TU Delft

Bachelor Thesis

CFD calculations on the helium cooling of the Pebble Bed Reactor Core

Author:
Thomas Frederikse

Supervisors:
Dr. Danny Lathouwers
Dr. Martin Rohde
Gert-Jan Auwerda

August 20, 2010

Abstract

The aim of this research is to investigate if the CFD code Fluidity, designed at Imperial college is capable of simulating the heat and mass transfer of the coolant in the Pebble Bed Very High Temperature Reactor (PB-VHTR). The PB-VHTR is one of the new designs proposed by the Generation IV intiatiative for safer and more fuel-efficient nuclear reactors, and is scheduled to be commercially exploitable in 2020. The Fluidity code is a finite element based CFD code, with special capabilities for adaptive remeshing. To verify the code, four test cases simulating both non-thermal and thermal flows have been investigated, as well as a simulation of flow trough a fraction of a pebble bed. The results show that for simple geometries, like 2D Poiseuille flow trough a tube, as well as more sophisticated flow patterns, like a Von Karman Vortex street behind a 2D cylinder, accurate results can be obtained, including correct predictions of the Strouhal number. Thermal flow simulations, like the 2D forced convection around a sphere have been tested and Nusselt relations have been verified. To get results in compliance with well-researched Nusselt relations, very fine grids were required, although the adaptive remeshing code can cut calculation time. The reactor core was simulated using an adaptive mesh containing 5 pebbles. The results showed that the power output results are in compliance with other research. The drawbacks of Fluidity are the lack of a dynamic LES model, which is particularly important when simulating thermal flows, since the dependence of the behavior of the heat flow on the Smagorinsky coefficient is large in such simulations, the impossibility to code the heat production equations inside the reactor pebbles (constant boundary temperatures were used in the simulation), and the lack of the possibility to directly calculate the steady-state flow pattern in a geometry.

Contents

Chapter 1

Introduction

The Chernobyl disaster brought the post-war nuclear era to an end: the public opinion changed radically, nuclear power was not an option anymore, and almost no new nuclear plants were built afterwards. However, the oil dependency of the modern economy and the signs of a climate change due to the emission of CO_2 caused by the burning of fossile fuels are asking for a wide investigation for alternatives to the current fossile fuel economy. One of the main alternatives is still nuclear power. The development of new reactor types, bundled in the Generation IV initiative, shows that a lot of the current objections to nuclear energy can be resolved, including safety concerns, nuclear waste production and non-proliferation. One of the most promising new designs is the Very High Temperature Reactor (VHTR), which uses graphite pebbles containing the nuclear fuel. This reactor type promises inherent safety characteristics and a high fuel efficiency rate.

1.1 The Very High Temperature Reactor

Around 1950, the first ideas arised of using spherical elements containing the fuel and the moderator. These spheres are resistant against very high temperatures up to 1600 Kelvin, a temperature that is much higher than the highest reached temperature during operation and in reactor accident scenarios.

1.1.1 Pebble bed reactors

In 1966 the first research reactor based on the pebble bed principle was built: the AVR (Arbeitsgemeinschaft Versuchsreaktor) in Jülich, West-Germany, which had a capacity of 45 Megawatt. Because of the capability of the spheri-

cal fuel elements to withstand high temperatures, this reactor was considered inherently safe: no measures had to be taken in case of accidents. Multiple tests were successfully run in this reactor, including loss-of-coolant accident scenarios. However, after the Chernobyl disaster and some operational problems the reactor was shut down. After the AVR, another commercial plant using spherical fuel elements was built in Germany: the 770 MW Thorium High Temperature Reactor, which was capable of breeding U^{235} out of thorium, a possible alternative for the widely used uranium, because Thorium is available in large amounts in the earth's bottom. However, the plant suffered from bad design and financial problems and was forced to shut down after 1 year of service and some accidents that released small amounts of radioactive gases into the environment.
In 2000 the Chinese research plant HTR-10, the 10 MW High Temperature Reactor, reached first criticality. This reactor has been built as a prototype for commercial-scale plants of the HTR Pebble-bed Module (HTR-PM) type: a 250 MW modular reactor, which is planned for completion in 2013. Also South-Africa is developing a Pebble-bed reactor, the 400MW Pebble Bed Modular Reactor. In figure 1.1, a schematic view of the HTR-10 reactor is shown. Helium is used as coolant. Because of it's low reactivity properties the helium won't become radioactive, and because it is in the gas phase during operations, no phase-change will occur, which prevents a fast increase in pressure. The continuous flow of pebbles through the reactor makes a continuous energy production possible, without the need of shutting down the reactor for refueling. The pebble flow causes a random stacking of the pebbles in the reactor core, which influences the heat production slightly during operation [27].

1.1.2 The Very High Temperature Reactor

In 2001, the Generation IV International Forum was found: an initiative of 13 member states (including the European Union) which aims at the development of the next-generation nuclear plants, featuring safer, more efficient and less waste-producing reactor designs. One of the designs is the Very High Temperature Reactor: a further development of the Pebble Bed Reactor, that features a higher outlet temperature, which allows for more efficient energy production. This reactor is not only capable of producting electricity, but also offers the possibility of producing hydrogen and heat for chemical processes, because of the high outlet temperature. The first commercial-scale operating plant design is expected around 2020.

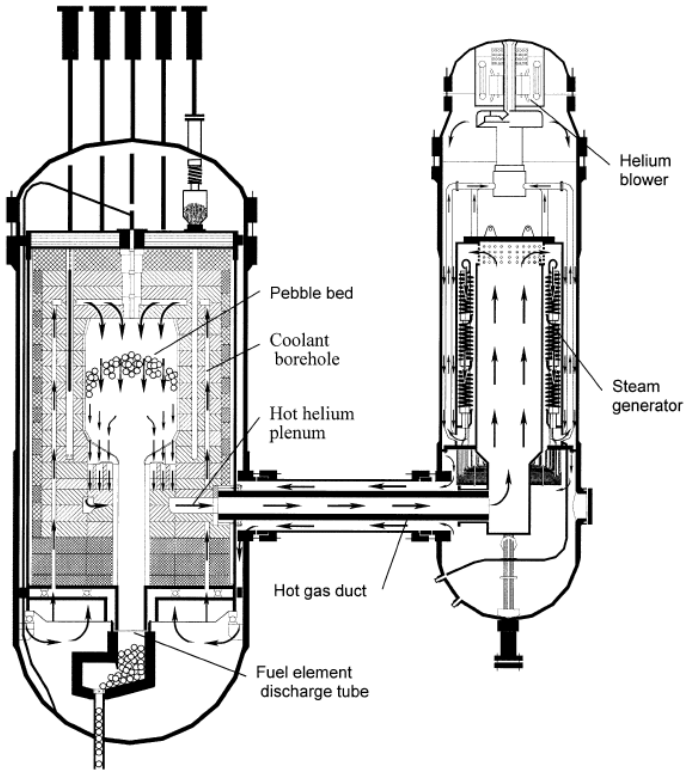

Figure 1.1: Schematic view of the HTR-10 reactor. Picture taken from [18]

1.2 Previous research

One of the focuses in the research of the Very High Temperature Reactors is the temperature profile in the reactor core during operation. Since the temperature in the core of the VHTR reaches significant higher values than in conventional Pebble Bed Reactors, accurate information is needed to be sure that the temperatures don't reach critical values, even in accident scenarios. Temperature calculations in the AVR and THTR reactor seemed to predict lower values of the temperature in the core than the highest temperatures in the pebbles reached at the AVR, which was measured after the reactor shutdown[16], which is unacceptable in the VHTR design. Since the design of the reactor includes a continuous process without the need of reactor shutdown for refueling, the pebbles are constantly moving through the reactor core and no devices are available to measure the core temperature profile. This creates the need of accurate calculation techniques. To describe the behavior of the helium flow between the pebbles, computational fluid dynamics are used. A typical VHTR design contains around 500.000 pebbles [14], which makes it impossible to calculate the flow through the whole reactor using CFD, since the computational power required for this kind of calculations is far out of reach. Different ways have been used to model the reactor core. For example [17] uses a 2-phase porous medium approach and compares the results with a conventional cfd-calculation over a small part of the pebble bed. They conclude that the porous-medium approach gives reasonable results for the average behavior, but local effects cannot be obtained using this approach. In [14] a more sophisticated CFD code is used to determine local effects in the core. However, it remained difficult to compare the computational results with experimental data.

1.3 Aim of the research

At the Applied Modelling an Computation Group from Imperial College in London, a new finite-element based CFD code has been developed: Fluidity. This code has the capability of dealing with complex geometries and offers features as adaptive remeshing and temperature calculations. The aim of this thesis is to compute the flow of helium and the transfer of heat in the pebble bed core using the Fluidity code and to determine if Fluidity is useful for reactor core calculations.

1.4 Outline of thesis

In chapter 2 the governing equations of thermal flow are discussed. Chapter 3 gives an overview of the numerical model Fluidity uses and the available options. In chapter 4 a few test cases are run to check the output results of Fluidity, as well as different calculation options. Chapter 5 shows the setup and the measurement results from the pebble bed calculations and chapter 6 shows the conclusions of the project as well as possibilities for further results.

Chapter 2

Governing equations

To be able to simulate flow and heat transfer phenomena in the pebble bed reactor we need a correct, complete and closed mathematical description of the phenomena. In this section the governing equations of mass, momentum and heat transport are being discussed. In section 2.1, the mass equations are discussed, in section 2.2 the momentum equations, and in 2.3 the energy equations are treated.

2.1 Mass conservation

The flow of fluids must obey a set of conservation laws. The important laws which are needed to describe the flow are conservation of mass, momentum and energy. In case of mass we can easily write

$$\frac{\mathrm{d}M}{\mathrm{d}t} = 0 \tag{2.1}$$

because we assume no destruction or production of mass during the process. To use this equation we need to use the concept of control volumes and control mass. A control volume is a arbitrarily chosen volume somewhere in the flow being analyzed, a control mass is the same, except being a mass rather than a volume. The conservation laws must hold for every chosen control volume. Now the conservation laws can be seen as a kind of bookkeeping. The change in the amount of the conserved quantity is what flows in minus what flows out. We define the intensive form of the conservation law, in which the conserved quantity gets independent of the size of the volume or mass. In the case of mass conservation we use the density (ρ). To re-obtain the total mass we just integrate over the volume Ω [2]:

$$\int_{\Omega} \rho \mathrm{d}\Omega = M \tag{2.2}$$

When applying the conservation law on a control volume we get the conservation equation. In case of mass it becomes [2]:

$$\frac{\partial}{\partial t}\int_{\Omega}\rho \mathrm{d}\Omega + \int_{\Gamma}\rho \boldsymbol{v}\cdot\boldsymbol{n}\mathrm{d}\Gamma = 0 \tag{2.3}$$

With Γ the complete boundary area of the control volume, and $\boldsymbol{v}\cdot\boldsymbol{n}$ the total velocity perpendicular to every boundary element. With Gauss' theorem this can be rewritten into the form:

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\boldsymbol{v}) = 0 \tag{2.4}$$

This equation is called the continuity equation.

