Modelling the Dewatering in the Forming Section of a Paper Machine

Kimmo Hentinen

Master’s thesis
University of Eastern Finland
Department of Physics and Mathematics
August 2010
Papermaking with modern paper machines is a big business around the world. In papermaking process wood fibres, additives and fines are mixed with water. This suspension is spread into a continuous layer. Water is then removed from the suspension layer and finally paper is formed. The water removal process is called dewatering.

Since the forming section is the first section of a paper machine participating in the dewatering process, the dewatering has a great impact on paper quality, and thus it is important to study the fluid flow in the forming section. In this thesis the dewatering process in the forming section of a paper machine is studied with modelling. Using a mathematical model the understanding about the dewatering can be increased and it is possible to affect the dewatering and thereby the produced paper.

Suspension and dewatering can be modelled using fluid dynamics. In this thesis the dewatering in the forming section is studied using computational fluid dynamics (CFD). Suspension flow in the forming section can be categorized as a multiphase flow. In addition the suspension flow geometry, especially at the beginning of the forming section, is not rigid but takes a shape according to the fluid flow. These properties make the modelling of the dewatering a difficult task. In this thesis two different models are used for describing fluid flow in the forming section: the boundary condition model (BCM) and the forming fabric model (FFM). For simplicity both models use a fixed geometry and treat the suspension only as water. Free open source program OpenFOAM is used as a modelling tool and one of the objectives is to study the usability of the program.

The numerical results show that modelling this phenomenon is a very challenging task. The results obtained with the boundary condition model do not compare that well with the previous studies. However, the results obtained with the forming fabric model compare better with the results from the previous studies. They also seem to be reasonable when compared to the reference results in two-dimensional channels with impermeable walls. Thus, the forming fabric model proved to be better for the dewatering modelling. In order to model this phenomenon more accurately fluid structure interaction and multiphase flow modelling would be required.
Acknowledgements

This work was carried out in the Department of Physics and Mathematics at the University of Eastern Finland, former University of Kuopio, between September 2008 and August 2010. Most of all I would like to thank my supervisor and leader of the Paper Physics Group Jari Hämäläinen for giving this opportunity to work with this thesis and of course for his guidance and support. I would also like to thank my other supervisors Heidi Niskanen and Elina Madetoja as well for their guidance and great support in this thesis and my studies in general. This would not have been possible without you. I want to give great shout-out for all the Paper Physics Group members that I have worked with. Their attitude and dedication to their work has also pushed me further in my doings.

I want to thank my family for their support and of course all of my friends who have helped me throughout these years. You know who you are. Last of all I would like to express my gratitude to the open source community in general. Not only for making this thesis possible but for your altruistic work in developing better and better tools and programs for everybody to use. Keep on rockin’ in the free world.

Kuopio, August 2010

Kimmo Hentinen
Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD</td>
<td>machine direction</td>
</tr>
<tr>
<td>CD</td>
<td>cross-machine direction</td>
</tr>
<tr>
<td>CFD</td>
<td>computational fluid dynamics</td>
</tr>
<tr>
<td>BCM</td>
<td>boundary condition model</td>
</tr>
<tr>
<td>FFM</td>
<td>forming fabric model</td>
</tr>
<tr>
<td>FVM</td>
<td>finite volume method</td>
</tr>
<tr>
<td>CV</td>
<td>control volume</td>
</tr>
<tr>
<td>2D</td>
<td>two-dimensional</td>
</tr>
<tr>
<td>3D</td>
<td>three-dimensional</td>
</tr>
<tr>
<td>p</td>
<td>pressure</td>
</tr>
<tr>
<td>ρ</td>
<td>density</td>
</tr>
<tr>
<td>µ</td>
<td>viscosity</td>
</tr>
<tr>
<td>µT</td>
<td>turbulent viscosity</td>
</tr>
<tr>
<td>µeff</td>
<td>effective viscosity</td>
</tr>
<tr>
<td>Re</td>
<td>Reynolds number</td>
</tr>
<tr>
<td>m</td>
<td>mass</td>
</tr>
<tr>
<td>Ω</td>
<td>arbitrary domain</td>
</tr>
<tr>
<td>x̄</td>
<td>position vector</td>
</tr>
<tr>
<td>t</td>
<td>time</td>
</tr>
<tr>
<td>Γ</td>
<td>domain boundary</td>
</tr>
<tr>
<td>u̅</td>
<td>velocity</td>
</tr>
<tr>
<td>n̅</td>
<td>surface normal vector</td>
</tr>
<tr>
<td>DDU</td>
<td>material derivative</td>
</tr>
<tr>
<td>P̅</td>
<td>momentum</td>
</tr>
<tr>
<td>f̅</td>
<td>external force vector</td>
</tr>
<tr>
<td>σ̅</td>
<td>stress tensor</td>
</tr>
<tr>
<td>σ̅k</td>
<td>k:th row of stress tensor σ̅</td>
</tr>
<tr>
<td>ˉɛ</td>
<td>rate of strain tensor</td>
</tr>
<tr>
<td>δij</td>
<td>Kronecker delta</td>
</tr>
<tr>
<td>k</td>
<td>turbulent kinetic energy in the k-ε model</td>
</tr>
<tr>
<td>ε</td>
<td>dissipation of turbulent kinetic energy in the k-ε model</td>
</tr>
<tr>
<td>g</td>
<td>acceleration of gravity</td>
</tr>
<tr>
<td>κ</td>
<td>permeability</td>
</tr>
<tr>
<td>ˉD</td>
<td>viscous loss tensor</td>
</tr>
<tr>
<td>ˉF</td>
<td>inertial loss tensor</td>
</tr>
<tr>
<td>i̅</td>
<td>surface tangential vector</td>
</tr>
<tr>
<td>u*</td>
<td>velocity predictor</td>
</tr>
<tr>
<td>α</td>
<td>under-relaxation factor</td>
</tr>
<tr>
<td>Dk</td>
<td>magnitude of the inverse of permeability in tangential direction</td>
</tr>
<tr>
<td>Dn</td>
<td>magnitude of the inverse of permeability in normal direction</td>
</tr>
</tbody>
</table>
1 Introduction

2 On industrial papermaking
  2.1 Dewatering in the forming section
  2.2 Components in the forming sections
    2.2.1 Forming fabrics
    2.2.2 Former types
  2.3 Papermaking process after the forming section

3 Fluid dynamics in the forming section
  3.1 Properties of fluid flows
  3.2 Conservation principles
    3.2.1 Mass conservation
    3.2.2 Momentum conservation
  3.3 The Navier-Stokes equations
    3.3.1 Turbulence modelling
  3.4 Modeling of the dewatering in the forming section
    3.4.1 Gravity
    3.4.2 Darcy’s law
    3.4.3 Governing equations
  3.5 Dewatering models
    3.5.1 The boundary condition model
    3.5.2 The forming fabric model

4 Numerical solution with CFD
  4.1 System of linear equations
    4.1.1 Defining a system of linear equations
    4.1.2 Linearization of the Navier-Stokes equations
    4.1.3 Derivative approximations
  4.2 Solving the Navier-Stokes equations with FVM
    4.2.1 Introduction to the finite volume method
4.2.2 Interpolation practices ........................................... 38
4.2.3 Approximation of the surface and volume integrals ......... 38
4.2.4 Velocity and pressure coupling ............................... 39
4.2.5 Under-relaxation .................................................. 40

5 Numerical experiments .................................................. 42
5.1 Reference simulations ............................................... 42
  5.1.1 Horizontal 2D channel ......................................... 42
  5.1.2 Gap former geometry with impermeable walls ............. 44
5.2 The boundary condition model .................................... 48
  5.2.1 Results ........................................................... 48
5.3 Forming fabric model .............................................. 51
  5.3.1 Results ........................................................... 51

6 Conclusions ............................................................. 54

References ................................................................. 57

Appendices
A OpenFOAM codes .................................................... 60
B OpenFOAM solver configurations ................................... 64
Despite the increase in digital storage of information, paper is still used every day. Thus paper industry is large business and papermaking process is widely studied [3, 4, 9, 10, 12, 13, 14, 15, 16, 18, 23, 26]. Paper is made first by mixing up water, wood fibres, additives and fines [27, 29]. This suspension is then spread on or between moving, porous fabrics. At the papermaking process water is removed from the suspension and fibres, additives and fines finally form the paper. In modern paper machines water is removed from the suspension through moving, porous fabrics. This water removal process is called dewatering and it starts at the forming section of a paper machine, where most of the dewatering takes place.

After the forming section fibres do not move respect to each other. Location and size of fibres defines the formation, i.e. small scale basis weight variations [23]. Eliminating these variations is important for all paper grades so the forming section becomes very important when the paper quality is concerned [29]. Formation of the paper depends on the fibre accumulation between the forming fabrics and accumulation is directly related to dewatering [27]. Thus by modelling the dewatering in the forming section we can gain knowledge about the fibre accumulation and the formation.

In this thesis dewatering occuring in the forming section is studied. The main focus is on the modelling of the dewatering process. The dewatering is modelled using computational fluid dynamics (CFD) and two different models for the dewatering are presented. First, we describe papermaking and dewatering in more depth, secondly fluid dynamics related to the dewatering and numerical solving with CFD are discussed. Then the models for the dewatering are presented and the modelling results are shown. Finally results are discussed and suggestions for future studies are presented.

Open source CFD toolbox OpenFOAM [24, 25] is used as the modelling tool in this thesis. In addition to modelling the dewatering one aim of this thesis is consider the usability of OpenFOAM. OpenFOAM is probably the most widely used open
source CFD code available and the abstraction level of the programming is high. Thus it should offer an efficient tool for creating or modifying already developed models. Furthermore, the actual source codes developed for the models used in this thesis are presented in appendices.
 Throughout the history mankind has been writing or drawing things it has experienced. For thousands of years paper, made out of different materials, has been used for this purpose. In future though, paper may have smaller role as a writing equipment than, for example, as a hygienic product or for packaging. Papermaking is a very large and sophisticated business nowadays. Due to the climate change, for example, the environmental issues are among the greatest challenges of the papermaking industry.

Paper is usually manufactured from wood fibres [27, 29]. The fibres are separated from wood by mechanical or chemical treatment. After the separation the aim is to distribute fibres so that they form a thin sheet. This sheet of wood fibres is called paper. In industrial papermaking fibres are mixed with water to form a fibre-water suspension or suspension for short. The suspension is spread on or between moving, porous fabrics which allow the water to be removed through the fabrics. Wood fibres pile up above or between the fabrics to form a continuous uniform fibre sheet. This sheet is processed further in different stages. In the last stage the continuous fibre sheet is reeled into huge rolls. This way a continuous papermaking process is achieved. Physical dimensions of a paper machine can be in a scale of hundred meters long and ten meters wide. In papermaking the longitudinal direction is referred as the machine direction (MD) and the width direction is the cross-machine direction (CD). Perpendicular with respect to both of this directions is the thickness direction of a paper sheet which is referred as ZD.

From wood fibres to ready paper the stages of industrial papermaking can be separated into the following:

- stock preparation
- headbox
- forming
2. On industrial papermaking

- wet pressing
- drying
- calendering and/or coating
- reeling.

In this thesis the focus is on the forming section. In the forming section the water is removed through the fabric from the suspension i.e. dewatering takes place and the fibres accumulate on or between the moving fabrics. In processes after the forming section there is some fluid removal occurring but these phenomena are not discussed in this thesis.

2.1 Dewatering in the forming section

If we want to understand dewatering phenomenon we must understand what has happened earlier in the papermaking process. Before the headbox and the forming section the stock preparation stage is the first stage of papermaking. In stock preparation wood fibres are mixed with water and possible additives and fines. From stock preparation the suspension is transported to the headbox of a paper machine.

The structure of a headbox in a modern paper machine is illustrated in Figure 2.1. First the suspension is led through a header from where the fibre suspension is led into manifold tube bank which spreads the suspension onto the whole width of the paper machine. Additional dilution water can be added to the suspension in manifold tube bank to control the suspension concentration in CD [27]. Manifold tube bank leads the suspension to an equalizing chamber from where the suspension

Figure 2.1: The structure of a modern headbox. By courtesy of Metso Paper, Inc.
continues to another set of pipes called a turbulence generator and after that to a slice channel. The slice channel ends to a slice opening from where the suspension is sprayed onto the forming section. Elastic plates, i.e. vanes, can be used in the slice channel to reduce large scale fluctuations and to maintain turbulence production [27]. Some older machines, or machines which produce special paper grades, can have another type of structure than headbox presented in Figure 2.1.

