Graz University of Technology
Faculty of Civil Engineering
Hydraulic Engineering and Water Resources Management

Large-eddy simulation (LES) of turbulent channel flow over rough beds

Master Thesis

by

Matthias Steger

first Supervisor : Schneider, Josef, Assoc.Prof. Dipl.-Ing. Dr.nat.techn.

second Supervisor : Shahriari, Shervin, M.Sc.

Graz, March 2019

Abstract

The turbulent flow over a rough bed is one of the main concerns in modern Computational Fluid Dynamics. In flows of hydraulic-engineering the walls of interest e.g. the river bank or the bed are often rough. To account for the roughness the Large-eddy simulation is used in this thesis. One of the challenges in Large-eddy simulation is the accurate inclusion of wall roughness. Similar as the knowledge of the velocity profile of a smooth wall, the velocity profile over a rough wall is needed. The specification itself is not an easy task. Especially the law of the wall for a rough bed is still subject of ongoing research. There are several techniques available to deal with this problem. In my Thesis a porous medium and artificially generated rough elements are used to simulate wall roughness.

This Thesis "Large-eddy simulation (LES) of turbulent channel flow over rough beds" is divided into five chapters whereas the first section gives a short introduction and literature review. The second chapter gives an overview of the theoretical background of the governing equations, turbulence, DNS, LES and RANS modeling. In the third chapter the generation of roughness elements by using the file boxex.C on a surface is shown in detail. The additional source terms are accounted by the file fvOptions. Where the case itself is based on the channel 395 in OpenFOAM. In the simulation cyclic boundary conditions in streamwise and spanwise directions are accounted. The velocity profiles for a smooth channel and for two different rough channels are compared. In the fourth chapter a trapezoidal channel with bed and bank roughness from a laboratory experiments are compared with the simulation. Whereas the velocity in this simulation is decreased because of too high computational power for an average computer at present. In the last chapter the summary and conclusions are presented. Attached to this thesis the files for generating the roughness elements, the domain, options, boundary and initial conditions are shown in Appendix A for the third chapter and Appendix B for the fourth chapter.

Acknowledgement

This thesis was written during my time at the Institute of Hydraulic Engineering and Water Resources Management - Graz University of Technology. Without the guidance and encouragement of Josef Schneider and Shervin Shahriari this thesis wouldn't exist:

I would like to thank M.Sc. Shervin Shahriari for his guidance and advice throughout this research. Your support in many numerical problems and confidence in my abilities is what made this possible. You are a excellent researcher and have the ability to get not confused with the complex numerical problems in hydraulic engineering.

I would like to thank my family for their love support, and constant encouragement during the whole study.

I would like to thank my colleges and friends for their camaraderie and collaboration over the years. Specially Lukas Deutschmann who offered me a place to sleep in the last semester.

I would like to extend my gratefulness to my girlfriend Magdalena Steiner for her love, encouragement, counseling and emotional care.

Statutory declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Monday 25th March, 2019 [illegible]

. . . .

Date *Signature*

Contents

Notation

t	time
ρ	density
ρ_r	reference density
$\mathbf{u}, u_i$	velocity vector
$\mathbf{x}, x_i$	spatial co-ordinate
u, v, w	velocity x, y, z-component
U, V, W	mean velocity x, y, z-component
u', v', w'	velocity fluctuation x, y, z-component
μ	dynamic viscosity
ν	kinematic viscosity
g	gravitational acceleration
Smx, Smy, Smz	source term x, y, z-component
Φ, ϕ	physical property
$S\phi$	source property
Re	Reynolds number
E	kinetic energy
k	wave number
Δ	filter width
τ_{ij}^{SGS}	LES SGS-stresses
τ_{ij}^{RANS}	RANS Reynolds stresses
q_i^{SGS}	SGS turbulent flux
S_{ij}	strain rate
δ_{ij}	Kronecker delta
ν_t, nut	artificial turbulent kinematic viscosity
ν_{total}	total kinematic viscosity
l	length scale
q	velocity scale
k	turbulent kinetic energy
d_{50}	characteristic sediment size
σ	standard deviation
e	minimal elevation

L	length
B	width
H	height
e_1, e_2, e_3	unit vector x, y, z-component
d	Darcy coefficient
f	Forchheimer coefficient
Co	Courant number
u_*, u_τ	friction velocity, shear velocity
Re_τ	Reynolds number based on friction/shear velocity
u_{bulk}, u_m	bulk/mean velocity in flow direction
Re_h	Reynolds number based on bulk velocity
k_s^+	dimensionless roughness height
τ_w, τ_0	wall shear stress
$\frac{\partial p}{\partial x}$	pressure gradient x-component
h	water depth
y^+	dimensionless wall distance
u^+	dimensionless velocity x-component
κ	von Karman constant
F_r	Froude number
λ	spanwise spacing
Q	flow discharge
U	bulk velocity (in the Trapezoidal channel)
E_s	energy slope
Re_*	Reynolds particle number
d_{bank}	grain size at bank
d_{bed}	grain size at bed
θ	bank inclination
$\Delta x, \Delta y, \Delta z$	grid size in x, y and z direction
$\Delta x^+, \Delta y^+, \Delta z^+$	dimensionless grid size in x, y and z direction

Chapter 1

Introduction

Nowadays with the high computational power, numerical simulation methods can be used for studying the physical phenomena of the flow, solving hydraulic and environmental engineering problems with an average personal computer. Turbulence plays a very important role in these problems and the effect of turbulence is essential. The analysis of fluid flow or heat transfer is called computational fluid dynamics or CFD. The fundamental equations for CFD had been well known in the 19th century. The numerical solution techniques was developed over 50 years ago in the 1950s and 1960s. CFD is used in many applications such as: Hydraulic Engineering, Aeorospace Enginneering, Automotive Engineering, Biomedical Engineering, Turbomachinery, Chemical reactions, Marine Engineering, Video Games, Movies, Meteorology and Sports (Versteeg and Malalasekera [41]).

It has been realized that CFD is an alternative of physical modeling which have in many fields their advantages seen as follows (Bates, Lane, and Ferguson [2]):

- **Costs:** Usually the costs of the simulation is lower than building a physical model. Basically the costs depends on the CFD licenses which are used for the simulation and of the size of the physical model.

- **Time:** The time for the simulation is much lower than measuring all flow parameters of the physical model or building the model which is also size depended.

- **Scale:** The scale of the hydraulic structure can be arbitrary. In CFD the geometry can be changed very fast and easy.