2.2 Momentum conservation

Momentum in the intensive form can be expressed by the density times the velocity:

$$p = \rho\boldsymbol{v} \tag{2.5}$$

In case of the momentum equations, we get an extra production term in the integral equation:

$$\frac{\partial}{\partial t}\int_{\Omega}\rho\boldsymbol{v}\mathrm{d}\Omega + \int_{\Gamma}\rho\boldsymbol{v}\boldsymbol{v}\cdot\boldsymbol{n}\mathrm{d}\Gamma = \Sigma\boldsymbol{f} \tag{2.6}$$

This production term is caused by 2 kinds of forces: forces acting on the volume (gravity, electromagnetic forces etc), and forces acting on the boundary (pressure terms, surface tension, shear stress etc).

2.2.1 The Newtonian Stress Tensor

In many cases we can make the assumption that the fluid we are analyzing is a Newtonian fluid. Such a fluid is defined[1] as a fluid for which holds:

$$\tau = \mu\frac{\partial v}{\partial y} \tag{2.7}$$

for a 2D flow in the x-direction, with τ being the shear stress term, and μ the dynamic viscosity. Now the Newtonian Stress Tensor can be constructed [2]:

$$\mathrm{T} = -\left(p + \frac{2}{3}\mu\nabla\cdot\boldsymbol{v}\right)\mathrm{I} + 2\mu\mathrm{D} \tag{2.8}$$

With I the unity tensor, p the static pressure, and D the tension tensor:

$$\mathrm{D} = \frac{1}{2}\left[\nabla\boldsymbol{v} + (\nabla\boldsymbol{v})^{T}\right] \tag{2.9}$$

2.2.2 The Navier-Stokes equations

When filling in the stress tensor in equation 2.6 we get the following result:

$$\frac{\partial}{\partial t}\int_{\Omega}\rho\boldsymbol{v}\mathrm{d}\Omega+\int_{S}\rho\boldsymbol{v}\boldsymbol{v}\cdot\boldsymbol{N}\mathrm{d}S=\int_{S}\mathrm{T}\cdot\boldsymbol{n}\mathrm{d}S+\int_{\Omega}\rho\boldsymbol{b}\mathrm{d}\Omega \tag{2.10}$$

With Gauss' theorem this can be rewritten into:

$$\frac{\partial\left(\rho\boldsymbol{v}\right)}{\partial t}+\nabla\cdot\left(\rho\boldsymbol{v}\boldsymbol{v}\right)=\nabla\cdot\mathrm{T}+\rho\boldsymbol{b} \tag{2.11}$$

In this equation all the forces acting on the volume (so-called body forces) are collected under the term $\boldsymbol{b}$. Equation 2.11 is called the Navier-Stokes Equation and is the basic equation for flowing phenomena in fluids. The equation is closed: all terms, except the velocity $\boldsymbol{v}$ have been defined, so in theory our problem is solved. Unfortunately, the Navier-Stokes equation is a nonlinear partial differential equation, for which no exact solution is known, except in rare easy cases. Even the existence and uniqueness of the solution have not been proved yet. To be able to use these equations for fluid dynamics, numerical methods must be employed to find a solution for the velocity field. This will be discussed in chapter 3.

2.3 Energy conservation

We're not only interested in the velocity field, but also in the temperature field in the flowing medium. To solve for the temperature we use the conservation law for energy, so the conservation equation can be used again. There are two ways heat is transported through the fluid: Diffusion and advection. Advection is the transport of heat due to the motion of the fluid. Diffusion is the phenomenon that heat is transported to the fluid on a microscale from molecule to molecule. It is described on macroscale by the Fourier heat conduction law[1], for which the 1-dimensional version reads:

$$\frac{\partial q}{\partial t}=-\lambda\frac{\partial T}{\partial x} \tag{2.12}$$

q is the heat flow per square meter, λ the thermal conductivity and T the temperature. Plugging in into the conservation equation leads to [2]:

$$\frac{\partial}{\partial t}\int_{\Omega}\rho T\mathrm{d}\Omega+\int_{\Gamma}\rho T\boldsymbol{v}\cdot\boldsymbol{n}\mathrm{d}\Gamma=\int_{\Gamma}\rho C_{p}\lambda\nabla T\cdot\boldsymbol{n}\mathrm{d}\Gamma \tag{2.13}$$

We make two approximations about the specific heat C_p of the fluid here. It's both constant in time and independent of pressure and temperature. Now

we have a closed equation, with only the temperature field T as unknown. Again we use Gauss' theorem to rewrite the equation:

$$\frac{\partial \rho T}{\partial t} + \nabla \cdot (\rho T \boldsymbol{v}) = \nabla \cdot (\rho C_p \nabla T) \tag{2.14}$$

When a source term is present it can be incorporated in the equation with an extra term s. This leads to a so-called advection-diffusion equation:

$$\frac{\partial \rho T}{\partial t} + \nabla \cdot (\boldsymbol{v} \rho T - \rho C_p \nabla T) = s \tag{2.15}$$

2.4 Initial and boundary conditions

We have derived the equations for the temperature and velocity field, but not yet treated the initial conditions to start the calculations. At first, the initial conditions for the fields must be set. Initial conditions define the values of the velocity and temperature at the start of the simulation. Secondly we define the boundary conditions, rules applied to specific areas in the simulation that must be met during the whole time the simulation runs. There are two different kinds of boundary conditions: Dirichlet and Neumann conditions. There are also two commonly used boundary conditions that represent more or less 'real' circumstances, which are based on Dirichlet and Neumann conditions: the no normal flow condition for vector quantities, like velocity and the zero flux condition for scalar quantities like temperature.

Dirichlet

A Dirichlet boundary condition imposes a defined value of the variable on the boundary, for example $T_{inflow} = 0$, meaning that the inflow temperature is zero. For the velocity field, we define this boundary condition as a vector, but we can also limit it to certain directions, for example $v_x = 1$, with no further constraints on v_y and v_z.

Neumann

A Neumann boundary condition defines the first deriverative of a quantity on the boundary:

$$\nabla T \cdot \boldsymbol{n} = q \qquad \text{On the boundary } \Gamma \tag{2.16}$$

This can be used for example to define a temperature gradient on a boundary, or a zero-flux condition, whereby the normal component of the gradient on the boundary must be equal to zero.

No normal flow

The no normal flow condition imposes a Dirichlet boundary condition with $v_n = 0$, to the flow perpendicular to the boundary. This is a way to simulate a domain boundary that is not a wall, but an infinite space. It also incorporates momentum conservation: when applied to a boundary, no momentum will flow through this boundary.

Zero flux

A zero flux boundary condition imposes a Neumann boundary condition of zero to the normal of the domain. Using such a boundary condition ensures no scalar quantity can flow out of or into the specified boundary. This is similar to an insulating boundary for e.g. the temperature field. It also acts as an energy conservation condition: no thermal energy can enter or leave through the boundary with an applied zero flux condition.

2.5 Approximations

To calculate all the effects in the fluid requires a lot of computational power. To avoid this, some approximations can be made in the Navier Stokes equation to speed up calculation. In this research two additional approximations are made for the Navier Stokes equation, besides the assumption that the fluid is Newtonian, which was used in paragraph 2.2.1 to obtain the Navier Stokes equations.

2.5.1 The Boussinesq approximation

The density of fluids encountered in many applications doesn't vary very much during the simulation. The Boussinesq approximation supposes that the variation in density of the fluid is only relevant when dealing with gravity terms and that the difference in inertia is negligible. The correctness of this assumption is mainly related to the temperature differences occurring in the flow. Due to this approximation, the Navier Stokes equations are simplified:

$$\rho_0 \frac{\partial \boldsymbol{v}}{\partial t} + \rho_0 \nabla \boldsymbol{v}\boldsymbol{v} = \nabla \cdot \mathrm{T} + \rho \boldsymbol{b} \tag{2.17}$$

in which ρ_0 represents the averaged density. The equations can be simplified even more when the density is considered as constant throughout the simulation. This will eliminate the density-dependent body force term in equation 2.17: $\rho \boldsymbol{b} \approx \rho_0 \boldsymbol{b}$ Now the density has become independent of the velocity field.

2.5.2 Incompressible fluids

Another approximation can be made for the fluid, namely the incompressibility of it. Most fluids and gases for which the Mach number $\mathrm{Ma} = v_{max}/v_{sound}$ is low ($\mathrm{Ma} \ll 0.3$) can be considered incompressible. This follows from the following relationship[24]:

$$\frac{\delta\rho}{\rho} \sim \mathrm{Ma}^2 \tag{2.18}$$

Then when $\mathrm{Ma} \ll 1$, the term $\delta\rho/\rho$ approaches zero. Incompressible fluids are fluids for which the divergence of the velocity field is zero: ($\nabla \cdot \boldsymbol{v} = 0$). Now the Newtonian stress tensor simplifies to

$$\mathrm{T} = -p\mathrm{I} + 2\mu\mathrm{D} \tag{2.19}$$

This is a simplification of equation 2.8.

Chapter 3

The numerical model

Equation 2.11 and equation 2.15 are closed equations and it should be possible to find a solution, when the boundary conditions are sufficient. The non-linearity of these equations makes it impossible to solve them analytically, as discussed before, so we have to use numerical methods to evaluate the equations. Computational fluid dynamics (CFD) software contains such numerical models. The research has been conducted with the Fluidity CFD code developed by the Applied Modelling an Computation Group at Imperial College in London. This chapter is dedicated to the numerical discretizations used by Fluidity to solve the differential equations. Both space and time must be discretized in order to solve the model numerically. In section 3.1 the spatial discretization , and in section 3.4 the temporal discretization is discussed.

3.1 Spatial discretization

There are a number of methods to discretize the domain spatially. Three of these methods are most widely used[3]:

1. Finite Difference Method
   The finite difference method makes use of the difference approximation of the deriveratives of a function: $f(a) \approx (f(a+h) - f(a))/h$. The domain on which the differential equation must be solved is divided in a grid, with the gridpoints as step size h of the difference equation.

2. Finite Volume Method
   The finite volume method divides the area of the flow on which the equations must be solved into small volumes. Every volume contains a node, that holds a value the quantities to be calculated, then the whole

volume is expected to hold that value. The time is also discretisized into small steps. At every timestep, the difference of nodal values at the boundary between 2 volumes determines the amount transported between volumes. Due to this transport, the nodal values change, and a new timestep can be started. This process can be repeated until a steady-state has been reached: a simulation has reached a steady state when the nodal values don't change anymore at new timesteps.

3. Finite Element Method
   The Finite Element Method is similar to the Finite Volume method in the way that the domain is spatially and temporally discretisized, but now, the nodal value is not supposed to be the value of the whole volume element. The method looks for a piecewise linear function that connects all the nodes and satisfies the differential equation. This leads to a function of the calculated quantity that is defined everywhere on the domain.

