LES Modelling of Turbulent Flow Through an Array of Cylinders Using OpenFOAM

Hanna Isaksson

Engineering Physics and Electrical Engineering, master's level
2019

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering
Preface

This report describes the work done during my master thesis project, which I performed at the division for Fluid Dynamics and Experimental Mechanics at Luleå University of Technology. I wish to thank Staffan Lundström, my examiner, for giving me the opportunity to perform this work on porous media. Thanks to Gunnar Hellström, with whom I had good discussions regarding my simulated output and validation of my simulation setup. Of course also many thanks to my supervisor Anna-Lena Ljung for helping me with everything regarding error search, discussing problem setups and giving feedback.
Abstract

The objective for this master thesis project was to perform simulations for high and low Reynolds number for a porous media modelled as quadratic packed cylinders. The simulations were to be performed by using the open source code OpenFOAM and the turbulence should be modelled by the LES turbulence model in order to resolve details about the turbulence. In Hellström et al. (2009) simulations were performed for a similar setup but for the $k - \Omega$ SST turbulence model. In this project simulations were performed not only for LES but also with the $k - \Omega$ SST model and for the laminar description in order to validate the LES problem setup. For the simulations Reynolds number was calculated based on the porosity and the cylinder diameter and was denoted $Re'$. The simulations were performed for a set of $Re'$ and for each $Re'$ the permeability was calculated based on Darcy’s law. The calculations of the permeability showed, in line with the results from Hellström, that at $Re'$ around 20 there is a drop in the permeability both for the LES and SST simulations. From the permeability simulations it was also seen that for low $Re'$ flows the LES and laminar descriptions agreed with each other, but that the SST simulations yielded lower values of the permeability. According to Hellström the permeability drop was due to onset of inertia effects. When looking at the resulting flow field plots it gave the idea that the amount of turbulence could also have influence on the resulting permeability.
# Contents

1 Introduction .................................................. 1
  1.1 About Porous Media ..................................... 1
  1.2 Previous Work ........................................... 1
  1.3 Purpose .................................................. 1
  1.4 Project limitations ..................................... 1

2 Theory ....................................................... 3
  2.1 Porous media ............................................ 3
  2.2 Fluid Mechanics ......................................... 3
    2.2.1 Governing Equations .............................. 3
  2.3 Computational Fluid Dynamics .......................... 4
    2.3.1 Spatial discretisation ............................ 4
    2.3.2 Interpolation schemes ........................... 5
    2.3.3 Time discretisation .............................. 5
    2.3.4 Pressure algorithm ............................... 5
    2.3.5 Mesh convergence ................................ 6
  2.4 Turbulence .............................................. 6
    2.4.1 RANS Equations .................................. 6
    2.4.2 Turbulence Models ............................... 7
    2.4.3 Shear-Stress Transport Model .................... 7
    2.4.4 Large Eddy Simulation ........................... 8
    2.4.5 Wall functions ................................... 9

3 Method ...................................................... 10
  3.1 Geometry ................................................ 10
    3.1.1 Meshing ........................................... 10
    3.1.2 Mesh study ....................................... 11
    3.1.3 Extrapolation .................................... 11
  3.2 Problem setup .......................................... 11
    3.2.1 Boundary conditions ............................ 11
    3.2.2 Solver ........................................... 11
    3.2.3 Turbulence models .............................. 12
    3.2.4 Discretisation .................................. 12
    3.2.5 Solution control ............................... 12
    3.2.6 Input/output control ............................. 12
    3.2.7 Running the simulation ........................... 12
  3.3 Post processing ........................................ 12
    3.3.1 Calculating variables ............................ 12
    3.3.2 Turbulence resolution ........................... 13
  3.4 Limitations and errors ................................ 13

4 Results ..................................................... 14
  4.1 Periodicity ............................................. 14
  4.2 Mesh study ............................................. 14
    4.2.1 Extrapolation ................................... 15
  4.3 Boundary layer resolution ............................ 15
  4.4 Turbulence resolution ................................ 16
  4.5 Permeability as function of Reynolds number ........ 16
  4.6 Flow at low Re' ....................................... 17
  4.7 Flow at high Re' ..................................... 19

5 Discussion .................................................. 22
  5.1 Geometry and mesh quality ............................ 22
  5.2 Comments on the mesh study .......................... 22
  5.3 Boundary layer resolution ............................ 22
  5.4 Turbulence resolution ................................ 22
  5.5 Permeability ........................................... 23
  5.6 Flow field .............................................. 23
  5.7 Discretisation and solution control .................. 24
  5.8 Error estimation ...................................... 24
  5.9 Conclusions ............................................ 25
  5.10 Further work ........................................... 25

6 APPENDIX .................................................... 26
  6.1 0 directory ............................................. 26
  6.2 constant directory .................................... 29
  6.3 system directory ..................................... 31
  6.4 myPisoFoam ............................................. 36
  6.5 Calculating variables, Python ......................... 37
1 Introduction

1.1 About Porous Media

Porous media are common materials that appear in a number of research areas. Therefore flow through porous media is a subject widely used in engineering and science e.g. ground water hydrology and reservoir engineering. Examples of porous media are soil, porous rocks, filter papers and a loaf of bread [1].

1.2 Previous Work

There has been a number of studies done for flow through porous media. From Seguin et al. (1998) the flow behaviour was experimentally examined for turbulent flows in a porous media by local measurements in the flow with probes. It was found that due to gradual transition to turbulence, laminar and turbulent flow might coexist. It was also discovered that the fluctuating velocity field stabilised at high Reynolds numbers. Stabilised velocity field at a probe corresponded to locally turbulent flow [2]. In Ghaddar (1995) numerical computations were performed for flow over arrays of cylinders with different arrangements including both random arrangement and regular arrangement covering hexagonal and quadratically packing. For the regular arranged cylinders it was discovered that the flows became unsteady at Reynolds number, Re’ [3], around or greater then 150 [4].

In Hellström et al. (2009) numerical studies were done for the array of quadratically packed cylinders in 2D. The permeability of the array was computed for a set of Reynolds number and so the behaviour of the permeability was examined at the onset of inertia effects and turbulence. This was visualised by plotting the permeability, K, as function of Re [5]. Reynolds number can be defined in different ways depending on the choice of characteristic length and fluid velocity. Reynolds number Re’ is defined by

\[ \text{Re'} = \frac{\rho D_p Q}{\mu} \frac{1}{1 - \phi} \]  

where \( \rho \) is the fluid density, \( D_p \) is the cylinder diameter, \( Q \) is the volumetric flux through the area \( A \), \( \mu \) is the dynamic viscosity and \( \phi \) is the porosity. The simulations showed that the permeability started to decrease at Re’ \( \approx 10 \). That corresponded to where the inertia effects set in. Simulations were run for laminar and turbulent flows, where the turbulent simulations were based on the k-\( \Omega \) SST model. The two models differed for high Re’, where the simulations from the turbulent case resulted in a larger drop in permeability for increasing Re’. The result is shown in fig. 1, where the the plot was reconstructed from the simulated data and the permeability was scaled by the true permeability for the laminar description. The true permeability was the permeability for creeping flows that follow Darcy’s law. For low Re both laminar and turbulent calculations agreed up to Re’ \( \approx 100 \), but for higher Re’ the laminar simulations differed from the turbulent formulation [5].

![Figure 1: Permeability plot from Hellström et al. (2009).](image)

1.3 Purpose

According to the results in Hellström et al. (2009) a turbulent description was needed for values of Re’ above 100 to capture the behaviour of the flow. Hellström stated that it was likely for large-scale eddies to be generated at high Re’, which might have affected the flow field. In this project LES simulations were done for the quadratically packed 2D cylinders in order investigate the influence and behaviour of large-scale eddies in the flow and to examine their affect on the permeability at high Re’. The simulations were carried out with the open source code OpenFOAM, and because of that it was also interesting to recreate the results for the cases with laminar and turbulent k-\( \Omega \) SST formulation for comparison between the turbulence models.

1.4 Project limitations

The simulations were performed on a 2D geometry, which means that no wall or surface effects were taken into account. In other words, both the thickness and the width of the porous media were omitted. The porous media was modelled by the same setup of quadratically packed cylinders as in Hellström et al. (2009). By these assumptions
about the geometry it was possible to perform the simulations for a cell by using periodic and symmetric conditions. The turbulence phenomena was modelled with turbulence models, which introduced approximations to the solutions.
2 Theory

2.1 Porous media

A porous media is a multiphase matter with at least two phases with one solid phase and one fluid phase. The parts in the porous media that are not occupied by the solid are called void space or pore space. Due to the void space, fluids are able to pass through the porous media. A porous media can be described by the term porosity, \( \phi \), which is a measure of how much void space there is in the medium. The porosity is expressed as

\[
\phi = \frac{U_v}{U_b},
\]

where \( \phi \) is the ratio of the volume of void space, \( U_v \), to the bulk volume, \( U_b \). Low \( Re \) flows through porous media can be described by Darcy’s law, which was discovered when letting water flow through sand filters. For a horizontal flow Darcy’s law can be expressed as

\[
K \frac{\Delta p}{L} = \frac{Q}{A}
\]

where \( K \) is the permeability, \( \mu \) is the dynamic viscosity, \( \Delta p \) is the pressure drop over a length \( L \) in the stream wise direction and \( Q \) is the flow rate through an area \( A \) [1].

2.2 Fluid Mechanics

Fluid mechanics is the branch of science covering moving and stationary fluids, which continuously deform when exposed for external shear stresses. It can also be said that a fluid does not have a preferred shape. In most cases it is of interest to obtain information of the macroscopic behaviour of the fluid by replacing the discrete microscopic molecules by a continuous distribution. This way of describing a fluid is called a continuum hypothesis [6].

2.2.1 Governing Equations

The governing principles for fluid mechanics are based on three conservation laws namely the conservation of mass, momentum and energy. In the following the derivations of these equations will be briefly described. For the derivations a material volume, \( V(t) \), is defined, which is occupied by mass elements. Because mass can neither be created nor destroyed, \( V(t) \) will contain a constant number of mass elements as it moves and deforms within the fluid. This gives that the surface \( A(t) \) of \( V(t) \) moves at the local fluid velocity \( u \). For the derivations two mathematical relations should also be presented, which are frequently used during the derivations. These are the Reynolds transport theorem for a control volume \( V'(t) \) with surface \( A'(t) \)

\[
\frac{d}{dt} \int_{V'(t)} F(x,t) dV = \int_{V'(t)} \frac{\partial F(x,t)}{\partial t} dV + \int_{A'(t)} F(x,t) b \cdot n dA,
\]

and Gauss’ divergence theorem

\[
\int \int \int_V \nabla \cdot Q dV = \int \int_A n \cdot Q dA.
\]

In eqs. (4) and (5) is \( n \) the outward pointing normal from \( A'(t) \), \( b \) is the local velocity of \( A'(t) \) and \( Q \) is the volumetric flux. Reynolds transport theorem is used to take care of the time derivatives of integrals and is implemented in the equations by letting the material volume and control volume instantaneously be equal. Gauss divergence theorem can be used to put terms in tensor form [6].

Conservation of Mass The conservation of mass for a material volume can be stated as

\[
\frac{d}{dt} \int_{V(t)} \rho(x,t) dV = 0.
\]

By rewriting eq. (6) using Reynolds transport theorem followed by Gauss’ divergence theorem it can be written as

\[
\frac{D \rho}{Dt} = \frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i} (\rho u_i) = 0.
\]

This equation is often referred to as the continuity equation. If the fluid is incompressible, which means that the density is constant in the whole fluid, the first term will equal zero and the equation can be written as

\[
\nabla \cdot u = 0 [6].
\]

Conservation of Momentum The conservation of momentum for a fluid is developed from Newton’s second law according to

\[
\frac{d}{dt} \int_{V(t)} \rho(x,t) u(x,t) dV = \int_{V(t)} \rho(x,t) g dV + \int_{A(t)} f(n,x,t) dA,
\]

where \( \rho u \) is the momentum per unit volume, \( g \) is the body force per unit volume, \( f \) is the surface force per unit area and \( n \) is the outward normal on \( A(t) \). By using Reynolds transport theorem on the left hand side and then Gauss’ divergence theorem eq. (9) can then with index notation be rewritten as

\[
\frac{\partial}{\partial t} (\rho u_i) + \frac{\partial}{\partial x_i} (\rho u_i u_j) = \rho g_j + \frac{\partial}{\partial x_i} (T_{ij}).
\]
This is called the Cauchy’s equation, where \( \rho g_j \) is the net body force and \( \frac{\partial T_{ij}}{\partial x_i} \) is the net surface force. The quantity \( T_{ij} \) is the stress tensor and can be written as \( T_{ij} = -\rho \delta_{ij} + \tau_{ij} \), where \( \rho \) is the thermodynamic pressure and \( \tau_{ij} \) is the deviatoric stress tensor. The term \( \tau_{ij} \) consists of one symmetric part \( S_{ij} \), strain rate tensor, and one anti-symmetric part \( R_{ij} \), rotation tensor. In the momentum equation only the strain rate tensor is taken into account since it contributes to fluid element deformation. The stress tensor can be expressed with \( S_{ij} \) according to

\[
T_{ij} = -\rho \delta_{ij} + 2\mu S_{ij} - \frac{1}{3} \Sigma_{mm} \delta_{ij} + \mu \Sigma_{mm} \delta_{ij}.
\]

The strain rate tensor for a Newtonian fluid ([7]) can be expressed as \( S_{ij} = \frac{1}{2} \Sigma_{ii} - \frac{D u}{D t} \). By using eq. (11) with the expression for \( S_{ij} \) in eq. (10) and by doing some simplifications as assuming small temperature fluctuations and incompressible fluid, Cauchy’s equation can then be written on the form

\[
\frac{D u}{D t} = -\nabla p + \mu \nabla^2 u,
\]

where the left hand side is rewritten by using the expression for the derivative in eq. (7). This is the Navier-Stokes momentum equation for an incompressible fluid [6].

