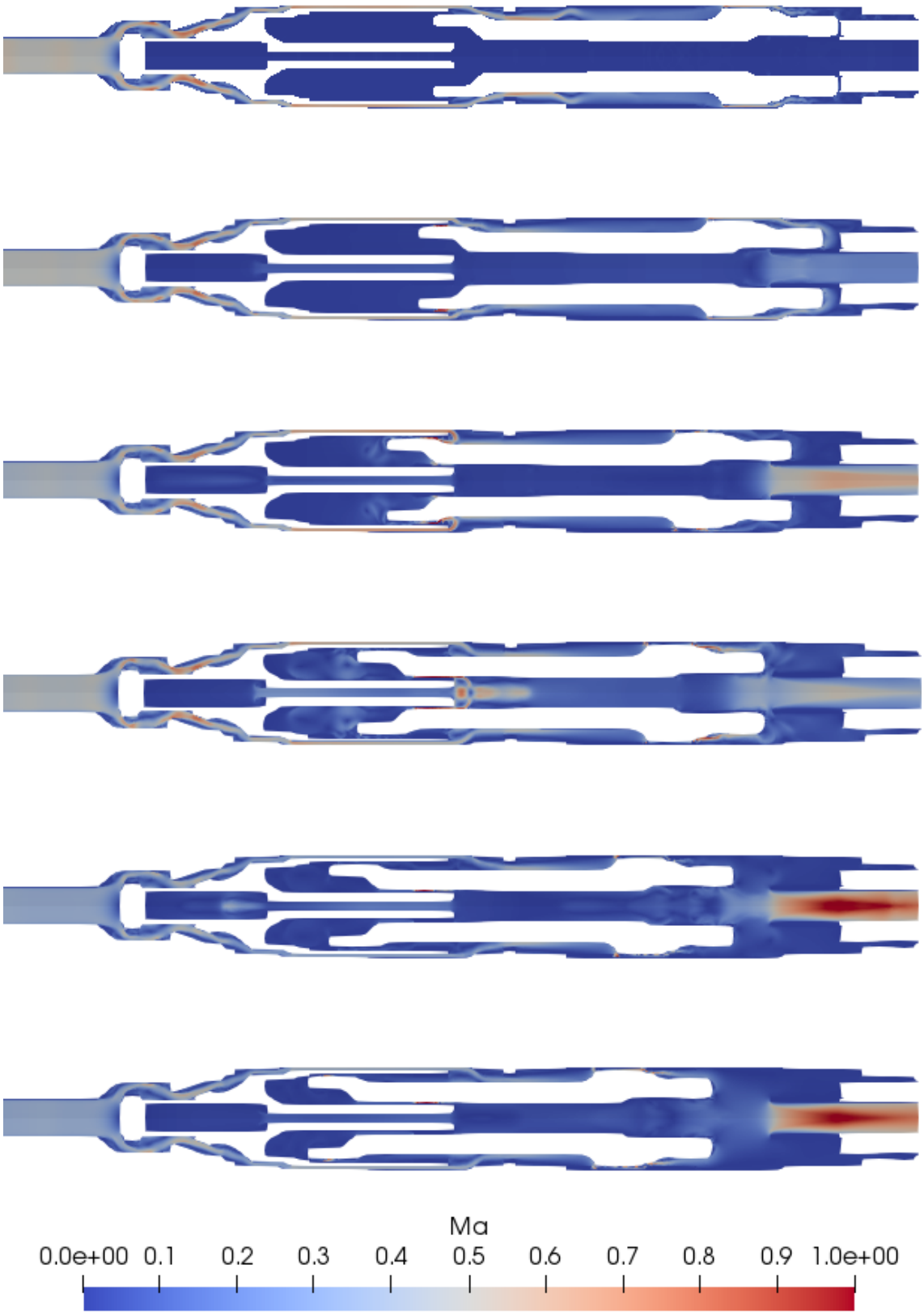

Figure 43. *Mach number in the hammer at time steps 2.5 ms, 5.0 ms, 10 ms, 15 ms, 20 ms and 24.5 ms ranging up to down*

Excluding numerical errors, only temperature reached limit of 400 K during the simulation. This could indicate that the limit is unphysical, and it should be set higher or that too much energy is converted to heat. Pressure and temperature are probed in the points indicated by the figure 44.