3.1.1 The finite element method

The finite element method is used by Fluidity to solve the Navier Stokes equations and the temperature equations. This method consists of a few steps. At first, the mathematical model must be constructed to define the physical problem. This has been done in chapter 2, and brought us to the equations just mentioned. After that, both space and time have to be discretized. The spatial discretization process is called meshing. When simulating a 2D flow problem in a square, a mesh could look like figure 3.1

3.1.2 Shape functions and the weak formulation

Take for example the 1-dimensional Fourier Heat Equation[1]

$$-K\frac{d^2T}{dx^2} = Q \tag{3.1}$$

on the domain $0 < x < l$. We have 2 boundary conditions: the Neumann boundary condition $-KdT/dx = q$ on $x = 0$ and the Dirichlet boundary condition $T = T_L$ on $x = L$. When Q is constant, the exact solution to this equation is known:

$$T(x) = T_L + \frac{q}{K}(L - x) + \frac{Q}{2K}(L^2 - x^2) \tag{3.2}$$

[1]The discussion of the Finite Element Method presented here is based on [25]

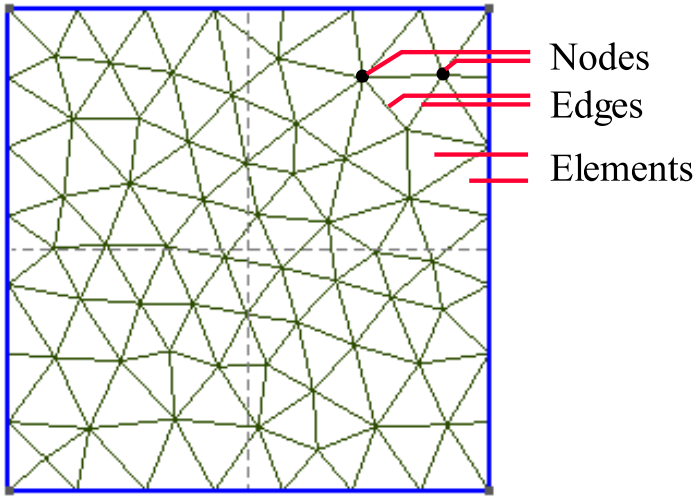

Figure 3.1: Example of a 2D mesh

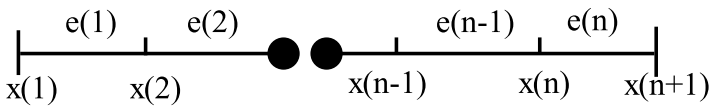

Figure 3.2: 1D finite element spatial discretization example

For most problems we have to solve, no exact solution is available. To solve such equations the same approach is used as given below, and is called the Galerkin weighted residual method. The first step is to discretize the domain in elements, noted e_k: $e_k = \{x : x_k \leq x \leq x_{k+1}\}$, see figure 3.2. The end points x_k and x_{k+1} are called nodes. The solution to equation 3.1 is now approximated over every element by a predefined set of functions dependent on x, denoted by $\phi_j(x)$. These functions are called shape functions, and their weighted sum must approach the final solution:

$$T(x) = \sum_{i=1}^{n+1} a_i \phi_i \tag{3.3}$$

with n the number of elements. In practice, the solution T obtained from equation 3.3 will not be exactly equal to the true solution of equation 3.1, so we can define a residual R:

$$R(T, x) \equiv -K \frac{d^2T}{dx^2} - Q \tag{3.4}$$

Now we introduce the weighting function $W(x)$ which aim is to make the residual zero in the following way:

$$\int_0^L W(x)R(T,x)dx = 0 \tag{3.5}$$

The Galerkin method is used to determine the weighting function W. The function is set equal to the shape function $\phi(x)$:

$$W_i(x) = \phi_i(x) \tag{3.6}$$

The number of unknown parameters a_j is equal to the number of shape functions ϕ_j. This guarantees existence and uniqueness of the solution. The next step is to determine the shape functions. The most simple case is the linear approximation: ϕ_j is linear over the corresponding element. However, when combining equations 3.5 and 3.4 we obtain

$$\int_0^L \phi(x)\left(-K\frac{d^2T}{dx^2} - Q\right)dx = 0 \tag{3.7}$$

This expression contains higher-order deriveratives, which are present in the 'real' solution T, but vanish when T is linearly approximated over the elements and gives discontinuities at the nodes. Integration by parts provides a solution. Using integration by parts, equation 3.7 can be rewritten into

$$\int_0^L K\frac{d\phi}{dx}\frac{dT}{dx}dx - \int_0^l \phi Q dx - K\phi\frac{dT}{dx}|_0^L = 0 \tag{3.8}$$

All higher-order deriverative terms have now disappeared, but this equation still contains exactly the same information as equation 3.1. This form of the differential equation is called the weak formulation. Linear approximations over the elements have now become useful, since they have a nonzero first deriverative. Adding equation 3.3 leads to the following expression, which is called the Galerkin form:

$$\sum_{i=1}^{n+1}\left[\sum_{j=1}^{n+1} K\left(\int_0^L \frac{d\phi_i}{dx}\frac{d\phi_j}{dx}dx\right)a_j - \int_0^L \phi_i Q dx + \phi_i\left(-K\frac{dT}{dx}\right)|_{x=0}^{x=L}\right] = 0 \tag{3.9}$$

When the shape functions are known, this problem only has $j+1$ unknowns: a_j. An example of a linear approximation over elements is shown in figure 3.3. The problem can now be written into a matrix equation. The procedure to do this is called the assembly procedure.

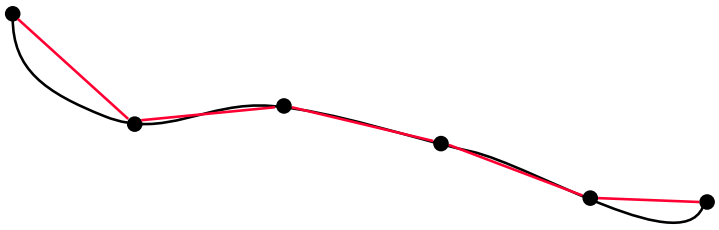

Figure 3.3: Example of a linear approximation in 1D

3.1.3 The assembly procedure

First we have to define our shape functions. These functions are linear and every node has his own shape function. Continuing the 1D example:

$$N_1 = \begin{cases} \frac{x_2 - x}{x_2 - x_1} & x_1 \leq x \leq x_2 \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

for the first element. For all the elements in between:

$$N_i = \begin{cases} \frac{x - x_{i-1}}{x_i - x} & x_{i-1} \leq x \leq x_1 \\ \frac{x_{i+1} - x}{x_{i+1} - x} & x_i \leq x \leq x_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

and for the final element:

$$N_{n+1} = \begin{cases} \frac{x - x_n}{x_{n+1} - x_n} & x_n \leq x \leq x_{n+1} \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

Now the weak form of the differential equation over the whole domain is divided into equations over every element:

$$\sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} \phi \left(-K \frac{\mathrm{d}^2 T}{\mathrm{d}x^2} - Q \right) \mathrm{d}x = 0 \tag{3.13}$$

and T can be expressed as:

$$T = \sum_{j=1}^{n+1} N_j(x) T_j \tag{3.14}$$

For simplicity in mathematical notation we now take 2 elements with length $L/2$. This leads to the following expression of the differential equation:

$$\sum_{i=1}^{3}\left\{\int_0^{L/2}\left[K\frac{\mathrm{d}N_i}{\mathrm{d}x}\left(\sum_{j=1}^{3}\frac{\mathrm{d}N_j}{\mathrm{d}x}T_j\right)-N_iQ\right]+\left[N_i\left(-K\frac{\mathrm{d}T}{\mathrm{d}x}\right)\right]_0^{L/2}\right.$$
$$\left.+\int_0^{L/2}\left[K\frac{\mathrm{d}N_i}{\mathrm{d}x}\left(\sum_{j=1}^{3}\frac{\mathrm{d}N_j}{\mathrm{d}x}T_j\right)-N_iQ\right]+\left[N_i\left(-K\frac{\mathrm{d}T}{\mathrm{d}x}\right)\right]_0^{L/2}\right\}=0 \quad (3.15)$$

This equation can be splitted in equations corresponding to each element. When these equations are combined, this leads to the following system of equations:

$$\frac{2K}{L}\begin{bmatrix}1&-1&0\\-1&2&-1\\0&-1&1\end{bmatrix}\begin{bmatrix}T_1\\T_2\\T_3\end{bmatrix}=\frac{Ql}{4}\begin{bmatrix}1\\2\\1\end{bmatrix}+\begin{bmatrix}q\\0\\-\left(-K\frac{\mathrm{d}T}{\mathrm{d}x}\right)_{x=l}\end{bmatrix} \quad (3.16)$$

This equation is solvable algebraically when T_3 is known ($T_{x=L} = T_3$), due to the boundary condition term in the right-hand side of the equation. The above method is easily expandable to higher-order shape functions and dimensions. This generally leads to very large systems of equations that must be solved.

3.1.4 Using the method in 3 dimensions

The example of the preceding paragraph is a 1D example, although the problems that must be solved are mostly in 3D. The finite element method can easily be used in 3D. To do so, the shape functions must be replaced by their 3D counterpart. When the domain is meshed by tetrahedra, every element is defined by 4 nodes: (a, b, c, d), and when linear shape functions are used, the nodal value ϕ is defined as:

$$\phi = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 z \quad (3.17)$$

This leads to an element matrix for the 4 nodes defining the element:

$$\begin{bmatrix}\phi_1\\\phi_2\\\phi_3\\\phi_4\end{bmatrix}=\begin{bmatrix}1&x_1&y_1&z_1\\1&x_2&y_2&z_2\\1&x_3&y_3&z_3\\1&x_4&y_4&z_4\end{bmatrix}\begin{bmatrix}\alpha_1\\\alpha_2\\\alpha_3\\\alpha_4\end{bmatrix} \quad (3.18)$$

To get the volume of every element the following relation is used:

$$V = \frac{1}{6}\det\begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{vmatrix} \tag{3.19}$$

3.2 Solvers

To solve the matrix equations of the form $A\boldsymbol{x} = \boldsymbol{b}$ arising from the finite element method, for example equation 3.16, the traditional method of Gaussian elimination would cost too much computational power, so iterative methods are used to approximate the exact solution. Fluidity uses the PETSc library of matrix solvers and preconditioners to solve the systems of equations. The goal of the solvers is to minimize the residual $\boldsymbol{r}_n$ of the approximate solution at the n-th iteration:

$$\boldsymbol{r}_n = \boldsymbol{b} - A\boldsymbol{x}_0 \tag{3.20}$$

After every iteration, a new guess is made for $\boldsymbol{x}$:

$$\boldsymbol{x}_{n=1} = \boldsymbol{x}_n + \boldsymbol{c}_n \tag{3.21}$$

When the residual is small enough, the approximate solution $\boldsymbol{x}_n$ is said to be converged.

3.2.1 Conjugate Gradient

The conjugate gradient method is an iterative method, which can be applied to real symmetric positive-definite matrices. The system of equations for the pressure field in fluidity is such a matrix. The algorithm from table 3.1 is used[7]. This algorithm requires A and P to be positive definite. P is the preconditioner matrix, which will be discussed in the next paragraph. The conjugate gradient method is fast and reliable, but not applicable to non-real, non-positive-definite matrices.