**Conservation of Energy**

Conservation of internal energy is defined in the first law of thermodynamics and its general form is according to

\[
\delta U = \delta W - \delta Q.
\]

where \( \delta U \) the infinitesimal change of internal energy, which depends on external forces \( \delta W \) acting on the body and the heat \( \delta Q \) leaving it [8]. When eq. (13) is applied on a material volume \( V(t) \) with surface area \( A(t) \) it can be written as

\[
\frac{d}{dt} \int_{V(t)} \rho \left( \epsilon + \frac{1}{2} |\dot{u}|^2 \right) dV = \int_{V(t)} \rho g \cdot \dot{u} dV + \int_{A(t)} f \cdot \dot{u} dA - \int_{A(t)} q \cdot \dot{n} dA,
\]

where \( \epsilon \) is the internal energy per unit mass and \( \frac{1}{2} |\dot{u}|^2 \) is the kinetic energy per unit mass. In this case the left hand side corresponds to the conservation of internal and kinetic energy. On the right hand side the two first terms are the work done by body forces and surface forces respectively, which is \( \delta W \) in eq. (13). The last term on the right hand side is the heat transferred out of \( V(t) \) and \( q \) is the heat flux vector. By using Reynolds transport theorem and Gauss’ divergence theorem on eq. (14) all integrals will be volume integrals and so it can be expressed with index notation instead. If the stress tensor is divided into terms of thermodynamic pressure and deviatoric stress the relation for mechanical and heat transferred out of \( V(t) \) and \( q \) is the heat flux vector. By using Reynolds transport theorem and Gauss’ divergence theorem on eq. (14) all integrals will be volume integrals and so it can be expressed with index notation instead. If the stress tensor is divided into terms of thermodynamic pressure and deviatoric stress the relation for mechanical and thermal energy can be written as

\[
\frac{D e}{D t} = \rho \frac{D v}{D t} \left( \epsilon + \frac{1}{2} |\dot{u}|^2 \right) + \left( \frac{\partial \delta_{ij}}{\partial \dot{x}_j} \frac{\partial \delta_{ij}}{\partial \dot{x}_i} \right) + \frac{\partial \delta q_i}{\partial \dot{x}_i}
\]

which can be divided into a pure mechanical part. If the mechanical contributions are subtracted from eq. (15) together with \( 1/\rho = v, \tau_{ij}(\partial u_i/\partial x_j) = \tau_{ij} \delta_{ij} \) and the continuity equation the expression for the internal energy can be written as

\[
\frac{D e}{D t} = -p \frac{D v}{D t} + \frac{1}{\rho} \tau_{ij} \delta_{ij} - \frac{\partial \delta q_i}{\partial \dot{x}_i}
\]

which is equivalent to the first law of thermodynamics - the change of energy of a system equals the work put into the system minus the heat lost by the system [6].

### 2.3 Computational Fluid Dynamics

In fluid dynamics it is of interest to obtain solutions to the governing fluid equations in order to get information about the flows behaviour. For many flows the fluid equations are difficult or even impossible to solve analytically due to complex geometries and turbulent flows. Since the fluid equations are based on partial differential equations, one way to get an approximate solution at some discrete point in the domain is to solve the equations numerically. The field that covers these methods are called computational fluid dynamics, CFD, [7].

#### 2.3.1 Spatial discretisation

In order to get a numerical solution a proper mathematical model has to be chosen. This can be a set of partial differential equations i.e. the conservation equations, together with simplifications such as incompressible flow. The models have to be discretised and one method for this is the finite volume method (FVM), which uses integral form of the conservation equations. In the FVM the geometrical domain divided into control volumes by a grid and a computational node is introduced at the centre of each cell. The conservation equations are satisfied at each cell centre. By Gauss theorem the integrals in the conservation equations can be rewritten as surface integrals, which are approximated as a sum of the integrals over each of the cells faces according to

\[
\int_S f dS = \sum_i \int_{S_i} f dS_i
\]

where \( f \) is the component of the flux vector normal to the cell face [7]. This kind of finite volume discretisation method is called Gaussian integration and since it uses the values on the cell faces, these values has to be interpolated from the cell centre [9].
2.3.2 Interpolation schemes

The interpolation of the integrand \( f \) from eq. (17) can be done according to different schemes for approximation. In upwind interpolation the interpolated value is given the same value as in the cell centre if the flow direction is out from the cell at the given face, and it is given the value from the neighbour cell centre if the flow direction is into the cell. If \( E \) is the current cell, \( P \) is a neighbour cell and \( e \) is the current cell face, then the approximation of the flux through that face can be defined as

\[
\phi_e = \begin{cases} 
\phi_P & \text{if } \mathbf{v} \cdot \mathbf{n} > 0 \\
\phi_E & \text{if } \mathbf{v} \cdot \mathbf{n} > 0 
\end{cases}
\]

(18)

where \( \mathbf{v} \) is the velocity field and \( \mathbf{n} \) is the face normal vector. This is a straight forward method which will not introduce oscillations, but it is numerically diffusive. The upwind scheme is a first order scheme since it retains only the first term in a Taylor approximation. Another interpolation method is the linear interpolation and, by using the same notation as for the upwind scheme, it uses the formulation

\[
\phi_e = \phi_E \lambda_e + \phi_P (1 - \lambda_e),
\]

(19)

where \( \lambda_e \) is a weighting factor depending on the distance between the cell centres and the position of the cell face. It can be defined by

\[
\lambda_e = \frac{x_e - x_P}{x_E - x_P},
\]

(20)

where \( x \) denoted the point at which the cell centre is located. The linear method is of second order but it may introduce oscillations to the solution [7]. In OpenFOAM there is also an interpolation scheme called linearUpwind. This is also second order and it uses the cell centre value from the current or the neighbouring cell according to the upwind method in 18, but it also uses the gradient of the flow to get the face value [9].

2.3.3 Time discretisation

For transient flows the time coordinate also has to be discretised. There are some different methods such as Euler forward and Euler backward, which are both of first order. Euler forward is an explicit method which is simple to implement and is cheap to run, but it is unstable for larger time steps. Euler backward is an implicit method and is more complex since it has to solve a large number of equations at each time step. An advantage with Euler backward is that it generates stable solutions. The dimensionless parameter Courant number defined as

\[
Co = \frac{u \Delta t}{\Delta x}
\]

(21)

is introduced during time discretisation, where \( u \) is the velocity, \( \Delta t \) is the time step and \( \Delta x \) is the typical grid size. The Courant number has to be smaller than one since a fluid particle should move to a maximum of one grid length in one time step.

2.3.4 Pressure algorithm

In the conservation equations there is no independent equation for the pressure. There are momentum equations for respective velocity component and the continuity equation does not contain the pressure. One idea for how to compute the pressure field is to construct pressure fields that satisfy the continuity equation. From the continuity equation in eq. (7) comes the idea of taking the divergence of the momentum equation, which gives Poisson’s equation for the pressure. By solving the pressure Poisson equation it gives a velocity field that fulfills the continuity equation which means it will be divergence free.

One method for computing the pressure field is an implicit pressure-correction method, called the PISO algorithm. This consists of one outer loop for the time step and one inner loop for the pressure correction, whose purpose is to find a pressure field that yields a velocity field that fulfills continuity. The inner loop starts with a start guess of the pressure, \( p^* \), from preceding time step, that is, from preceding outer iteration. Because the pressure was obtained from the former outer iteration the computed velocity field \( u_* \) in the inner loop does usually not satisfy the continuity equation. To enforce continuity at each time step the velocities have to be corrected, which is done by calculating the velocity field again from the pressure Poisson equation but with a modified pressure field

\[
p** = p^* + p',
\]

(22)

where \( p' \) is the pressure correction. The procedure is to first solve the momentum equation for the initial pressure \( p^* \). Then the pressure-correction equation is solved for \( p' \), which roughly is solving the momentum equation for \( p' \). This yields the velocity-correction \( u'_* \), which gives the resulting velocity field

\[
u_{**} = u_* + u'_*,
\]

(23)

The corrected velocities yielded from the corrected pressure will satisfy the continuity equation, but maybe not the momentum equation. In order to also satisfy the momentum equation another turn in the inner loop has to be done, which means inner iterations are performed until the corrections are small enough. In the PISO algorithm a second pressure-correction step is performed, which then gives

\[
p*** = p** + p',
\]

(24)

where \( p' \) is the new pressure-correction and \( p** \) is the pressure that will be initial guess in next iteration. One method for improving the convergence is to introduce an under-relaxation parameter \( \alpha \) according to
\[ p \ast \ast = p \ast + \alpha p', \quad (25) \]

where \( 0 \leq \alpha \leq 1 \). This means that only a part of the pressure correction is added and this should assure the convergence [7].

### 2.3.5 Mesh convergence

The numerical solution has to be convergent and a fine grid will for a convergent method give a solution that approaches the exact solution. The numerical iteration should continue until a certain convergence criterion is fulfilled. It is important to have a grid-independent solution so that the result does not depend on the fineness of the grid [7]. In order to validate how close the solution is to a mesh independent solution an extrapolation can be performed. One method for extrapolating values is the Richardson extrapolation described in [10] and [11]. As a first step the representative grid sizes are defined by \( h_1 < h_2 < h_3 \) together with the grid refinements factors \( r_1 = h_2/h_1 \) and \( r_2 = h_1/h_2 \). The apparent order of accuracy can be determined by the fixed point iteration

\[ p = \frac{\ln|e_2/e_1 + q(p)|}{\ln(r_1)}, \quad (26) \]

\[ q(p) = \ln \left( \frac{r_1^p - s}{r_2^p - s} \right), \quad (27) \]

and

\[ s = \text{sign} \left( \frac{e_2}{e_1} \right), \quad (28) \]

where \( e = \phi_{x1} - \phi \) for the flow parameter \( \phi \) that is to be extrapolated. The extrapolation can be considered to work if the calculated apparent order of accuracy agrees with the formal order of accuracy. The extrapolated value can be calculated from

\[ \phi_{ext1} = r_1^p \phi_1 - \phi_2 \quad r_1^p - 1 \quad (29) \]

and the approximated error according to

\[ e_{a1} = \frac{\phi_1 - \phi_2}{\phi_1}. \quad (30) \]

To evaluate how far from the extrapolated value the simulated values are, the extrapolated error can be calculated according to

\[ e_{ext1} = \frac{\phi_{ext1} - \phi_1}{\phi_{ext1}}. \quad (31) \]

### 2.4 Turbulence

The behaviour of the flow, laminar or turbulent, can be characterised by the non-dimensional parameter called Reynolds number defined by

\[ Re = \frac{UL}{v}, \quad (32) \]

where \( U \) and \( L \) are characteristic velocity and length scales of the flow, and \( v \) is the kinematic viscosity. When for an example the fluid velocity increases, \( Re \) does also increase and the flow goes to turbulent flow [12]. There are some different ways to define \( Re \), which is done by making different choices of the characteristic parameters. The definition of \( Re \) may for an example give different results for critical velocities for onset of turbulence and inertia effects. One way to define \( Re \) for flow in porous media is \( Re' \), which was shown in eq. (1) in the introduction [5].

Turbulent motion range in size from the width of the flow to smaller scales, and the scales becomes smaller at higher \( Re \). From this it can be said that turbulence consists of eddies of different sizes. According to Richardson’s idea about energy cascade, large eddies are unstable and break up into smaller eddies. Energy is transferred from the large eddies to the smaller and this process continues until \( Re \) for the eddies are sufficiently small so that the eddy motion is stable. At this smallest scale the energy is dissipated by viscous action [12].