- **Information:** With CFD all flow parameters are solved in the entire domain and this gives more insight of the flow behavior. The user can decide how deep the simulation should be done.

- **Repeatability:** The simulation is repeatable. Some physical models are limited by carrying out the data.

Hydraulic Engineering is concerned with the flow of water. The most common applications for hydraulics are: hydro power plants, dealing with floodings, transport of sediments, pipe flows, pumping, pipelines, open channel flows, dam constitutions, spillways, outlets, drainages, sewerage, aqueducts, etc. All these topics underly the fluid mechanics. Hydraulic engineers uses CFD to accurately predict flow characteristics. There are different numerical solution techniques in CFD available to compute the flow physic of a fluid and they are discussed in Chapter 2 (Houghtalen, Akan, and Hwang [16]).

In an open channel turbulent flow is strongly influenced by the roughness on the channel bed. In rivers, streams and estuaries the roughness conditions near the bed can vary significantly. The flow is spatially inhomogeneous. I.e. time-averaged statistics are dependent of the location in an open channel flow. A velocity profile can be spatially averaged within a predefined area which is larger than the roughness elements. To define this formulation the roughness geometry has to be known. Most simulations based on the Reynolds-averaged Navier-Stokes equations (RANS) have not advanced the understanding of flows over rough beds. Direct numerical simulations (DNSs) and large-eddy simulations (LESs) are improved in revealing details of turbulent flows. The numerical effort in RANS is essentialy lower than in LES and DNS. In many LES and DNS simulations an exact defined "roughness element" were performed and in other simulations a virtual boundary method has been used. All these predefined "roughness elements" provide statistical data and gives insight into turbulence structures (Stoesser [38]). A broad knowledge of wall turbulence in water flow has been obtained by using a hot-film anemometer or a hydrogen-bubble method. Nakagawa, Nezu, and Ueda [26] used point measurements and flow visualization to characterize the water shear flow. Nezu and Rodi [28] established that an open channel flow consists of two regions. Where the inner region is controlled by kinematic viscosity and friction velocity and the outer region near the free surface is controlled by the flow depth and maximum velocity. Lyn [21] used a two-component laser-Doppler velocimetry (LDV) to obtain a better understanding of the flow and transport implications of dunes and ripples. Tominaga et al. [39] investigated secondary currents on an open and closed channel with smooth and rough boundaries. Blanckaert, Duarte, and Schleiss [4] investigated the influence of the bank roughness, bank inclination, shallowness and boundary shear stress on the variability of flow patterns. Nikora et al. [29] used a roughness geometry function to mimic the roughness of a natural channel bed. Nikora et al. [29] found out that the roughness geometry functions from water-worked gravel beds of New Zealand rivers behave similar to the unworked gravel beds created manually in a flume. In LES a wavey surfaced bed was used by Calhoun and Street [6] to investigate neutrally stratified flow. In DNS square bars consisted with a cavity width to roughness height ratio was considered by Leonardi et al. [20] for a rough turbulent channel flow. The study has shown that recirculation zones occur upstream and downstream of each element with a width to height ratio of over seven. As an less expensive alternative is the virtual boundary method, which does not resolve the roughness explicitly. The rough wall is embedded in a Cartesian grid and the no-slip condition is imposed by body forces. In the virtual boundary method Bhaganagar, Kim, and Coleman [3] used a no-slip surface consists of a 3D "egg-carton"-shaped surface. The shape, height and distribution of the "roughness elements" were a priori known in these studies. However in an open channel the detailed bathymetry of the rough bed is hardly known. Nakayama [27] suggested in the virtual boundary method to add additional dispersive stress terms to the momentum equations to overcome these limitations. From DNS Nakayama [27] determined that this method requires a priori knowledge of the magnitude of the dispersive stresses. In LES a fixed two-dimensional dune in a turbulent open channel flow was studied by Yue, Lin, and Patel [43]. Scotti [35] placed randomly ellipsoids of a certain height on a smooth wall to mimic sandpaper roughness. This method is in LESs performed and compared with laboratory statistical data of an natural channel-bed to validate the method. Singh, Sandham, and Williams [37] presents in DNS the results of a turbulent flow of an open channel over the rough bed, which consists of spheres in a hexagonal arrangement. Stoesser [38] proposed to modify the virtual sandpaper method from Scotti [35]. To mimic the roughness of a realistic natural channel bed a roughness geometry function as presented by Nikora et al. [29] is employed.

1.1 Objective of the work

The aim of this study is to implement wall roughness with the porous medium approach with the Large-eddy simulation on a surface in a turbulent channel flow by applying the approach from Stoesser [38]. This study follows the report from Margalit [22]. With this technique dimensionless velocity profiles are generated for a smooth and for two rough cases. With these profiles the porous medium approach can be checked if this technique works. Where the case set up of the rough elements is based on channel 395 tutorial in OpenFOAM. The model has cyclic boundary conditions in the streamwise and spanwise directions. Additionally the results in a trapezoidal channel from laboratory experiments by Blanckaert, Duarte, and Schleiss [4] and Tominaga et al. [39] are compared with the simulation. This experiment shows multi-cellular secondary currents in a straight trapezoidal channel. At present the features of secondary flow are quite unknown. OpenFOAM and Paraview is able to visualize the multi-cellular secondary currents in a smooth and rough trapezoidal channel.

1.2 Methodology

This thesis is divided into five chapters. Where the first chapter shows a short literature review of many techniques which are concerned for simulating roughness. In the second chapter the importance of the Large-eddy simulation is presented. In the third chapter the roughness generation itself is discussed and computed by following the report from Margalit [22]. Where in this Thesis the simulations of rough- and smooth channels are carried out by the open-source code OpenFOAM. The "pimpleFoam" solver has been used by solving the channel flow. To account for the roughness elements the fvOptions file in OpenFOAM is used. To post process the results in OpenFOAM the program Paraview has been used. Whereas the point data which are used for plots are generated with Matlab. In the fourth chapter a trapezoidal channel with bed and bank roughness from laboratory experiments from Tominaga et al. [39] and Blanckaert, Duarte, and Schleiss [4] are compared with the simulation. Whereas the velocity in this simulation is decreased because of too high computational power for an average computer at present. The simulation is compared with the experiments and the differences between the smooth and rough walls are shown. In the last chapter the summary and conclusions are presented.