The main task of the headbox is to produce the right type of jet for the forming section. This means even mass distribution in CD, turbulence generation for breaking up fibre flocs, i.e. small fibre aggregations, and producing stable jet by stabilizing pressure. Jet leaving from the slice opening has a certain fibre to water mass ratio (concentration) depending on the paper grade. Usually this is $0.5 - 1.0\%$ [29].

From the headbox the fibre suspension is sprayed onto the forming section which consists of a continuous moving fabric or fabrics called forming fabrics (or wires). Dewatering of the suspension is the main task of the forming fabrics in the forming section. Another important task of the forming fabrics and the forming section is to pass the developed sheet of fibres forward in the process. In modern gap formers the forming section consists of two forming fabrics and the fibre suspension is sprayed between these two fabrics. Traditional fourdrinier former consists of only one fabric on which the suspension is sprayed. Therefore in fourdrinier former the dewatering occurs only in one direction through the forming fabric [27].

In papermaking process most of the dewatering occurs in the forming section; up to 98% of the whole mass flow is removed. When the suspension leaves the forming section it has a concentration between $15 - 22\%$ depending on the paper grade [29]. Dewatering can be enforced by vacuum on the other side of the forming fabric. Vacuum can be generated inside the forming roll, separate suction boxes or zones. In addition, vacuum can be produced using blades in contact with the forming fabrics. Dewatering components will be discussed more when different former types are discussed.

Dewatering phenomenon has a major influence on how evenly the fibres are distributed in the paper sheet. In the forming section accumulation of fibres forms the initial sheet of fibres which is called the wet web. This sheet of fibres is then processed forward in the process. After the forming section fibre displacement or orientation do not change that much and thus formation depends greatly on the dewatering. The formation is one measure for the paper quality. Thereby it is obvious that paper industry is very interested in what happens in the forming section. Furthermore, the energy needed for dewatering increases as the water content in the paper web decreases [18]. In the press section dewatering is based on mechanical pressure but at the drying section dewatering happens through evaporation. In the drying section rolls have to be heated and that requires a lot of energy. Removing water as much as possible mechanically, reduces the energy cost of the dewatering by drying.
2. On industrial papermaking

2.2 Components in the forming sections

2.2.1 Forming fabrics

Forming fabrics are planar, continuous plastic wovens revolving inside a paper machine [27]. They act as a smooth support base for the fibre suspension and as a filtration medium. The fibres that build up above or in between the forming fabrics form the wet web. Thus, the forming fabrics have a major effect on the paper quality.

Commonly used fabric structures are so called single-layer (SL), double-layer (DL), triple-layer (TL), triple-weft (TW) and self support binding (SSB) structures [29]. Names refer to the number of fabric filament layers used to weave the fabric. Single-layer forming fabric consists of only one layer of filament in each of the two directions. Figure 2.2 shows both sides of a single-layer forming fabric. Paper side is the side in contact with the suspension and fibres and wear side is the side in contact with the machine. Cross-section views of DL, TL and SSB structures can be seen in Figure 2.3. With different weaving methods different dewatering and structural properties for the fabric are achieved.

The forming fabric has two important purposes: dewatering and retention. By retention we mean the ratio of how much of fibre medium is left for forming of the wet web and how much is wasted during dewatering [27]. At the beginning of the dewatering a layer of fibres is quickly formed on or between the fabrics. This layer acts as a filtration base for later dewatering and wet web forming. Dewatering characteristics depend greatly on the former design, forming fabric structure, stock preparation and running parameters of the forming section.

![Single-layer forming fabric](image-url)
As well as dewatering, the retention depends on the machine running parameters and the former design. The most important thing is the forming fabric structure and especially the pore size. With too big pore size the dewatering would be fast but too much fibres would be wasted with the removed water. Too small pore size means better retention but dewatering would be slower [27]. Thus, a compromise has to be made between these two objectives. As one can see, the forming section also affects the efficiency of the papermaking process and thus this makes it even more important stage in the process.

### 2.2.2 Former types

**Fourdriner former**

Fourdriner is the oldest former type in papermaking process. In this kind of former the jet from the slice opening is sprayed on top of the forming fabric moving only in horizontal direction. Figure 2.4 shows an example of a fourdriner former. Headbox is on the left and continuous forming fabric revolves in the former in clockwise direction. At the end of the forming section wet web leaves the former and is transported forward in the process. In this type of former dewatering occurs only
2. On industrial papermaking

downwards through the forming fabric. Though fourdriner is the oldest former type (introduced around 1820), it is still widely used. Originally gravity was the only force that generated dewatering in the forming section. Later on there has been other dewatering elements such as foil elements, dry and wet suction boxes [27], which can be seen as yellow ”boxes” in Figure 2.4. In addition rolls can be used to press wet web to squeeze water out through the fabric. Fourdriner formers have the advantage of a gentle dewatering and a long dewatering time which are required e.g. for some special paper grades. The problem with fourdriner formers is the one-sided dewatering which causes the paper to have different properties on different sides.

TWIN-WIRE FORMERS

Along with the higher productivity needs came the need for a higher machine speed. With the fourdriner former there was a problem with stable dewatering at higher running speeds. Solution was to use two forming fabrics, one on both side of the wet web to stabilize the dewatering [27]. With this solution the problem related to the free surface between the wet web and air creating friction was avoided. At the same time dewatering became faster because water was removed into two directions.
First twin-wire formers were developed in 1950s. Nowadays these formers can be divided in two types: hybrid formers and gap formers. As the name indicates the hybrid formers use both the traditional fourdriner forming and twin-wire forming. An example of this kind of former can be seen in Figure 2.5. Apart from the upper forming fabric the structure of this hybrid former is quite similar to the fourdriner former in Figure 2.4.

![Figure 2.5: An example of a hybrid former. © KnowPap.](image)

In gap formers the jet is sprayed directly between the two forming fabrics. Dewatering takes place through both fabrics. Thus, the forming section length can be much shorter than with fourdriner or hybrid formers. There is no need for horizontal
forming fabrics and therefore the headbox angle can vary a lot. In Figure 2.6 there is an example of a gap former. The headbox is down on the centre and it is followed by two rolls which create the gap for the jet. The left roll is called the forming roll. In gap formers the dewatering is mainly generated by dewatering elements and the fabric tension instead of gravity. Similarly to hybrid formers there are dewatering elements on both sides of the wet web. Dewatering can be increased by creating a high vacuum inside the forming roll. Initial dewatering in a modern gap former right after the headbox slice opening can be seen in Figure 2.7.

**Figure 2.6:** An example of a gap former. By courtesy of Metso Paper, Inc.

**Figure 2.7:** Jet from the headbox between the two forming fabrics and initial dewatering. By courtesy of Metso Paper, Inc.

The use of twin-wire formers, such as gap former, gives the following advantages:
increased dewatering capacity, more symmetric top and bottom side of the paper, lower basis weight variability, better formation and lower linting [27].

One important property in the forming section is the jet-to-wire ratio. When the headbox jet and the forming fabrics have different speeds the ratio is called the jet-to-wire speed ratio and it has a major effect on fibre orientation anistropy, for example. The preferred jet-to-wire ratio depends on the produced paper grade and the forming fabric type. For example, speed of $27 \frac{m}{s}$ for the fabrics and $30 \frac{m}{s}$ for the jet gives jet-to-wire ratio 1.11. Typically the ratio has a value close to 1.

### 2.3 Papermaking process after the forming section

After the forming section the wet web is called paper web. In wet pressing the paper web is mechanically pressed using rolls. This process squeezes the water out from the paper web. At the drying section paper web is dried by heating it with steam-heated rolls to cause evaporation [16]. After the wet pressing the dry material mass concentration can be up to 50%, and in the drying section it is increased further to preferred level depending on the paper grade [29].

After the drying section paper can be calendared and/or coated. In calendering paper is pressed between series of rolls to make it smoother. Paper can be coated with some material in order to have certain quality properties, e.g. smoothness and gloss, for the ready paper. Last stage of the continuous papermaking process is reeling where paper is rolled and removed from the process. After this the paper is cut and processed further to be delivered to the customers.
Chapter 3

Fluid dynamics in the forming section

Fluid behaviour can be studied with fluid mechanics. The study of fluid mechanics can be divided into two parts: fluids in motion (fluid dynamics) and fluids at rest (fluid statics) [2]. With these definitions it is obvious that fluid mechanics is present in various phenomena e.g. breathing, blood flow, fans, airplanes and swimming. The suspension flow in paper machines is a phenomenon which can be seen as a part of fluid dynamics. Fluid dynamics provides the theory for describing the dewatering of the fibre suspension in the forming section. Therefore it is essential for this thesis to consider fluid dynamics.

Fluid dynamics can be divided into three parts: analytical fluid dynamics (AFD), experimental fluid dynamics (EFD) and computational fluid dynamics (CFD). AFD would be the most accurate method but it can be used only in some special cases. With EFD we would get flow properties, e.g. velocity and pressure, in actual fluid flow domain or in a scale model if that is used. In EFD the objective is to obtain accurate results with measurement methods without affecting the fluid flow. These properties are quite hard to attain at the same time. In addition EFD often requires a lot of work and equipment which makes it is financially expensive.

In CFD computers are used to obtain approximate solution for fluid flow. With nowadays computers it is possible to solve CFD models in reasonable time and therefore it is quite cheap and effective way to study fluid flow. Next we discuss basic fluid properties and characteristics as well as equations that govern the fluid flow. When a mathematical model for the fluid flow is obtained, an approximate solution can be obtained by using numerical methods to solve the model. Fluid mechanics is a very broad field and it would take several books to cover all the theory. Thus, we concentrate on the issues being important for the model used in this thesis.
3. Fluid dynamics in the forming section

3.1 Properties of fluid flows

For fluid mechanics all matter consists of solid or fluid. According to one definition a solid can resist a shear stress by a static deflection but a fluid cannot [30]. Fluid can refer to gas or liquid. The difference between these phases are the cohesive forces between the fluid particles. These particles are usually atoms or molecules. In liquids particles are closely packed and cohesive forces are strong. With these properties liquids tend to retain their volume. In gases cohesive forces are not that strong and particles can move quite freely.

Fluids have many different properties but when fluid flow is considered, the most important features are density and viscosity. Density has the units of \( \rho = \text{kg/m}^3 \), and it is affected by the temperature and the internal pressure of the fluid. The pressure effect is called compressibility of fluids. Gases are more compressible than liquids which are nearly incompressible.

Viscosity is denoted by \( \mu \) and has the dimension of \( \mu = \text{Pa s} = \text{kg/ms} \) in SI units. It can be described as the ”fluidity” of the fluid [21]. Viscosity describes how much the fluid resists the deformation caused by internal or external forces.

Fluids can be classified into Newtonian, non-Newtonian and generalized Newtonian fluids. In Newtonian fluid the shearing stress is linearly related to the rate of shearing strain. For non-Newtonian or generalized Newtonian fluids these are not necessarily linearly related. Generalized Newtonian fluids can be shear thickening or shear thinning. One example of generalized Newtonian fluid is called Bingham plastic fluid [2]. A certain amount of shear stress has to be applied on the Bingham plastic fluid to get it into motion and after that the relation is linear.

Fluids obey Newton’s laws in the same way as solids. Forces acting on fluid can be divided to surface forces and body forces. Gravity is a good example of a body force and wind blowing on the lake is an example of a surface force. Usually in man made machines the fluid flow is caused by pressure difference between certain points. Fluid flow is said to be natural or forced depending on the reason of the flow [2]. Natural flows are caused by natural means such as gravity or buoyancy effect. Instead, forced flows are caused by external means such as a pump or a fan. No matter what causes the fluid flow, the flow depends on certain fluid properties and it can be studied with equations derived from the conservation principles.

Depending on the fluid velocity the flow can be described as laminar or turbulent. At lower velocities the flow is laminar, meaning smooth and steady. At higher velocities the flow becomes fluctuating and unsteady, then flow is said to be turbulent. Nature of the flow depends also on the dimensions of the flow domain. Fluid flow can be described using a dimensionless number called the Reynolds number defined as

\[
Re := \frac{\rho UL}{\mu},
\]
where $U$ is the mean fluid velocity and $L$ is the characteristic length. At low Reynolds numbers flows are laminar and at higher values flows are turbulent. Change between laminar and turbulent flow is not distinct. There is a transition which means that there is no accurate value for Reynolds number where a flow can be considered to be laminar or turbulent. In general it can be said that transition to turbulent happens at Reynolds number between $1000 - 10000$ [30].