Figure 44. *Probe locations*

Outflow of the hammer is studied by inspecting mass flow and average temperature at the outlet. Mass outflow is presented in the figure 45.

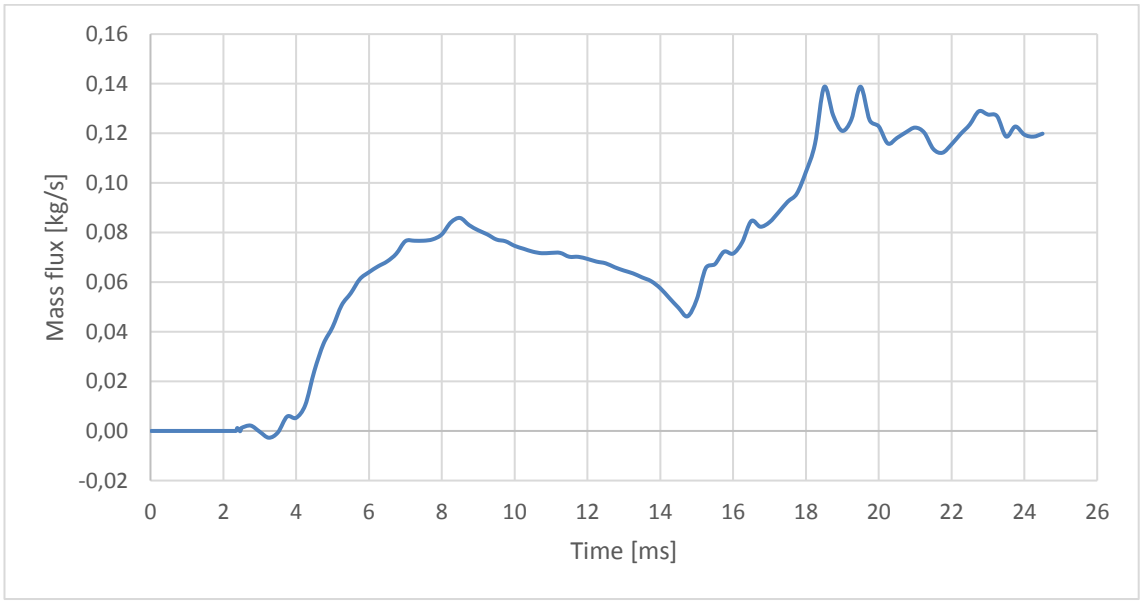

Figure 45. *Mass outflow at the outlet*

From the figure 45, mass flow at the outlet increases as the pressure accumulates in the lower pressure chamber and starts to decrease when the piston blocks the side channels. Mass outflow increases again when the piston moves past the foot valve at t = 14 ms. Surprisingly, mass flow does not decrease after the piston has blocked the side walls and moved past the foot valve. This could be caused by inaccuracies in modelling clearances with porous zones using low-resolution grid, but it could also illustrate the effect of leakage through the gaps. Averaged temperature at the outlet is investigated and shown in the figure 46.