Chapter 2

Theory

The fluid flow is completely described by the Navier-Stokes equations together with the continuity equation. In the case of mass or heat transfer a scalar transport equation has to be added. For simple geometries and flow conditions analytical solutions are available. At a high Reynolds number the flow becomes turbulent and the analytical solution cannot be obtained. Solving the Navier-Stokes equations at a high Reynolds number with all length scales requires high computational power. Until 2080 optimistic predictions believe that the Direct Numerical Solution (DNS) which solve all fluid quantities at all scales wont be available for common engineering problems. Until 2045 Large Eddy Simulation (LES) which models the small scales and resolve the large eddies wont be available for common engineering problems. The most used approach at current generation is the Reynolds Averaged Navier Stokes (RANS) simulation [42]. To solve the eddies at a certain scale, the mesh must be smaller than the size of the eddy. Also the calculation has to be 3D and therefore the number of grid points and the computing cost required increase with Re^3. Even for a medium Reynolds number of 87 000 based on channel depth and bulk velocity a supercomputer of 2048 processors needs a half year to perform a DNS with $1.8 * 10^{10}$ grid points [34].

2.1 Governing equations

The fundamental governing equations for modeling the fluid flow were derived from the principles of conservation of mass, momentum, energy, Stoke's hypothesis and an equation of state. For an incompressible fluid the energy equation is not taken into account because without density variations there is no linkage between the mass, momentum and energy equations. The governing equations of an incompressible Newtonian fluid in a Cartesian coordinate system are given in Eq. 2.1 to Eq. 2.4. (Versteeg and Malalasekera [41])

$$\text{Continuity:} \qquad div(\mathbf{u}) = 0 \tag{2.1}$$

$$\text{x-Momentum :} \qquad \frac{\partial u}{\partial t} + div(u\mathbf{u}) = -\frac{\partial p}{\partial x}\frac{1}{\rho} + div(\nu\, grad(u)) + S_{mx} \tag{2.2}$$

$$\text{y-Momentum:} \qquad \frac{\partial v}{\partial t} + div(v\mathbf{u}) = -\frac{\partial p}{\partial y}\frac{1}{\rho} + div(\nu\, grad(v)) + S_{my} \tag{2.3}$$

$$\text{z-Momentum:} \qquad \underset{\text{(I)}}{\frac{\partial w}{\partial t}} + \underset{\text{(II)}}{div(w\mathbf{u})} = \underset{\text{(III)}}{-\frac{\partial p}{\partial z}\frac{1}{\rho}} + \underset{\text{(IV)}}{div(\nu\, grad(w))} + \underset{\text{(V)}}{S_{mz}} \tag{2.4}$$

The first term (I) is defined as the rate of change term, (II) is the convection term, (III) is the pressure gradient over density term, (IV) the diffusion term and (V) is the source term. Turbulence models uses extra stress terms in the Navier-Stoke equations. In tensor notation the continuity equation is as follows [34]:

$$\text{Continuity equation:} \qquad \frac{\partial u_i}{\partial x_i} = 0 \tag{2.5}$$

The momentum conservation in index notation is as follows [34]:

$$\text{Momentum equation:} \qquad \frac{\partial u_i}{\partial t} + \frac{\partial u_i u_j}{\partial x_j} = -\frac{1}{\rho_r}\frac{\partial p}{x_i} + \nu\frac{\partial^2 u_i}{\partial x_j \partial x_j} + g_i\frac{\rho - \rho_r}{\rho_r} \tag{2.6}$$

In Eq. 2.6 the Boussinesq approximation has been made so that the influence of variable density appears only on the last term $g_i\frac{\rho-\rho_r}{\rho_r}$ which is the buoyancy term. ρ_r is the reference density and g_i is the gravitational acceleration in direction x_i. Scalar quantities such as pollutant concentration and temperature etc. are indicated as a general variable ϕ and can be written in the following form:

$$\text{transport equation :} \qquad \frac{\partial \phi}{\partial t} + div(\phi\mathbf{u}) = div(\Gamma\, grad(\phi)) + S_{\phi} \tag{2.7}$$

In tensor notation the transport equation is as follows [34]:

$$\text{transport equation :} \qquad \frac{\partial \phi}{\partial t} + \frac{\partial u_i\, \phi}{\partial x_i} = \Gamma\frac{\partial^2 \phi}{\partial x_i \partial x_i} + S_{\phi} \tag{2.8}$$

Γ is the diffusivity of ϕ in Eq. 2.7 and Eq. 2.8.

2.2 Turbulence modeling

Turbulence has important effects on the flow phenomena that plays a major role in hydraulic engineering. The fluctuating turbulent motion causes an increase in momentum transfer and increases friction on solid boundaries. Turbulence causes losses around structures and flows through conduits. Another governing influence of turbulence is that the pressure and velocity distribution varies along the flow domain. This creates unsteady forces on the boundaries and for example in pipes the velocity distribution is much more uniform than in laminar pipe flow. Also in an straight open channel turbulence causes secondary motions. In the river turbulence keeps sediment particles in suspension and erodes particles from the bed. Therefore the suspended and bed load transport is caused by turbulence [34]. The term turbulence and laminar flow is important to distinguish. At a certain Reynolds number the flow become unstable and changes from laminar to turbulent flow. Reynolds [33] attempts to quantify turbulence. The Reynolds number is defined in Eq. 2.9. $|\mathbf{u}|$ is the velocity in $\frac{m}{s}$, L the length in m and ν the kinematic viscosity in $\frac{m^2}{s}$.

$$\text{Reynolds number:} \qquad Re = \frac{|\mathbf{u}|L}{\nu} = \frac{inertial forces}{viscous forces} \tag{2.9}$$

Many flows in Hydraulic Engineering are turbulent. At a high Reynolds number the flow becomes turbulent and at a low Reynolds number the flow is laminar. Turbulence is the chaotic change of field values like pressure or velocity in space and time. With low viscosity and high velocity the Reynolds number is very high which causes turbulence. With no changes in time of the field values the flow behave laminar. A typical velocity structure of a turbulent flow are shown in Fig. 2.1. [41] In this figure the velocity is decomposed into a steady mean value U and a fluctuating component $u'(t)$ The decomposition is shown in Eq. 2.10. The velocity u is defined in the x-direction. Where v and w are defined in y-, and z-direction.