3.2 Conservation principles

Fluid flows can be considered using equations derived from mass and momentum conservation principles. Next, conservation principles for mass and momentum are derived. Main references for this section are [2, 11].

3.2.1 Mass conservation

Let $\Omega$ be a body with a constant volume. In addition let $\Omega$ be a closed system, and thus the mass $m$ of the body is constant. This means that the time derivative of the mass is zero, which can be expressed as

$$\frac{dm_{\Omega(t)}}{dt} = 0.$$  \hspace{1cm} (3.2)

Using Reynolds transport theorem [2] the derivative of the mass can be written as

$$\frac{dm_{\Omega(t)}}{dt} = \frac{d}{dt} \int_{\Omega(t)} \rho \, d\Omega + \int_{\Gamma(t)} \rho(\vec{u} \cdot \vec{n}) \, dS = 0,$$  \hspace{1cm} (3.3)

where $\rho = \rho(\vec{x}, t)$ is the density of a fluid in point $\vec{x}$ at time $t$, $\Gamma(t)$ is the boundary of $\Omega(t)$, $\vec{u}$ is the velocity and $\vec{n}$ is the normal vector pointing outwards from the surface $\Gamma$ (see Figure 3.1). Volume of the body does not depend on time, so the derivative can be taken inside the integral. We assume that $\Omega(t)$ is a compact subset of $\mathbb{R}^n$, $\Gamma(t)$ is a piecewise smooth surface and $\rho(\vec{u} \cdot \vec{n})$ is a continuously differentiable
3. Fluid dynamics in the forming section

function in \( \Omega \). With these assumptions we can use Gauss’ divergence theorem to transform the surface integral into a volume integral. Hence, Equation (3.3) can be written as

\[
\int_{\Omega(t)} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) \, d\Omega = 0. \quad (3.4)
\]

Equation (3.4) must be valid for arbitrary volume \( \Omega(t) \), therefore there must apply that

\[
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0. \quad (3.5)
\]

This equation is called the continuity equation. By using three-dimensional (3D) Cartesian coordinates Equation (3.5) can be written in a form

\[
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = \frac{\partial \rho}{\partial t} + \sum_{i=1}^{3} \frac{\partial (\rho u_i)}{\partial x_i} = 0. \quad (3.6)
\]

If the fluid density is assumed to be constant, Equation (3.5) simplifies

\[
\nabla \cdot (\rho \vec{u}) = 0 \Rightarrow \nabla \cdot \vec{u} = 0. \quad (3.7)
\]

Equation (3.4) can also be written as

\[
\int_{\Omega(t)} \left( \frac{\partial \rho}{\partial t} + \nabla \rho \cdot \vec{u} + \rho \nabla \cdot \vec{u} \right) \, d\Omega = 0. \quad (3.8)
\]

From the previous equation the first and the second term inside the integral define the material derivative

\[
\frac{D \rho}{Dt} := \frac{\partial \rho}{\partial t} + \nabla \rho \cdot \vec{u}. \quad (3.9)
\]

Material derivative will be used later in the derivation of the momentum conservation principle.

3.2.2 Momentum conservation

According to Newton’s second law the derivative of the momentum \( \vec{p}_{\Omega(t)} \) respect to time must be equal to the sum of external forces, which can be divided into body forces and surface forces written as

\[
\frac{d\vec{p}_{\Omega(t)}}{dt} = \int_{\Omega(t)} \rho \vec{f} \, d\Omega + \int_{\Gamma(t)} \vec{\sigma} \cdot \vec{n} \, dS, \quad (3.10)
\]

where \( \vec{f} = (f_1, f_2, f_3)^T \) is the sum of body forces per unit mass and surface forces are denoted by the stress tensor \( \vec{\sigma} \).
3. Fluid dynamics in the forming section

Equation (3.10) can be written using 3D Cartesian component as

$$
\frac{d\vec{p}_{k}(\Omega(t))}{dt} = \int_{\Omega(t)} \rho f_k \, d\Omega + \int_{\Gamma(t)} \vec{\sigma}_k \cdot \vec{n} \, dS, \quad k = 1, 2, 3, \tag{3.11}
$$

where $\vec{\sigma}_k$ is the $k$:th row of tensor $\vec{\sigma}$. Using Reynolds transport theorem, Gauss’ divergence theorem, Equation (3.8) and material derivative notation the momentum derivative respect to time can be written as

$$
\frac{d\vec{p}_{k}(\Omega(t))}{dt} = \int_{\Omega(t)} \left[ D\left( \rho u_k \right) + (\rho u_k)(\nabla \cdot \vec{u}) \right] \, d\Omega
$$

$$
= \int_{\Omega(t)} \left[ u_k \left( \frac{D\rho}{Dt} + \rho \nabla \cdot \vec{u} \right) + \rho \frac{Du_k}{Dt} \right] \, d\Omega
$$

$$
= \int_{\Omega(t)} \rho \frac{Du_k}{Dt} \, d\Omega, \quad k = 1, 2, 3, \tag{3.12}
$$

where $u_k$ is the $k$:th velocity component. We use Gauss’ divergence theorem to write the second term on the right hand side of Equation (3.11) as

$$
\int_{\Gamma(t)} \vec{\sigma}_k \cdot \vec{n} \, dS = \int_{\Omega(t)} \nabla \cdot \vec{\sigma}_k \, d\Omega. \tag{3.13}
$$

Substituting Equations (3.12) and (3.13) into Equation (3.11) gives

$$
\int_{\Omega(t)} \rho \frac{Du_k}{Dt} \, d\Omega = \int_{\Omega(t)} \rho f_k \, d\Omega + \int_{\Omega(t)} \nabla \cdot \vec{\sigma}_k \, d\Omega, \quad k = 1, 2, 3. \tag{3.14}
$$

This equation has to apply for an arbitrary volume $\Omega(t)$, thus the momentum equation is written as

$$
\rho \frac{Du_k}{Dt} = \rho f_k + \nabla \cdot \vec{\sigma}_k, \quad k = 1, 2, 3. \tag{3.15}
$$

When this is replaced into Equation (3.14) and the equation is written in a vector form, we get

$$
\rho \frac{\partial \vec{u}}{\partial t} + \rho \nabla \vec{u} \cdot \vec{u} = \rho \vec{f} + \nabla \cdot \vec{\sigma}. \tag{3.16}
$$

For Newtonian fluids the stress tensor $\vec{\sigma}$ can be written as

$$
\vec{\sigma} = 2\mu \vec{\varepsilon} - \left( p + \frac{2}{3} \mu \nabla \cdot \vec{u} \right) I, \tag{3.17}
$$

where $I$ is the unit tensor and $p$ is the static pressure. Assuming constant density and using the continuity equation (3.7) the second term inside the brackets is zero. Tensor $\vec{\varepsilon}$ is the rate of strain defined as

$$
\varepsilon_{ij} = \frac{1}{2} \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \quad i, j = 1, 2, 3. \tag{3.18}
$$
The stress tensor components can be now written as

$$\sigma_{ij} = 2\mu\epsilon_{ij} - p\delta_{ij} = \mu \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - p\delta_{ij}, \quad (3.19)$$

where $\delta_{ij}$ is the Kronecker delta defined as

$$\delta_{ij} = \begin{cases} 
1, & \text{if } i = j \\
0, & \text{if } i \neq j. 
\end{cases} \quad (3.20)$$

The divergence of the stress tensor $\bar{\sigma}$, appearing in Equation (3.16), can be written as

$$\left( \nabla \cdot \bar{\sigma} \right)_i = 3 \sum_{j=1}^{3} \left[ \frac{\partial}{\partial x_j} \left[ \mu \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right] - \frac{\partial p}{\partial x_j} \right], \quad i = 1, 2, 3. \quad (3.21)$$

If we assume the viscosity to be constant, the previous equation can be written in a form

$$\left( \nabla \cdot \bar{\sigma} \right)_i = 3 \sum_{j=1}^{3} \left[ \mu \left( \frac{\partial^2 u_i}{\partial x_j \partial x_j} + \frac{\partial^2 u_j}{\partial x_j \partial x_i} \right) - \frac{\partial p}{\partial x_j} \right], \quad i = 1, 2, 3, \quad (3.22)$$

and further this can be written in a vector form as

$$\nabla \cdot \bar{\sigma} = \mu \left( \nabla^2 \bar{u} + \nabla \cdot (\nabla \cdot \bar{u}) \right) - \nabla p. \quad (3.23)$$

Again using the continuity equation (3.7) the previous equation can be written in a form

$$\nabla \cdot \bar{\sigma} = \mu \nabla^2 \bar{u} - \nabla p. \quad (3.24)$$

Substituting Equation (3.24) into Equation (3.16) gives the momentum conservation equation

$$\rho \frac{\partial \bar{u}}{\partial t} + \rho \nabla \bar{u} \cdot \bar{u} = \rho \bar{f} + \mu \nabla^2 \bar{u} - \nabla p, \quad (3.25)$$

which is also called the momentum equation.

### 3.3 The Navier-Stokes equations

The Navier-Stokes equations are the continuity equation and the momentum equation defined for fluid. For Newtonian fluid stress tensor is assumed to be as in Equation (3.17) and with this assumption the momentum equation is written as in Equation (3.25). In addition in the derivation of the momentum equation we assumed density to be constant which gives incompressible form of the Navier-Stokes equation

$$\begin{cases} 
\rho \frac{\partial \bar{u}}{\partial t} + \rho \nabla \bar{u} \cdot \bar{u} - \mu \nabla^2 \bar{u} = \nabla p + \rho \bar{f} \\
\nabla \cdot \bar{u} = 0. 
\end{cases} \quad (3.26)$$
3. Fluid dynamics in the forming section

These equations are now in general 3D form. If we consider steady-state flow, i.e. velocity field is time independent, Equation (3.26) can be written as

\[
\begin{align*}
\rho \nabla \ddot{u} \cdot \ddot{u} - \mu \nabla^2 \ddot{u} &= -\nabla p + \rho \ddot{f} \\
\nabla \cdot \ddot{u} &= 0.
\end{align*}
\]

(3.27)

Term \( \rho \ddot{f} \) is called the source term. It describes the body forces acting on the fluid.

### 3.3.1 Turbulence modelling

The Navier-Stokes equations apply also for turbulent flows but it is very time-consuming, even for a small volumes, to achieve a numerical solution for the Navier-Stokes equations accurate enough for turbulence modeling. In many applications turbulent flows are present. Thus, turbulent modelling is needed for modelling flows with a high Reynolds number. Many models have been developed to approximate turbulent flows and those models are used in CFD calculations. Most commonly used turbulence models can be classified into three category: direct numerical simulation (DNS), large eddy simulation (LES) and Reynolds averaged Navier-Stokes (RANS) models. Model most commonly used is the \( k-\epsilon \) model, which is a RANS model. The \( k-\epsilon \) model will also be used in this thesis.