#### 2.4.1 RANS Equations

Equations of motions for turbulent flows can be obtained by starting from Navier-Stokes equation in eq. (12). For turbulent flows the velocity \( u(x, t) \) can be decomposed into a mean \( \langle U(x, t) \rangle \) part and a fluctuating part \( \tilde{u}(x, t) \). This can be written as

\[ u(x, t) = \langle U(x, t) \rangle + \tilde{u}(x, t) \quad (33) \]

and this is called Reynolds decomposition. Reynolds decomposition follows from the continuity equation for incompressible flows, eq. (8), and it can be written as

\[ \nabla \cdot u = \nabla \cdot (\langle U \rangle + \tilde{u}) = \nabla \cdot (\langle U \rangle) + \nabla \cdot \tilde{u} = 0. \quad (34) \]
Since the divergence of $\mathbf{u}$ is zero from continuity and the mean part of the velocity field yields zero, then also the divergence of the fluctuating part is zero. In order to obtain momentum equations the mean, or time-average, of Navier-Stokes equation in eq. (12) is taken. First the mean is calculated for the derivative on left side of Navier-Stokes equation, and by using Reynolds decomposition the derivative can be expressed as

$$\frac{D \langle u_i \rangle}{Dt} = \frac{\partial \langle u_i \rangle}{\partial t} + \langle u_i \rangle \frac{\partial \langle \mathbf{u} \cdot \mathbf{u} \rangle}{\partial \mathbf{x}_j} + \frac{\partial}{\partial \mathbf{x}_j} \langle \mathbf{u} \cdot \mathbf{u} \rangle = \frac{D \langle u_i \rangle}{Dt} + \frac{\partial}{\partial \mathbf{x}_j} \langle \mathbf{u} \cdot \mathbf{u} \rangle.$$

(35)

This expression can now be used when taking the mean of the whole Navier-Stokes equation and the resulting expression is

$$\frac{D \langle u_i \rangle}{Dt} = -\frac{1}{\rho} \frac{\partial (\rho)}{\partial \mathbf{x}_j} + \nu \nabla^2 \langle u_i \rangle - \frac{\partial}{\partial \mathbf{x}_j} \langle \mathbf{u} \cdot \mathbf{u} \rangle.$$

(36)

and is called the Reynolds-averaged Navier-Stokes (RANS) equation. The quantity $\langle \mathbf{u} \cdot \mathbf{u} \rangle$ is called the Reynolds stresses and it is this term that forms the difference between Navier-Stokes and RANS equations. The Reynolds stresses together with the density, $\rho \langle \mathbf{u} \cdot \mathbf{u} \rangle$, corresponds to apparent stress from the fluctuating velocity field. From above it is given that turbulent flows can be described by the mean continuity equation, eq. (34), and the RANS, eq. (36), which in three dimensions gives four equations. Except from the four quantities ($\rho$) and $\langle \mathbf{u} \rangle$, there are also the Reynolds stresses, which gives an underdetermined system of equations. This means that the RANS equations cannot be solved unless the Reynolds stresses can be determined and therefore this method needs turbulence models to approximate the Reynolds stresses and close the equation system [12].

### 2.4.2 Turbulence Models

Since turbulent flows are good at mixing fluids and due to the viscous action from the eddies, it can be assumed that the effect of turbulence can be represented as increased viscosity. This leads to the hypothesis about the eddy-viscosity model for Reynolds stresses. The eddy-viscosity model needs to be described by two equations. Common two-equation eddy-viscosity models are the $k-\epsilon$ model and the $k-\Omega$ model [7]. The quantity $k$ is used to represent the eddy velocity scale and $\epsilon$ and $\Omega$ are used for representing the eddy length scales [13]. The Reynolds stresses can be determined according to

$$-\rho \langle \mathbf{u} \cdot \mathbf{u} \rangle = \mu_t \left[ \frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial \langle u_j \rangle}{\partial x_i} \right] - \frac{2}{3} \Omega \mu_t, k,$$

(37)

where the turbulent kinetic energy $k$ is expressed as

$$k = \frac{1}{2} \langle \mathbf{u} \cdot \mathbf{u} \rangle,$$

(38)

and $\mu_t$ is the eddy viscosity. In the case with the $k-\epsilon$ model the eddy viscosity is expressed as

$$\mu_t = \rho C_{\mu} \frac{k}{\epsilon},$$

(39)

where $C_{\mu}$ is a dimensionless constant and the eddy dissipation rate $\epsilon$ is given by

$$\epsilon = k^{3/2} / L,$$

(40)

where $L$ is the eddy length scale. The second common two-equation model is the $k-\Omega$ model, which uses the inverse time scale $\Omega$ as the second variable and it is expressed as

$$\Omega = \rho \frac{k}{\mu_t} [7].$$

(41)

These turbulence models uses wall functions to describe the flow at the boundary close to walls [13].

### 2.4.3 Shear-Stress Transport Model

The two models $k-\epsilon$ and $k-\Omega$ have different advantages close to and far away from walls. This gives the idea of combining these two models to get the best out of both. This is called the shear-stress transport (SST) model because it takes care of the transport of the principal turbulent shear stress. In the sublayer and the logarithmic region the $k-\Omega$ model is used and in the wake and free shear layer outside the inner region the $k-\epsilon$ model is used. The $k-\epsilon$ model is transformed into a $k-\Omega$ formulation and then a blending function, $F_\Omega$, is introduced to handle the transition between the two models. It is applies according to

$$\phi = F_\Omega \phi_1 + (1 - F_\Omega) \phi_2,$$

(42)

where $\phi_1$ represents the $k-\Omega$ model and $\phi_2$ the transformed $k-\epsilon$ model. The function $F_\Omega$ is designed to be 1 in the sublayer and logarithmic region so that only $k-\Omega$ is active close to the wall, and then successively go to zero in the wake region. After having transformed $k-\epsilon$ and multiplied the equations with the blending function the equation for $k$ looks like

$$\frac{D \rho k}{Dt} = \gamma_{ij} \frac{\partial u_i}{\partial x_j} \beta^* \rho \omega^2 \beta^* \frac{\partial \rho}{\partial x_j} \left( \mu + \sigma_{\omega} \mu_t \frac{\partial k}{\partial x_j} \right),$$

(43)

and the equation for $\Omega$ is written as

$$\frac{D \rho \omega}{Dt} = \frac{\gamma_{ij}}{\nu_t} \frac{\partial u_i}{\partial x_j} \beta^* \rho \omega^2 \beta^* \frac{\partial \rho}{\partial x_j} \left( \mu + \sigma_{\omega} \mu_t \frac{\partial \omega}{\partial x_j} \right) + 2 \sigma (1 - F_\Omega) \sigma_{\omega}^2 \int \frac{\partial \rho}{\partial x_j} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j} [14].$$

(44)
2.4.4 Large Eddy Simulation

In turbulent flows the motion on larger scales are generally more energetic than the small ones. This means that the large eddies are the most effective transporters of conserved quantities and thus have more influence on the flows behaviour. From this argumentation it may be reasonable to use a simulation method that resolves the larger eddies [7]. Such a method is called large eddy simulation, LES, which in computational expense lies between DNS and the Reynolds-stress models [12]. This method uses a low pass filter, which by filtering the velocity field from eddies with size smaller than a length scale \( \Delta \), gives a velocity field where the large scales are resolved. The filtered velocity is defined as in

\[
\langle u_i(x) \rangle = \int_G G(x, x') u_i(x') dx'.
\]

When filtering the Navier-Stokes equation for incompressible flow the set of equations obtained looks like

\[
\frac{\partial (\rho \langle u_i \rangle)}{\partial t} + \frac{\partial (\rho \langle u_i u_j \rangle)}{\partial x_j} = - \frac{\partial \rho}{\partial x_i} + \frac{\partial}{\partial x_j} \left[ \mu \left( \frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial \langle u_j \rangle}{\partial x_i} \right) \right].
\]

Due to linearity of the continuity equation, it is not changed by filtering. In the filtered Navies-Stokes equation in eq. (46) the quantity \( \langle u_i u_j \rangle \) appears. This quantity is not easy to compute so it is approximated by the model

\[
\tau_{ij} = -\rho (\langle u_i u_j \rangle - \langle u_i \rangle \langle u_j \rangle),
\]

where \( \tau_{ij} \) is called the subgrid-scale Reynolds stresses or residual-stress tensor, though it is not a physical stress. The residual-stresses have to be modelled and the earliest and most common model is the Smagorinsky model. The Smagorinsky model is an eddy viscosity model and such models assume that the effects from the residual stresses are increased transport and dissipation. A model for this could be written as

\[
\tau_{ij} = S_{ij} - \frac{1}{2} \rho S_{kk} \delta_{ij},
\]

where \( S_{ij} \) is the residual-stress tensor from eq. (47), \( \mu_t \) is the eddy viscosity and \( \langle S_{ij} \rangle \) is the strain rate of the resolved field (the large scales). The eddy viscosity can be written as

\[
\mu_t = C_S S_{kk}^{1/2},
\]

where \( C_S \) is a dimensionless model parameter called Smagorinsky coefficient and \( \Delta \) is the filter length scale [7]. According to [15] the eddy viscosity can be rewritten in terms of the residual turbulent kinetic energy as

\[
\mu_t = C_S \rho \Delta \sqrt{k_{res}},
\]

The Smagorinsky coefficient is dependent of Re according to [7], and proportional to the filter width \( \Delta \) [12]. The coefficient can be modelled by using van Driest damping according to

\[
C_S = C_{pol} \left( 1 - e^{-y^{+} / A^{+}} \right)^2
\]

where \( y^{+} \) is the dimensionless wall distance and \( A^{+} \approx 25 \) is a constant [7]. The van Driest damping is a function that handles the grid size, which is connected to the eddy length scale, and it matches the mixing lengths for the log law in the buffer region [12].

**Turbulence resolution** In order to be able to resolve the residual-stresses a fine mesh size, \( h \), is required since the condition \( \Delta > h \) has to be fulfilled [7]. Another requirement on the grid is that it has to be fine enough to be able to resolve about 80 % of the kinetic energy. The turbulence resolution can be calculated according to

\[
M(x, t) = \frac{k_{res}(x, t)}{k_{pol}(x, t) + k_{res}(x, t)}
\]

where \( M(x, t) \) lies between 0 and 1 and is the measure of turbulence resolution, \( k_{res}(x, t) \) is the turbulent kinetic energy of the resolved motions and \( k_{pol}(x, t) \) is the kinetic energy of the residual motions corresponding the the residual-stress tensor. The turbulent kinetic energy for the resolved motions can be calculated according to

\[
k_{res}(x, t) = \frac{1}{2} \langle (U - \langle U \rangle) \cdot (U - \langle U \rangle) \rangle
\]

For an 80 % turbulence resolution the turbulence-resolution tolerance is set to \( \epsilon = 0.2 \), which gives

\[
M(x, t) \leq \epsilon.
\]

This relation is maintained by adjusting the grid according to \( \Delta > h \). Also if \( M(x, t) \) equals zero the simulation corresponds to DNS and if \( M(x, t) \) is 1 all turbulent kinetic energy is modelled and so the simulation corresponds to RANS [16].
2.4.5 Wall functions

Turbulent flows close to walls e.g. a cylinder surface are divided into two regions, one outer region corresponding to the free stream flow and one inner region close to the wall. The position of the transition region between the outer and inner regions is defined by a dimensionless distance called $y^+$ defined as

$$y^+ = \frac{u_\tau y}{v}. \quad (55)$$

The variable $u_\tau$ is the shear velocity and the other variables are the physical distance and the dynamic viscosity. The inner region consists of the viscous sublayer for $y^+ < 5$, the buffer layer for $5 < y^+ < 30$ and the log-law region for $30 < y^+ < 50$. The log-law is defined as

$$u^+ = \frac{1}{\kappa} \ln y^+ + B, \quad (56)$$

and relates the dimensionless velocity $u^+$ to $y^+$ [12]. The constant $\kappa$ is von Kármán $\kappa = 0.41$ and $B$ is an empirical constant related to the thickness of the viscous sublayer [7]. The purpose of the wall functions is that they links the shear stress to the instantaneous velocity through logarithmic functions. The log-laws are empirical descriptions of the flow behaviour. They do not fully resolve the whole boundary layer but represent the properties of turbulent boundary layers [13]. In order for a CFD simulation to resolve the boundary layer properly the cell closest to the wall should be within the viscous sub-layer. According to [17] and [13] this corresponds to an $yPlus$ around 1 for both $k - \Omega$ and LES simulations. If wall functions are used for modelling the flow close to a surface the first cell centre should be placed in the log-law region, which corresponds to $30 < y^+ < 50$. 