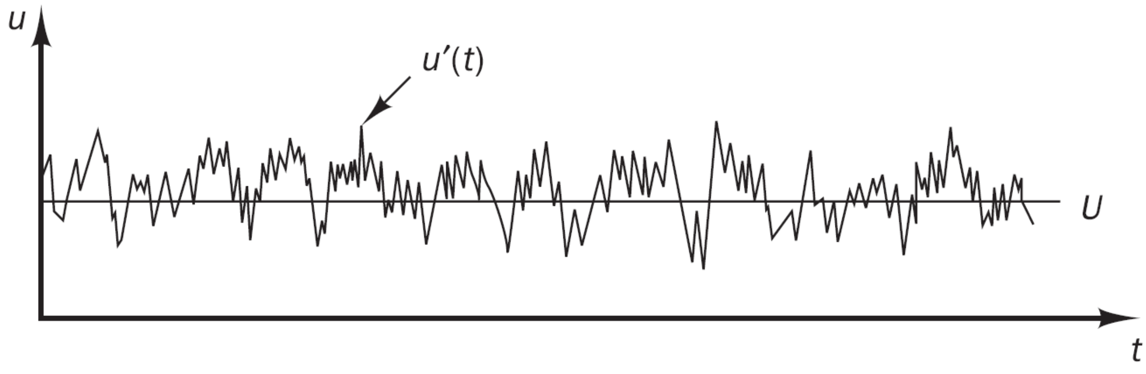


Fig. 2.1: velocity measurement in turbulent flow [41]

$$\text{Reynolds decomposition:} \qquad u(t) = U + u'(t) \tag{2.10}$$

The velocity fluctuations $u'(t), v'(t)$ and $w'(t)$ give additional stresses which are called Reynolds stresses. The main characteristics of turbulence are listed as follows: [41], [34]

- **Occurrence:** At a high Reynolds number the flow is turbulent. The interaction occurs of when the inertia or convection is high and when the viscosity or diffusion is low.
- **Irregularity:** The turbulent flow is a chaotic field change of pressure and velocity in space and time, manifest by fluctuations. Turbulent motions are very complex, unsteady and always three-dimensional.
- **Diffusivity:** Turbulence causes strong momentum, heat and mass transfer.
- **Dissipative:** At very small-scale eddy motions the turbulent kinetic energy gets converted into heat due to viscous stresses.
- **Rotational:** Turbulence incorporate a wide range of very small to large vorticity with rotational axes in all directions.
- **Continuum:** Turbulence is a continuum phenomena.

The visualization in Fig. 2.2 shows a fully turbulent flow inside the jet region where the size of eddies range from small to large [41].

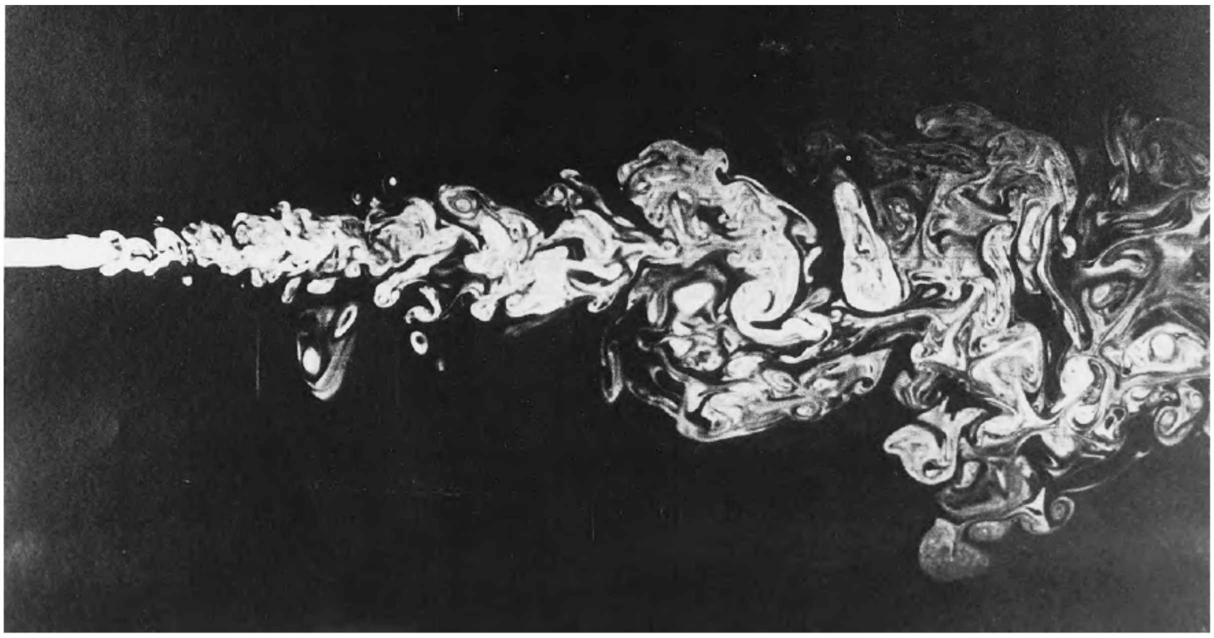


Fig. 2.2: Visualization of turbulence of a jet flow [41]

The small eddies corresponds to high frequency fluctuations. In the small scales viscous forces are acting and dissipation takes place. In Fig. 2.3 a turbulent spectrum with wave number k and kinetic energy E is shown. the most energetic eddies are the large ones and extract energy from the mean motion. Where the big motions transfer their energy to the smaller eddies with higher frequency fluctuations. This transfer is called energy cascade. At very small scales viscous forces become active which causes the dissipation of fluctuation energy. At very high Reynolds numbers the conversion into heat takes place at smaller eddies. The indication T in the figure shows that at sufficiently high Reynolds numbers the eddies are only transfered from larger to smaller ones which is called inertial sub-range [34].