3.2.2 GMRes

To solve matrices which cannot be handled by the conjugate gradient algorithm, the Generalised Minimum Residual method can be used. The method constructs a basis of orthogonal vectors based on the Arnioldi iteration process. For a complete description of this method, see [19]. This method gives accurate results, but is significantly slower than the conjugate gradient

Table 3.1: Algorithm for the Conjugate Gradient method

1	$\boldsymbol{x}_0 = 0, \boldsymbol{r}_0 = b, \boldsymbol{t}_0 = \boldsymbol{s}_0 = P^{-1}\boldsymbol{r}_0, k = 0$
2	**while** $(\boldsymbol{r}_k, \boldsymbol{t}_k) > \epsilon^2(\boldsymbol{r}_0, \boldsymbol{t}_0)$
3	$\alpha_k = (\boldsymbol{r}_k, \boldsymbol{t}_k)/(\boldsymbol{s}_k, A\boldsymbol{s}_k)$
4	$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha k \boldsymbol{s}_k$
5	$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha k A \boldsymbol{s}_k$
6	Solve $P\boldsymbol{t}_{k+1} = \boldsymbol{r}_{k+1}$
7	$\beta_k = (\boldsymbol{r}_{k+1}, \boldsymbol{t}_{k+1})/(\boldsymbol{r}_k, \boldsymbol{t}_k)$
8	$\boldsymbol{s}_{k+1} = \boldsymbol{t}_{k+1} + \beta_k \boldsymbol{s}_k$
9	$k = k + 1$
10	**end while**

method, so it is only useful for solving non-real and non-positive definite systems of equations, when the conjugate gradient method is not capable of solving the system. This is the case for the velocity and temperature field.

3.2.3 Preconditioning

Even with an iterative method, solving a system of equations can consume a lot of time. To reduce the amount of time needed for calculation, preconditioners are used. The aim of a preconditioner is to create a new system of equations with a lower condition number. The condition number of a matrix is the ratio between the largest and smallest eigenvalue. The lower the condition number, generally the faster the system can be solved. In Fluidity the main preconditioner is SOR: Successive Over Relaxation:

$$M = (D + L)D^{-1}(D + L)^T \tag{3.22}$$

Solving the new system $M^{-1}A\boldsymbol{x} = M^{-1}\boldsymbol{b}$ will reduce calculation time significantly.

3.3 Meshing and adaptive remeshing

To get accurate results using the Finite Element method, a suitable mesh must be created. By using a finer mesh, results will become closer to the real solution, however, the calculation time will grow when the mesh is refined. So the mesh must be chosen in a way that the results are close enough to the real solution, but small enough to keep the time needed to process within bounds. There are 2 ways of using meshes in the simulation:

1. Using a predefined mesh: before the simulation starts, define a mesh and use this mesh throughout the simulation.

2. Adaptive remeshing: adapt the mesh during the processing to capture details where needed and use bigger elements where appropriate.

3.3.1 The predefined mesh

The Fluidity code can read meshes from the 'GMSH' mesh format: meshes created by the open source code GMSH [2]. With this program, the geometry is defined and can be meshed. The geometry is defined with points and lines connecting the points. The characteristic element size can be defined on every point, so the mesh can be refined or coarsened at certain points. GMSH can mesh the geometry using a variety of algorithms:
2D algorithms

1. Meshadapt

2. Delaunay

3. Frontal

3D algorithms

1. Delaunay

2. Frontal

Since the aim of this research is 3D simulations, we briefly discuss the Delaunay and Frontal algorithms here.

Frontal meshing

Frontal meshing is the most straightforward method. Define nodes at all the boundaries of the domain and then start to expand into the domain, until it's completely covered.

Delaunay meshing

The Delaunay mesh algorithm makes use of Delaunay the triangulation principle [23]. A triangulation of a collection of nodes V is the collection of triangles (or tetrahedra in 3D) that connects all vertices without any line of any triangle intersecting any other. A circle is empty when there is not any

[2] See http://geuz.org/gmsh/

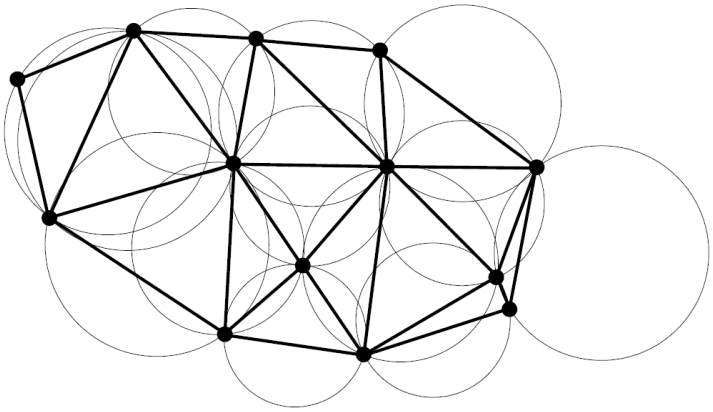

Figure 3.4: Example of a Delaunay triangulation, picture from [23]

node inside (Nodes on the boundary are allowed). Now, an edge for which a circle exists that circumscribes this edge and is empty is called Delaunay. When every edge has a empty circumcircle, the triangulation is called a Delaunay triangulation. See for an example figure 3.4 The Delaunay meshing algorithm now looks for non-Delaunay edges in the mesh, and changes the mesh, until it's completely Delaunay. There are a few algorithms to do so. See for example [23]. The advantage of a Delaunay algorithm is that is maximizes the minimum angle between edges, so the aspect ratio of the elements remain small. Small aspect ratios give more accurate results in the Finite Element Method[10].

3.3.2 Adaptive remeshing

One way of improving the calculation while minimalizing the computational power required is adaptive remeshing: changing the mesh in a way that the mesh is detailed where needed and coarse where possible. The scheme used by fluidity is developed by C.C. Pain,, A.P. Umpleby, C.R.E. de Oliveira and A.J.H. Goddard and documented in [11]. A brief overview (based on [11]) is given here. To calculate the quality of the mesh, the following functional is to be minimized:

$$\mathfrak{F} = \|F\|_p \tag{3.23}$$

In which F is a vector containing all elements of the domain, defined by

$$F_e = \frac{1}{2}\sum_{l\in\mathfrak{L}_e}(\delta_l)^2 + \mu(q_e)^2 \tag{3.24}$$

where $\mathfrak{L}_e$ is the set of all edges in element e and μ is equal to 1. In this functional, the first term describes the size of the element, and the second term describes the shape.

The metric

The metric used in equation 3.24 is made dependent on the desired error. In a multi-dimensional grid, the metric is defined as

$$\hat{M} = \frac{\hat{\gamma}}{\hat{\epsilon}} |H| \tag{3.25}$$

with $\gamma = 1$, and ϵ the desired interpolation error. H is the Hessian of the field that is adapted to.
The term δ_l from equation 3.24 is defined as $\delta_l = r_l - 1$ with r_l the length of edge δ_l with respect to the metric M. It's constructed in such a way that the edge length l becomes 1 with respect to the metric M when the desired interpolation error ϵ has been reached. The shape function q_e is defined as:

$$q_e = \frac{\alpha}{\rho_e} - 1 \tag{3.26}$$

α being $1/(2\sqrt{6})$ and ρ_e the radius of the inscribed sphere of element e with respect to the metric. Now, the adaptive remeshing code makes changes to the mesh and compares the changed mesh with the old mesh. When the change of the mesh results in a lower value of $\mathfrak{F} = \|F\|_p$, the change is accepted, otherwise, it's rejected. This process is repeated until the desired interpolation error bound has been reached.

3.4 Temporal discretization

Time is also discretisized in the simulation. Unlike some other CFD codes, Fluidity can't be used to do time-independent steady-state calculations. Since the amount of transferred quantity is time-dependent, it's necessary to discretisize the time in a way the solution of the computational problem represents the real fluid behavior. Fluidity works with timesteps, at which the equations are solved. After a sufficiently converged solution is calculated for a certain timestep, a new timestep will start.

3.4.1 Explicit and implicit control

The timestep discretization process is the algorithm that replaces the time-dependent deriveratives in the governing equations by a numerically solvable

system. One of these methods is the so called Θ-method. The Θ-method replaces the time deriverative by a simple difference:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{u(t^{n+1}) - u(t^n)}{\Delta t} \tag{3.27}$$

The Θ-method uses the parameter Θ to control the discretization process:

$$\frac{u(t^{n+1}) - u(t^n)}{\Delta t} = \Theta u^{n+1} + (1 - \Theta)u^n \tag{3.28}$$

The parameter Θ must be on the interval $[0, 1]$. Three choices of Θ are common:

$$\begin{array}{ll} \Theta = 1 & \text{Backward implicit method} \\ \Theta = 1/2 & \text{Crank-Nicolson-Galerkin method} \\ \Theta = 0 & \text{Explicit Euler forward scheme} \end{array} \tag{3.29}$$

For all the values of Θ except $\Theta = 0$, the value of u^{n+1} must be obtained implicitly from this equation. The methods differ in accuracy and stability, and all methods with $\Theta < 1/2$ are not unconditionally stable. The most widely used method is the Crank-Nicolson-Galerkin method, which is always stable, and gives accurate results when the timestep is small enough. The timestep size is discussed in the next paragraph. When the timestep becomes too big, the method will show oscillatory behavior, which is a non-physical outcome of the equations[25]. The explicit timestepping algorithm ($\Theta = 0$), is only conditionally stable, and the numerical accuracy is lower[2].

3.4.2 Courant Number and adaptive timestepping

When the timesteps are getting too large, the distance some quantities advect along the elements, can get longer than the cell size. This will make the results obtained in the affected area unusable, since they don't represent a mathematical correct solution. To avoid this situation, the Courant Friedrichs Lewy condition must be met [9]:

$$\boldsymbol{v}\frac{\Delta t}{l} < 1 \tag{3.30}$$

with l the smallest length element in the mesh, and Δt the timestep. To get accurate results, but optimize calculation speed, the timestep size can be adapted to meet this condition. For the Crank-Nicolson method, the Courant number:

$$\mathrm{CFL} = \boldsymbol{v}\frac{\Delta t}{l} \tag{3.31}$$

must also be equal to or smaller than 1 at every element to prevent oscillatory behavior of the solution[2].

3.5 Turbulence modelling

The flows which are going to be simulated are in the high-Reynolds-region, which imposes that turbulence effects will occur. The multiscale character of turbulence makes it impossible so solve for the smallest vortices in the flow, due to the limitations in computational power. To include the effects of turbulence into the simulation, a model is needed to incorporate the turbulent strains.

3.5.1 Large Eddy Simulation

Since the larger eddies in a turbulent region depend mostly on the geometry, and the smaller eddies don't, the aim is to solve only for the large eddies and use a sub-gridscale model for the smaller eddies. The Navier Stokes equation in incompressible form is rewritten [2] into

$$\rho\frac{\partial\tilde{\boldsymbol{v}}}{\partial t}+\rho\nabla\cdot(\tilde{\boldsymbol{v}}\tilde{\boldsymbol{v}})=\nabla\cdot\tilde{\mathrm{T}}+\rho\boldsymbol{b} \tag{3.32}$$

where $\tilde{\boldsymbol{v}}$ is the above-gridscale (average) velocity. The stress tensor T is then defined as

$$\mathrm{T}=-p\mathrm{I}+2(\mu+\mu_t)\mathrm{D} \tag{3.33}$$

The next step is to find the correct value of μ_t, the sub-gridscale eddy viscosity. The most widely used method to determine μ_t is the Smagorinsky model.

3.5.2 The Smagorinsky model

The Smagorinsky model used by Fluidity is[4]

$$\mu_t=\rho C_s^2\hat{M}^{-1}\left|\tilde{D}\right| \tag{3.34}$$

The metric $\hat{M}$ from the adaptive remeshing process described in paragraph 3.3.2 is used as the filter scale in Fluidity. C_s is the Smagorinsky coefficient and $\tilde{D}$ is the above-filterscale tension tensor, as defined in equation 2.9.