3 Method

3.1 Geometry

The geometry was defined in OpenFOAM’s `blockMeshDict`, where the porous media was modelled as quadratically packed cylinders. Since wall effects and effects from inlet and outlet were neglected it could be assumed that the flow through the media was periodic in the flow direction and symmetric in the direction perpendicular to the flow. Because of these assumptions about periodicity and symmetry the calculations could be performed for a cell containing a half cylinder, see fig. 2. The fact that wall effects were neglected the simulations could be done in 2D. The porosity was determined to be 40%, which corresponded to a cylinder radius of 2.185 m for a cell with dimensions \(5 \times 2.5\) m. The porosity was calculated according to eq. (2), where \(U_v\) was the area (volume in 3D) of the void space, that is the cell area minus the area of the cylinder, and \(U_b\) was the area of the whole cell.

![Figure 2: Geometry used for the cell.](image)

3.1.1 Meshing

Also the mesh was defined in `blockMeshDict`. When meshing with `blockMesh`, the geometry was divided into hexahedral blocks defined by eight vertices, one for each corner of the hexahedron. The faces of the blocks consisted of straight lines and arcs. The mesh was generated by defining the number of elements along one block face. The number of elements on one block face had to match with the number of elements on the neighbouring block face. The blocking lead to areas with larger elements at the upper corners. In order to get higher resolution at the walls an area along the cylinder wall was defined where the elements were smaller and the resulting mesh is shown in fig. 3. The mesh was built by using the `blockMesh` command and in order to control that the building went out well the command `checkMesh` was run. The mesh was only one element thick in the depth direction. OpenFOAM can interpret this as an infinite extension in that direction if the solution of Navier-Stokes equations is turned off in this direction. Such a treatment generated a 2D solution.

![Figure 3: Mesh.](image)

3.1.2 Mesh study

In order to obtain solutions that were independent of the grid size a mesh convergence analysis was carried out for the \(k-\Omega\) SST turbulence model. For a fine enough mesh the values of the flow parameters will not vary with an increasing number of elements. The convergence study was done by calculating the permeability from eq. (3) and Reynolds number according to eq. (1) for a given mean velocity but for meshes with different elements sizes ranging from 24400 elements to 96250 elements. The mesh study was performed for the mean velocity \(1e-2\) m/s since there were more fluctuations to take care of in this case due to the turbulence.

3.1.3 Extrapolation

To estimate how far the simulated values of the permeabilities were from the real value, a Richardson extrapolation was performed according to the guidelines in [10] and [11] and it was performed in MATLAB R2019a. The extrapolation was performed on three meshes, namely those with 32000, 49800 and 75650 elements. The equations used for the
calculations were presented in the theory part. As representative grid sizes, \( h_1 < h_2 < h_3 \), the mean face areas at the inlet were used. The apparent order of accuracy was obtained by using the fixed point iteration according to eqs. (26) and (28). In the fixed point iteration \( \varepsilon = K_{i+1} - K_i \) was based on the permeability. If there is agreement between the formal and apparent order of accuracy it is an indication if the grid was in the asymptotic range [10].

The extrapolated value of the permeability was calculated according to eq. (29) and the relative and extrapolated errors was calculated according to eq. (30) and eq. (31) respectively.

The resulting values from the mesh study and the extrapolation is shown in the results part in section 4.2. From these results the mesh with 75650 elements was chosen for the simulations. That choice was a compromise between how much numerical accuracy that was actually possible to obtain from the given mesh, how the permeability seemed to have stabilised for smaller elements sizes and to reduce the simulation time.

3.2 Problem setup

The simulations in this project were performed with OpenFOAM v-1812. The directory structure for OpenFOAM is shown in fig. 4. In the following the method for the case setup will be described. For details the files in the cases and the code for the modified solver are shown in the appendix in section 6.

![Figure 4: Directory setup in OpenFOAM [9].](image)

The simulations were performed on the mesh chosen from the convergence study. In order to study the effects on the permeability at increasing flow velocity a parameter study was carried out for a range of different Re, from creeping flow at low Re to turbulent flows at high Re. The flow was driven by a momentum source called \texttt{meanVelocityForce} set in \texttt{fvOptions} and it was defined for the whole cell domain. The momentum source was set by defining a mean velocity, which was the parameter used for the parameter study. A force was calculated and applied to drive the flow, which generated a corresponding pressure gradient. The parameter study was performed for each of the cases \( k – \Omega \) SST turbulence model and LES Smagorinsky, both set in \texttt{turbulenceModels}, which made it possible to compare results from different turbulence models. The kinematic viscosity \( \nu \) was defined in \texttt{transportProperties} and was set to 1e-6 m\(^2\)/s for all cases, which corresponds to water [18]. Also set in the \texttt{transportProperties} was the \texttt{transportModel} chosen to be Newtonian.

3.2.1 Boundary conditions

The boundary conditions were chosen to correspond to the assumptions about symmetry and periodicity. In the flow direction the patches for inlet and outlet of the cell were chosen as \texttt{cyclic}. The \texttt{cyclic} boundary condition meant that the two patches inlet and outlet were interpreted to be in physical contact. For the up and down patches the \texttt{symmetry} condition was used, and at the cylinder wall \texttt{noSlip} condition was used for the velocity field and \texttt{zeroGradient} for the pressure field. The turbulent quantities were modelled with wall functions with fixed values at the wall. The wall functions used were \texttt{kqRWallFunction} for \( k \), \texttt{nutkWallFunction} for \( \nu_t \) and \texttt{omegaWallFunction} for \( \Omega \). The wall functions for \( k \) and \( \Omega \) were chosen to be able to handle high Re and the wall function for \( \nu_t \) was a wall function that is based on the kinematic [19]. The initial values for the wall functions were \( k=0.03 \text{ m}^2/\text{s}^2 \), \( \Omega=300 \text{ Hz} \) and \( \nu_t=0 \text{ m}^2/\text{s} \) according to the setup in [5].

3.2.2 Solver

When choosing a solver in OpenFOAM the characteristics of the flow had to be defined. The flow through the quadratically packed cylinders was assumed to be incompressible and Newtonian. Since the parameter study was done up til high Re, the solver should be able to model turbulence. The solver also had to be able to handle transient flows even if the flow underwent a transition period until it had developed full turbulence. The solver chosen for this problem was the \texttt{pisoFoam} solver. It solved the Navier-Stokes equations by using the PISO algorithm for the pressure, it was a transient solver and it could model turbulence for incompressible flows. For the LES simulations the \texttt{pisoFoam} solver was slightly modified in \texttt{myPisoFoam} to give the turbulent kinetic energy as output. This was done by defining the unresolved turbulent kinetic energy \( k_{sgs} \) as a field function object in \texttt{createFields.H} and then in the \texttt{myPisoFoam.C} file add the line \( k_{sgs} = turbulence-> k() \); before \texttt{runTime.write();} in the runtime loop. For details, see section 6.
3.2.3 Turbulence models

Simulations were run for the turbulence models $k − \Omega$ SST and LES Smagorinsky, but also for a laminar description, which were set in turbulenceProperties. For the LES simulation the Smagorinsky model was used together with the van Driest damping function in eq. (51). The van Driest damping helped the Smagorinsky model to yield an eddy viscosity that was zero at the walls, which was according to the definition of the eddy viscosity since it should be zero in regions with no turbulence.

3.2.4 Discretisation

In the fvSchemes file the schemes for discretisation and interpolation were set. The time was discretised by the Euler backward scheme, which is an implicit, first order discretisation method. The spatial discretisation was done by using Gaussian integration according to eq. (17), which needed interpolation to the cell faces. The interpolation schemes used were the second order linear interpolation according to eq. (19) for the gradient and laplacian terms and linearUpwind for the divergence terms.

3.2.5 Solution control

The file fvSolution handled settings about equation solvers, tolerances and algorithm control. The keyword nCorrectors in the PISO subdirectory specified the number of times for which the pressure was solved in the PISO algorithm. In this work it was set to two times. Relaxation was introduced in the subdirectory relaxationFactors and it was set to 0.5 for all field variables. The same tolerance criterion was used for all variables, it was set to 1e-6 with relative tolerance 0.1 except for when solving the final pressure since then the relative tolerance was set to zero. The meaning of this is that the first time the pressure was solved it was during the condition of 0.1 relative tolerance but the second and last time the relative tolerance should be zero.

3.2.6 Input/output control

Handling input and output control was done in the controlDict file. Here the run time and the time step were defined and also how often data should be written to file. The simulations were run until steady state, which in the field plots was seen as when the field started to fluctuate around a certain state. The choice of time step depended on the convergence and the size of Co. Also the choice of solver had to be stated in controlDict, and it was pisoFoam for the SST and lamian simulations and myPisoFoam when running for LES. In controlDict functions could be added. The functions used here was residuals, vorticity and for the LES simulations also the fieldAverage was used on the and U field for time averaging.

3.2.7 Running the simulation

The simulation time was sped up by running the simulations in parallel. The geometry was decomposed with the scotch method in the decomposeParDict. This method attempted to minimise the number of processor boundaries. In this case the geometry was divided into six cases and it was decomposed without weighting in scotch. The decomposition was performed by the command decomposePar and after the simulations were run the geometry was reconstructed by reconstructPar. The simulations were run with the command mpirun -np 6 <solverName> -parallel > log.pisoFoam, where <solverName> was pisoFoam for SST and laminar, and the modified myPisoFoam for the LES case. The > log.pisoFoam formulation yielded a file where simulation information at each time step was saved.

3.3 Post processing

Residuals from the simulations were extracted by using includeFunc residuals as a function in the controlDict file, which generated a file with the residuals for each time step in the postProcessing directory. Values of the residuals for pressure and the velocity in x and y directions were extracted together with corresponding time step. The value of the Courant number was calculated at each time step and both max and mean values were given. The solver information was saved by using the line pisoFoam > log.pisoFoam in the Allrun script, where the Courant numbers could be viewed. yPlus was calculated by using the command pisoFoam -postProcess -func yPlus, which gave the y+ value at each output time. The computed field variables were plotted by using ParaView 5.4.1.

3.3.1 Calculating variables

Reynolds number was calculated according to eq. (1) and the permeability according to eq. (3), where the pressure gradient in eq. (3) was taken from the momentumSourceProperties generated in each time directory during run time due to the momentum source. Since the pressure in the incompressible solvers was on the kinematic form the pressure gradient had to be multiplied by the fluid density to obtain the dynamic form of the pressure in [Pa]. The volumetric flux could be computed by using the command postProcess -latestTime -func “patchAverage(name=INLET,phi)”. This command calculated the average of the volumetric flux at the inlet patch based on an arithmetic average of the cell faces at the inlet. The variable phi was the volumetric flow though a cell face so in order to get the volumetric flow for the whole inlet patch, phi was multiplied by the number of elements on the inlet patch. For the pressure gradient, time averaging was used over a number of time steps were the solution had reached a steady state. When plotting the permeability K as function of Re, the permeability was normalised by scaling with the permeability corresponding to Darcian flow, that is laminar flow at low Re. The permeability at low Re was called true permeability and the permeability calculated during the parameter sweep was called apparent permeability. Code for the post processing of these variables is found in the appendix.
3.3.2 Turbulence resolution

The turbulence resolution was calculated in the case for LES simulations. This was done based on the mean value of the velocity field, \(U_xMean\) and \(U_yMean\), calculated as field average in controlDict and the unresolved kinetic energy \(k_{sgs}\) from the myPisoFoam solver. The resolved turbulent kinetic energy was calculated according to eq. (53) written as

\[
k_{res} = 0.5 \ast \langle (U_x - U_xMean) \ast (U_x - U_xMean) + (U_y - U_yMean) \ast (U_y - U_yMean) \rangle\]

where the mean values came from the field averaging and the angle brackets corresponded to time average. The time average was calculated for both turbulent kinetic energies before calculating the turbulence resolution. The turbulence resolution was then calculated straight from eq. (52) and then a spatial mean was computed in order to get an approximate overall value. The turbulence tolerance was chosen to be 0.2, which guaranteed that at least 80% of the turbulence was resolved.

3.4 Limitations and errors

When performing numerical calculations it is important to keep in mind that the solutions are not analytically exact but numerical approximations of the governing equations. The numerical modelling consists of the discretisation of the equations, but also use of turbulence models and wall functions have impact on the accuracy of the solution. In the solution schemes the time was discretised of first order. The interpolation for the spatial schemes was for some variables linear, which is of second order but can introduce oscillations to the solution. For the pressure algorithm the number of turns in the inner loop was set to two. The numerical and modelling errors can be reduced by the mesh convergence analysis. Other assumptions about the flow that was done was incompressible and Newtonian flow.

The problem was also limited regarded the geometry. The thickness of the material was completely neglected by assuming 2D geometry so no effects from top or bottom walls was taken into account. The assumptions about periodicity and symmetry meant that effects from in- and outflow was neglected so the flow out from the simulated cell was bounded to be the inflow to a similar following cell. The quadratic packing of the cell is also a simplification since it assumes a porous media that is ordered is a specific way.
4 Results

4.1 Periodicity

In fig. 5a and fig. 5b is the velocity magnitude and \( \nu_t \) plotted respectively along the inlet and the outlet patches in order to prove that the periodicity condition was fulfilled. The values were taken for the case with driving mean velocity 1e-6 m/s (Re' \( \approx \) 0.1457) for the SST simulation. As seen in the figures both the velocity and the turbulent quantity agreed at the inlet and outlet patches, which confirmed that the periodicity condition was fulfilled.