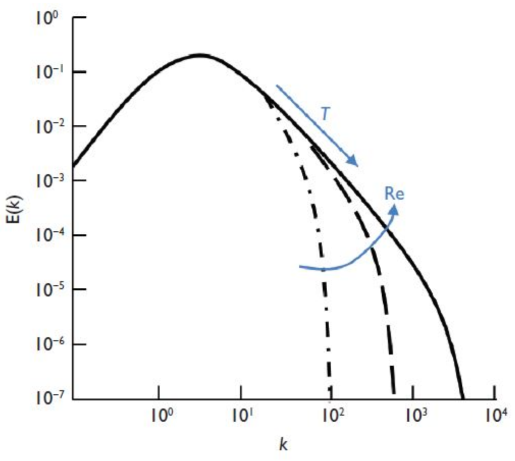


Fig. 2.3: Spectra of isotropic turbulence with increasing Reynolds number [34]

2.2.1 DNS

For an incompressible turbulent flow four unknowns u, v, w and p can be directly simulated with the instantaneous continuity and Navier-Stokes equations Eq. 2.1 to Eq. 2.4. Direct numerical simulation (DNS) has to solve the quantities at very small time steps and at a very fine spatial mesh in which the Kolmogorov length scales are reached. In this small scales the energy dissipation takes place. DNS resolve the smallest turbulent eddies. Moin and Mahesh [24] support the DNS regarding of the precise details of turbulent parameters. These are for example used by validation new turbulent models. High Reynolds numbers require a very fine mesh in each direction and small time steps to describe the processes at all length scales. The computation time for turbulent flows at high Reynolds numbers in DNS at present is not practicable. Low Reynolds numbers in DNS does not predict a realistic turbulence flow in engineering practice. An alternative is the Large-eddy simulation (LES) which has not that high computational effort as in DNS [41].

2.2.2 LES

Large-eddy simulation (LES) is used to simulate turbulent flows. The large non universal turbulence motions are directly resolved whereas the small eddies which have a universal character are modeled. DNS is highly computational expense and it increases as the cube of Reynolds number whereas LES is motivated by the limitations of DNS. On the other hand the Reynolds-stress models (RANS) are less reliable and accurate than the LES. In DNS almost all computational effort is expected on the smallest dissipative motions. The larger dynamic scale motions which are effected by the flow geometry are computed directly by LES [32]. The behavior of the smaller eddies are nearly isotropic whereas the large eddies which extract energy from the mean flow are more anisotropic and their behavior depend on the geometry of the domain. The cutoff width determines the eddies which are resolved. The information about the smaller eddies is destroyed beneath the cutoff width. In Fig. 2.4 the concept of LES is illustrated. From the mean flow the large eddies extract energy and transfers it into smaller scales. This process is called energy cascade. For selecting the cut-off width at least $80-90\%$ of the energy should be resolved. In LES the motion is as much simulated as it is possible on a affordable grid [34].

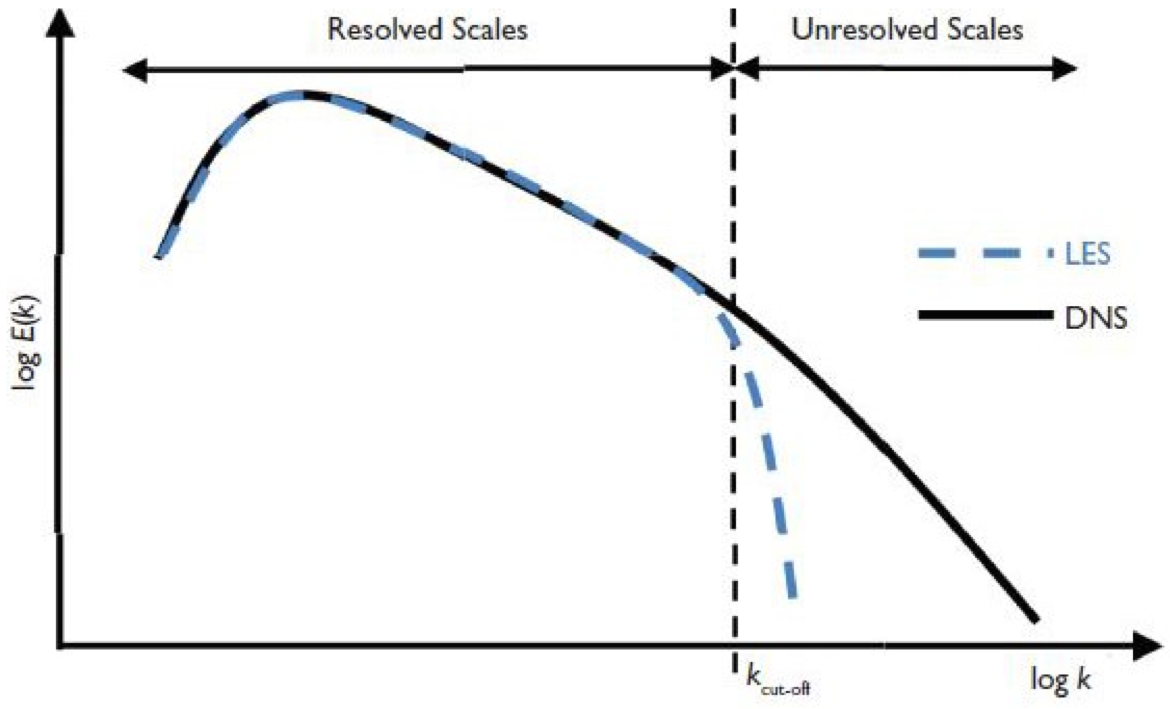


Fig. 2.4: Concept of Large Eddy Simulation [34]

The small eddies are accounted by using a subgrid-scale model (SGS model). The spatial filtering in LES can be represented by a filter function as seen in Eq. 2.11 [41]. This spatial averaging filter remove the small scale motions. Only the motions are resolved which are larger than the mesh size.

$$\text{filtering:} \qquad \bar{\phi}(\mathbf{x},t) \equiv \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} G(\mathbf{x},\mathbf{x'},\Delta)\phi(\mathbf{x'},t)dx_1'dx_2'dx_3' \tag{2.11}$$

Here $\mathbf{x}$ is the location where $\bar{\phi}$ is to be determined. In the spatial integration $\mathbf{x}'$ is the location where ϕ is considered. $\bar{\phi}(\mathbf{x},t)$ is the filtered function, $G(\mathbf{x},\mathbf{x'},\Delta)$ is the filter function and $\phi(\mathbf{x'},t)$ is the original unfiltered function. The filter function $G(\mathbf{x},\mathbf{x'},\Delta)$ is normalized so that the integration of the filter function is 1. The most used three-dimensional LES filtering functions are the box filter, Gaussian filter and spectral cutoff filter shown in Fig. 2.5. The term $\bar{\phi}(\mathbf{x},t)$ represents a three-dimensional and time dependent filtered component. The decomposition of ϕ is defined in Eq. 2.12 which is similar as the Eq. 2.21. The important difference is that the filtered residual $\bar{\phi}'(\mathbf{x},t)$ is not zero and applying the filter twice smoothes further the function $\phi(\mathbf{x'},t)$.