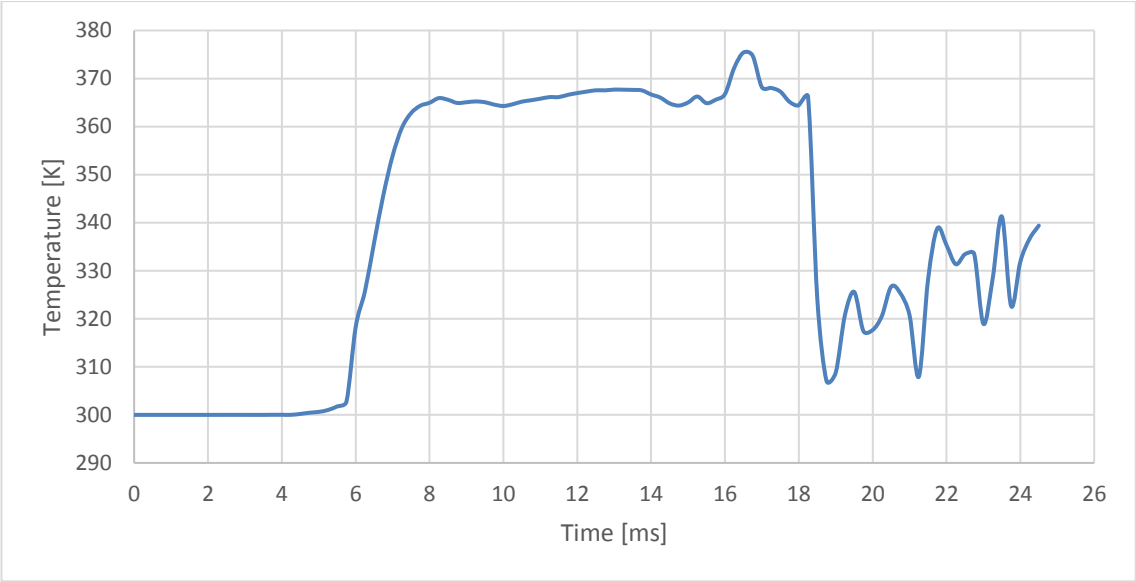

Figure 46. *Averaged temperature at the outlet*

Temperature has a square pulse like behaviour from t = 5.5 ms to t = 18.5 ms and after that, fluctuations occur. The Mach number is investigated at t = 20 ms as mass flow has large values at that instant. Ma exceeding one is also shown in the figure 43. Ma isocontours are presented in the figure 47.

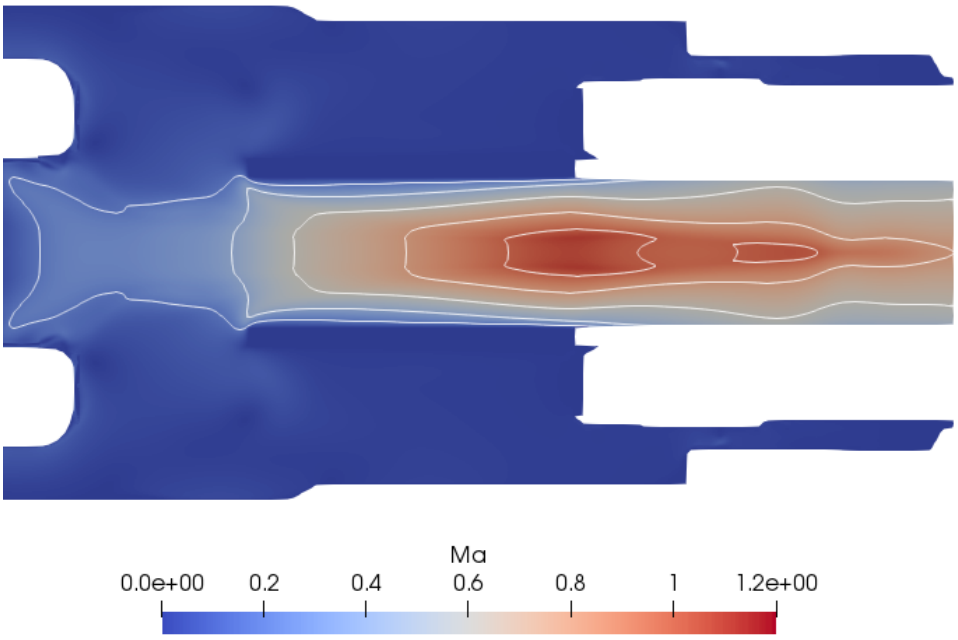

Figure 47. *Transonic flow at the outlet with Ma isocontours*

Flow at the outlet is transonic as Ma locally exceeds one which is allowed without flow being supersonic. Also, it seems that temperature fluctuations are related to the transonic flow starting from t = 18 ms. However, transonic flow as a phenomenon is complicated. Further studies would be needed to determine whether it is significant for DTH hammer operation.

9. CONCLUSIONS

In this thesis project a simulation approach for down-the-hole hammers and pneumatic devices with moving components in general was developed. Simulation for a well hammer was done using rigid body dynamics with 2D axisymmetric overset grid with two different approaches. First, simulation was run using high-resolution grid in the clearances between the piston and the hammer walls. However, this approach failed due to overset grid assembly robustness issues in the near wall regions in parallel simulation. Second simulation used high-permeability porous zones in near wall regions instead of wall boundaries. With this approach overset grid assembly issues could be eliminated entirely.