3.5.3 The Smagorinsky coefficient

To close the Smagorinsky model, the value of C_s needs to be determined. Entering the correct value of C_s is important: the value of the coefficient directly influences the rate of turbulent dissipation. There is, however, no

'correct' value of the constant. One of the used expressions is for example [22]:

$$C_s = \frac{1}{\pi}\left(\frac{2}{3\alpha}\right)^{3/4} \tag{3.35}$$

Where $\alpha \approx 1.5$ is the Kolmogorov constant, which leads to a value for C_s of 0.17. There are, however, various studies which report values between 0.065 and 0.25, for example [4], [2] and [12].

Chapter 4

Model validation

To check if the results obtained from the CFD code are in accordance with real results, four test case have been run. The first two tests (in paragraph 4.1 and 4.2) are 2D tests to check if the calculations done on the velocity field lead to useful results. The last two tests are a 2D (paragraph 4.3) and a 3D (paragraph 4.4) test in which the temperature field is checked.

4.1 2D Laminar flow through a tube.

The first test case is the laminar flow through a tube at low Reynolds numbers (RE << 2000). In such case, a Poiseuille flow will occur in the tube. For the Poiseuille flow, an exact solution of the Navier Stokes equation exists, so it can be easily compared.

4.1.1 Physical parameters and geometry

For this case the physical parameters are used as in table 4.1. For such a flow, the solution to the Navier Stokes equation is known as the Hagen Poiseuille

Table 4.1: Physical parameters for the 2D tube test case

Property	Dimension	Value
Tube length	m	1
Tube diameter	m	0.1
Inflow velocity	m/s	$1.0 \cdot 10^{-3}$ uniform over the inflow
Dynamic viscosity	$Pa \cdot s$	$1.0 \cdot 10^{-3}$
Density	kg/m^3	1000

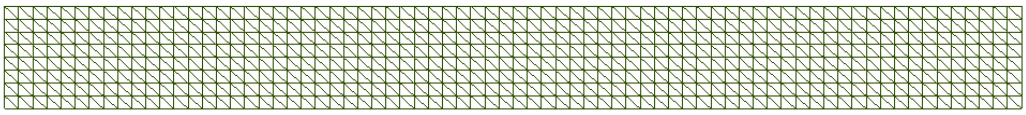

Figure 4.1: Mesh of the 2D tube case

Law[1]:

$$\Delta p = \frac{32 \mu v_{inflow} L}{d^2} \tag{4.1}$$

with Δp the pressure drop over the tube, d the tube diameter, v_{inflow} the mean inflow velocity and L the tube length. Filling in the values of table 4.1 in equation 4.1 leads to an expected pressure drop of $3.2 \cdot 10^{-3}$ Pa.

4.1.2 Numerical discretizations

The mesh used in this calculation is shown in figure 4.1. A timestep size of 0.01 seconds is used and the simulation stops after 10 seconds. As temporal discretization, the Crank-Nicolson scheme is used for it's numerical stability and reliability.

4.1.3 Results

The simulation returns a pressure drop of $3.1 \cdot 10^{-3}$ Pa, which is close to the expected value. Also the characteristic parabolical shape of the velocity field over the tube diameter is observable, when compared to the theoretical graph, see figure 4.2. The difference between the expected result and the computed result, although not big may have a few causes, the numerical discretization may be too coarse, and the Hagen Poiseille law only holds for tubes where the inflow is already fully developed, e.g. not uniform, but parabola-shaped. Since the inflow in this case is uniform and has to develop, the pressure drop may be different.

4.2 2D Flow past cylinder

The second test case is the simulation of a 2D non-thermal flow. This is a standard CFD test case, so both numerical and experimental data are widely available. This makes verification of the obtained results very easy.

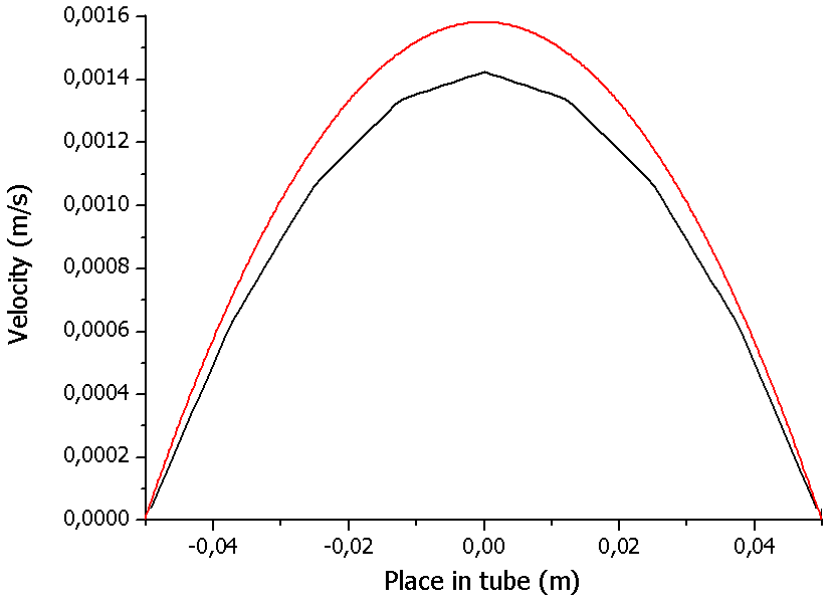

Figure 4.2: Flow velocity over the tube diameter (black) compared with the theoretical solution (red)

Table 4.2: Physical parameters of the 2D flow past cylinder test case

Property	dimension	value
Inflow velocity	m/s	1
Density	kg/m^3	0.01
Dynamic viscosity	$Pa \cdot s$	$2.8 \cdot 10^{-5}$
Reynolds number	-	100

4.2.1 Physical parameters and geometry

A non-thermal flow is being simulated. The geometry is shown in figure 4.3. The fluid properties of this case are listed in table 4.2. In this test case, a periodic oscillation is expected to occur in the domain[5]. This oscillation is called vortex shedding and causes a Von Kárman vortex street in the wake behind the cylinder. To verify this case, the Strouhal-relation is used. The Strouhal number is the dimensionless parameter of the frequency of the oscillation and is defined as[6]:

$$\mathrm{St} = \frac{fd}{\bar{v}} \tag{4.2}$$

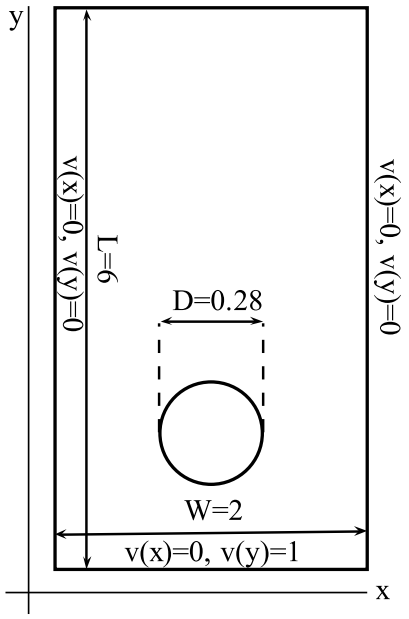

Figure 4.3: The used geometry with the boundary conditions

Table 4.3: Settings for the 2D flow past cylinder case

Test run no.	Interpolation errorbound	Timestep
1	0.04	0.001
2	0.06	0.001
3	0.06	0.005

Where f is the oscillation frequency and $\bar{v}$ the mean inflow velocity. There is a strong empirical relationship between the Strouhal and Reynolds number, which is discussed in section 4.2.3.

4.2.2 Numerical discretisations

In this test case the adaptive remeshing option of Fluidity is tested, as described in section 3.3.2. Three test runs have been computed, each with a different value of the desired velocity interpolation error bound. The timestep size is $1{\cdot}10^{-4}$s. The Crank-Nicolson scheme is used as temporal discretization as in the previous test cases. Three test cases have been run. The description of the test cases is listed in table 4.3. To simulate sub-gridscale turbulence, a large eddy simulation model is used, as described in section 3.5.1. The Smagorinsky coefficient C_s used is 0.1, which is the recommended value for

Table 4.4: Test results for the 2D flow past cylinder case

Test run nr.	Run time	Maximum no. of nodes	Strouhal-number
1	1330 min	19500	0.224
2	754 min	11810	0.243
3	502 min	103500	err

most cases [4].

4.2.3 Results

The results of the 3 test runs are listed in table 4.4. The timestep size of test run 3 was too big, so the Courant number exceeded 1. This caused an overshoot around the cylinder, which resulted in a non converging solution after $t = 5.57$. Because of this overshoot, no vortex street was visible in the wake after the cylinder. It also led to a rise in the number of nodes needed in calculation. Because of this, this measurement must be considered as unusable. The other two simulations showed the expected vortex street behavior in the cylinder wake, see figure 4.4. The Strouhal number of the two simulations gave values around 0.2, which is in accordance with the various existing computational and experimental data available, which predict values of St between 0.16 and 0.31 for Re = 100[6][5]. The difference in literature data may be caused by small geometry differences and boundary condition settings. Most of the simulations use a no-normal-flow condition for the side boundary instead of a no-slip condition. However, the obtained results are in accordance with the expected values. The problems in test run 3 show that it is important to watch the Courant number carefully, and make sure it does not exceed 1.

4.3 2D Forced convection around cylinder

After the verification of the velocity field, the thermal flow model must be validated. The 2D forced convection around a cylinder has been chosen to do this job, because extensive research had been conducted about the Nusselt relation of this case[8], so the results can be easily checked.