$$\text{decompositon:} \qquad \phi(\mathbf{x},t) = \bar{\phi}(\mathbf{x},t) + \phi'(\mathbf{x},t) \tag{2.12}$$

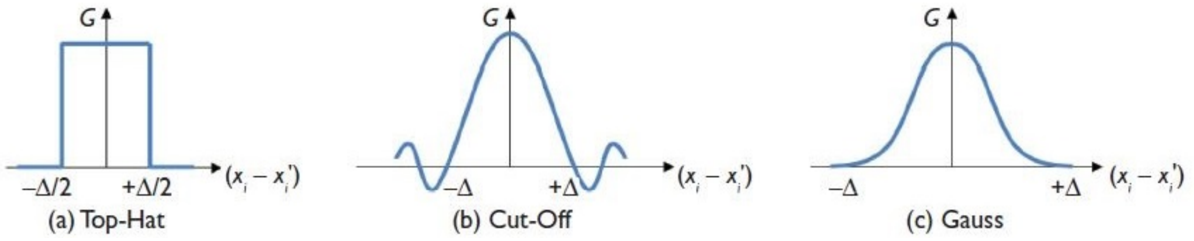


Fig. 2.5: Filter functions used in LES [34]

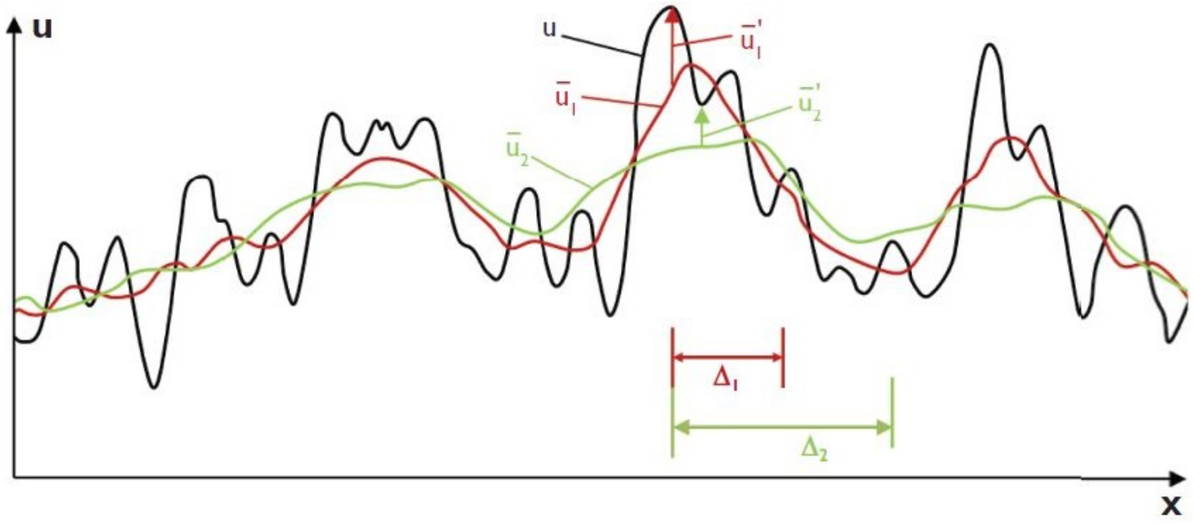


Fig. 2.6: Filtered functions $\bar{u}$ of different filter widths Δ [34]

In Fig. 2.6 the effect of a top-hat (box-filter) filter differs with the widths Δ. The cutoff width Δ is often taken into account as follows:

$$\text{cutoff:} \qquad \Delta = \sqrt[3]{\Delta x \Delta y \Delta z} \tag{2.13}$$

Filtering of the continuity and Navier-Stokes equations of an incompressible fluid from equation Eq. 2.1 to Eq. 2.4 yields as follows: [41]

$$\text{LES Continuity equation:} \qquad div(\rho\bar{\mathbf{u}}) = 0 \tag{2.14}$$

$$\text{LES x-Momentum:} \quad \frac{\partial \rho \bar{u}}{\partial t} + div(\rho\bar{u}\bar{\mathbf{u}}) = -\frac{\partial \bar{p}}{\partial x} + div(\mu\, grad(\bar{u})) - (div(\rho\overline{u\mathbf{u}}) - (div(\rho\bar{u}\bar{\mathbf{u}})) \tag{2.15}$$

$$\text{LES y-Momentum:} \quad \frac{\partial \rho \bar{v}}{\partial t} + div(\rho\bar{v}\bar{\mathbf{u}}) = -\frac{\partial \bar{p}}{\partial y} + div(\mu\, grad(\bar{v})) - (div(\rho\overline{v\mathbf{u}}) - (div(\rho\bar{v}\bar{\mathbf{u}})) \tag{2.16}$$

$$\text{LES z-Momentum:} \quad \frac{\partial \rho \bar{w}}{\partial t} + div(\rho\bar{w}\bar{\mathbf{u}}) = -\frac{\partial \bar{p}}{\partial z} + div(\mu\, grad(\bar{w})) - (div(\rho\overline{w\mathbf{u}}) - (div(\rho\bar{w}\bar{\mathbf{u}})) \tag{2.17}$$

(I) (II) (III) (IV) (V)

The over bar in Eq. 2.14 to Eq. 2.17 indicates a spatially filtered flow variable. The terms (V) are caused by filtering the convective terms of the Navier-Stokes equations thus:
$div(\rho\,\overline{\phi\mathbf{u}}) = div(\rho\,\bar{\phi}\bar{\mathbf{u}}) + (div(\rho\,\overline{\phi\mathbf{u}}) - div(\rho\,\bar{\phi}\bar{\mathbf{u}}))$
The terms (V) can be represented as a set of divergence stresses which are termed as the **sub-grid-scale stresses** (SGS-stresses). In tensor notation the continuity equation is as follows [34]:

$$\text{Continuity equation:} \qquad \frac{\partial \overline{u_i}}{\partial x_i} = 0 \tag{2.18}$$

The momentum conservation in index notation is as follows [34]:

$$\text{Momentum equations:} \quad \frac{\partial \bar{u}_i}{\partial t} + \frac{\partial \bar{u}_i \bar{u}_j}{\partial x_j} = -\frac{1}{\rho_r}\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu \frac{\partial u_i}{\partial x_j}\right) - \frac{\partial \tau_{ij}^{SGS}}{\partial x_j} + g_i \frac{\overline{\rho} - \rho_r}{\rho_r} \tag{2.19}$$