In the \( k-\epsilon \) model the time-averaged Navier-Stokes equations are used. This means that velocity and pressure are presented as a sum of the average value and a fluctuation component as

\[
\begin{align*}
u_i &= \bar{u}_i + \ddot{u}_i \\
p &= \bar{p} + \ddot{p},
\end{align*}
\]

where \( \bar{u}_i \) and \( \bar{p} \) are the average values and \( \ddot{u}_i \) and \( \ddot{p} \) are the fluctuation component values. Substituting the previous expressions into Equation (3.27) and integrating the achieved equations with respect to time gives

\[
\begin{align*}
-\nabla \cdot (2\mu \bar{\tau} + \bar{\tau}) + \rho \nabla \ddot{u} \cdot \ddot{u} &= -\nabla \bar{p} + \rho \ddot{f} \\
\nabla \cdot \ddot{u} &= 0,
\end{align*}
\]

(3.29)

where \( \bar{\tau} \) is the fluctuation term known as the Reynolds stress term \([11]\) and \( \bar{\epsilon} \) is the time averaged rate of strain tensor. Next, Reynolds stress is approximated by using the Boussinesq hypothesis

\[
\bar{\tau} = 2\mu_T \bar{\epsilon} - \frac{2}{3} \rho k I,
\]

(3.30)

where \( \mu_T \) is the turbulent viscosity (also eddy viscosity) and \( k \) is the turbulent kinetic energy. By leaving out the average notation bar the time-independent Navier-Stokes...
3. Fluid dynamics in the forming section

Equations are now written as

\[
\begin{align*}
-\nabla \cdot \left[2(\mu + \mu_T)\vec{\epsilon}\right] + \rho \nabla \vec{u} \cdot \vec{u} &= -\nabla (p + \frac{2}{3} \rho k) + \rho \vec{f} \\
\nabla \cdot \vec{u} &= 0.
\end{align*}
\] (3.31)

From this we can define the effective viscosity as \( \mu_{eff} := \mu + \mu_T \) and effective pressure as \( P := p + \frac{2}{3} \rho k \). Variables \( \mu_T \) and \( k \) are linked together with the equation

\[
\mu_T = \rho C_\mu \frac{k^2}{\epsilon},
\] (3.32)

where \( C_\mu = 0.09 \). The scalar variable \( \epsilon \) must not be confused with the rate of strain tensor \( \vec{\epsilon} \). \( \epsilon \) describes the dissipation of the turbulent kinetic energy. Variables \( k \) and \( \epsilon \) are coupled with two partial differential equations called the \( k-\epsilon \) equations. These equations are not discussed here but can be studied e.g. from [11]. Usually \( \frac{2}{3} \rho k \) is multiple order of magnitudes smaller than static pressure \( p \), hence \( P \approx p \). With these assumptions Equation (3.31) can be written as

\[
\begin{align*}
-\nabla \cdot (2\mu_{eff}\vec{\epsilon}) + \rho \nabla \vec{u} \cdot \vec{u} &= -\nabla p + \rho \vec{f} \\
\nabla \cdot \vec{u} &= 0.
\end{align*}
\] (3.33)

The \( k-\epsilon \) equations include velocity and thus Equation (3.33) and \( k-\epsilon \) equations should be solved simultaneously. However, in this thesis we use iterative numerical methods for solving these equations and they are solved consecutively.

3.4 Modeling of the dewatering in the forming section

Dewatering in the forming section through the forming fabrics obeys the Navier-Stokes equations. In the forming section gravity and resistance due to porosity are the body forces acting on the fluid. By introducing these forces into the Navier-Stokes equations and assigning boundary conditions for the variables both in the Navier-Stokes equations and in the turbulence model, we can formulate a mathematical model for the dewatering process. Next, body forces are derivated and added to the Navier-Stokes equations.

3.4.1 Gravity

Due to gravity fluid’s own weight generates pressure inside the fluid. The pressure is always present in reality. Often when modelling fluid flow with CFD one has to make simplifications. Sometimes gravity can be left out from the mathematical model, but we take gravity into account. Pressure due to fluid’s own weight can be calculated as follows

\[
p_{hyd} = \rho gh,
\] (3.34)
3. Fluid dynamics in the forming section

where \( g \) is acceleration of gravity, \( h \) is the observation depth measured from the fluid surface and subscript hyd refers to the pressure caused by gravity. For water this pressure is called the hydrostatic pressure. By taking the gradient of the previous equation it can be written as

\[
\nabla p_{\text{hyd}} = \rho g \nabla h, \tag{3.35}
\]

when density and acceleration of gravity are assumed to be constant in the solution domain. Equation (3.35) describes the density times force per unit mass, and thus it is equivalent to the source term \( \rho \dddot{f} \) in Equation (3.27).

3.4.2 Darcy’s law

Theory of fluid flow in porous medium is based on Darcy’s law from year 1856: ”The rate of flow \( Q \) of water through the filter bed is directly proportional to the area \( A \) of the sand and to the difference \( \Delta h \) in the height between the fluid heads at the inlet and outlet of the bed, and inversely proportional to the thickness \( L \) of the bed” [1]. Darcy’s law came up from the experiments concerning earth science, but it can be used for any material with porous properties. Mathematically law can be expressed as

\[
Q = -\frac{CA\Delta h}{L}, \tag{3.36}
\]

where \( C \) is a coefficient describing the porosity of the medium and other terms are described above. For our purposes it is more convenient to interpret the height between the fluid heads as a pressure difference \( \Delta p \). This form is often presented in literature instead of Equation (3.36). Subscript \( p \) refers to the pressure difference over the porous area. Furthermore coefficient \( C \) is now defined as \( C = \frac{\kappa}{\mu} \), where \( \kappa \) is the permeability of the porous medium, and thus

\[
Q = -\frac{\kappa A \Delta p}{L}. \tag{3.37}
\]

By writing the rate of flow as

\[
Q = UA, \tag{3.38}
\]

where \( U \) is the average magnitude of the velocity component perpendicular to area \( A \), we obtain

\[
\frac{\Delta p}{L} = -\frac{1}{\kappa} U. \tag{3.39}
\]

Using more general notation and by assuming \( U \) to be equal to \( \vec{u} \) Equation (3.39) can be written as

\[
\nabla p = -\frac{1}{\kappa} \vec{u}. \tag{3.40}
\]
3. Fluid dynamics in the forming section

Permeability used in the Darcy’s law describes the ability of porous medium to transmit fluid. It has the units of \( [\kappa] = \text{m}^2 \). Smaller the value greater the ability to resist the fluid flow.

In 1901 Philippe Forchheimer discovered that there is nonlinear relationship between the flow rate and the change of pressure at sufficiently high velocity. Nonlinear term added to the Darcy’s law is \( \alpha u^2 \). Later this nonlinear term was replaced by notation \( \beta \rho u^2 \), where \( \beta \) is called the inertial factor. The Darcy-Forchheimer or simply Forchheimer (also Forchheimer-Dupuit [20]) equation is

\[
\nabla p_p = - \left( \frac{1}{\kappa} \bar{u} + \beta \rho \bar{u}^2 \right).
\]

(3.41)

Linear term in (3.41) is called the viscous loss term and nonlinear term is called the inertial loss term [8]. Equation (3.41) represents the force per unit mass times density. Thus, it describes the resistance due to homogenous porosity and it equals to the source term \( \rho \bar{f} \) in Equation (3.27). If we want to define different values for porosity in different directions we must define source term as

\[
\rho \bar{f} = - \left( \mu \bar{D} + \rho |\bar{u}| \bar{F} \right) \bar{u},
\]

(3.42)

where \( \bar{D} \) and \( \bar{F} \) are tensors.

Resistance due to porosity can be modelled by adding right hand side of Equation (3.42) to the source term of the Navier-Stokes equations. In the forming section the forming fabrics are thin porous layers at the edge of our area of interest. Thus, we can describe the porosity of the forming fabrics with a boundary condition derived from the Darcy-Forchheimer equation (3.41). In this thesis both of these methods are used. Boundary condition is derived from Equation (3.41) first by ignoring the first term on the right hand side and then writing the equation in a form

\[
u = \sqrt{- \frac{\Delta p_p}{L \mu \frac{1}{\kappa}}},
\]

(3.43)

where \( \nu \) refers to the magnitude of the velocity component perpendicular to porous area. Using shorter notation \( L \mu \frac{1}{\kappa} =: R_d \) for the dewatering resistance, Equation (3.43) can be written as

\[
u = \sqrt{\frac{\Delta p_p}{R_d}},
\]

(3.44)

which states that the velocity is calculated using the pressure difference over the porous boundary.

Forming fabrics have also other properties than porosity, such as elasticity, stability and stiffness. As far as we know, these properties have only a small influence on dewatering and thus are ignored here. In this thesis forming fabrics are treated as a porous medium having a fixed permeability through the fabric.
3. Fluid dynamics in the forming section

3.4.3 Governing equations

In the models used in this thesis the source term $\rho \vec{f}$ in Equation (3.27) includes hydrostatic pressure caused by gravity. The resistance due to porosity is also included into the source term in one of the two models. In the other model porosity is described with a boundary condition. Let us first derive the governing equations for the model where porosity is included into the source term. External forces acting on the fluid can be now written as a sum of the right hand sides of Equations (3.35) and (3.42)

$$\rho \vec{f} = \rho g \nabla h - \left(\mu \bar{D} + \rho |\bar{u}| F\right) \bar{u}. \quad (3.45)$$

Substituting Equation (3.45) into Equation (3.31), the Navier-Stokes equations with the $k-\epsilon$ model can be written as

$$\begin{cases} -\nabla \cdot \left(2 \mu_{\text{eff}} \bar{\epsilon}\right) + \rho \nabla \bar{u} \cdot \bar{u} = -\nabla p + \rho g \nabla h - \left(\mu \bar{D} + \rho |\bar{u}| F\right) \bar{u} \\ \nabla \cdot \bar{u} = 0. \end{cases} \quad (3.46)$$

Integrating previous equations over the volume $\Omega$, the integral form of the time independent Navier-Stokes equations with the $k-\epsilon$ model can be written as

$$\begin{cases} -\int_{\Omega} \left(\nabla \cdot \left(2 \mu_{\text{eff}} \bar{\epsilon}\right)\right) d\Omega + \int_{\Omega} \left(\rho \nabla \bar{u} \cdot \bar{u}\right) d\Omega \\ = -\int_{\Omega} \nabla p d\Omega + \int_{\Omega} \left(\rho g \nabla h - \left(\mu \bar{D} + \rho |\bar{u}| F\right) \bar{u}\right) d\Omega \end{cases} \quad (3.47)$$

which can also be written as

$$\begin{cases} \int_{\Omega} \left(\rho \nabla \bar{u} \cdot \bar{u}\right) d\Omega \\ = -\int_{\Omega} \left(2 \mu_{\text{eff}} \bar{\epsilon} + p I\right) d\Omega + \int_{\Omega} \left(\rho g \nabla h - \left(\mu \bar{D} + \rho |\bar{u}| F\right) \bar{u}\right) d\Omega \end{cases} \quad (3.48)$$

With certain assumptions about the functions inside the integrals we can use Gauss’ divergence theorem and the previous equations can be written as

$$\begin{cases} \int_{\Gamma} \left(\rho \bar{u} \bar{\n} \cdot \bar{n}\right) d\Omega \\ = -\int_{\Gamma} \left(2 \mu_{\text{eff}} \bar{\epsilon} + p I\right) \cdot \bar{n} d\Omega + \int_{\Omega} \left(\rho g \nabla h - \left(\mu \bar{D} + \rho |\bar{u}| F\right) \bar{u}\right) d\Omega \end{cases} \quad (3.49)$$
This is the equation used in the model in which porosity is treated as a source term. In the model where the porosity is modelled using a boundary condition the governing equations are written in a form

\[
\begin{align*}
\int_{\Gamma} (\rho \vec{u} \cdot \vec{n}) \, d\Omega &= - \int_{\Gamma} \left( 2\mu_{\text{eff}} \hat{\varepsilon} + pI \right) \cdot \vec{n} \, d\Omega + \int_{\Omega} \left( \rho g \nabla h \right) \, d\Omega \\
\int_{\Gamma} (\vec{u} \cdot \vec{n}) \, d\Omega &= 0,
\end{align*}
\]

which is obtained simply by ignoring the porosity from the source term. On the porous boundary the tangential velocity is fixed and normal velocity component is calculated using Equation (3.44). In two-dimensional (2D) case this can be written as

\[
\vec{u}_{\Gamma} = \vec{u}_t + \vec{u}_n, \tag{3.51}
\]

where \( \Gamma \) refers to the boundary, \( \vec{u}_t \) is tangential velocity (forming fabric velocity) and \( \vec{u}_n \) is normal velocity.

### 3.5 Dewatering models

In previous studies [3, 4, 12, 16, 18, 19, 28] the dewatering is modelled with different methods, depending on the aim of study. Dewatering in gap formers is modelled e.g. in [4, 19]. In [4] forming fabrics are not fixed but take shape according to the suspension flow. Fibre accumulation between the fabrics is also included in the model. Dewatering in fourdriner formers is modelled in [3, 12, 28]. The dewatering of suspensions in general is analyzed in [6]. Many studies such as [9, 14, 16, 17, 18] concentrate on fibre accumulation and its effect on dewatering. The Darcy’s law is often used for modelling the dewatering, for example see [19, 28]. Experimental studies of dewatering are presented in [13, 31]. There exist also a review of forming and dewatering, see [23].