![Figure 5: Plots for U and \( \nu_t \) at inlet and outlet respectively.](image)

4.2 Mesh study

In table 1 are the values of Re' and K at the mean velocity 1e-2 m/s for meshes with increased fineness. In fig. 6a was Re' calculated for the different meshes and plotted against corresponding mean face area at the inlet. It could be seen that the fluctuations of Re' between the different cases were small and the percentage difference between the largest and smallest values was 0.0425 %, which was considered negligible small. The behaviour of the permeability was more depending of the elements size and in fig. 6b it was evaluated for the different meshes and plotted against corresponding mean face area at the inlet. It was seen that the value of the permeability increased for the smaller element sizes, but for the meshes with 235 and 245 elements at the inlet the change in value seemed to flatten out.

<table>
<thead>
<tr>
<th>Re'</th>
<th>K, SST model [m^2]</th>
<th>Number of elements</th>
<th>Elements at inlet</th>
<th>Mean face area at inlet [m^2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1457.10</td>
<td>8.42476e-06</td>
<td>24400</td>
<td>140</td>
<td>0.0003571</td>
</tr>
<tr>
<td>1456.67</td>
<td>8.44621e-06</td>
<td>32000</td>
<td>160</td>
<td>0.0003125</td>
</tr>
<tr>
<td>1456.56</td>
<td>8.68617e-06</td>
<td>43250</td>
<td>185</td>
<td>0.0002703</td>
</tr>
<tr>
<td>1457.18</td>
<td>9.08281e-06</td>
<td>49800</td>
<td>190</td>
<td>0.0002632</td>
</tr>
<tr>
<td>1457.10</td>
<td>9.42134e-06</td>
<td>60050</td>
<td>205</td>
<td>0.0002439</td>
</tr>
<tr>
<td>1457.09</td>
<td>9.41414e-06</td>
<td>66100</td>
<td>215</td>
<td>0.0002326</td>
</tr>
<tr>
<td>1457.01</td>
<td>9.46632e-06</td>
<td>75650</td>
<td>225</td>
<td>0.0002227</td>
</tr>
<tr>
<td>1457.07</td>
<td>9.98657e-06</td>
<td>83590</td>
<td>235</td>
<td>0.0002128</td>
</tr>
<tr>
<td>1457.15</td>
<td>1.00039e-05</td>
<td>96250</td>
<td>245</td>
<td>0.0002040</td>
</tr>
</tbody>
</table>

![Figure 6: Mesh study of Re' and K as function of face area at the inlet patch.](image)
4.2.1 Extrapolation
In the extrapolation of the permeability the grid refinement factors were determined to \( r_1 = h_2/h_1 = 1.1847 \) and \( r_2 = h_3/h_2 = 1.2015 \), which were considered valid since the refinements factors were recommended to be larger than 1 or close to 1.3 according to the given guidelines. The apparent order of accuracy was evaluated to 2.4108, and since the formal order of accuracy was 1 in time and 2 in space the calculated apparent order was taken to be good enough. The extrapolated value was evaluated to \( K_{ext1} = 2.0452 \times 10^{-4} \), which gave an approximate relative error of 4.05 % and an extrapolated error of 6.08 %, which corresponded to the error between the the mesh with 75650 elements and a mesh with infinite small elements. In fig. 7 the extrapolated value plotted together with the points used for the extrapolation. The extrapolated error was considered to be small enough and when looking at the value of the extrapolated \( K \) in fig. 6b, it corresponded quite well to the value from the finest grid used in the mesh study, which was in the area where the behaviour of the permeability had flattened out. Since that difference was small and the extrapolated error was small enough the mesh with 75650 was chosen as a compromise between accuracy and simulation time.

![Figure 7: Extrapolation of the permeability as function of mean face area at the inlet.](image)

### 4.3 Boundary layer resolution
In order to validate the resolution of the boundary layer a study was performed on the yPlus value and on the flow profiles. The study was performed for the \( k-\Omega \) SST model for the meshes with 32000, 49800 and 75650 elements with mean velocity \( 1 \times 10^{-2} \) m/s and the yPlus values were calculated by using `pisoFoam -postProcess -func yPlus`. The values of yPlus for the latest time step are shown in table 2 and according to the guidelines in [17] the mesh with 49800 elements would have been enough for the boundary layer resolution since the mean value of yPlus was close to 1 and the maximum value 4.17102 was still within a range that could be accepted. But in fact, since wall functions were used to model the flow close to the cylinder, all three meshes yielded too low yPlus values since a value closer to 30 would have been desired when using wall functions [13].

#### Table 2: yPlus values for mean velocity \( 1 \times 10^{-2} \) m/s.

<table>
<thead>
<tr>
<th>elements</th>
<th>yPlusMin</th>
<th>yPlusMax</th>
<th>yPlusAvg</th>
</tr>
</thead>
<tbody>
<tr>
<td>32000</td>
<td>0.0301441</td>
<td>7.24999</td>
<td>2.45907</td>
</tr>
<tr>
<td>49800</td>
<td>0.0100404</td>
<td>4.17102</td>
<td>1.20558</td>
</tr>
<tr>
<td>75650</td>
<td>0.0040643</td>
<td>2.49643</td>
<td>0.67122</td>
</tr>
</tbody>
</table>

In order to examine the effects from use of wall functions an additional simulation was done for the case with 75650 elements but without wall functions. In fig. 8a and fig. 8b are the velocity component in the y-direction and the pressure plotted for the three different meshes from the yPlus study together with the extra case without wall functions. The profiles were taken at the top of the cylinder between the wall and the upper symmetry plane. As seen in both fig. 8a and fig. 8b the cases with 75650 elements coincided quite well despite the difference in use of wall functions. In fig. 8a is the flow behaviour in the free flow was similar in all cases but the 75650 elements cases deviated more in the boundary layer region. When looking at the pressure profiles in fig. 8b was it also the 75650 elements cases that deviated and the other two cases did better agree with each other. From this study it could be seen that the use of wall functions did not seem to have that much impact of the flow behaviour, but the result from different meshes differed quite much.
4.4 Turbulence resolution

Another quantity that depended on the grid size was the turbulence resolution for LES simulations according to the expression in eq. (52), since the unresolved turbulent kinetic energy depended on the grid size due to the filtering. In order to confirm that the turbulence was resolved properly for the LES simulations, the resolution was computed as a spatial average for the mesh with 75650 elements that was chosen from the convergence study. It turned out that the mean resolution for this mesh was 95.34 % when the mean flow velocity was set to 1e-5 m/s ($Re' \approx 1.457$) and 96.81 % for 1e-3 m/s ($Re' \approx 145.7$). This meant that the turbulence resolution was quite the same for low and higher $Re$ but it also meant that the calculated resolution was considerably higher than the required 80 % criterion. The high value of the turbulence resolution gave the idea of performing a simulation for the coarser mesh with 49800 elements for the mean velocity 1e–5 m/s in order to validate the elements size dependence of the turbulence resolution. The resolution calculated for this case was 91.26 %, which also this was a higher resolution than 80 %, but still not as high as for the 75650 elements cases.

4.5 Permeability as function of Reynolds number

In fig. 9a and fig. 9b was the permeability plotted as a function of Reynolds number. The values shown as points were reconstructed from Hellström et al. (2009). The values from Hellström were scaled with respect to the permeability for creeping flow for the laminar flow description, called true permeability. The simulated values for SST and LES were in fig. 9a analogous scaled by the simulated value for laminar description from table 3. In fig. 9b was the values from the LES simulation scaled by the LES simulated permeability for $Re=0.1457$ and the values from the SST simulation were scaled by the SST simulated value for $Re=1.457$. The scaling was performed in order to be able to compare the permeability behaviour with respect to $Re$.

As fig. 9b showed was the behaviour of the permeability at middle high $Re'$ similar for all cases and for the simulations with turbulence descriptions the permeability approached zero at high $Re'$. The permeability did drop in value for $Re'$ around 10 and 20 and this value seemed to be true for all cases. This permeability drop was, according to Hellström, the onset of inertia effects. From table 3 and fig. 9a it is seen that the LES simulation yielded higher values of the permeability than the SST simulation, both for high and low $Re'$. In fig. 9a and in the table it was also seen that for low $Re'$ the LES and laminar descriptions seemed to agree quite well with each other but the values for the SST deviated with almost 50 % from these. In addition did the SST description yield a permeability drop at low $Re'$. In both figs. 9a and 9b it can be seen that the new simulated values yielded relative lower values of the permeability after the permeability drop compared to Hellström.
Table 3: Values of the apparent permeability for different Re.

<table>
<thead>
<tr>
<th>Re'</th>
<th>K SST</th>
<th>K LES</th>
<th>K laminar</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.145710</td>
<td>0.00012902</td>
<td>0.00028991</td>
<td>0.00029092</td>
</tr>
<tr>
<td>1.457090</td>
<td>0.00019875</td>
<td>0.00029079</td>
<td>0.00029081</td>
</tr>
<tr>
<td>14.57000</td>
<td>0.00019954</td>
<td>0.00027993</td>
<td>0.00028211</td>
</tr>
<tr>
<td>145.7101</td>
<td>7.0223e-05</td>
<td>0.00012100</td>
<td>-</td>
</tr>
<tr>
<td>1457.090</td>
<td>9.4668e-06</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

4.6 Flow at low Re'

In this section are the field plots shown for the case with mean velocity 1e-6 m/s at steady state, which corresponds to Re' ≈ 0.1457. For this low Re' flow are the fields for pressure and velocity examined.

Pressure field In fig. 10a and fig. 10b is the pressure field plotted for LES and SST respectively. In both cases it was seen that there was an area with higher pressure before the cylinder and there was a pressure drop over the cylinder. When looking at the magnitude of the pressure field it seemed as if the pressure was higher in absolute value in the SST case, which could be viewed on as a larger pressure drop over the cylinder. For the SST simulation it could therefore be stated the the pressure gradient was larger. It was also be seen that the pressure field at the bottom of the cell was higher for the LES case than for the SST.

Velocity field In figs. 11a and 11b are the velocity fields plotted for mean velocity 1e-6 m/s, that is Re' = 0.1457, and from these figures the velocity fields looked quite similar. In figs. 11c and 11d are the velocity fields shown together with the vector field, which shows the flow direction that was from left to right and that for the SST simulation there were tendencies for recirculation at the bottom of the cell. The recirculation is better viewed in the streamline plots in figs. 11e and 11f. These plots showed that there was no recirculation in the LES case but that the SST case had developed a recirculation zone at the bottom of the cell, which was seen at the bottom right and the bottom left.

Figure 10: Pressure field for LES and SST simulations.
Figure 11: Velocity field for LES and SST simulations.
4.7 Flow at high Re' 

In this section are field plots presented for pressure, velocity, vorticity and the turbulent viscosity for the mean velocity 1e-3 m/s that is for Re'=145.7. For these simulations steady state might not be truly fulfilled due to long simulation time and convergence problems.

Pressure field In figs. 12a and 12b are the pressure fields for LES and SST respectively. Both cases had an area with higher pressure before the cylinder. In the SST case there was an area with low pressure behind the cylinder that almost looked as if it may have been composed of two eddies. In the LES case there was one small area of low pressure behind the cylinder and another area with low pressure close to the inlet. In the LES case there was also an additional high pressure area at the upper right corner and there were also two additional smaller eddies with medium high pressure, one at the top of the cylinder and one at the inlet/outlet.

![Pressure field for LES](image1.png) ![Pressure field for SST](image2.png)

(a) Pressure field for LES.  (b) Pressure field for SST.

Figure 12: Pressure field for LES and SST simulations.

Velocity field For the velocity fields in figs. 13a and 13b the mean velocity was higher, Re'=145.7. For this case there were more differences between the velocity fields. In fig. 13a was it possible to distinguish two eddies, one in front of the cylinder and one after. This observation was confirmed by the vector plots in figs. 13c and 13d, where the circulation in the eddies are shown together with the flow direction. In the streamline plots in figs. 13e and 13f the recirculation are even more clear. In the LES case the two eddies were visible and close to the one behind the cylinder was a third smaller eddy. In the SST case there was one larger eddy that spanned from behind the cylinder to the front of it. There was also a small eddy bellow the large eddy behind the cylinder. Compared to the velocity field for the Re'=0.1457 the eddies for higher Re' seemed to appear higher up on the cylinder and they were larger.
Figure 13: Velocity field for LES and SST simulations.
**Vorticity**  In figs. 14a and 14b is the vorticity field shown, which is computed as the rotation of the velocity field according to
\[
\omega = \nabla \times \mathbf{u}.
\] (58)
the vorticity showed were the largest velocity gradients were together with the rotation of the velocity field. As seen in the figures there was more rotation for the LES simulation. If one of the velocity derivatives in the vorticity is zero there will be shear instead of rotation in the field. The areas with vorticity coincided quite well with the recirculation areas in the field plots for the velocity field.