The LES SGS-stresses are defined as follows: [41]

$$\text{LES SGS-stresses:} \qquad \tau_{ij}^{SGS} = \overline{u_i u_j} - \bar{u}_i \bar{u}_j \tag{2.20}$$

For dimensional reasons $\rho(\overline{u_i u_j} - \bar{u}_i \bar{u}_j)$ are the stresses. The LES SGS-stresses can be decomposed with the following equation:

$$\text{Reynolds decomposition:} \qquad \phi(\mathbf{x}, t) = \bar{\phi}(\mathbf{x}, t) + \phi'(\mathbf{x}, t) \tag{2.21}$$

The SGS stresses can be decomposed into the following form:

$$\text{SGS stresses decomposed:} \qquad \tau_{ij}^{SGS} = \rho\overline{\bar{u}_i \bar{u}_j} - \rho\bar{u}_i \bar{u}_j + \rho\overline{\bar{u}_i u'_j} + \rho\overline{u'_i \bar{u}_j} + \rho\overline{u'_i u'_j} \tag{2.22}$$

There are three groups of the SGS stresses in Eq. 2.22. Where $\rho\overline{\bar{u}_i \bar{u}_j} - \rho\bar{u}_i \bar{u}_j$ are the Leonard stresses (L_{ij}), $\rho\overline{\bar{u}_i u'_j} + \rho\overline{u'_i \bar{u}_j}$ are the cross stresses (C_{ij}) and $\rho\overline{u'_i u'_j}$ are the LES Reynolds stresses (R_{ij}). Similar as in the Reynolds stresses in the RANS equations the R_{ij} stresses which are caused by convective momentum transfer must be modeled. However most of the SGS models uses the whole stress in Eq. 2.20 as a single entity and models τ_{ij}^{SGS} by means of a single SGS turbulence model [41]. For a filtered scalar transport equation as in Eq. 2.8 a term q_i^{SGS} appears.

$$\text{SGS turbulent flux:} \qquad q_i^{SGS} = \overline{u_i \phi} - \bar{u}_i \bar{\phi} \tag{2.23}$$

2.2.2.1 Subgrid-Scale (SGS) Models

In hydraulic-flow calculations τ_{ij}^{SGS} will be modeled of an explicit SGS model. There is an other approach (Implicit Large-Eddy Simulation) available but not discussed. A successful SGS model should dissipate from the large resolved scales the correct amount of energy. The interactions between the largest unresolved and smallest resolved scales are very important to be modeled. The boundary between the unresolved and resolved scales are shown in Fig. 2.4, indicated by the dashed line. In LES the the energy dissipation is much faster beyond the dashed line than in the DNS. For a Very-Large-Eddy simulation (VLES) a more sophisticated SGS model is required. A physically correct dissipation is important to remove the turbulent kinetic energy from the resolved scales which is shown in Fig. 2.7. The tensor τ_{ij}^{SGS} is decomposed into an anisotropic and isotropic part as follows: [34].

$$\text{LES SGS-stresses decomposed:} \qquad \tau_{ij}^{SGS} = \tau_{ij} + \frac{1}{3}\tau_{kk}^{SGS}\delta_{ij} \tag{2.24}$$

Where δ_{ij} is the Kronecker Delta.

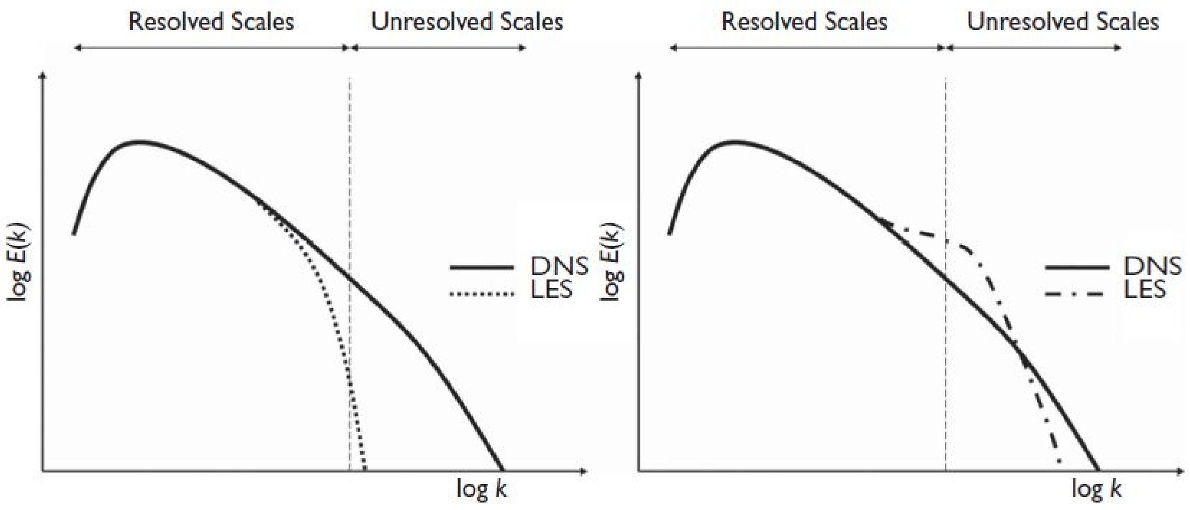


Fig. 2.7: Highly dissipative SGS model (left) and low dissipative SGS model (right) [34]

SMAGORINSKY MODEL:

Smagorinsky (1963) suggested that the Boussinesq hypothesis provide a good model for describing the effects of the unresolved eddies, since the smallest turbulent eddies are almost isotropic [41]. The anisotropic stress tensor τ_{ij} in Eq. 2.24 is approximated by a strain rate $\bar{S}_{ij}$ and artificial turbulent viscosity ν_t.

$$\text{Anisotropic stress tensor:} \qquad \tau_{ij} = -2\nu_t \bar{S}_{ij} \tag{2.25}$$

The strain rate is defined as follows:

$$\text{Strain rate:} \qquad \bar{S}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) \tag{2.26}$$

ν_t is not a fluid property and it depends on the resolved velocity field $\bar{u}_i$. From dimensionless analysis ν_t can be decomposed into a characteristic length scale l and velocity scale (q) as follows:

$$\text{turbulent viscosity:} \qquad \nu_t = l\, q \tag{2.27}$$

Where the length scale is defined to be the size of the filter width Δ times a Smagorinsky constant C_s

$$\text{lenth scale:} \qquad l = C_s \Delta \tag{2.28}$$

The velocity scale q can be determinate similar to Prandtl's mixing length theory:

$$\text{velocity scale:} \qquad q = l\,\sqrt{2\bar{S}_{ij}\bar{S}_{ij}} \tag{2.29}$$