In this thesis we study gap formers with a fixed geometry and the Darcy-Forchheimer equation (3.42) is used for describing the porosity of the forming fabrics. There exists more realistic models, such as flexible geometry used in [4], for example, but these models are also more complex. Our aim is to test a simple model. Thus for simplicity we use water properties to model the properties of the fibre suspension. Furthermore fibre accumulation is ignored in our model even if it affects the dewatering especially at the end of the forming section. Models used in this thesis are the boundary condition model (BCM) and the forming fabric model (FFM). These models are presented next.

#### 3.5.1 The boundary condition model

In the boundary condition model the porosity of the forming fabrics is treated with the boundary condition (3.51). Governing equations for this model are presented in
3. Fluid dynamics in the forming section

Equation (3.50). Next, the geometry and boundary conditions used in this model are discussed more in depth.

**Gap former geometry and computational domain**

Schematic drawing from the beginning of the forming section of a gap former and the computational domain of a gap former (dashed line) are presented in Figure 3.2. Parametrization of the computational domain is presented in Figure 3.3. Angles $\theta_k$, $k = 0, \ldots, 4$ are defined as angles from positive $x$-axis to counter clockwise direction and there must apply that $\theta_k < \theta_{k+1}$ for all $k = 0, \ldots, 3$. The forming roll radius is
3. Fluid dynamics in the forming section

denoted by \( r \). In reality the thickness of the suspension, between the forming fabrics depends on the fabric properties and paper machine running parameters. Modelling this phenomenon would require fluid structure interaction (FSI) modelling. For simplicity we use a fixed geometry and the suspension thickness is approximated in five different locations, defined with the angles \( \theta_k \). Thicknesses are denoted by \( h_i, i = 0, \ldots, 4 \). The computational domain boundaries and mesh are presented in Figure 3.4.

![Figure 3.4: The computational domain boundaries and mesh in BCM.](image)

**Figure 3.4:** The computational domain boundaries and mesh in BCM.

**Boundary conditions**

There are usually three types of boundary conditions used with the Navier-Stokes equations: Dirichlet, Neumann and mixed of these two. In the Dirichlet boundary conditions a fixed value for a certain property at the domain boundary is set. This means e.g. setting a velocity value at the flow inlet. In the Neumann boundary condition the gradient of a certain property is set on a domain boundary.

In this model the boundary conditions for the computational domain are derived from paper machine properties. At the boundaries two different types of boundary conditions are used: fixed value (Dirichlet) and zero gradient (Neumann). At the zero gradient boundary condition the partial derivative into the direction of the surface normal of a property is set to zero.
3. Fluid dynamics in the forming section

Boundary conditions for $k$ and $\epsilon$ at the inlet are calculated using equations presented in [10, 25]. $k$ is calculated as

$$k_{in} = \frac{3}{2} I_t \overline{u}^2,$$

(3.52)

where $\overline{u}$ is the mean velocity and $I_t$ is the turbulence intensity. $\epsilon$ is calculated using the value of $k$ as

$$\epsilon_{in} = \frac{C_\mu^{0.75} k^{1.5}}{l},$$

(3.53)

where $l$ is the length scale which we define to be 20% of the tube (channel) diameter similarly to [25] and $C_\mu$ is constant presented earlier in Section 3.3.1. At the outlet boundary the Neumann boundary condition is used. At the porous walls fixed inlet values are used to model the turbulence.

In BCM the mathematical model to be solved is Equation (3.50) with the following boundary conditions

$$\begin{cases}
\bar{p} = p_{out}, & \frac{\partial u}{\partial n} = 0 \ \forall \ l = 1, 2, 3, \ \frac{\partial k}{\partial n} = 0, \ \frac{\partial \epsilon}{\partial n} = 0 \text{ on } \Gamma_{in} \\
p = 0, \ \bar{u}_t = u_t \cdot \bar{t}, \ \bar{u}_n = \left(\sqrt{\frac{\Delta p_{df}}{R_d}}\right) \cdot \bar{n}, & \text{ on } \Gamma_{p, u} \text{ and } \Gamma_{p, j}, \\
\bar{u} = \bar{u}_t + \bar{u}_n, \ k = k_{in}, \ \epsilon = \epsilon_{in} \text{ on } \Gamma_{p, u} \text{ and } \Gamma_{p, j},
\end{cases}$$

(3.54)

where $\bar{t}$ is the tangential vector, $u_t$ is the magnitude of the tangential velocity (forming fabric velocity), $\Delta p_{df}$ is the pressure difference over the boundary (forming fabric) and $R_d$ is the dewatering resistance. Outside of the forming fabrics the pressure is assumed to be zero.

3.5.2 The forming fabric model

The governing equations for the forming fabric are presented in Equation (3.49). In this model the porosity is modelled as a source term in the Navier-Stokes equations. The geometry and the boundary conditions used in this model are discussed next.

GAP FIFER GEOMETRY AND COMPUTATIONAL DOMAIN

The gap former geometry is the same as in Figure 3.2. The computational domain is the same as in the boundary condition model except that the forming fabrics are also modelled. The parametrization of the computational domain in this model is presented in Figure 3.5. Both forming fabrics have the same thickness $h_{df}$.

The computational domain boundaries for the FFM are presented in Figure 3.6. In the figure green area is the area of the suspension denoted by $\Omega_s$ and white areas are the forming fabrics denoted by $\Omega_{ff}$ and thus $\Omega = \Omega_s \cup \Omega_{ff}$. Height of the suspension area and forming fabrics are not in scale.
Boundary conditions

The mathematical model of FFM contains Equation (3.49) with the following boundary conditions
The boundary conditions for the $k$-$\epsilon$ model are calculated the same way as in the boundary condition model.

By solving the models discussed earlier, we obtain the pressure and velocity field in the computation domain. Governing equations in both models describe a system of partial differential equations. In next chapter we discuss the solving of a system of partial differential equations numerically.
The Navier-Stokes equations describe the fluid flow with a system of partial differential equations. Only in very simple cases it is possible to solve the Navier-Stokes equations analytically [5]. When analytical solution cannot be obtained, one way to solve a system of partial differential equations is to use numerical solution methods. Numerical solution is usually obtained with computers. The derivatives in the Navier-Stokes equations are usually not known so we have to approximate the values using the known values e.g. for the pressure and velocity. These approximated derivative values are then used to obtain an approximate solution for the Navier-Stokes equations. This is actually the purpose of CFD: approximate real life fluid flow in certain finite number of points in the solution domain.

Approximation of the derivatives at a certain point is done using variable values at neighbourhood points. When these approximated derivatives are included in the mathematical model, we obtain a mathematical equation. In the neighbour point we do the same and the obtained equation for this point includes at least one term which is also present in the equation for the previous point. From these equations we can form a system of equations. If the approximations and the equations are linear, the obtained system of linear equations can be written in a matrix form. A solution is obtained by solving the matrix equation with some numerical method. Main reference for this section is [5].

CFD calculations in this thesis are done using the finite volume method (FVM). In FVM the computational domain is divided into control volumes (CV:s) and it uses the integral forms of the Navier-Stokes equations to form a system of linear equations (SLE). In OpenFOAM many pre-configured solvers use FVM and thus it is used for solving the dewatering models presented earlier.
4. Numerical solution with CFD

4.1 System of linear equations

CFD calculations are usually done by formulating a system of linear equations which is then solved to obtain the values for the variables (e.g. pressure and velocity). Next, we discuss defining a system of linear equations, linearization of nonlinear terms and derivative approximation.

4.1.1 Defining a system of linear equations

A system of linear equations is defined as

\[
\begin{align*}
    a_{11}\phi_1 + a_{12}\phi_2 + \cdots + a_{1N}\phi_N &= c_1 \\
    a_{21}\phi_1 + a_{22}\phi_2 + \cdots + a_{2N}\phi_N &= c_2 \\
    \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots &= \vdots \\
    a_{N1}\phi_1 + a_{N2}\phi_2 + \cdots + a_{NN}\phi_N &= c_N,
\end{align*}
\]

where \(a_{ij}\) are known coefficients and \(\phi_i\) are the unknown variables. The aim is to solve the unknowns so that all the equations are valid. The equations above can be written in a matrix form as

\[ A\phi = c, \]

where

\[
A = \begin{bmatrix}
    a_{11} & a_{12} & \cdots & a_{1N} \\
    a_{21} & a_{22} & \cdots & a_{2N} \\
    \vdots & \vdots & \ddots & \vdots \\
    a_{N1} & a_{N2} & \cdots & a_{NN}
\end{bmatrix}, \quad (4.3)
\]

\[ \phi = \begin{bmatrix}
    \phi_1 \\
    \phi_2 \\
    \vdots \\
    \phi_N
\end{bmatrix}, \quad (4.4) \]

and

\[ c = \begin{bmatrix}
    c_1 \\
    c_2 \\
    \vdots \\
    c_N
\end{bmatrix}. \quad (4.5) \]

For solving this type of matrix equations there exists numerous numerical methods. Methods can be divided in two categories: direct methods and iterative methods [7]. Direct methods are based on the Gaussian elimination in which terms below the diagonal are eliminated (set to zero) by multiplying and substracting rows from
each other. Many variations of the Gauss elimination have been derived. LU decomposition is one variant which is often used for CFD calculations.

Iterative methods are based on the idea that one guesses a solution and then uses a certain equation to improve the solution systematically. After \( n \) iterations we have solution \( \phi^n \) which does not satisfy the matrix equation exactly so we can write Equation (4.2) in a way that there is non-zero residual \( r^n \)

\[
A\phi^n = c - r^n.
\]  

(4.6)

The aim is to minimize the magnitude of \( r^n \). If the residual is small enough, we can write

\[
\phi^n \approx \phi.
\]  

(4.7)

Examples of iterative methods are Jacobi method, Gauss-Seidel, Stone’s method and conjugate gradient methods. In this thesis conjugate gradient methods are used. Basic idea is that the matrix equation is transformed into minimization problem. Conjugate gradient method is then applied to it in order to reach the solution. More information about iterative methods can be found e.g. from [5, 7].

In CFD the matrix \( A \) in Equation (4.2) is usually sparse meaning that there are only few non-zero elements at each row. A special case of a sparse matrix is a band matrix in which there are few non-zero terms before and after the non-zero diagonal. A band matrix is generated when structured grid is used. Structured grid means usually that hexagon elements in 3D or quadrangle elements in 2D case are used in order to divide the computational domain into control volumes. Sparse, but not necessarily a band matrix, is generated when unstructured grid is used. In unstructured grid tetrahedral (in 3D case) or triangular (in 2D case) elements are usually used. Compared with the direct methods the iterative methods do not change the coefficients of matrix \( A \) and thus, they are more suitable for solving matrix equations including sparse matrices. Iterative methods can be more efficient in CFD calculations than direct methods but on the other hand they are also more complex to implement [7].

4.1.2 Linearization of the Navier-Stokes equations

In Equations (3.49) and (3.50) the convection term \( \rho \vec{u} \vec{u} \cdot \vec{n} \) is nonlinear. Thus the Navier-Stokes equations are nonlinear. In addition, Darcy-Forchheimer equation includes nonlinear term \( \beta \rho \vec{u}^2 \). In order to use solvers suitable for systems of linear equations we have to linearize these terms or treat them with Newton-like methods. Newton-like methods require the calculation of the Jacobian matrix, which costs computationally more than using linearization and solving problem with iterative methods [5]. Linearization is also used in this thesis.
Linearization can be done e.g. by evaluating the nonlinear terms using values from the previous iteration for one of the velocity term, which can be written as

$$\rho \vec{u}^m \vec{u}^m \cdot \vec{n} = \rho \vec{u}^{(m-1)} \vec{u}^m \cdot \vec{n},$$

where $m$ is the iteration index [11]. More simple method is to calculate the nonlinear terms using velocity values only from the previous iteration. With this kind of method the nonlinear terms are treated explicitly. One should take into account that this kind of method converges only for flows with very small Reynolds numbers [11].