As the main interest was to simulate piston cycle during continuous operation and not the initial transient, a rebound velocity of 5 m/s was given to the piston. However, initial velocity was too high, and the piston collided to the upper pressure chamber. Rebound velocity was an educated guess based on semi-analytical DTH hammer design tool not specifically made for well hammers. In the design tool, the air volume in the upper pressure chamber approaches zero as the piston moves upwards causing piston deceleration to approach infinity. However, this is not the case for studied well hammer which has finite minimum upper pressure chamber volume.

For the results, displacement, velocity and acceleration data for the piston were acquired which can be used to tune the DTH hammer design tool for well hammers. Also, data for pressure, temperature and flow velocity are gathered from the simulation. Several different flow phenomena are noticed. Most notably flow at the outlet gradually develops to transonic speed producing temperature and mass flow fluctuations.

As for the future development of overset analysis for OpenFOAM, few suggestions are made. In parallel, overset grid assembly should be made robust. Also, zero gap implementation, like in STAR-CCM+, would be useful to simulate moving bodies in proximity when gap flow in the clearances is negligible. Lastly, plane restraint would be useful to restrain translation in one dimension by allowing movement only up to a defined plane.

For future work, several new studies should be made. First, simulation should be run with lesser initial piston velocity to simulate the whole piston cycle up to the piston impact to the bit. Results can be used to tune already existing design tools to correlate better with well hammers. Second, simulation should be run in 3D with *rhoPimpleFoam* or *rhoCentralFoam* solvers without overset grid. Also, pressure accumulation in the lower pressure chamber and piston acceleration should be studied with and without clearances.

Third, gap flow causing leakage should be studied analytically and with CFD with different grid resolutions to investigate the possibility to use low-resolution gap mesh better. The possibility of replacing clearances with solid walls and boundary conditions with polynomial data fits for gap flow should be studied also. If clearances could be removed without compromising moving mesh, it would greatly increase minimum cell size and mesh quality. Fourth, if previous suggested studies are successful, LES could be used instead of RANS in static cases if even more accurate results are required. Lastly, simulation model must be verified and validated properly with DTH test bench before further conclusions can be made.

BIBLIOGRAPHY

[1] D. Bruce, R. Lyon, S. Swartling, The history of down-the-hole drilling and the use of water-powered hammers, 2014.

[2] J. Silver, WH4 Technical Brochure, 2017.

[3] C. Song, K. Kwon, D. Shin, W. Hwang, J. Lim, J. Cho, Trend analysis of drilling technology for top-hammer drilling machine, 2013.

[4] P. Bordi, Top hammer workshop 2, 2018.

[5] J. Sokka, Design and implementation of factory layout renewal in South Korea, 2012.

[6] Halco, A-Z of DTH drilling, 2016.

[7] V. Pohja, General sales kit, 2011.

[8] Robit, Well drilling rig, 2018.

[9] J. Silver, Hyper 41 manual, 2018.

[10] J. Blazek, Computational fluid dynamics: principles and applications, 2005.

[11] W. Sutherland, The viscosity of gases and molecular force, Philosophical Magazine, 1893, pp. 507-531.

[12] Y. Cengel, M. Boles, Thermodynamics: an engineering approach, 7th ed. 2010.

[13] F, Menter, Two-equation eddy-viscosity turbulence models for engineering applications, AIAA Journal, Vol. 32, Iss. 8, 1994, pp. 1598-1605.

[14] F, Menter, M, Kuntz, R, Langtry, Ten years of industrial experience with the SST turbulence model, Turbulence, Heat and Mass Transfer 4, 2003, pp. 625 - 632.

[15] OpenCFD Ltd, OpenFOAM v1712 Extended Code Guide, 2017.

[16] D. Spalding, A single formula for the “law of the wall”, Vol. 28, Iss. 3, 1961, pp. 455-458.

[17] H. Hafsteinsson, Porous Media in OpenFOAM, 2009.

[18] A. Bejan, Convection Heat Transfer, 1984.

[19] The OpenFOAM Foundation, OpenFOAM v5 User Guide, 2017.

[20] T. Holzmann, Mathematics, numerics, derivations and OpenFOAM®, 2017.

[21] J. Crank, P. Nicolson, A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type, Mathematical Proceedings of the Cambridge Philosophical Society, Vol. 43, Iss. 1, 1947, pp. 50-67.