![Vorticity for LES and SST simulations](image)

(a) Vorticity for LES.  
(b) Vorticity for SST.

Figure 14: Vorticity for LES and SST simulations.

**Turbulent viscosity**  As seen in figs. 15a and 15b the LES simulation resolved more of the turbulence phenomena than the simulation run for the SST turbulence model, that is, more details could be viewed using the LES model. It should be noted that the scale differed between the plots, which means that there was more turbulence in the SST case. For the LES case the transition to the coarser mesh is shown at the upper right corner.

![Turbulent viscosity for LES and SST simulations](image)

(a) Turbulent viscosity for LES.  
(b) Turbulent viscosity for SST.

Figure 15: Turbulent viscosity for LES and SST simulations.
5 Discussion

5.1 Geometry and mesh quality

The mesh used for the simulations was built in BlockMeshDict, but for a start attempts were done for constructing the mesh using ICEM 19.2. In ICEM the mesh was constructed with fine inflation layers and good element quality and the boundary conditions were set before the mesh was exported. The mesh was exported as fluent from ICEM and was converted into an OpenFOAM mesh by the command fluentMeshToFoam <name>.msh. The conversion worked well and the resulting mesh could be viewed in ParaView and simulations were able to run, but the flows did not look as expected. It turned out that when converting a fluent mesh into a mesh in OpenFOAM the information about the boundary conditions was lost so all boundaries were set to wall patches. When changing the boundary condition type in constant/polyMesh/patchname to cyclic the matching between the inlet and the outlet was reversed. This was due to that when OpenFOAM did the numbering of the faces on inlet and outlet the numbering was done from the bottom and up on the inlet patch and from the top and down on the outlet patch and when the matching in the cyclic condition was done it seemed as if the face with lowest number in the inlet patch was matched with the face with lowest number in the outlet patch. This yielded a boundary condition that was reversed cyclic since the flow that went in the upper corner came back in the lower corner. To obtain a mesh that could manage the cyclic boundary condition the mesh was built in blockMeshDict instead. This caused a loss of accuracy when building the mesh due to limitations in controlling the elements and also due to time limit, and so the mesh quality did not become as good as in ICEM. The periodicity condition was checked for the new mesh and in figs. 5a and 5b this was proven.

The resulting mesh from blockMeshDict did not have smooth inflation layers due to problems with getting simpleGrading working on the circular geometry. There were also two areas with coarser mesh positioned at the upper corners. That was because the mesh was bounded to the position of the blocks, which were probably not defined at the most optimal positions. The loss of inflation layers might have affected the resolution at the walls due to loss of smooth transitions, but according to the study made in figs. 8a and 8b the resolution at the walls seemed to work. In addition the overall mesh was quite fine, which was confirmed by the turbulence resolution study.

The upper corners with coarser mesh affected the possibility of resolving the turbulence in these areas and the abrupt transition between the elements sizes might also have had some impact on the result. As seen in figs. 13e and 13f some of the turbulence phenomena occurred at these areas. Though since the spatial average of the turbulence resolution was as high as 96.81 %, probably somewhat lower in these areas according to the turbulence resolution study, the turbulence resolution could still be considered to be quite good. It is also worth to mention the elements in the lower corners since they were squeezed tight together. This caused some elements with bad shape, which could have affected the result, but since the flows were slow at these areas it was probable that not so much information about the turbulence and flow behaviour was lost.

5.2 Comments on the mesh study

From the mesh convergence study, which was performed on the permeability for the given mean velocity 1e-2 m/s, it was seen that the value of the permeability was changing when decreasing the cell size. To reduce the number of simulations in the mesh study the extrapolation was performed in order to estimate the error between the current value of the permeability and the extrapolated value for a mesh with infinite small elements, which would almost correspond to an analytic solution. With the given choice of meshes the extrapolation worked quite well when looking at the apparent order of accuracy and the grid refinements factors since they were of a size close to given guidelines. Because of this the extrapolated value should have been able to give a quite reliable hint about how far the simulations were from the analytic solution. This means that the 6 % error for the mesh chosen for the simulations was considered to be a reliable error estimation, and also an error small enough to use for the simulations.

During the extrapolation the mean face area at the inlet was used as the representative grid size. This might have introduced some error to the extrapolation process since a more proper measure of the representative size would have been the cell height, obtained by taking the square root of the face area. It is difficult to tell how much this affected the result from the extrapolation but it means that the extrapolated values are not to be taken for being completely true.

From the extrapolation the mesh with 75650 elements was still quite coarse but there are some arguments for why the chosen mesh could be used despite the 6 % error. As already mentioned the element quality was limited, which gave that for some areas, where the elements were coarser or of bad shape, the numerical solution was less accurate. Due to this phenomenon there would have been areas with bad resolutions even if the number of elements was increased further. From this and the time limit factor it was decided that a mesh with 75650 elements was considered to be a good enough compromise between simulation time and numerical precision.

5.3 Boundary layer resolution

From the mesh convergence study the solution was taken to be "accurate enough" for a mesh with 75650 elements and this fine mesh yielded small yPlus values. For the simulations wall functions were used, but when calculating the yPlus value it turned out to be too low for using wall functions so the optimal choice for the simulations would have been not using wall functions at all. The boundary layer study in figs. 8a and 8b showed that the flow profile depended on the mesh fineness, but not so much on whether wall functions were used or not. The profiles for the coarser meshes seemed to agree with each other at a higher level than with the finest mesh, but since the mesh study was already performed it was decided to take the profile for the finest mesh as the true result. From this study it could be stated that the use wall functions did not affect the result so much. It was also seen that the result differed quite much with respect the elements size, but that has already been discussed in the mesh study.

5.4 Turbulence resolution

The calculated mean turbulence resolution for the 75650 elements mesh and Re' = 1.457 was 95.34 %, which for an LES simulation is a high value. The turbulence resolution was also calculated for the case 49800 elements with Re' = 1.457
in order to examine the dependence of the grid size. A low flow velocity was used since it from the 75650 elements case turned to be quite small difference in resolution and so the slower mean velocity was chosen in order to reduce simulation time. The turbulence resolution for the 49800 elements case was 91.26%, which confirmed the theory that predicted the mesh dependence of the turbulence resolution. According to [16] a value of 100% would correspond to DNS, but when looking at the mesh quality and the numerical precision in the simulations it was not possible to treat the given LES solutions to be close to DNS solution. It should be mentioned that the given turbulence resolution was a spatial mean so there were areas with both higher and lower resolution, but since the mean was still quite far from 80% the resolution could be taken to be high despite effects from the spatial mean.

With respect to the turbulence kinetic energy resolution a coarser mesh could have been used instead, but then the mesh convergence study has to be taken into account. A coarser mesh gives less precision so information about the flow would be lost. It should be mentioned though that the convergence study was performed for the SST case so it is possible that the result would have been a bit different if computed for LES. Another possibility is that the use of wall functions affected the turbulence resolution, but this statement is yet to be proved.

5.5 Permeability

When calculating Re' the velocity was based on the volumetric flux over a characteristic area according to eq. (1). In OpenFOAM the flux was taken from the patchAverage function on the variable phi at the inlet. This variable turned out to be the volumetric flux per face area at the given patch, so when multiplying the resulting value from the patchAverage function with the number of elements at the inlet patch the mean velocity at the inlet was obtained. This was confirmed by using the patchAverage function on the velocity at the inlet, which gave a velocity similar to the calculated velocity from the flux.

The calculated values of the permeabilities from Hellström et al. (2009) and the simulations in this project differed from each other with about an order of $10^{-4}$. This was a large difference but it can be explained by the fact that Hellström simulated on a geometry with dimension in [cm] and in this project the geometry was defined in [m], which gave a difference of order $10^{-2}$. Since the area is the length scale squared the difference in area was of order $10^{-4}$, the same order as for the permeability. When computing the velocity in Re' the area is used together with the volumetric flux, so it is possible that this was were the order difference set in.

The permeabilities from Hellström agreed with each other and their difference to the new values could partly be explained by the geometry scale, but there were also differences in the new simulations between SST and LES. This difference was harder to explain, but since the laminar simulation agreed with the LES simulation, the values from LES were taken to be more reliable. In the SST simulation the permeability had lower values than for LES and laminar and for the lowest Re' the permeability for SST was lower than the other SST permeabilities at low Re'. When looking at the pressure gradients the SST simulations resulted in larger pressure gradients than in the LES cases. In addition, the first value for the SST simulation at Re' = 0.1457 had even higher pressure gradient relative the other SST simulations for low Re' flow, which resulted in the lower permeability. Without further examinations it might be hard to explain these deviations in the pressure gradient, but some error sources could have been errors in the problem setup or modelling error in the SST model for low Re', possibly connected to erroneous use of wall functions. All simulated cases were set up in the same way, but one thing that would have been interesting to examine is the dependence of use of wall functions for low Re'. It is possible that the slower flows in the SST model were more sensitive to wall functions and that could explain the deviation of the permeability for the lowest Re', but not the deviation between LES and SST.

5.6 Flow field

For the simulations for low Re' it was seen from the pressure field plots that there was larger pressure difference over the cylinder for the SST case than for the LES simulations. This meant that the SST simulations yielded a larger pressure gradient, which, as already stated, resulted in lower permeability for the SST since when calculating the permeability there was division by the pressure gradient. It should be noted though that the scale of the pressure for low Re' flows was of order $10^{-11}$. This was small so it is possible that the result was affected by the numerical precision, but since the values were small for both LES and SST this should not have been the reason to the difference between them. The low pressure values were due to the large geometry scale so low pressure was needed in order to yield low Re'.

For the SST simulations at low Re' it was also seen that a recirculation zone was created at the bottom of the cell. This recirculation was missing in the corresponding LES case, which in the pressure field can be explained by higher pressure at the bottom of the cylinder for the LES. The recirculation might have affected the pressure field, and so also the outcome of the permeability calculations. As discussed during the permeability analysis the LES simulations were considered to be more reliable than the SST so in fact the recirculation at low Re' in the SST case might not exist in a real flow.

In the pressure field for SST at high Re' there was a low pressure area at the inlet/outlet, which in the LES simulations seemed to have been resolved into two separate eddies. This might have been due to the ability of LES to more properly resolve the turbulence, but it should be noted that the position of the low pressure areas differed between SST and LES. The difference might have depended on the different turbulence models, but maybe also on that the fields shown for high Re' might not be proper steady state due to convergence problems during the simulations. In the velocity field plots for high Re' the eddies were shown more clear and it was seen that compared to low Re' the recirculation zone seemed to climb up on the cylinder and they were also split into several eddies. This observation agreed with the conclusions drawn in Hellström et al. (2009) about the different turbulence models.

As seen in the field plots for high Re' the recirculation zones resided higher up on the cylinder and were therefore in the area where the mesh was coarser than in the rest of the geometry. This might have caused loss of information about the flow field in these areas, but since the mean turbulence resolution for LES was considerable higher than the recommended value it is possible that the area with coarse mesh still was fine enough to resolve the flow information. The optimal mesh might have been designed with areas with finer mesh where most of the turbulent phenomena appeared to minimize the loss of information in these zones.
From the plots for the turbulent viscosity it was seen that the turbulence in the SST case was almost 70% higher than for the LES case. The higher amount of turbulence coincided with the lower value of permeability for the SST simulations. This gives the idea that maybe the permeability is affected not only by inertia effects, but also by turbulence. In the turbulent viscosity plots for LES the transition between the finer and coarser mesh was visible. Since this was only seen for the $v_1$ variable and it did not seem to affect the flow, this effect was considered to be negligible.

### 5.7 Discretisation and solution control

The discretisation used for the simulations was of first order in time and of second order in space. Using discretisation of second order lead to improved numerical accuracy, even if there was a possibility for getting numerical oscillations when using the linear interpolation [7]. When running turbulent simulations, and especially for LES simulations, a high resolution is often preferred and the resolution becomes more accurate when using a higher order discretisation scheme. The disadvantage with using higher order schemes is longer simulation time. In a case where the performance of the mesh is uncertain, which it was in this case, there has to be a compromise how much energy can be put into the numerical schemes. If a mesh is of bad quality it does not help to use a fifth order scheme since the resolution would be limited by badly shaped or coarse elements.