4.3.1 Physical parameters and geometry

The geometry is a simple rectangle with a cylinder in it, see figure 4.5. In contrast to the non-thermal flow validation case from paragraph 4.2 the outside

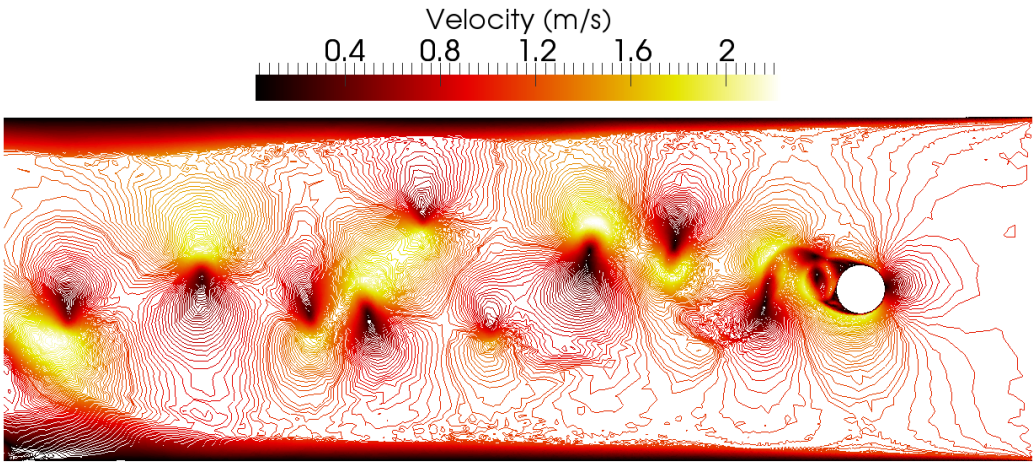

Figure 4.4: Contour plot of the magnitude of the velocity on the 2D flow past cylinder case, test run 2, after 800 timesteps

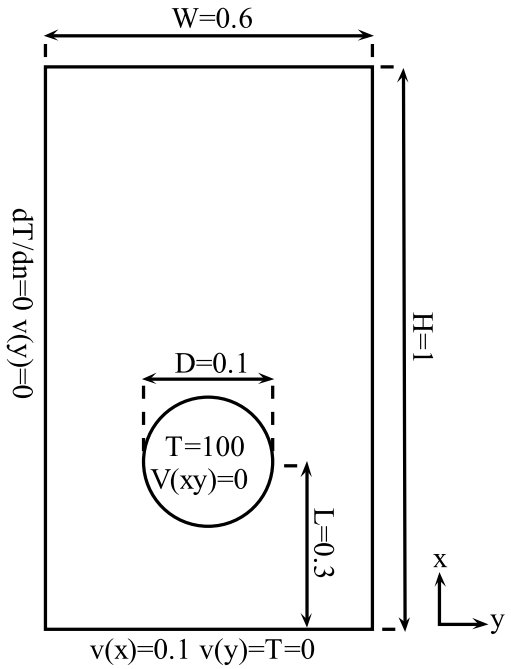

Figure 4.5: Geometry and boundary conditions of the 2D forced convection case

Table 4.5: Physical and model properties of the 2D forced convection case

Quantity	Value
Kinematic viscosity	$2 \cdot 10^{-5}$
Thermal diffusivity	$2 \cdot 10^{-5}$
Prescribed density	1
Reynolds	500
Prandtl	1
Cylinder perimeter	0.1π

Table 4.6: 2D forced convection case test settings

Test Run	Abs. Vel. interpl. Errorbnd.	Errorbnd. as % of v_{inflow}
1	0.001	1%
2	0.002	2%
3	0.008	8%

wall boundary condition has been changed from a no-slip to a no-normal-flow condition. Using this boundary conditions, all the heat transferred to the fluid must flow out of the boundary at the top when steady state has been reached. All the relevant physical and model parameters are listed in table 4.5. This case is extensively researched, and the following Nusselt relation has been found [8], assuming constant viscosity:

$$\text{Nu} = (0.4\text{Re}^{1/2} + 0.06\text{Re}^{2/3})\text{Pr}^{1/4} \tag{4.3}$$

leading to a expected Nusselt number of 12.7. The Nusselt number can be extracted from the steady state simulation result. To do so, the surface integral of the velocity times temperature over the outflow must be taken, which yields the total heat flux: $\phi_Q = \rho c_p \int_O v_x(T - T_0)\mathrm{d}A$. Then Nusselt is given by:

$$\text{Nu} = \frac{\phi_Q}{\pi(T_{\text{sphere}} - T_{\text{inflow}})\lambda} \tag{4.4}$$

4.3.2 Numerical discretizations

The simulation has been run several times, with different setting for the adaptive remeshing error bound. The different settings are listed in table 4.6. To overcome the problems with the Courant number exceeding 1, as in the 2D flow past cylinder case, an adaptive timestepping algorithm is used in this case, which changes the timestep in such a way that the Courant number never exceeds a selected value. In this test case, the maximum

Table 4.7: Results of the 2D forced convection case

Test Run	No. of nodes	Mean timestep	Averaged Nusselt number
1	17000	0.0015	15.6
2	11000	0.002	16.3
3	2500	0.0035	18.3

allowed Courant number is 0.9. The same LES model is used as in the 2D flow past cylinder case with the same value for the Smagorinsky coefficient: $C_s = 0.1$.

4.3.3 Results

As expected in the 2D cylinder case, vortex shedding occurred, so there is no steady state, and the energy outflow is not constant over time. The mean output energy over one period had to be calculated. This led to the results in table 4.7. As seen from the results, the Nusselt number seems to converge to a lower value as the number of nodes increases, and test run 1 is within a fair range of 25 percent of the measured value, the same variance as reported in [8]. In literature, different values of the Smagorinsky coefficient are used. When the grid is coarse, the Smagorinsky model will have a greater impact in the calculation results than when the grid is fine. It is difficult to obtain a good value of this coefficient which gives accurate results for a specific case. Recently new models have been developed, which use a local value for the Smagorinsky coefficient at each node, see for example [14]. Unfortunately, the Fluidity code doesn't contain such a model. Since the results of the simulation strongly depends on the subgridscale turbulence modelling, which in itself depends on the value of the Smagorinsky coefficient, the value of this number must be chosen carefully.

4.4 3D Forced convection around sphere

The last test case is a 3D sphere with a uniform temperature placed in a flow with a constant inflow temperature. Since a Nusselt relation in known for this case, it is a good check for all the parameters in 3D.

4.4.1 Physical parameters and geometry

This simulation uses the geometry and boundary conditions as in figure 4.6. The temperature gradient at the outer walls is zero (Neumann boundary

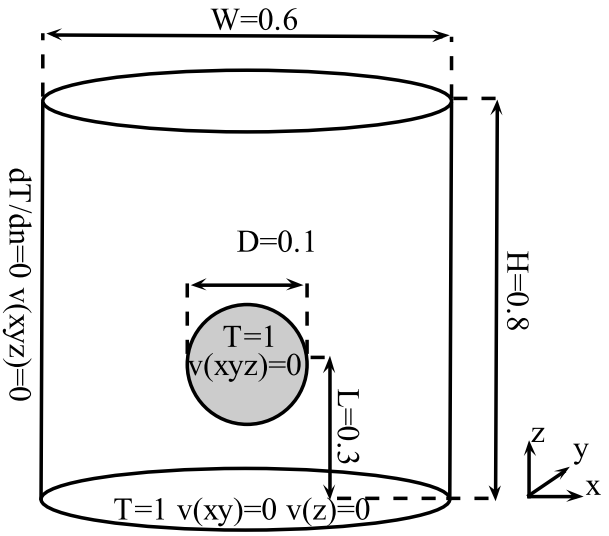

Figure 4.6: Geometry of the 3D sphere forced convection case

Table 4.8: Used constants at the 3D sphere forced convection case

Dynamic viscosity	$2 \cdot 10^{-5}$
Density	1
Thermal diffusivity $\left(\frac{\lambda}{\rho C_p}\right)$	$2 \cdot 10^{-5}$
Pr	1
Re	5000
Pe	5000

condition). The inflow temperature is zero and the sphere temperature is 100. All the used physical values are listed in table 4.8. This case has been extensively researched, which has led to the following Nusselt relation (under the assumption that the viscosity is constant i.e. not varying with temperature)[8]:

$$\mathrm{Nu} = 2 + \left(0.4\mathrm{Re}^{1/2} + 0.06\mathrm{Re}^{2/3}\right) \cdot \mathrm{Pr}^{0.4} \tag{4.5}$$

In this case Nu is determined by measuring the total energy outflow:

$$\phi_Q = \rho C_p \int_\Omega v_z \cdot T \mathrm{d}\Omega \tag{4.6}$$

Now the heat transfer coefficient h is defined as [8]:

$$h = \frac{\phi_Q}{4\pi d^2 \Delta T} \tag{4.7}$$

Table 4.9: Test results of the 3D forced convection case

Test case	Velocity error bound	Temp. error bound	Nodes	Nu
1	0.6	45	11000	34,3
2	0.4	30	36000	26.2
3	0.2	15	154000	41.8

And Nu is defined as

$$\mathrm{Nu} = \frac{hd}{\lambda} \tag{4.8}$$

In this case, equation 4.5 gives Nu=46.

4.4.2 Numerical discretizations

In this simulation, adaptive remeshing is used with various parameters. The Smagorinsky LES model is used to simulate turbulence effects, which are expected due to the high Reynolds number (Re = 5000). The value of the Smagorinsky coefficient $C_s = 0.17$, the same as used in the Pebble Bed simulation in [14]. Three simulations have been run with a varying interpolation error bound, so coarse and fine meshes were checked.

4.4.3 Results

The results of the various measurements are listed in table 4.9. As can be seen, the values for Nu are below the expected value, but when the grid becomes fine enough, the value of Nu approaches this value. Run 2 gives an unexpected result: the Nusselt number lies 42% below the expected value, while the highest measured deviation from equation 4.5 reported by [8] is 30%. So, to make accurate predictions of the temperature field, the mesh bust be very fine. The interpolation error bound can be calculated as a percentage of the input velocity and temperature. This leads to a relative value of the interpolation error bounds: $\epsilon^{\mathrm{rel}}_{\mathrm{velocity}} = \epsilon^{\mathrm{abs}}/v_{\mathrm{inflow}}$ and $\epsilon^{\mathrm{rel}}_{\mathrm{temperature}} = \epsilon^{\mathrm{abs}}/T_{\mathrm{inflow}}$ which leads to $\epsilon^{\mathrm{rel}}_{\mathrm{velocity}} = 0.2$ for the velocity interpolation error bound and $\epsilon^{\mathrm{rel}}_{\mathrm{temperature}} = 0.15$ for the temperature interpolation bound for test case 3, which returned adequate results.

Chapter 5

Pebble Bed simulation

The main aim of this research is to determine if Fluidity can be used to do thermal calculations on the coolant flow in VHTR reactors. In order to check this, a small part of a pebble bed is simulated. In this chapter, the set-up and results of the pebble bed simulation will be discussed. The following paragraphs are about the geometry, boundary conditions and simulation settings. The results are discussed in paragraph 5.5.

5.1 Geometry

Since the computational power required to do a complete DNS calculation over the whole pebble bed is too big, even for the world's fastest computers, a small domain of the pebble bed is taken for simulation. The geometry of the simulation is a 3D non-random pebble stacking, consisting of 9 pebbles, 6 half pebbles, 12 quarter pebbles and 8 1/8 pebbles, as shown in figure 5.1. The pebbles have a diameter of 5 centimeter and a space between them of 1 centimeter. These values have been chosen, because Fluidity is not capable of processing two pebbles touching eachother(Processing such a geometry leads to non-converging systems of equations), which would be the case if 6 cm pebbles would have been chosen in the same domain. This geometry can be considered as a sort of unit cell: when the same geometry is placed on top of or next to the first one, it still represents a pebble bed. This pebble bed has a packing fraction of 45 %.

5.2 Properties of helium

Helium is used as coolant in the pebble bed reactor. The properties of helium are listed below and are valid in the ranges $1 \cdot 10^5 \text{Pa} < p < 1 \cdot 10^7 \text{Pa}$ and

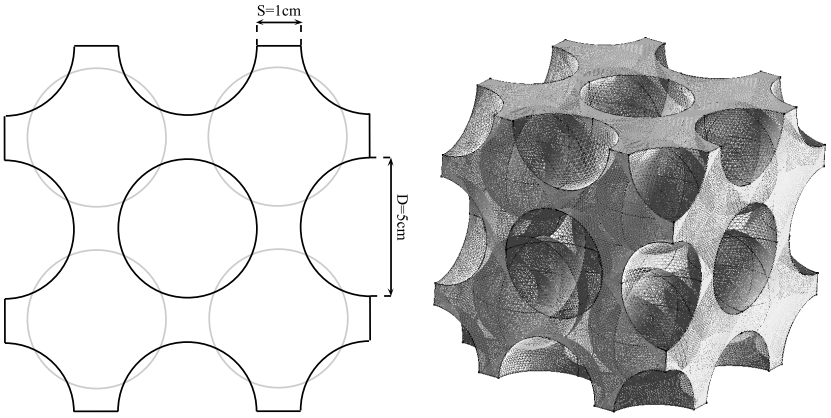

Figure 5.1: 2D cross section and 3D view of the geometry

$239\text{K} < T < 1773\text{K}$.