### 4.1.3 Derivative approximations

To obtain linear equations we must somehow approximate the partial derivatives in the Navier-Stokes equations. Derivative approximation can be done e.g. with Taylor series expansion. Continuously differentiable function $\phi(x)$ can be expressed in the vicinity of point $x_i$ as a Taylor series

$$\phi(x) = \phi(x_i) + (x-x_i) \left( \frac{\partial \phi}{\partial x} \right)_i + \frac{(x-x_i)^2}{2!} \left( \frac{\partial^2 \phi}{\partial x^2} \right)_i + \frac{(x-x_i)^3}{3!} \left( \frac{\partial^3 \phi}{\partial x^3} \right)_i + \cdots.$$  

Approximation for first order derivative is reached by truncation of the Taylor series after the second term. By doing this the derivative can be written as

$$\left( \frac{\partial \phi}{\partial x} \right)_i \approx \frac{\phi(x) - \phi(x_i)}{(x-x_i)}.$$  

Replacing $x_i$ by $x_{i-1}$ or $x_{i+1}$ and $x$ by $x_i$ we can reach two different approximations

$$\left( \frac{\partial \phi}{\partial x} \right)_i \approx \frac{\phi(x_i) - \phi(x_{i-1})}{(x_i-x_{i-1})},$$  

and

$$\left( \frac{\partial \phi}{\partial x} \right)_i \approx \frac{\phi(x_{i+1}) - \phi(x_i)}{(x_{i+1}-x_i)}.$$  

One more expression is obtained by using both $x_{i-1}$ and $x_{i+1}$ for approximation which can be defined as

$$\left( \frac{\partial \phi}{\partial x} \right)_i \approx \frac{\phi(x_{i+1}) - \phi(x_{i-1})}{(x_{i+1}-x_{i-1})}.$$  

These are called backward (BDS), forward (FDS) and central (CDS) difference schemes, respectively [5], and they are illustrated in Figure 4.1 with a simple example. These approximations are often called finite difference approximation. It
is obvious that the truncation produces an error but usually compromise between accuracy and computational time has to be made in order to reach some solution within a finite time. The denser the mesh the more accurate the approximation is, but it takes more time to compute approximations for all nodes. Some other, more complex and more accurate, methods could also be used to approximate the first order derivative \cite{5}, but we will not discuss those in this thesis.

If second order derivatives are needed, the simplest method is to take the derivative of the first order derivative in the same way as presented above. This method requires velocity values in at least three nodes but this is still computationally quite cheap. For higher order approximations more node values would be required.

\section*{4.2 Solving the Navier-Stokes equations with FVM}

\subsection*{4.2.1 Introduction to the finite volume method}

In the finite volume method the computational domain is divided into finite number of contiguous control volumes. The Navier-Stokes equations must apply to the control volumes the same way as to the whole computational domain. FVM uses integral form of the Navier-Stokes equations as a starting point \cite{5}. Surface and volume integrals in Equation (3.49) or (3.50) must be approximated in order to obtain a linear equation for every CV. Interpolation is needed because the computational node is at the centre of each CV. After obtaining the linear equation for every CV, the equations can be combined to form a system of linear equations. When Navier-Stokes equations are discussed, $\phi$ in Equation (4.2) includes the unknown variables i.e. velocity $\vec{u}$ and pressure $p$ in each computational node.
4.2.2 Interpolation practices

The Navier-Stokes equations (3.49) and (3.50) include surface integrals but the actual velocity and pressure values are calculated at the CV centres. To calculate surface integrals one have to interpolate values at the CV surfaces. Interpolation is done using the CV centre values.

There are multiple methods for interpolating values of the variables between two CV centres [5, 7]. Upwind differencing scheme (UDS) is quite equivalent to backward and forward difference approximation. In UDS the value on the CV face depends on the fluid flow direction. The variable value at a certain face is assumed to have the same value as the centre node which is at the upwind direction of the flow.

Another simple method is to linearly interpolate the value between the CV centre. This corresponds to the central difference scheme of the first order derivative. One way is to assume parabolic shape for the variable values between the CV centres in order to obtain some approximation for the value at the face. There are also other more accurate higher order methods developed for interpolation [5, 7]. These methods interpolate variable values using more than two computational nodes and by doing this the computational costs are increased.

4.2.3 Approximation of the surface and volume integrals

In order to obtain linear equation we have to approximate the values of the integrals in the Navier-Stokes equations (3.49) and (3.50). In this section we discuss some approximation methods briefly using [5, 24] as main references. In 2D case the surface integrals are one-dimensional integrals and volume integrals are actually area integrals. In 3D case these are real surface and volume integrals.

Surface integrals in control volumes can be written as a sum of surface integrals over every surface of the control volume

\[ \int_{\Gamma} f dS = \sum_k \left( \int_{\Gamma_k} f dS \right), \]  

(4.14)

where

\[ \bigcup_k \Gamma_k = \Gamma. \]  

(4.15)

Evaluation of \( f \) at the CV face was discussed in the previous sections. The simplest way to approximate the integral is the midpoint rule [5]

\[ \int_{\Gamma_k} f dS \approx f_k S_k, \]  

(4.16)
where $f_k$ is the value obtained from the interpolation and $S_k$ is the area of surface $\Gamma_k$. Other methods, such as trapezoid rule or higher order approximations can also be used.

Volume integrals can be approximated as a product of the CV volume (CV area in 2D) and the mean value of $f$ as

$$\int \Omega f d\Omega \approx f_\Omega V_\Omega,$$

where $f_\Omega$ is the mean value of $f$ over the CV volume $V_\Omega$. Higher order methods could be also used for volume integral approximations but we will not discuss them here.

### 4.2.4 Velocity and pressure coupling

The Navier-Stokes equations are complicated due to the pressure and velocity coupling. There is no independent equation for the pressure. Instead, it is included in the momentum equation. An iterative implicit pressure-correction method can be used to solve the coupling problem [5]. In the implicit pressure-correction method the momentum equation for a CV is written as a function of the velocity when the pressure values and the source term values are taken from the previous iteration. Each linear equation derived from the momentum equation can be written as

$$A_P u^m_P + \sum_l A_l^m u^m_l = -\frac{\partial p^{m-1}}{\partial x_i} + Q^{m-1}_i, \forall i = 1, 2, 3, \ P = 1, \ldots, N,$$

(4.18)

where $P$ is the index of the computational node, $l$ refers to the neighbouring nodes, $m$ is the iteration index, $i$ is the coordinate component index and $Q$ includes all terms which do not include velocity ($u^*$) or pressure ($p$) [5]. When these equations are derived for every CV, they can be combined to form a system of linear equations from where the velocity $u^*$ can be solved. Notation $u^*$ refers to the fact that the velocity solved from the above equation satisfies only the momentum equation and thus is only a predictor for the velocity satisfying the Navier-Stokes equations. In the implicit pressure-correction method the velocity field is corrected to satisfy the continuity equation using the pressure field which satisfies the continuity equation. Then the next question is, how to solve a pressure field which satisfies the continuity equation if the continuity equation does not even include a pressure term? Pressure can be solved using the following method. After the predictor for the velocity is obtained an equation for the pressure using Equation (4.18) can be given as

$$A_P u^m_P + \sum_l A_l^m u^m_l = -\frac{\partial p^m}{\partial x_i} + Q^m_i.$$

(4.19)
Furthermore this can be written in a form

\[
 u_{iP}^{m*} = - \sum_l A_u^{u_i u_i} u_{il}^{m*} A_u^{u_i P} - \frac{1}{A_u^{u_i P}} \frac{\partial p_i^m}{\partial x_i} + \frac{Q_i^{m-1}}{A_u^{u_i P}},
\]  

(4.20)

which is achieved only by rearranging the terms. If we substitute the velocity \( u_{iP}^{m*} \) to the continuity Equation (3.7), we can write

\[
 \nabla \cdot (u_{iP}^{m*}) = \nabla \cdot \left( - \sum_l A_u^{u_i u_i} u_{il}^{m*} A_u^{u_i P} - \frac{1}{A_u^{u_i P}} \frac{\partial p_i^m}{\partial x_i} + \frac{Q_i^{m-1}}{A_u^{u_i P}} \right) = 0.
\]  

(4.21)

Now using the above equation we can write the Poisson equation for the pressure as

\[
 \nabla \cdot \left( \frac{\partial p_i^m}{\partial x_i} A_u^{u_i P} \right) = \nabla \cdot \left( - \sum_l A_u^{u_i u_i} u_{il}^{m*} A_u^{u_i P} + \frac{Q_i^{m-1}}{A_u^{u_i P}} \right).
\]  

(4.22)

The only unknown variable in the equation is the pressure. The same way as for the velocity components in the momentum equation, the discretation of the pressure leads to a system of linear equations. After solving the pressure the velocities are corrected using equation

\[
 u_{iP}^{m} = - \sum_l A_u^{u_i u_i} u_{il}^{m*} A_u^{u_i P} - \frac{1}{A_u^{u_i P}} \frac{\partial p_i^m}{\partial x_i} + \frac{Q_i^{m-1}}{A_u^{u_i P}}.
\]  

(4.23)

We use the term "inner iteration" to refer to the velocity field correction by solving the pressure. After the velocity field is corrected, we move to the next "outer iteration", in which a new velocity field satisfying the momentum equation is solved and the correction procedure is repeated. Finally, after multiple outer iterations, a solution which satisfies the Navier-Stokes equations is reached. This can be regarded as solving a time-dependent problem until a steady-state is reached. In time-dependent case the choice of time step is important in order to obtain accurate history of the flow. In time-independent case the objective is to reach convergence as fast as possible.

OpenFOAM uses SIMPLE algorithm (see [5]) for velocity and pressure coupling. This algorithm is based on the method presented above and uses the same idea to first construct a velocity field satisfying momentum equation. After that the velocity field is corrected using the pressure field solved using the continuity equation.

### 4.2.5 Under-relaxation

Consecutive solution method for velocity and pressure coupling described above can be unstable. This instability can be reduced using under-relaxation. In the under-relaxation method the change of a certain property between two consecutive outer
iterations is limited, e.g. the velocity field for the next outer iteration is calculated as

\[ \bar{u}^n = \bar{u}^{n-1} + \alpha_\delta(\bar{u}^{\text{new}} - \bar{u}^{n-1}), \]  

(4.24)

where \( \alpha_\delta \) is the under-relaxation factor for velocity and \( \bar{u}^{\text{new}} \) is the velocity solution of the current outer iteration [5]. The under-relaxation method causes a slower but more steady convergence.
In this chapter simulations and modelling results for the dewatering in the forming section are presented. In addition results for flows in simple geometries with impermeable walls are presented. This is done in order to give some reference results for the actual dewatering modelling. First results are presented for flows in a horizontal 2D channel with a constant height and a horizontal 2D contracting channel. Next results for flows in gap former geometries with impermeable walls are presented. Finally, results obtained with BCM and FFM are presented.

All the simulations were performed using OpenFOAM version 1.6. This open source CFD toolbox includes numerous pre-configured solvers, pre- and post-processing utilities. For mesh generation OpenFOAM’s blockMesh tool was used. Simulations were performed in a PC with Intel Pentium M 760 (2 GHz) processor and 512 MB (DDR2) memory.

5.1 Reference simulations

None of the built-in solvers in OpenFOAM was suitable for our needs so a solver for these reference simulations and for BCM was created by modifying the simpleFoam solver. This is a steady-state solver for incompressible flows with the option to include a turbulence model. Solver was modified by adding gravity to the pre-configured solver. See Appendix A for the source code. In the reference simulations Equation (3.50) was solved without any porosity modelling at the walls.

5.1.1 Horizontal 2D channel

Fluid flow was simulated in a 0.79 m long and 9 mm high channel with zero velocity at the walls. These dimensions were selected in order to make comparison with the gap former geometry easy. Left end of the channel was the inlet and right end was the outlet. Inlet velocity was 15 m/s into positive x-direction and fixed pressure at the outlet was 17.5 kPa. Water properties were used for the fluid in the simulations.
and k-ε turbulence model was included in the model. In Figure 5.1 the pressure field is shown and in Figure 5.2 the velocity field is presented. Pressure loss in Figure 5.1 is approximately the same as the loss calculated analytically with the Darcy-Weisbach equation [2, 22].

For simulations in contracting channel the same solver, boundary conditions and fluid properties were used. The geometry was similarly 0.79 m long channel with 9 mm high inlet but the outlet height was 3 mm instead of 9 mm used in the previous simulation. The lower wall of the channel was kept horizontal while the upper wall was placed in an angle so that the channel contracted. The pressure field and the velocity field for this contracting channel case can be seen in Figures 5.3 and 5.4.

**Figure 5.1:** The pressure field in a horizontal 2D channel with a constant height.

**Figure 5.2:** The velocity field in a horizontal 2D channel with a constant height.

**Figure 5.3:** The pressure field in a horizontal 2D contracting channel.

**Figure 5.4:** The velocity field in a horizontal 2D contracting channel.
5. Numerical experiments

Analytical approximation for pressure loss can be calculated using the Bernoulli equation [2]. The pressure loss in Figure 5.3 is approximately the same as the analytically calculated loss. The velocity fields obtained in these simulations seem to be very reasonable and thus can be assumed to be correct.