For the solution control the tolerance was set to 1e-6, which meant that for a converged solution the residuals should have fallen bellow that value. For the simulations with higher Re it was sometimes more difficult to fulfil that convergence criterion for the pressure field. Smaller time step was used, which yielded better values on the residuals but it also generated very low Courant number. In order to avoid mean Co smaller than 1e-4 some solutions were used even if the pressure residuals still was of the size 5e-6. The argumentation was that a 5e-6 residual has fallen more than five decades and was close to 1e-6 and so it was chosen as a compromise for avoiding even lower Co and longer run time. With this said the solutions for high Re should not be viewed on as completely reliable solutions.

For the PISO loop the number of correctors was set to two, which equalled two turns in the inner pressure loop, that is, the number of times for which the pressure was solved at each time step. If solved several times the accuracy would increase, but it would also increase the simulation time. For PISO simulations in OpenFOAM it seemed to be quite common to use two turns in the inner loop [9], but it should also be considered where to put in energy to get the most accurate result. Maybe it would give better result to use second order discretisation in time instead of running the inner pressure loop a third time.

The relaxation factors were set to 0.5. Relaxation factors are used in order to improve stability and convergence to the solutions, but if they are set to be too low it might have prohibited the iteration process to move forward and it would have taken longer time to get convergence. Since there were some problem with getting convergence on the pressure field, the relaxation factor for the pressure field might have been to small so the iterations in the inner loop might have been too restricted. This might have caused problems to obtain convergence in the pressure field but it might also have helped to guarantee convergence but at the cost of simulation time.

### 5.8 Error estimation

When building the geometry the symmetry and periodicity boundary conditions were assumed. This meant that the flow was supposed to expand infinite in the flow direction and in the direction perpendicular to the flow. There can be some discussion about these assumptions when dealing with turbulent flows. Turbulence is a flow phenomenon with an irregular behaviour and with fluctuating velocity fields [12]. By assuming that the flow would look the same for all cells might be limiting for turbulent flows since the random motions would lead to some variation in the flow field between different cells. Another limiting factor when modelling turbulence was that the simulations were run in 2D. Turbulence is a 3D phenomenon so when one direction was removed some of the information was lost. In the case of LES simulations the filter width was calculated by the scaling according to the change in geometry. The scaling might have been important at least for \( \Omega \) when the geometry was defined to be 2D the LES is not strictly applicable and the solver was warned for this before running.

In the boundary layer resolution study it was seen that for SST at Re\(^*\) = 1457 there was little difference in the flow with or without wall functions. From this the conclusion was drawn that the solution was good even though wall functions was used for yPlus much lower than 30. The SST values differed considerably from the LES and laminar simulations, which gave the idea of doubting whether the wall functions really had no effects since the study was only performed on the SST case. Another point that could have affected the accuracy or performance of the wall functions was that the boundary values for the \( k \) and \( \Omega \) used in Hellström et al. (2009) was used as initial values in the wall functions. Here it might have been two errors and one of them was that the values in Hellström were used for no wall functions so they might not have been valid when using wall functions. The other possible error was that these values was not scaled according to the change in geometry. The scaling might have been important at least for \( k \) since the unit of \( k \) was depending on the length scale.

In the above the turbulence resolution was computed and considered to be of high (maybe too high) quality. The computation of the turbulence resolution was based on the unresolved kinetic energy, the sub-grid scale kinetic energy, and this was obtained by slightly modifying the pisoFoam solver to give this variable as output. When modifying the solver some of the assumptions of having a solver that solves correct problem is lost. This means that there might be some uncertainty about the variable \( k_{res} \), but factors that supports the theory that this really was the correct variable are that it has correct unit, \([m^2/s^2]\) and it depended on the grid size, which the sub-grid scale kinetic energy was supposed to do, and it yielded a turbulence resolution of a reasonable order.

The convergence criterion was set to 1e-6 for the simulations and it turned out that since the elements were small a quite small time step was demanded to give convergence, which resulted in very low Courant numbers. For a solution to be valid the Courant number should be lower than 1 otherwise the information in cells were Co>1 would be lost. If one cell in the domain does not fulfil this criterion the solution is not reliable. For the simulated cases the maximum Courant number was around the order 1e-2, which was smaller than 1 but a really low value. For larger time steps either the convergence criterion was not fulfilled or the Courant number started to diverge to very high numbers. A solution to this could have been to use a coarser mesh, which would also have reduced the turbulence resolution, but
then again the results from the convergence study have to be taken into account. Another factor that could have affected the convergence was the use of relaxation factors. These were introduced to reduce the problem with increasing Co during a simulation but they might also have been the cause of the difficulty to obtain convergence.

5.9 Conclusions

In this master thesis project simulations were performed for high and low Reynolds number for a porous media modelled as quadratically packed cylinders. In addition the simulations were performed by using OpenFOAM. The turbulence was modelled by the LES turbulence model and it was validated by additional simulations for laminar and \( k - \Omega \) SST description. Reynolds number was defined based on the porosity and the cylinder diameter and it was denoted \( Re' \). The simulations were performed for a set of \( Re' \) and for each \( Re' \) the permeability was calculated based on Darcy’s law. The calculations showed that there was a drop in the permeability at \( Re' \approx 20 \), which according to Hellström is due to onset of inertia effects. From the permeability it was also seen that for low \( Re' \) flows the LES and laminar descriptions agreed with each other, but that the SST simulations yielded lower values of the permeability. Examining the resulting flow field plots it gave the idea that the amount of turbulence could also have influence on the resulting permeability.

5.10 Further work

Regarding the mesh convergence study it might have been of more interest to perform the study for lower \( Re' \) based on the outcome from the low \( Re' \) simulations for the SST case since the permeability for the lowest \( Re' \) deviated strongly from the other permeability for low \( Re' \). It would also have been interesting to perform the convergence study for the LES case since the SST yielded permeability on another scale than the LES and laminar cases. The mesh study can be complemented for by looking at other variables than the permeability such as the turbulent quantities \( k \) or \( \nu_t \). Also the extrapolation can be improved by using the more accurate choice of representative cell height by taking the square root of the areas used in this project. There are also possibilities for improving the mesh quality regarding inflation layers and finer elements in areas with high turbulence.

In order to obtain more information about the turbulence phenomenon, especially regarding the LES turbulence description, simulations should be performed in 3D. It is also possible that by using the periodicity the flow field was limited by the geometry. If a number of cells were used instead it is possible that the field would have been able to develop the turbulence more freely due to the random behaviour of turbulence, which could give a slightly other behaviour of the field. It would also have been interesting to use another type of packing to see how the flow field depends on the packing geometry.

The geometry of the cylinder itself could also be modified by for an example introducing a ragged surface on the cylinder to see how the behaviour of the eddies in the flow changes. In [12] it is written about flows over different spherical objects such as different sports balls. It is stated that a tennis ball, which has a ragged surface, obtains a lower critical \( Re \) (lower onset of turbulence) than a smooth sphere. It is probable that a similar argumentation holds for flow over a cylinder, which would mean that a cylinder with a ragged surface would yield turbulence at lower \( Re \). By using ragged surfaces, or surfaces modified by some other design, it would be possible to control the flow around the cylinder and by that maybe also control the behaviour of the permeability. Therefore it could be possible that, depending on the application for the simulated porous media, the surface could be designed to let different amounts of the fluid pass through the media. This argumentation is be supported by the possible correlation between higher turbulent viscosity to lower permeability.
6 APPENDIX

6.1 0 directory

```
FoamFile
{
    version 2.0;
    format ascii;
    class volScalarField;
    object p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions [0 2 -2 0 0 0 0];
internalField uniform 0;
boundaryField
{
    INLET
    {
        type cyclic;
    }
    OUTLET
    {
        type cyclic;
    }
    UP
    {
        type symmetry;
    }
    DOWN
    {
        type symmetry;
    }
    CYLINDER
    {
        type zeroGradient;
    }
    frontAndBackPlanes
    {
        type empty;
    }
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

U