Density

The density of helium in the reactor core is given by[15]:

$$48.14 \cdot 10^{-6} \frac{p}{T} \left(1 + 0.4446 \cdot 10^{-6} \frac{p}{T^{1.2}}\right)^{-1} \tag{5.1}$$

Specific heat

The specifc heat of helium is given by[15]:

$$c_p = 5195 \frac{\text{J}}{\text{KgK}} \tag{5.2}$$

Dynamic viscosity

The dynamic viscosity is given by[15]

$$\mu = 3.674 \cdot 10^{-7} T^{0.7} \tag{5.3}$$

Thermal conductivity

The thermal conductivity of helium is given by[15]:

$$\lambda = 2.682 \cdot 10^{-3} \left(1 + 1.123 \cdot 10^{-9} p\right) T^{0.71\left(1 - 2 \cdot 10^{-10} p\right)} \tag{5.4}$$

Table 5.1: Physical parameters of the core calculations

Quantity	Value
Background pressure	3 MPa
Mean temperature	773K
Diffusivity	$3.1 \cdot 10^{-5} \frac{\mathrm{m}^2}{\mathrm{s}}$
Density	$1.86 \frac{\mathrm{kg}}{\mathrm{m}^3}$
Dynamic viscosity	$3.86 \cdot 10^{-5} \mathrm{Pa} \cdot \mathrm{s}$

Other physical parameters

The viscosity, density and thermal diffusivity terms are constant, using the equations from section 5.2 and a temperature of 773K and an inlet pressure of 3MPa. The diffusivity term is defined as:

$$a = \frac{\lambda}{\rho C_p} \tag{5.5}$$

This leads to the values as in table 5.1.

5.3 Boundary and initial conditions

Velocity boundary conditions

The coolant is modeled to flow downwards. At the top of the geometry, a Dirichlet Boundary condition is applied with $v_{in} = 0.71\mathrm{m/s}$. This value has been chosen to represent the flow rate in the Chinese HTR-10 test reactor, which uses a mass flow of 4.3kg/s[18], which leads to an average flow speed of 0.6m/s in the pebble bed reactor at a packing fraction of 60 %. Aiming at the same mean helium velocity at this simulation leads to an inflow velocity of 0.71m/s, because the inflow area is relatively small compared to the average flow area. At the pebble boundaries, a no-slip condition is applied, and at the side boundaries a no-normal-flow condition is applied: the normal component of the velocity at the side boundaries is zero. This approximates the periodic character of the geometry. The background pressure is 3MPa, a typical value for a pebble bed reactor[18].

Temperature boundary conditions

The pebble boundaries are at a constant temperature of 800 K, which is a typical value for the temperature inside the reactor core, since the aim of the VHTR reactor is to reach temperatures between 500 and 1200K. The helium

Table 5.2: Adaptive remeshing settings for the core calculations

Setting	Value
Velocity interpolation error bound	0.15 m/s
Temperature interpolation error bound	4K
Minimum edge length	$4 \cdot 10^{-4}$m
Remeshing period in timesteps	20

inflow temperature is 773K. The temperature difference of 27K is typical for the pebble bed reactor and calculated in [21]. The low temperature difference between the helium and the pebbles makes it possible to neglect the temperature dependency of the physical properties of helium and take them as constant. The side boundaries carry a zero-flux boundary condition, so no heat is transferred from or into the domain.

5.4 Numerical model

The simulation uses the adaptive remeshing code of Fluidity , with the settings as listed in table 5.2. The interpolation error bound settings are based on the same relative errors as in the 3D forced convection case in paragraph 4.4.3: $\epsilon^{rel}_{temperature} = 0.15$ of the the temperature difference of 27K leads to an errorbound of 4K and $\epsilon^{rel}_{velocity}$ of the inflow velocity of 0.71 m/s leads to a bound of 0.15 m/s. To deal with turbulence effects, large eddy simulation is used, with the Smagorinsky coefficient $C_s = 0.17$, the same value as used in the LES simulation of a pebble bed in [14]. An adaptive timestepping is used, with a maximum Courant number of 1.

5.5 Measurement results

The simulation has been run using the setting from the preceding sections on the TUDelft HPC11 server cluster using 2 cores for faster calculation times. After 2200 variable timesteps, a total time of 0.5 seconds has been simulated. This led to a mesh with 350.000 nodes. The calculation took 14 days. The main fields that have been calculated are the pressure, velocity and temperature fields, and the results of each field will be discussed here.

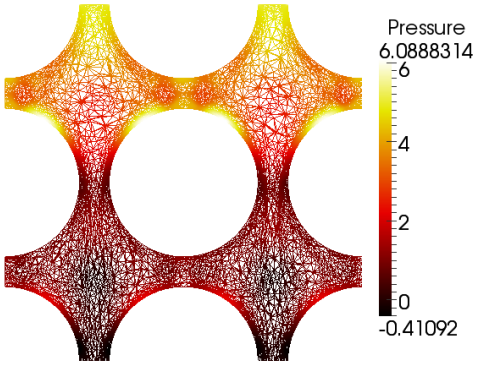

Figure 5.2: Pressure field at x-normal

5.5.1 Pressure field

The pressure field at the x-normal at the middle of the domain in the steady state is shown in figure 5.2 The pressure drop over the simulation area is about 7 Pascal. For packed beds, the average pressure drop can be estimated by the Ergun relation[28]:

$$\frac{\Delta p}{\Delta x} = -\left(\frac{150\mu}{d_{\text{pebble}}}\frac{(1-\gamma)^2}{\gamma^3} + \frac{1}{2}\rho\frac{3.5}{d_{\text{pebble}}}\frac{(1-\gamma)}{\gamma^3}\bar{v_x}\right)\bar{v_x} \tag{5.6}$$

with γ the packing fraction. This leads for the parameters of our simulation to an expected pressure drop of 17 Pascal, a value within the order of magnitude, but still about a factor 2 higher than the simulation value. This may be explained by the fact that our domain is very small, containing only a few pebbles, and the Ergun relation, which is based on the average pressure drop over large packed beds, could break when used in very small domains. Also the fact that the simulated domain is a set of ordered pebbles not touching eachother could cause differences with the expected value, which is based on a randomly filled packed bed. The fact that our simulation domain is small, and the velocity boundary condition is uniform at the inflow could cause differences, because the flow pattern must develop.

5.5.2 Velocity field

The z-velocity fields of the x-normal at the origin and $x = 0.04$ are shown in figure 5.3. The z-velocity varies between -2.9 m/s and +1.1 m/s. The presence of positive values of the x-velocity points at the occurrence of vortices in the simulation. There are indeed areas in the domain which show such

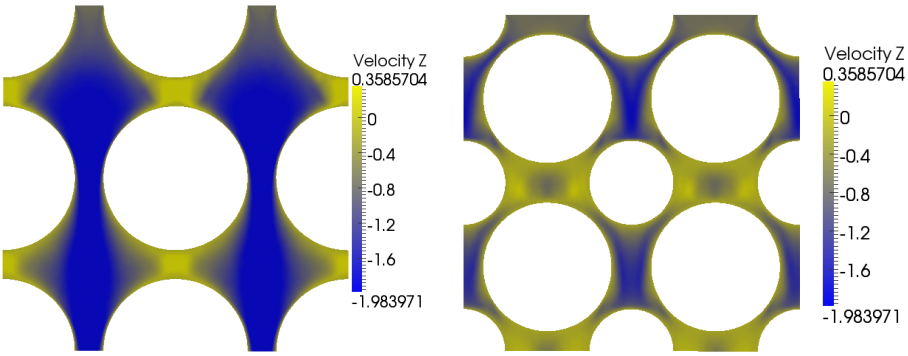

Figure 5.3: Z-velocity at the x-normal at the origin(left) and $x = 0.04$(right)

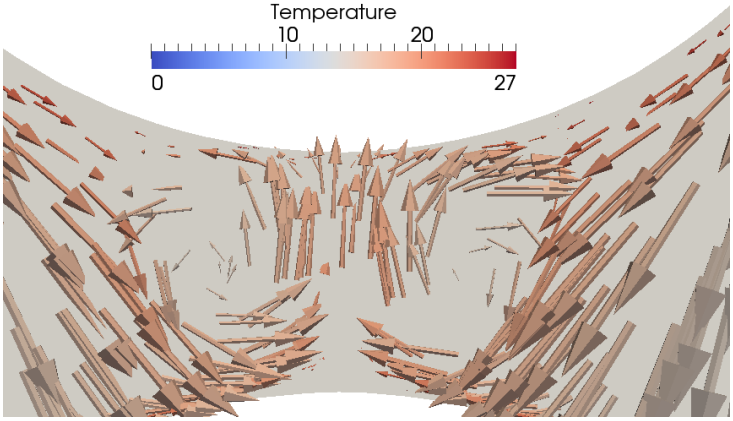

Figure 5.4: Typical vorticity behavior between 2 vertically adjacent pebbles

behavior, especially in the small areas between 2 vertically adjacent pebbles. A detailed view of such a vortex is shown in figure 5.5.2.

5.5.3 Temperature field

A x-normal cross section of the temperature field at $x = 0$ and $x = 0.04$ is shown in figure 5.5 The difference between the inflow temperature and the mean outflow temperature is 16K. A Nusselt relationship for pebble bed flow

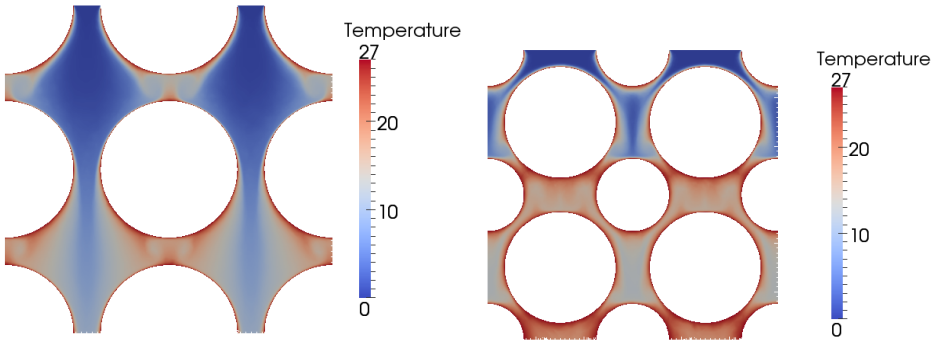

Figure 5.5: Temperature field at the x-normal at the origin(left) and $x = 0.04$(right)

has been proposed by Gnielinski[29]:

$$\mathrm{Nu} = (1 + 1.5\,[1-\gamma]) \times \left(2 + \sqrt{\left[0.664\frac{\mathrm{Re}^{\frac{1}{2}}}{\gamma}\mathrm{Pr}^{\frac{1}{3}}\right]^2 + \frac{0.037\frac{\mathrm{Re}}{\gamma}^{0.8}\mathrm{Pr}}{1 + 2.443\frac{\mathrm{Re}}{\gamma}^{-0.1}\left(\mathrm{Pr}^{\frac{2}{3}} - 1\right)}}\right) \tag{5.7}$$

Filling in all values, with $\Delta T = 27$ leads to $\mathrm{Nu} = 64,3$, with $\mathrm{Re} = 1446$ and $\mathrm{Pr} = 0.67$, which results in $h = 162\mathrm{W/m^2K}$. The measured value, obtained in the same way as the 3D forced convaction case from paragraph 4.4, is $\Phi_Q = 676.4\mathrm{J/s}$. This results in a heat coefficient $h = 199.3\mathrm{W/m^2K}$, which is somewhat higher than the expected value. According to the fact that the Nusselt relation is based on large randomly-filled packed beds, a deviation between the calculated results and the Nusselt relationship of equation 5.7 can be expected. However, it is within the range of 20%, which, taking all the simplifications into account, is acceptable.