5.1.2 Gap former geometry with impermeable walls

In this section the simulation results for two gap former geometries (see Figures 3.2 and 3.3) are presented. First, water flow was simulated in a gap former geometry with a constant height. Inlet angle \( \phi_0 = 270^\circ \), outlet angle \( \phi_4 = 360^\circ \), forming roll radius \( r = 0.5 \text{ m} \) and 9 mm constant height were used. Zero velocity was defined for walls. Inlet velocity was 15 m/s into positive \( x \)-direction and 17.5 kPa was the pressure at the outlet. Pressure and velocity fields with this set-up are presented in Figures 5.5 and 5.6.

\[ \begin{align*}
\text{p (kPa)} & = 101.583 \\
& = 100 \\
& = 80 \\
& = 60 \\
& = 40 \\
& = 20 \\
& = 16.3842
\end{align*} \]

\[ \begin{align*}
\text{Figure 5.5: } & \text{ The pressure field of a gap former geometry with a constant height.}
\end{align*} \]

Secondly fluid flows were simulated in a contracting gap former geometry. The gap former geometry parameters are presented in Tables 5.1 and 5.2. Walls were impermeable and zero velocity was defined at the walls. Inlet velocity was again 15 m/s into positive \( x \)-direction and outlet pressure was fixed to 17.5 kPa. Results are shown in Figures 5.7 and 5.8. In the last reference simulation fluid flow was simulated in a gap former geometry with the same geometry parameters and also the same boundary conditions as in the previous simulation, except 15 m/s fixed
Figure 5.6: The velocity field of a gap former geometry with a constant height.

Table 5.1: The geometry angle parameters used in the simulations with the gap former geometry.

| Parameter | value  \\
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi_0$</td>
<td>270°</td>
</tr>
<tr>
<td>$\phi_1$</td>
<td>292.5°</td>
</tr>
<tr>
<td>$\phi_2$</td>
<td>315°</td>
</tr>
<tr>
<td>$\phi_3$</td>
<td>337.5°</td>
</tr>
<tr>
<td>$\phi_4$</td>
<td>360°</td>
</tr>
</tbody>
</table>

tangential velocity was defined at the walls. The results can be seen in Figures 5.9 and 5.10.

Accurate analytical solutions for pressure losses in these geometries are very difficult to obtain, thus here the results can be compared only with the results in the straight horizontal channel. The pressure losses in Figures 5.5 and 5.7 are larger than in Figures 5.1 and 5.3 due to gravity and the curved shape. The pressure loss in Figure 5.9 is slightly smaller than in Figure 5.7 as it should be due to the moving walls. Again, the velocity fields in these simulations seem to be reasonable. Maximum velocity in Figure 5.10 is smaller compared to Figure 5.8 because there is more fluid flowing out from the domain at the vicinity of the walls, and velocity
5. Numerical experiments

Table 5.2: The geometry parameters used in the simulations with the gap former geometry.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td>0.5 m</td>
</tr>
<tr>
<td>$h_0$</td>
<td>9 mm</td>
</tr>
<tr>
<td>$h_1$</td>
<td>7.5 mm</td>
</tr>
<tr>
<td>$h_2$</td>
<td>6 mm</td>
</tr>
<tr>
<td>$h_3$</td>
<td>4.5 mm</td>
</tr>
<tr>
<td>$h_4$</td>
<td>3 mm</td>
</tr>
</tbody>
</table>

Figure 5.7: The pressure field of a gap former geometry with the zero velocity at the walls.

can be smaller at the centre of the channel in order to obtain the same mass flux as in inlet. Simulations in the contracting gap former geometry with the zero velocity at the walls and with the moving walls are good reference results for the dewatering studies.
5. Numerical experiments

Figure 5.8: The velocity field of a gap former geometry with the zero velocity at the walls.

Figure 5.9: The pressure field of a gap former geometry with the fixed $15 \, \text{m/s}$ tangential velocity at the walls.
5. Numerical experiments

Figure 5.10: The velocity field of a gap former geometry with the fixed 15 m/s tangential velocity at the walls.

5.2 The boundary condition model

The solver used in the reference simulations was also used in BCM with the porosity boundary condition. The boundary condition defined in Equation (3.51) was not available in OpenFOAM so it was created. Built-in boundary condition fixedValue was used as a basis for the new boundary condition, see Appendix A for the source code. The number of elements used in the simulation was 1000×80 (length × thickness, as shown in Figure 3.4) and the number of outer iterations was 5000. See Appendix B for the solver configurations used in the simulation.

5.2.1 Results

As mentioned in Section 3.5 the suspension was modelled as water. Thus, for density the value $\rho = 1 \cdot 10^3 \frac{\text{kg}}{\text{m}^3}$ and for viscosity the value $\mu = 1 \cdot 10^{-3} \frac{\text{kg}}{\text{m}\cdot\text{s}}$ were used. Gap former geometry parameter values used in the simulation with BCM were the same as the ones used in the gap former reference simulations, presented in Tables 5.1 and 5.2. The parameter values were chosen in a way that they approximate the real geometry of a gap former. Boundary condition values used in the simulation are presented in Table 5.3. Dewatering resistance $R_d = 4 \cdot 10^{11} \frac{\text{kg}}{\text{m}^2\cdot\text{s}}$ was used for
Table 5.3: The boundary condition values used in BCM.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vec{u}_{in}$</td>
<td>$(15, 0, 0) \text{ m/s}$</td>
</tr>
<tr>
<td>$k_{in}$</td>
<td>0.03375 $\text{ m}^2/\text{s}$</td>
</tr>
<tr>
<td>$\epsilon_{in}$</td>
<td>0.566 $\text{ m}^2/\text{s}$</td>
</tr>
<tr>
<td>$p_{out}$</td>
<td>17.5 kPa</td>
</tr>
<tr>
<td>$u_t$</td>
<td>15 $\text{ m/s}$</td>
</tr>
</tbody>
</table>

the porous boundaries. In the calculations for the k-\( \epsilon \) model boundary conditions, turbulence intensity value $I_t = 1\%$ was used and characteristic length was 20\% of the inlet height: $l = 0.2 \cdot 9 \text{ mm}$. The inlet velocity, the tangential velocities of the forming fabrics and the pressure at the outlet were chosen in a way that they were the same order of magnitude as in [4]. Pressure and velocity field are presented in Figures 5.11 and 5.12. A close-up from the outlet velocity field can be seen in

Figure 5.11: The pressure field obtained with BCM.

Figure 5.13 and dewatering profiles are shown in Figure 5.14.

The results for the pressure and velocity fields obtained with BCM are larger than ones obtained in simulations with impermeable walls in Figures 5.9 and 5.10. If there is water removed through the walls, the pressure loss should be smaller and
the velocities at the outlet should also be smaller. Thus, BCM does not seem to work as it should.
5. Numerical experiments

5.3 Forming fabric model

For FFM built-in rhoPorousSimpleFoam solver in OpenFOAM was modified. The original solver applied to compressible fluid flows in porous media. It was modified to apply for incompressible fluids and gravity was included in the solver, see Appendix A for the source code. Porosity modelling is actually programmed in the porousZone class. This class was modified in a way that the porosity (permeability) was defined separately in the normal and tangential direction inside the forming fabric, see Appendix A for the source code. The number of elements used in the simulation was 1000 \times 100 and the number of outer iterations was 25000. The solver configurations are presented in Appendix B.

5.3.1 Results

Fluid density and viscosity values were the same as in BCM. Geometry parameter values were also the same as the ones used in BCM and presented in Tables 5.1 and 5.2. Completed with the forming fabric thickness \( h_{ff} = 1 \) mm. Boundary condition values used in the simulations are presented in Table 5.4. The k-\( \epsilon \) model properties were the same as in BCM. Pressure and velocity fields are presented in Figures 5.15 and 5.16, and dewatering profiles are shown in Figure 5.17.

When the pressure field obtained with FFM in Figure 5.15 is compared with the results obtained from the simulation with the impermeable moving walls in Figure 5.9, one can clearly see that FFM works as it should; dewatering causes the pressure loss to be smaller. The pressure loss is the same order of magnitude as in Figure 5.5 which is reasonable. Figure 5.15 shows that the pressure increases after the inlet and

\[ u \]
5. Numerical experiments

Table 5.4: The boundary conditions for FFM. Subscript \( t \) refers to the tangential direction and \( n \) to the normal direction.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \vec{u}_{\text{in}} )</td>
<td>( (15, 0, 0)^T \text{ m/s} )</td>
</tr>
<tr>
<td>( k_{\text{in}} )</td>
<td>( 0.03375 \text{ m}^2 \text{s}^{-1} )</td>
</tr>
<tr>
<td>( \epsilon_{\text{in}} )</td>
<td>( 0.566 \text{ m}^2 \text{s}^{-3} )</td>
</tr>
<tr>
<td>( p_{\text{out}} )</td>
<td>( 17.5 \text{ kPa} )</td>
</tr>
<tr>
<td>( D_t(=\frac{1}{\kappa_t}) )</td>
<td>( 0 \text{ m}^{-2} )</td>
</tr>
<tr>
<td>( D_n(=\frac{1}{\kappa_n}) )</td>
<td>( 33 \text{ m}^{-2} )</td>
</tr>
</tbody>
</table>

Figure 5.15: The pressure field obtained with FFM.

reaches its maximum right before the middle point of the computational domain. After the maximum point, the pressure decreases towards the outlet. This type of behaviour for the pressure is somewhat similar to the pressure profiles presented in [4]. Dewatering profile, instead, differs from the profile presented in [4]. In the results obtained with FFM there is a distinct maximum point in the dewatering velocity profile in the middle of the computational domain. This dewatering profile correlates well with the obtained pressure field. The order of magnitudes of the dewatering velocities are correct compared to [4, 12, 13, 31] but the shape of the dewatering velocity profile is different. In previous studies the maximum of the dewatering
5. Numerical experiments

Figure 5.16: The velocity field obtained with FFM.

Figure 5.17: The dewatering velocity profiles obtained with FFM.

velocities is at the point where the jet from the headbox hits the forming fabrics. Figure 5.16 shows that the velocity of the suspension varies only little at the length of the computational domain.
In this thesis the objective was to study the dewatering in the forming section of a paper machine. In the dewatering process water is removed from fibre suspension through porous forming fabrics. The dewatering was studied using computational fluid dynamics. Thus mathematical models and simulations were used. Two CFD models were used for studying the dewatering at the beginning of the forming section: the boundary condition model and the forming fabric model. These models were implemented into open source CFD program OpenFOAM.

Both, the boundary condition model and the forming fabric model use the Navier-Stokes equations and k-ε turbulence model. BCM describes the fluid flow in porous media using a boundary condition derived from the Darcy-Forchheimer equation. With this model we can fix the velocity of the forming fabrics. The Darcy-Forchheimer equation is also used in FFM but forming fabrics are modelled as a porous media inside the computational domain. The dewatering right after the point where the jet from the headbox hits the forming fabrics in the forming section was modelled with both of the models. Modelling usually requires simplifications. In the models used in this thesis two major simplifications are made: fixed geometry and treatment of the fibre suspension as water. Geometry of the computational domain is fixed where as in reality the forming fabrics take the shape according to the fluid flow from the headbox. Also, in reality the suspension would have different properties than water and the accumulation of fibres would have an effect on the dewatering. Both of these simplifications were made in order to keep the models as simple as possible for preliminary studies with OpenFOAM.

For the dewatering simulations, reference simulations in 2D channels with impermeable walls were performed. The results in these simulations indicate that the boundary condition model with impermeable walls works as it should and it provides good results. When the boundary condition model results are compared with the reference simulation results, it is clear that BCM does not describe the dewatering well. Furthermore, if the resistance to the fluid flow was decreased at
6. Conclusions

the porous walls in order to gain greater dewatering velocities, the solver started to oscillate between two solutions. Thus, convergence was not reached with this model when larger dewatering velocities were modelled. This kind of oscillating should not occur and the problem is most probably in the numerical solution of the problem. Different solvers for the system of linear equations, solver parameters, discretation schemes, more simple geometries and excluding the k-ε model were tried out, but none of these solved the problem. Convergence was not reached either with smaller under-relaxation factors. In addition, it turned out that boundary conditions for k and ε at the porous walls had a huge impact on the pressure and velocity fields. We do not know what those values really are, so different boundary conditions were tried out (e.g. zero gradient and very small fixed values). The best results were obtained with using the same values for k and ε as in the inlet.