[22] E. Ekedahl, 6-DOF VOF-solver without damping in OpenFOAM, 2009.

[23] J. Steger, J. Benek, On the use of composite grid schemes in computational aerodynamics, Computer Methods in Applied Mechanics and Engineering, 1987, pp. 301-320.

[24] OpenCFD Ltd, New and improved numerics, 2017.
URL https://www.openfoam.com/releases/openfoam-v1706/numerics.php.

[25] G. Zagaris, M. Campbell, D. Bodony, E. Shaffer, M. Brandyberry, A toolkit for parallel overset grid assembly targeting large-scale moving body aerodynamic simulations, Springer Berlin Heidelberg, 2010, pp. 385-401.

[26] B. Roget, J. Sitaraman, Robust and efficient overset grid assembly for partitioned unstructured meshes, Journal of Computational Physics, 2014, pp. 1-24.

[27] R. Meakin, Object X-rays for cutting holes in composite overset structured grids, 2001.

[28] Sharcnet, Diagnosing overset interface issues, 2016. URL https://www.sharcnet.ca/Software/Ansys/17.2/en-us/help/flu_ug/flu_ug_sec_overset_issue_diagnosis.html.

[29] E. Quon, M. Smith, Advanced data transfer strategies for overset computational methods, Computers & Fluids, 2015, pp. 88-102.

[30] K. Krebelj, wedgePlease, 2017. URL https://github.com/krebeljk/wedgePlease.

[31] CFD Online, Forum discussion, 2017. URL https://www.cfd-online.com/Forums/openfoam-meshing/186049-2d-axisymmetric-model-openfoam-4-0-a.html#post645900.

[32] H. Jasak, Error analysis and estimation for the finite volume method with applications to fluid flows, 1996.

[33] A. Mikkonen, Discussion with thesis instructor, 2018.

[34] OpenCFD Ltd, Overset virtual course, 2018.

APPENDIX A: OVERSET AND DYNAMIC SETUP

Settings for overset simulation with moving grid are presented briefly. Overset simulation setup is started by selecting overset grid and it is most easily done with *topoSet*. For a single overset grid on a background mesh, *topoSetDict* can be quite generic if `insidePoints` is not defined in the overset cell.

```
actions
(
      {
            name            c0;
            type            cellSet;
            action          new;
            source          regionsToCell;
            sourceInfo
            {
                  insidePoints    ((0.001 0 0));
            }
      }
      {
            name            c1;
            type            cellSet;
            action          new;
            source          cellToCell;
            sourceInfo
            {
                  set       c0;
            }
      }
      {
            name            c1;
            type            cellSet;
            action          invert;
      }
);
```

First the background mesh is selected, and then duplicate cell set is created with the name c1. After that, c1 is inverted so that it only contains overset region and selected cell sets are given field values of 0 and 1 with *setFields*. Lastly, overset patch must be first in the *boundary* dictionary to properly activate explicit interpolation for field values when needed. Dynamic settings for high-resolution grid simulations are presented below in *dynamicMeshDict*.

```
motionSolverLibs
(
      "libsixDoFRigidBodyMotion.so"
      "libfvMotionSolvers.so"
);
dynamicFvMesh                   dynamicOversetFvMesh;
solver                          sixDoFRigidBodyMotion;
sixDoFRigidBodyMotionCoeffs
{
      patches         (pistonWalls innerClearance outerClearance);
      innerDistance             100.0;
      outerDistance             101.0;
      mass                      -----;
      centreOfMass              (0 -0.2327 0);
      momentOfInertia           (0.318 0.0054 0.0625);
      //velocity                (0 0 0);
      report                    on;
      accelerationRelaxation    0.95;
      accelerationDamping       0.98;

      solver
      {
            type Newmark;
      }

      constraints
      {
            fixedLine1
            {
                  sixDoFRigidBodyMotionConstraint  line;
                  direction                        (0 1 0);
            }
            fixedLine2
            {
                  sixDoFRigidBodyMotionConstraint  line;
                  direction                        (1 0 0);
            }
            fixedOrientation
            {
                  sixDoFRigidBodyMotionConstraint  orientation;
            }
      }
}
```

In overset simulation inner distance and outer distance are specified far away from the computational domain to prevent mesh morphing. In the initial position, the piston is fixed with three constraints which limit the degrees of freedom from six to zero. Fixing the piston is necessary since physical walls are not allowed to collide. Constraint `fixedLine2` can be removed to allow one dimensional translation when acceleration of the piston has positive values in the y-direction. Initial velocity will be added to model the rebound of the piston after hitting the bit.