Chapter 6

Conclusion

The aim of the research is to investigate if the Fluidity CFD code is capable of doing calculations on pebble bed cores. The research contains two parts, the 4 test cases and the pebble bed calculations.

6.1 Test cases

The four test cases have shown a few requirements for the simulations to produce useful results. The momentum calculations are in accordance with the expected results, but the thermal calculations must be treated carefully: coarse grids can cause results that differ significantly from experimental data. The lack of a local adaptive Smagorinsky coefficient model makes results of the LES turbulence model questionable. However, with a fine grid and the appropriate value of C_s acceptable results are produced. The adaptive remeshing code is proven to produce usable meshes, however, the interpolation error bound setting ϵ must be set carefully, because too big values render results which differ from expected outcomes. It showed that $\epsilon^{rel}_{temperature} = 0.15$ and $\epsilon^{rel}_{velocity} = 0.20$ provide results which are within 25% of the expected value, which is within the error marge expected in these cases.

6.2 Pebble bed calculations

The core calculation in this project is a simplified model. The main simplifications are:

- The temperature of the pebbles is uniform at the boundary, in contrast with the real situation, where the boundary of the pebble varies with the local flow regime and the properties of the fission process inside.

- The pebble stacking in the simulation consists of pebbles which do not touch each other and are not randomly stacked. This led to a packing fraction of 45 percent, which is lower then real pebble beds with percentages of about 55-60 percent packing fraction.

- The simulated domain is a 12 by 12 centimeter cube, which is only a small part of a reactor.

Taking these simplifications into account, reasonable results were obtained for the pressure drop and the heat transfer coefficient (within the 20% range). This shows that Fluidity is capable of doing calculations on such geometries and generate acceptable results. Still, the simplifications used in this research, makes the obtained results unusable for real reactor core calculations. This means that, in order to be able to use Fluidity for this kind of calculations, which require a more complete physical model without the simplifications used here, the code needs some additional features.

6.3 Recommendations

Some missing features of the current Fluidity code can be implemented to improve the results of this simulation, including:

- A dynamic Smagorinsky model: such models have already been developed, and are proven to improve the obtained results.

- A new way for handling heated or cooled objects: now only Neumann and Dirichlet boundary conditions can be applied to such objects. The addition of customizable heating and cooling options, such as a constant heat production option, could make the software more applicable to more problems.

- The equation of state options in Fluidity are not capable of dealing with more complex state equations, such as temperature dependency of the density and diffusivity. Coupling these equations could improve results.

- Periodic boundary conditions: Instead of a no normal flow velocity and zero flux temperature conditions at the side boundaries, the boundary values at each side could coupled to the other side, by using periodic boundary conditions. Using this approach could deliver more accurate results.

- The ability to create meshes with pebbles touching eachother will make simulations with higher packing fractions possible.

Another issue of the used CFD code is the fact that there is interaction between the fissile process inside the pebbles and the temperature of the coolant and the pebbles. In order to make the calculations on heat production and heat transfer more accurate, the energy production process equations, the thermohydraulic calculations inside the pebbles and the coolant flow behavior could be coupled.

Appendix A

Used symbols

A.1 Units

Symbol	definition	dimension
a	Thermal diffusivity	m^2/s
$\boldsymbol{b}$	Body force	N
C_p	Specific heat	J/KgK
C_s	Smagorinsky coefficient	-
d	Diameter	m
D	Tension tensor	-
e	Element number	-
f	Frequency	hz
$\boldsymbol{f}$	Force	N
g	Gravitational acceleration	$\frac{m}{s^2}$
h	Heat transfer coefficient	$\frac{W}{m^2K}$
I	Unity tensor	-
L	Object length	m
M	Mass	kg
M	Metric	-
$\boldsymbol{n}$	The normal	-
N_j	Node number	-
p	Pressure	Pa
$\tilde{p}$	Mean pressure over time	Pa
p	Momentum	kg/m^2s
q	Heat flow	J/m^2
Q	Heat generation per unit volume	J/m^3s
s	Source term	-
t	Time	s
T	Temperature	K
W	Weight function	-

Symbol	definition	dimension
T	Newtonian Stress tensor	-
v	Velocity	m/s
$\boldsymbol{v}$	Velocity (vector form)	m/s
$\bar{v}$	Mean velocity over area	m/s
$\tilde{v}$	Mean velocity over time	m/s
x, y, z	Cartesian coordinates	m
α	Kolmogorov constant	-
γ	Packing fraction	-
Γ	Domain boundary	-
ϵ	Interpolation errorbound	-
λ	Thermal conductivity	W/mK
μ	Dynamic viscosity	$\mathrm{Pa} \cdot \mathrm{s}$
ν	Kinematic viscosity	m^2/s
ρ	Density	kg/m^3
ρ_0	Bulk density	kg/m^3
τ	Shear stress	N/m^2
ϕ	Shape function	-
ϕ_Q	Heat flux	J/s
Ω	The domain	-

A.2 Dimensionless numbers

Symbol	Name	Definition
CFL	Courant number	$\boldsymbol{u}\frac{\Delta t}{l}$
Ma	Mach	$\frac{v}{v_{sound}}$
Nu	Nusselt	$\frac{hd}{\lambda}$
Pr	Prandtl	$\frac{\nu}{a}$
Re	Reynolds	$\frac{\rho v d}{\mu} = \frac{vd}{\nu}$
St	Strouhal	$\frac{fd}{\bar{v}}$

Bibliography

[1] Van den Akker, Mudde: *Fysische transportverschijnselen*, VSSD, derde druk (2008)

[2] Ferziger, Peric: *Computational methods for Fluid Dynamics*, Springer, 3rd edition (2002)

[3] Rannacher: *Finite Element Methods for the Incompressible Navier-Stokes Equations*, college-verslag, Institute of Applied Mathematics University of Heidelberg (1999)

[4] Applied Modelling and Computation Group: *Fluidity Manual*, Imperial College (2010)

[5] Leontini, Thompson, Hourigan: *A numerical study of global frequency selection in the time-mean wake of a circular cylinder*, Journal of Fluid Mechanics vol. 645 pp. 435-446 (2010)

[6] Garcia, Pavlidis e.a. : *A two-phase adaptive finite-element method for solid-fluid coupling in complex geometries*, International journal for numerical methods in fluids, in Press (2010)

[7] Van Kan, Segal, Vermolen: *Numerical Methods in Scientific Computing*, VSSD, first edition (2005)

[8] Whitaker: *Forced convection heat transfer correlations for flow in pipes, past flat plates, single cylinders, single spheres, and for flow in packed beds and tube bundles*, American Institute of Chemical Engineers Journal vol. 18, No.2 pp. 361-371

[9] Courant, Friedrichs, Lewy: *On the Partial Difference Equations of Mathematical Physics*, Mathematischen Annalen 100, pp. 32-74 (1928)

[10] Zienkiewicz, Taylor, Nithiarasu: *Finite Element Method for Fluid Dynamics*, Elsevier, 6th edition (2005)

[11] Pain, Umpleby, de Oliveira, Goddard: *Tetrahedral mesh optimisation and adaptivity for steady-stae and transient finite element calculations*, Computational methods in Applied Mechanical Engineering 190 pp. 3771-3796 (2001)

[12] Mason, Callen: *On the magnitude of subgrid-scale eddy coefficient in large-eddy simulations of turbulent channel flow*, Journal of Fluid Mechanics 162 pp. 439-462 (1986)

[13] Petrila, Titus: *Basics of fluid mechanics and introduction to computational fluid dynamics*, Springer, 2005

[14] Hassan: *Large eddy simulation in pebble bed gas cooled core reactors*, Nuclear Engineering and Design vol. 238 pp. 530-537 (2008)

[15] Geschäftsstelle des Kerntechnischen Ausschusses beim Bundesambt für Strahlenschutz: *Safety standards of the Nuclear Safety Standards Commission (KTA) 3102.1 Reactor Core Design for High-Temperature Gas-Cooled Reactor*, Bundesambt fuï Strahlenschutz (1978)

[16] Moorman: *A safety re-evaluation of the AVR pebble bed reactor operation and its consequences for future HTR concepts*, Berichte des Forschungszentrums Jülich (2008)

[17] Wu, Ferng, Chieng, Liu: *Investigating the advantages and disadvantages of realistic approach and porous approach for closely packed pebbles in CFD simulation.* Nuclear Engineering and design vol. 240 pp. 1151-1159 (2010)

[18] Gao, Shi: *Thermal Hydraulic calculation of the HTR-10 for the initial and equilibrium core* Nuclear Engineering and Design 218 npp. 51-64 (2002)

[19] Saad, Schultz: *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems* Journal of Scientific Computing 7 pp. 856-869 (1986)

[20] Barrett, Berry, Chan, Demmel, Donato, Dongarra, Eijkhout, Pozo, Romine, Van der Vorst: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM (1994)

[21] Boer: *Thermal Hydraulic Reactor Model of the Pebble Bed Modular Reactor in Flownet*, Faculty of Engineering Potchefstroom University for CHE, Africa (2003)

[22] Meyers, Sagaut: *On the model coefficients for the standard and the variational multi-scale Smagorinsky model*, Journal of Fluid Mechanics vol. 569 pp. 287-319 (2006)

[23] Shewchuk: *Lecture Notes on Delaunay Mesh Generation*, Department of Electrical Engineering and Computer Science, University of California at Berkeley (1999)

[24] Lewalle: *Lecture Notes in Incompressible Fluid Dynamics: Phenomenology, Concepts and Analytical Tools*, Syracuse University (2006)

[25] Pepper, Heinrich: *The finite element method: Basic Concepts and Applications*, Taylor and Francis, Second edition (2006)

[26] Donea, Huerta: *Finite element methods for flow problems*, Wiley (2003)

[27] Achenbach: *Heat and flow characteristics of packed beds*, Experimental Thermal and Fluid Science vol. 10, issue 1, pp. 17-27 (1995)

[28] Ergun: *Fluid flow through packed columns* Chemical Engineering Progress 48, pp. 89-94 (1952)

[29] Gnielinski: *Equations for the calculation of heat and mass transfer during flow through stationary spherical packings at moderate and high Peclet numbers* International Chemical Engineering 21, pp 378-383 (1981)