FFM proved to be better for modelling the dewatering. When mass fluxes through computational domain boundaries are concerned, FFM results are very reasonable. This can be stated also when results are compared with the reference simulation results. As mentioned, the FMM dewatering velocities have the same order of magnitude as the velocities in previous studies but the shape of the velocity profiles are different. Also the pressure field values differ from the pressure values in previous studies [4, 12, 13, 31]. Differences are most probably caused by the fixed geometry. The velocity values remain about the same throughout the computational domain but it is hard to say, are these velocity variations also present in real gap formers due to fixed geometry used in the simulation.

These results show that it is a difficult task to model the dewatering in the forming section even with simplified models. BCM, having the property of fixed forming fabric velocity, would have been better for modelling the dewatering but FFM proved to work better in this case. One weakness of FFM is the high number of outer iterations needed for the steady state solution. The simplifications in these models might be the reason why they do not describe the dewatering in the forming section that well. In addition, the approach which the pre-configured solvers of OpenFOAM use to solve the Navier-Stokes equations could be inappropriate for the boundary condition model. It would require fluid structure interaction and multiphase modelling in order to obtain better models and results. However, these models can be used as a starting point for developing better models.

Another objective in this thesis was to study the usability of OpenFOAM for CFD. Basic use of OpenFOAM (e.g. running tutorial cases) requires basic knowledge of a Linux operating system. Having only a command line user interface, OpenFOAM requires commitment from the user to get familiar with the use. However, command line usage is made very easy with simple commands. In order to create new solvers or modify pre-configured ones, the user has to have basic knowledge on programming and compiling. The weakness of OpenFOAM compared to commercial softwares is the lack of documentation. User’s Guide [25] is available
but it provides only a limited documentation of OpenFOAM. OpenFOAM Programmer’s Guide [24] which has some additional documentation was discontinued after the release of OpenFOAM version 1.5. When developing solvers with OpenFOAM the user can use the C++ Source Guide for OpenFOAM. This is very useful but also very laborious tool when solver development is concerned. Actual code writing process is quite easy due to extensive selection of built-in algorithms for different types of calculations needed in fluid dynamics and finite volume method calculations. Of course the ease of code writing depends on how familiar the user is with the C++ syntax.
REFERENCES


[29] VTT. Knowpap 5.0 - learning environment.

The boundary condition model: UEqn.H

```cpp
// Solve the Momentum equation
tmp<fvVectorMatrix> UEqn
(
    fvm::div(phi, U)
    + turbulence->divDevReff(U)
);
UEqn().relax();
eqnResidual = solve
(
    UEqn() == -fvc::grad(p) + fvc::grad(gh)
).initialResidual();
maxResidual = max(eqnResidual, maxResidual);
```

The boundary condition model: pEqn.H (modifications to `simpleFoam` solver)

```cpp
... // Momentum corrector
U -= (fvc::grad(p) - fvc::grad(gh))/AU;
...
```

The boundary condition model: `porousWallTangentialVelocity` constructor

```cpp
porousWallTangentialVelocityFvPatchVectorField::tangentialVelocityFvPatchVectorField
(
```
const fvPatch& p,
const DimensionedField<vector, volMesh>& iF,
const dictionary& dict

: 
fixedValueFvPatchVectorField(p, iF),
tU(0), //Initialization
un(p.nf()), //Patch normal unit vectors
ut(p.size(),vector(0,0,0)), //Initialization
dres(0), //Initialization
dspeed(p.size(),scalar(0)),
pldspeed(p.size(),scalar(0)),
alpha(0)
{

tU=dict.lookupOrDefault("tangVel",tU); //Reading tangential velocity value
dres=dict.lookupOrDefault("dres",dres); //Reading dewatering resistance value
alpha=dict.lookupOrDefault("alpha",alpha); //Reading under-relaxation factor
vector e2(0, 1, 0); //Unit vector into Y-direction
ut=e2^un; //Unit vector in tangential direction SET TANGVEL TO POSITIVE OR
NEGATIVE VALUE DEPENDING WHAT DIRECTION DO YOU WANT TO USE
if (dict.found("value")) //Checking and reading latest field if present
{
    fvPatchVectorField::operator=(vectorField("value", dict, p.size()));
    //Setting up latest field values
}
else //No latest field found, we create a new one
{
    fvPatchVectorField::operator=(ut*tU); //Velocity at the patch is
tangential velocity
}
}

The boundary condition model: porousWallTangentialVelocity class, updateCoeffs method

void porousWallTangentialVelocityFvPatchVectorField::updateCoeffs()
{
    if (updated())
    {
        return;
    }
    const volScalarField& pressure = db().lookupObject<volScalarField>("p");
    //Pressure field
    scalarField outside_p=scalarField(p.size(),0);
    //Pressure value outside the patch
    scalarField inside_p=scalarField(pressure.boundaryField()[patch().index()]);
    //Pressure value at the boundary
dspeed=(inside_p-outside_p)/dres;
//Calculation of the dewatering speed through the wire
for (int i=0;i<patch().size();i++)
//Checking that there is no inflow through the patch
{
    if (neg(dspeed[i]))
    {
        dspeed[i]=0;
        //Setting up zero velocity through the patch if inflow is detected
    }
}
pIdspeed=(vectorField(*this)) & un; //Current velocity normal to the patch
Info << "Dewatering speed, mean: " << average(pIdspeed) " m/s, " << "min: " << min(pIdspeed) " m/s, " << "max: " << max(pIdspeed) " m/s" << endl; //Dewatering info
dspeed=pIdspeed+(sqrt(dspeed)-pIdspeed)*alpha; //Under-relaxation and square root operation
fvPatchVectorField::operator=(ut*tU+un*dspeed); //Assigning of new velocity
field on the patch
fixedValueFvPatchVectorField::updateCoeffs();

The forming fabric model: UEqn (modifications to \texttt{rhoPorousSimpleFoam} solver)

\[
U = (\text{tr} \text{tu}() & \text{UEqn().H}() - \text{gradp}/\text{rho} + \text{fvc::grad(gh)})
\]

The forming fabric model: pEqn (modifications to \texttt{rhoPorousSimpleFoam} solver)

\[
tpEqn = (\text{fvm::laplacian(} \text{tr} \text{tu}()/\text{rho}, \text{p}) = \text{fvc::div(} \text{phi} - \text{fvc::div(} \text{tr} \text{tu}()&\text{fvc::grad(gh)})\]);

\[
U -= \text{tr} \text{tu}()&(\text{fvc::grad(p)}/\text{rho} - \text{fvc::grad(gh)})
\]

The forming fabric model: \texttt{porousZone} class, \texttt{porousZoneTemplates} template, \texttt{addViscousInertialResistance} method
template<class RhoFieldType>

void Foam::porousZone::addViscousInertialResistance
(

tensorField& AU,
const labelList& cells,
const RhoFieldType& rho,
const scalarField& mu,
const vectorField& U
) const
{

vectorField surfaceVec = mesh_.boundary()[cellZoneID_].Sf(); //Surface
area vectors
vectorField surfaceNormVec = surfaceVec/mag(surfaceVec); //Unit surface
area vectors
vector e1(0, 1, 0); //Unit vector into Y-direction
vectorField unitSurfaceTang = e1^surfaceNormVec; //Unit tangent vector
tensor localE(0, 0, 0, 0, 0, 0, 0, 0, 0);
short xcelln=mesh_.boundary()[cellZoneID_].size(); //Number of cells in
tangential direction
short counter=0;
forAll (cells, i)
{

localE.x()=e1;
localE.y()=unitSurfaceTang[counter];
localE.z()=surfaceNormVec[counter];
tensor D=(localE.T() & D_.value() & localE); //Calculating porosity
tensor F=(localE.T() & F_.value() & localE); //directions
AU[cells[i]] += mu[cells[i]]*D + (rho[cells[i]]*mag(U[cells[i]]))*F;
//Resistance due to porosity
counter++;
if (counter==xcelln)
{

counter=0;
}
}
}
The boundary condition model: *fvSolution* file

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

solvers
{
    p GAMG
    {
        tolerance 1e-5;
        relTol 1e-4;
        smoother GaussSeidel;
        nPreSweeps 0;
        nPostSweeps 2;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 100;
        agglomerator faceAreaPair;
    }
```

B. OpenFOAM solver configurations

mergeLevels 1;
};
U PBiCG
{
  preconditioner DILU;
  tolerance 1e-06;
  relTol 0;
};
k PBiCG
{
  preconditioner DILU;
  tolerance 1e-05;
  relTol 0;
};
epsilon PBiCG
{
  preconditioner DILU;
  tolerance 1e-05;
  relTol 0;
};
SIMPLE
{
  nNonOrthogonalCorrectors 0;
  pMin pMin [1 -1 -2 0 0 0] -1e10;
}
relaxationFactors
{
  p 0.3;
  U 0.7;
  k 0.7;
  epsilon 0.7;
}

// ************************************************************************* //
The boundary condition model: *fvSchemes* file
B. OpenFOAM solver configurations

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

// *** * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default steadyState;
}

gradSchemes
{
    default Gauss linear;
    grad(p) Gauss linear;
    grad(U) Gauss linear;
    grad(gh) Gauss linear;
    snGradCorr(U) Gauss linear;
    snGradCorr(p) Gauss linear;
}

divSchemes
{
    default none;
    div(phi,U) Gauss upwind;
    div(phi,k) Gauss upwind;
    div(phi,epsilon) Gauss upwind;
    div((nuEff*dev(grad(U).T()))) Gauss linear;
}

laplacianSchemes
{
    default none;
    laplacian(nuEff,U) Gauss linear corrected;
    laplacian((1|A(U)),p) Gauss linear corrected;
    laplacian(DkEff,k) Gauss linear corrected;
    laplacian(DepsilonEff,epsilon) Gauss linear corrected;
    laplacian(DREff,R) Gauss linear corrected;
}

interpolationSchemes
{
    default linear;
B. OpenFOAM solver configurations

```plaintext
interpolate(U) linear;
}

snGradSchemes
{
    default corrected;
}

fluxRequired
{
    default no;
P;
}

// ************************************************************************* //

The forming fabric model: fvSolutions file

```
B. OpenFOAM solver configurations

```plaintext
nCellsInCoarsestLevel 100;
agglomerator faceAreaPair;
mergeLevels 1;
};
U PBiCG
{
  preconditioner DILU;
tolerance 1e-06;
relTol 0;
};
k PBiCG
{
  preconditioner DILU;
tolerance 1e-05;
relTol 0;
};
epsilon PBiCG
{
  preconditioner DILU;
tolerance 1e-05;
relTol 0;
};
}
SIMPLE
{
  nNonOrthogonalCorrectors 0;
pMin pMin [1 -1 -2 0 0 0] -1e10;
}
relaxationFactors
{
  p 0.7;
  U 0.85;
  k 0.85;
epsilon 0.85;
}

// ************************************************************************* //

The forming fabric model: fvSchemes file

/*--------------------------------*- C++ -*----------------------------------*

B. OpenFOAM solver configurations

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

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default steadyState;
}

gradSchemes
{
    default Gauss linear;
    grad(p) Gauss linear;
    grad(U) Gauss linear;
    grad(gh) Gauss linear;
    snGradCorr(U) Gauss linear;
    snGradCorr(p) Gauss linear;
}

divSchemes
{
    default none;
    div(phi,U) Gauss upwind;
    div(phi,k) Gauss upwind;
    div(phi,epsilon) Gauss upwind;
    div((nuEff*dev(grad(U).T()))) Gauss linear;
    div((rAU&grad(gh))) Gauss linear;
}

laplacianSchemes
{
    default none;
    laplacian(nuEff,U) Gauss linear corrected;
    laplacian((1|A(U)),p) Gauss linear corrected;
    laplacian(DkEff,k) Gauss linear corrected;
    laplacian(DepsilonEff,epsilon) Gauss linear corrected;
    laplacian(DREff,R) Gauss linear corrected;
    laplacian((rAU&grad(gh))) Gauss linear corrected;
}
B. OpenFOAM solver configurations

}\n
interpolationSchemes
{\
  default linear;
  interpolate(U) linear;
}\n
snGradSchemes
{\
  default corrected;
}\n
fluxRequired
{\
  default no;
  P;
}\n
// ************************************************************************* //