APPENDIX B: ADDITIONAL PIMPLE SOLVER SETTINGS

Additional solver settings used in the simulations are presented. These settings are turned off by default.

- *momentumPredictor*: solves a momentum predictor using momentum equation and pressure from a previous time step. Momentum predictor is solved at the start of a time step before pressure equation and it can help convergence and stability in some cases.
- *transonic*: sets to solve the transonic algorithm that helps mass conservation in transonic flows.

```
if (pimple.transonic())
{
     phid
     (
          "phid",
          (fvc::interpolate(psi)/fvc::interpolate(rho))*phiHbyA
     );

     phiHbyA -=
     fvc::interpolate(psi*p)*phiHbyA/fvc::interpolate(rho);

     fvScalarMatrix pDDtEqn
     (
          fvc::ddt(rho) + psi*correction(fvm::ddt(p))
          + fvc::div(phiHbyA) + fvm::div(phid, p)
          ==
          fvOptions(psi, p, rho.name())
     );
```

  Transonic algorithm makes the solution sensitive to boundary conditions and requires care when used. If non-orthogonality correction is used, additional term is added to the transonic algorithm.
- *oversetAdjustPhi*: flux correction for overset simulations. As the overset method is non-conservative and may lead to pressure fluctuations in closed domain, use of flux adjustment is recommended.
- *rhoMin* and *rhoMax*: can be used to limit minimum and maximum values of density. Limiting density is a useful option to prevent unrealistic values from numerical errors. Especially negative density values lead to solver crash.
- *turbOnFinalIterOnly*: sets to solve turbulence equations after each SIMPLE loop increasing solution accuracy and stability. By default, turbulence equations are solved only once at the end of the PIMPLE loop.

APPENDIX C: FVSCHEMES

```
FoamFile
{
     version         2.0;
     format          ascii;
     class           dictionary;
     object          fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
     default                    Euler; //CrankNicolson 0.5;
}

gradSchemes
{
     default                    cellLimited Gauss linear 0.5;
     grad(U)                    cellLimited Gauss linear 0.5;
}

divSchemes
{
     div(phi,U)                 Gauss linearUpwind grad(U);
     div(phi,h)                 Gauss linearUpwind grad(h);
     div(phi,K)                 Gauss linearUpwind grad(K);
     div(phiv,p)                Gauss linearUpwind;
     div(phi,B)                 Gauss linearUpwind;
     div(meshPhi,p)             Gauss linearUpwind grad(p);
     div(B)                     Gauss linearUpwind;
     div(rhoPhi,U)              Gauss linearUpwind;
     div(U)                     Gauss linearUpwind;
     div(phid,p)                Gauss linearUpwind grad(p);

     div(phi,omega)             Gauss upwind default;
     div(phi,k)                 Gauss upwind grad(k);

     div(((rho*nuEff*dev2(T(grad(U))))    Gauss linear;
}

laplacianSchemes
{
     default                    Gauss linear limited 0.5;
}
```

```
interpolationSchemes
{
      default                     linear;
}

snGradSchemes
{
      default                     limited 1.0;
}

oversetInterpolation
{
      method                      inverseDistance;
}

wallDist
{
      method                      Poisson;
}
```