```
FoamFile
{
    version 2.0;
    format ascii;
    class volVectorField;
    object U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions [0 1 -1 0 0 0 0];
internalField uniform (0 0 0);
boundaryField
{
```
INLET
{
    type cyclic;
}
OUTLET
{
    type cyclic;
}
UP
{
    type symmetry;
}
DOWN
{
    type symmetry;
}
CYLINDER
{
    type noSlip;
} frontAndBackPlanes
{
    type empty;
}

FoamFile
{
    version 2.0;
    format ascii;
    class volScalarField;
    object k;
}

dimensions [0 2 -2 0 0 0 0];
internalField uniform 0.03;
boundaryField
{
    INLET
    {
        type cyclic;
    }
    OUTLET
    {
        type cyclic;
    }
    DOWN
    {
        type symmetry;
    }
    UP
    {
        type symmetry;
    }
    CYLINDER
    {
        type kqRWallFunction;
        value uniform 0.03;
    }
}
FoamFile
{
    version 2.0;
    format ascii;
    class volScalarField;
    object nut // nuSgs;
}

dimensions [0 2 -1 0 0 0 0];

internalField uniform 0;

boundaryField
{
    INLET
    {
        type cyclic;
    }
    OUTLET
    {
        type cyclic;
    }
    UP
    {
        type symmetry;
    }
    DOWN
    {
        type symmetry;
    }
    CYLINDER
    {
        type nutkWallFunction;
        value uniform 0;
    }
    frontAndBackPlanes
    {
        type empty;
    }
}

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

Ω
format ascii;
class volScalarField;
location "0";
object omega;
}
// ****************************************************** //
dimensions [0 0 -1 0 0 0 0];
internalField uniform 300;
boundaryField
{
INLET
{
    type cyclic;
}
OUTLET
{
    type cyclic;
}
DOWN
{
    type symmetry;
}
UP
{
    type symmetry;
}
CYLINDER
{
    type omegaWallFunction;
    value uniform 300;
}
frontAndBackPlanes
{
    type empty;
}
}
// ****************************************************** //

6.2 constant directory

fvOptions
{
format ascii;
class dictionary;
location "constant";
object fvOptions;
}

momentumSource
{

type meanVelocityForce;
active yes;

meanVelocityForceCoeffs
{
    selectionMode all; // whole domain
    fields (U);
    Ubar (1e-4 0 0); // mean velocity
    relaxation 1.0;
}
transportProperties

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

transportModel Newtonian;

nu 1e-06; // almost water

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

turbulenceProperties

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

simulationType LES;

LES
{
    LESModel Smagorinsky;
    turbulence on;
    printCoeffs on;
    delta vanDriest;
    
    vanDriestCoeffs
    {
        delta cubeRootVol;
        cubeRotVolCoeffs
        {
            deltaCoeff 1;
        }
        Aplus 26;
        Cdelta 0.158;
    }
}

// or

simulationType RAS;

RAS
RASModel kOmegaSST;
turbulence on;
printCoeffs on;

// or
simulationType laminar;

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

6.3 system directory

blockMeshDict

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

scale 1;

vertices
{
// point 0, back
(4.685 0 -0.01) // 0
(4.85 0 -0.01) // 1
(5 0 -0.01) // 2
(5 1.662 -0.01) // 3
(4.162 1.662 -0.01) // 4
(4.045 1.545 -0.01) // 5
(5 2.5 -0.01) // 6
(4.162 2.5 -0.01) // 7
(2.5 2.5 -0.01) // 8
(2.5 2.35 -0.01) // 9
(2.5 2.185 -0.01) // 10
(0.315 0 -0.01) // 11
(0.15 0 -0.01) // 12
(0 0 -0.01) // 13
(0 1.662 -0.01) // 14
(0.838 1.662 -0.01) // 15
(0.995 1.545 -0.01) // 16
(0 2.5 -0.01) // 17
(0.838 2.5 -0.01) // 18

// point 19, front
(4.685 0 0.01) // 19
(4.85 0 0.01) // 20
(5 0 0.01) // 21
(5 1.662 0.01) // 22
(4.162 1.662 0.01) // 23
(4.045 1.545 0.01) // 24
(5 2.5 0.01) // 25
(4.162 2.5 0.01) // 26
(2.5 2.5 0.01) // 27
(2.5 2.35 0.01) // 28
(2.5 2.185 0.01) // 29
(0.315 0 0.01) // 30
(0.15 0 0.01) // 31
(0 0 0.01) // 32
(0 1.662 0.01) // 33
(0.838 1.662 0.01) // 34
(0.995 1.545 0.01) // 35
(0 2.5 0.01) // 36
}
blocks
{
  hex (5 4 9 10 24 23 28 29) (40 160 1) simpleGrading (1 1 1) // 0
  hex (0 1 4 5 19 20 23 24) (40 160 1) simpleGrading (1 1 1) // 1
  hex (1 2 3 4 20 21 22 23) (65 160 1) simpleGrading (1 1 1) // 2
  hex (4 3 6 7 23 22 25 26) (65 65 1) simpleGrading (1 1 1) // 3
  hex (9 4 7 8 28 23 26 27) (160 65 1) simpleGrading (1 1 1) // 4
  hex (15 16 10 9 34 35 28 23) (40 160 1) simpleGrading (1 1 1) // 5
  hex (12 11 16 15 31 30 35 34) (40 160 1) simpleGrading (1 1 1) // 6
  hex (13 12 15 14 32 31 34 33) (65 160 1) simpleGrading (1 1 1) // 7
  hex (14 15 18 17 33 34 37 36) (65 65 1) simpleGrading (1 1 1) // 8
  hex (15 9 8 18 34 28 27 37) (160 65 1) simpleGrading (1 1 1) // 9
};

edges
{
  // front
  arc 0 5 (4.392 1.0925 -0.01)
  arc 5 10 (3.5925 1.892 -0.01)
  arc 1 4 (4.535 1.175 -0.01)
  arc 4 9 (3.675 2.035 -0.01)
  // back
  arc 19 24 (4.392 1.0925 0.01)
  arc 24 29 (3.5925 1.892 0.01)
  arc 20 23 (4.535 1.175 0.01)
  arc 23 28 (3.675 2.035 0.01)
  // front
  arc 11 16 (0.608 1.0925 -0.01)
  arc 16 10 (1.4075 1.892 -0.01)
  arc 12 15 (0.465 1.175 -0.01)
  arc 15 9 (1.325 2.035 -0.01)
  // back
  arc 30 35 (0.608 1.0925 0.01)
  arc 35 29 (1.4075 1.892 0.01)
  arc 31 34 (0.465 1.175 0.01)
  arc 34 28 (1.325 2.035 0.01)
};

boundary
{
  DOWN
  [ type symmetry; faces
    (0 1 20 19)
    (1 2 21 20)
    (12 11 30 31)
    (13 12 31 32)
  ];

  INLET
  [ type cyclic;
    neighbourPatch OUTLET;
    faces
      (2 3 22 21)
      (3 6 25 22)
  ];

  UP
  [ type symmetry;
    faces
      (7 8 27 26)
  ]
}
OUTLET
{
    type cyclic;
    neighbourPatch INLET;
    faces
    {
        (14 13 32 33)
        (17 14 33 36)
    };
}

CYLINDER
{
    type wall;
    faces
    {
        (10 5 24 29)
        (5 0 19 24)
        (16 10 29 35)
        (11 16 35 30)
    };
};
mergePatchPairs
;

// decomposeParDict

FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location "system";
    object decomposeParDict;
}

numberOfSubdomains 6;
method scotch;
distributed no;
roots ()
;

// fvSchemes


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

// ******************************************************************** //
ddtSchemes
{
  default Euler;
}

gradSchemes
{
  default Gauss linear;

  limited cellLimited Gauss linear 1;
  grad(U) $limited;
  grad(k) $limited;
  grad(omega) $limited;
}

divSchemes
{
  default none;

  div(\( \phi, U \)) bounded Gauss linearUpwind unlimited;
  turbulence bounded Gauss linearUpwind limited;
  div(\( \phi, k \)) $turbulence;
  div(\( \phi, \omega \)) $turbulence;
  div(\( \phi, \epsilon \)) $turbulence;
  div((nuEff*dev(T(\( \nabla U \)))) bounded Gauss linear;
  div((nuEff*dev2(T(\( \nabla U \)))) bounded Gauss linear;
}

laplacianSchemes
{
  default Gauss linear corrected;
}

interpolationSchemes
{
  default linear;
}

snGradSchemes
{
  default corrected;
}

// ?? calculated distance to wall

wallDist
{
  method meshWave;
}

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

fvSolution

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

// ********************************************** //
solvers
{
p
    solver GAMG;
    smoother GaussSeidel;
    tolerance 1e-6;
    relTol 0.1;
}
pFinal
{
    Sm:
    relTol 0;
}
"(U|k|omega|epsilon)"
{
    solver smoothSolver;
    smoother symGaussSeidel;
    tolerance 1e-6;
    relTol 0.1;
}

relaxationFactors
{
    fields
    {
        p 0.5;
    }
    equations
    {
        U 0.5;
        "(k|omega|nut).f" 0.5;
    }
}
PISO
{
    nCorrectors 2;
    nNonOrthogonalCorrectors 0;
    pRefCell 0;
    pRefValue 0;
}

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

controlDict

FoamFile
{
    version 2.0;
    format ascii;
    class dictionary;
    location "system";
    object controlDict;
}

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

application pisoFoam://myPisoFoam;
startFrom latestTime://startTime;
startTime 0;
stopAt endTime;
endTime 3000;
deltaT 0.05;
writeControl timeStep;
writeInterval 200;
purgeWrite 0;
writeFormat ascii;
writePrecision 6;
writeCompression off;
timeFormat general;
timePrecision 6;
runTimeModifiable true;

// *-----------------------------------------------------------------------

functions {
  vorticity1 {
    type vorticity;
    libs (libfieldFunctionObjects.so);
    executeControl outputTime;
    writeControl writeTime;
  }
  fieldAverage1 {
    type fieldAverage;
    libs (libfieldFunctionObjects.so);
    enabled true;
    executeControl outputTime;
    writeControl writeTime;
    fields {
      ksgs {
        mean on;
        prime2Mean off;
        base time;
      }
      U {
        mean on;
        prime2Mean off;
        base time;
      }
    };
  }
  
  #includeFunc residuals
}
// *-----------------------------------------------------------------------

6.4 myPisoFoam

createFields.H

...
IOobject
(
    "ksgs",
    runTime.timeName(),
    mesh,
    IOobject::NO_READ,
    IOobject::AUTO_WRITE
),

mesh,

dimensionedScalar("ksgs", dimVelocity*dimVelocity, 0.0)
);

myPisoFoam.C

... ksgs_=turbulence->k(); runTime.write(); ...

6.5 Calculating variables, Python

Parameter sweep, permeability

manageData.py

import subprocess
import os
import sys
import matplotlib.pyplot as plt
import csv
import numpy as np

# List of files to be copied
os.mkdir('plotData')
cases=['case1e6', 'case1e5', 'case1e4']
time=' 2910 2920 2930 2940 2950 2960 2970 2980 2990 3000'
phi='8.89147E-11 8.89147E-10 8.89147E-09 '

```py
cases=['case1e3']
time=' 9910 9920 9930 9940 9950 9960 9970 9980 9990 10000'
phi='8.89154E-08 '
```

cases=['case1e2']
time=' 2910 2920 2930 2940 2950 2960 2970 2980 2990 3000'
phi='8.89147E-07 '

```
fceNum=225
# Copy files to plotData directory
b=0
pathCopy='python copyFile.py ' for i in range(1,len(cases)+1):
    # call copyFile.py
    arg= faceds [ i-1 ]+time
    cmdCopy=pathCopy+arg
    os.system(cmdCopy)
    b=b+1
    print (b)

# Write values to csv files
c=0
d=0
pathCalc='python csvWrite.py ' for j in range(1,len(cases)+1):
    # intPoints as arg and fix number in loops
    phactor=float( phi [d:d+12])+faceNum
    arg=cases [ j-1 ]+time+’ ’+str(phactor)
    cmdCalc=pathCalc+arg
    os.system(cmdCalc)
    d=d+12 # phi
c=c+1
    print (c)
```
# copies files (phi and deltaP) from case directories to plotData directory
# input is casename and time directories

import shutil # copy files
import os # create directory
import sys # input arguments

# copying files from case to plotData directory
arg=str ( sys.argv [1]) # name of case
# Create directory for each case
os.mkdir('plotData/'+arg)
print(sys.argv)

# Loop over relevant time directories
for j in range(2,len(sys.argv)):
    time=str (sys.argv[j])
    # Create directory for each time step
    os.mkdir('plotData/'+arg+'/'+time)
    # Copy files
    shutil.copy(arg+'/'+time+'/uniform/momentumSourceProperties','plotData/'+arg+'/'+time+'/momentumSourceProperties')

# writes values to the csv files phiMean, deltaP and values (Re,K)
# input is casename and time directories

import csv
import sys
import numpy as np
name=sys.argv[1]
print(name)

# deltaP
# Open file with dP for all time steps
deltaP=0
a=0
for i in range(2,len(sys.argv)-2):
    time=str (sys.argv[i])
    fileWayP="plotData//+name+'/'time+''/momentumSourceProperties"
    fileP=open(fileWayP,"r")
    lines=fileP.readlines()
    fileP.close()
    gradLine=lines[-4] # Line with dP value
    # Pick value deltaP as string
    pString=gradLine.rsplit(None,1)[1]
    strP1=pString.replace('e ','E')
    strP2=strP1.replace('; ','') # line ends with a semi colon
    # Save float into vector or as a sum
    deltaP=deltaP+float(strP2)
    a=a+1

# Calculate mean of deltaP
deltaPmean=deltaP/a

# Write MEAN of deltaP to file
fileP=open("deltaP.csv","a")
fileP.write(str(deltaPmean)+')
fileP.close()

# Constant parameters
porous=0.4
Dp=2*2.185
nu=1E-6
rho=997
mu=rho*nu # dynamic viscosity
# Calculate variables
phi=float (sys.argv[-1])
U=phi/A

Re=np.divide (rho*Dp, mu*(1-porous))
K=np.divide (mu*U, deltaPmean*rho)

# write Re and K to file
fileValue=open("values.csv","a")
fileValue.write(str(Re)+","+str(K))
fileValue.close()

print(’dP =’,deltaPmean)
print(” phi =”,phi)
print(”U =”,U)
print(”Re =”,Re)
print(”K =”,K)

Turbulence resolution
createkres.py
# calculate kres from UMean
import csv
import numpy as np

m=13106 # internal points

# Calculate kres as an average over a set of time steps
kresSum=0
ksgsSum=0
# start time loop
for j in range(1,len(time)):
    # open files
    fileUMean=open(time[j-1]+’/UMean’) # time average
    linesUMean=fileUMean.readlines ()
    fileUMean.close()

    fileU=open(time[j-1]+’/U’)
    linesU=fileU.readlines ()
    fileU.close()

    fileksgs=open(time[j-1]+’/ksgs’) # or ksgsMean
    linesksgs=fileksgs.readlines ()
    fileksgs.close()

    kresInst=np.array([])
ksgsInst=np.array([])
a=0
    for i in range(23,m):
        # managing files
        # UMean
        linePar=linesUMean[i]
        line1=linePar.replace(‘(‘, ‘ ‘)
        line=line1.replace(‘)’, ‘ ‘)
        UxMeanstr=line.rsplit(None,2)[0]
        UyMeanstr=line.rsplit(None,2)[1]

        UxMeanE=UxMeanstr.replace(‘e’, ‘E’)
        UxMean=float(UxMeanE)
        UyMeanE=UyMeanstr.replace(‘e’, ‘E’)
        UyMean=float(UyMeanE)

        # U
        linePar=linesU[i]
        line1=linePar.replace(‘(‘, ‘ ‘)
        line=line1.replace(‘)’, ‘ ‘)
Uxstr = line.rsplit(None, 2)[0]
Uystr = line.rsplit(None, 2)[1]

UxE = Uxstr.replace('e', 'E')
Ux = float(UxE)

UyE = Uystr.replace('e', 'E')
Uy = float(UyE)

# ksgs
line = lineskgs[i]
lineE = line.replace('e', 'E')
lineF = float(lineE)

# Instantaneous energy for each cell, write to a vector
inst = [0.5 * ((Ux - UxMean) * (Ux - UxMean) + (Uy - UyMean) * (Uy - UyMean))]

kresInst = np.append(kresInst, inst)
ksgsInst = np.append(ksgsInst, lineF)

# vector with sum of energy for each element
kresSum = kresSum + kresInst
ksgsSum = ksgsSum + ksgsInst

# mean values over time for each cell
kres = kresSum / len(time)
ksgs = ksgsSum / len(time)

# write to file
filekres = open("kres.csv", "a")
fileksgs = open("ksgs.csv", "a")

for k in range(1, len(kres) - 1):
    filekres.write(str(kres[k - 1]))
    filekres.write('
')
    fileksgs.write(str(ksgs[k - 1]))
    fileksgs.write('
')

filekres.close()
fileksgs.close()

# calculate resolution of turbulence
import csv
import matplotlib.pyplot as plt

filekres = open("kres.csv", 'r')
fileksgs = open("ksgs.csv", 'r')
lineskres = filekres.readlines()
linesksgs = fileksgs.readlines()

a = 0
M = []
Msum = 0

for row in linesksgs:
    kres = float(lineskres[a])
    ksgs = float(linesksgs[a])
    M.append(ksgs / (ksgs + kres))

    # turbulence resolution for each cell
    fileM = open("M.csv", 'a')
    fileM.write(str(M[a]))
    fileM.write('
')

    # spatial average
    Msum = Msum + (ksgs / (ksgs + kres))
    a = a + 1

fileM.close()
print('Mavg = ', Msum / a)
References


41