APPENDIX D: FVSOLUTION

```
FoamFile
{
      version         2.0;
      format          ascii;
      class           dictionary;
      object          fvSolution;
}
// * * * * * * * * * * * * * * *  * * * * * * * * * * * * * * * //

solvers
{
      p
      {
            solver                      PBiCGStab;
            preconditioner              DILU;
            tolerance                   1e-07;
            relTol                      0;
      }

      pFinal
      {
            solver                      PBiCGStab;
            preconditioner              DILU;
            tolerance                   1e-07;
            relTol                      0;
      }

      "(k|omega)"
      {
            solver                      smoothSolver;
            smoother                    symGaussSeidel;
            tolerance                   1e-6;
            relTol                      0;
      }

      "(k|omega)Final"
      {
            solver                      smoothSolver;
            smoother                    symGaussSeidel;
            tolerance                   1e-6;
            relTol                      0;
      }
```

```
    rho
    {
        solver                  smoothSolver;
        smoother                symGaussSeidel;
        tolerance               1e-8;
        relTol                  0;
    }

    rhoFinal
    {
        solver                  smoothSolver;
        smoother                symGaussSeidel;
        tolerance               1e-8;
        relTol                  0;
    }

    hFinal
    {
        solver                  smoothSolver;
        smoother                symGaussSeidel;
        tolerance               1e-6;
        relTol                  0;
    }

    h
    {
        solver                  smoothSolver;
        smoother                symGaussSeidel;
        tolerance               1e-6;
        relTol                  0;
    }

    yPsi
    {
        solver                  PBiCGStab;
        preconditioner          DILU;
        tolerance               1e-6;
        relTol                  0;
    }
}

PIMPLE
{
    momentumPredictor           false;
    transonic                   true;
    nOuterCcorrectors           15;
    nCorrectors                 1;
    nNonOrthogonalCorrectors    0;
    oversetAdjustPhi            true;
```

```
    rhoMin rhoMin                   [1 -3 0 0 0] 0.7;
    rhoMax rhoMax                   [1 -3 0 0 0] 11.0;
    turbOnFinalIterOnly             false;

    residualControl
    {
        h
        {
            relTol              0;
            tolerance           1e-3;
        }
        p
        {
            relTol              0;
            tolerance           5e-3;
        }
        "(k|omega)"
        {
            relTol              0;
            tolerance           5e-3;
        }
        rho
        {
            relTol              0;
            tolerance           5e-3;
        }

    }
}

relaxationFactors
{
    fields
    {
        p                   1;
        pFinal              1;
        rho                 1;
        rhoFinal            1;

    }
    equations
    {
        h                   0.9;
        hFinal              1;
        "k|omega"           0.9;
        "(k|omega)Final"    1;
    }
}
```

APPENDIX E: FVOPTIONS

```
FoamFile
{
      version         2.0;
      format          ascii;
      class           dictionary;
      location        "constant";
      object          fvOptions;
}
// * * * * * * * * * * ** * * * * * * * * * * * * * * * * * * //

porosity1
{
      type            explicitPorositySource;
      explicitPorositySourceCoeffs
      {
            selectionMode              cellZone;
            cellZone                   porousZone;
            type                       DarcyForchheimer;
            d                          (5e13 5e13 5e13);
            f                          (0 0 0);
            coordinateSystem
            {
                  type                 cartesian;
                  origin               (0 0 0);
                  coordinateRotation
                  {
                        type           axesRotation;
                        e1             (1 0 0);
                        e2             (0 0 1);
                  }
            }
      }
}

source1
{
      type                       limitTemperature;
      selectionMode              all;
      active                     true;
      min                        270;
      max                        400;
}
```

```
source2
{
      type                          limitVelocity;
      selectionMode                 all;
      active                        true;
      min                           0;
      max                           400;
}
```

APPENDIX F: DYNAMICMESHDICT

```
FoamFile
{
      version         2.0;
      format          ascii;
      class           dictionary;
      object          dynamicMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

motionSolverLibs
(
      "libsixDoFRigidBodyMotion.so"
      "libfvMotionSolvers.so"
);

dynamicFvMesh                   dynamicOversetFvMesh;
solver                          sixDoFRigidBodyMotion;
sixDoFRigidBodyMotionCoeffs
{
      patches         (pistonWalls innerClearance outerClearance);
      innerDistance         100.0;
      outerDistance         101.0;
      mass                  -----;
      centreOfMass          (0 -0.2327 0);
      momentOfInertia (0.318 0.0054 0.0625);
      velocity              (0 5 0);
      report                          on;
      accelerationRelaxation          1;
      accelerationDamping             0.99;
      solver
      {
            type            Newmark;
      }
      constraints
      {
            fixedLine1
            {
                  sixDoFRigidBodyMotionConstraint  line;
                  direction (0 1 0);
            }

            fixedOrientation
            {
                  sixDoFRigidBodyMotionConstraint  orientation;
            }
      }
}
```