The turbulent viscosity yields as:

$$\text{turbulent viscosity:} \qquad \nu_t = (C_s\Delta)^2\sqrt{2\bar{S}_{ij}\bar{S}_{ij}} \tag{2.30}$$

The adjustable parameter C_s is assumed to be constant and for a channel flow this value is found to be $0.065 - 0.1$. For highly three dimensional complex problems the Smagorinsky constant is impossible to determine a priori. For the implementation into a code the total viscosity term is defined as follows:

$$\text{total viscosity:} \qquad \nu_{total} = \nu_t + \nu \tag{2.31}$$

There are three modified turbulent viscosity models available to alleviate the classical Smagorinsky model. The WALE model does not need a wall damping, the dynamic Smagorinsky model calculates the model coefficient and the one equation SGS model uses a transport equation to solve the SGS kinetic energy k^{SGS} which take care of the largest unresolved scales. There are also SGS models available which are not based on the eddy viscosity concept [34].

ONE EQUATION SGS MODEL: (Yoshizawa, 1986, Ghosal et al., 1994, Menon and Kim, 1996)
For the Large-Eddy Simulation of turbulent channel flow over rough beds the k-Equation in OpenFOAM is used which represents a one equation SGS model. To account for the history effects on the SGS stresses the turbulent kinetic energy is used [34].

$$\text{turbulent kinetic energy:} \qquad k = \frac{1}{2}\tau_{kk} \tag{2.32}$$

For the total amount of energy in the unresolved eddies the velocity scale is defined as follows:

$$\text{velocity scale:} \qquad q = k^{1/2} \tag{2.33}$$

The eddy viscosity is defined as follows:

$$\text{turbulent viscosity:} \qquad \nu_t = C_v\Delta k^{1/2} \tag{2.34}$$

Where τ_{ij} is as follows:

$$\text{Anisotropic stress tensor:} \qquad \tau_{ij} = -2C_v\Delta k^{1/2}\bar{S}_{ij} \tag{2.35}$$

To determine the SGS kinetic energy k, a transport equation is used.

$$\text{transport equation:} \frac{\partial k}{\partial t} + \frac{\partial(\bar{u}_j k)}{\partial x_j} = \frac{\partial}{\partial x_j}\left[(\nu + C_k \Delta k^{1/2})\frac{\partial k}{\partial x_j}\right] + 2C_v \Delta k^{1/2}\bar{S}_{ij}\bar{S}_{ij} - C_e \frac{k^{3/2}}{\Delta} \tag{2.36}$$

2.2.3 RANS

In the engineering practice it is mostly unnecessary to solve the turbulent fluctuations in detail. With the mean pressure or mean stresses etc. the engineers are satisfied. The majority of turbulent computations are based on the Raynolds Averaged Navier-Stoke equations where the fluctuations are time filtered. The RANS turbulence models has proved to be elusive to capture the large eddies. In Eq. 2.10 the flow variables like the velocity $\mathbf{u}(t)$ can be represented in general as a property of $\varphi(t)$. Therefore $\varphi(t)$ is the sum of the mean value Φ and the fluctuation term $\varphi'(t)$ with the following equation: $\varphi(t) = \Phi + \varphi'(t)$. The mean value Φ is defined as follows:

$$\text{Mean flow variable:} \qquad \Phi = \bar{\varphi} = \frac{1}{\Delta t}\int_0^{\Delta t} \varphi(t)dt \tag{2.37}$$

Δt is the averaging time which should be small than the time scale of the mean motion but larger than the time scale of the turbulent fluctuations. The average of the fluctuating part $\varphi'(t)$ is defined as follows:

$$\text{Average of the fluctuation:} \qquad \overline{\varphi'} = \frac{1}{\Delta t}\int_0^{\Delta t} \varphi'(t)dt \equiv 0 \tag{2.38}$$

The average of two multiplied flow variables becomes to:

$$\overline{\varphi(t)\psi(t)} = \overline{\varphi'(t)\psi'(t)} + \overline{\psi'(t)\Phi} + \overline{\varphi'(t)\Psi} + \overline{\Psi\Phi} = \Psi\Phi + \overline{\varphi'(t)\psi'(t)} \tag{2.39}$$

By time averaging the governing equations Eq. 2.1 to Eq. 2.4 becomes to: [41]

$$\text{Continuity:} \qquad div(\mathbf{U}) = 0 \tag{2.40}$$

$$\text{x-Mom.:} \frac{\partial \rho U}{\partial t} + div(\rho U \mathbf{U}) = -\frac{\partial P}{\partial x} + div(\mu\, grad(U)) + S_{mx} + \left[-\frac{\partial(\rho\overline{u'u'})}{\partial x} - \frac{\partial(\rho\overline{u'v'})}{\partial y} - \frac{\partial(\rho\overline{u'w'})}{\partial z}\right] \tag{2.41}$$

$$\text{y-Mom.: } \frac{\partial \rho V}{\partial t} + div(\rho V \mathbf{U}) = -\frac{\partial P}{\partial y} + div(\mu\, grad(V)) + S_{my} + \Bigg[-\frac{\partial(\rho \overline{u'v'})}{\partial x} \tag{2.42}$$

$$- \frac{\partial(\rho \overline{v'v'})}{\partial y} - \frac{\partial(\rho \overline{v'w'})}{\partial z} \Bigg] \tag{2.43}$$

$$\text{z-Mom.: } \frac{\partial \rho W}{\partial t} + div(\rho W \mathbf{U}) = -\frac{\partial P}{\partial z} + div(\mu\, grad(W)) + S_{mz} + \Bigg[-\frac{\partial(\rho \overline{u'w'})}{\partial x} \tag{2.44}$$

$$- \frac{\partial(\rho \overline{v'w'})}{\partial y} - \frac{\partial(\rho \overline{w'w'})}{\partial z} \Bigg] \tag{2.45}$$

(I) (II) (III) (IV) (V) (VI)

The capital letters P, U, V, W, $\mathbf{U}$ and the over bar indicates a time-averaged variable. In (VI) the terms do not appear in Eq. 2.2 to Eq. 2.4 which are derived by time-averaging the nonlinear convective term. These extra turbulent stress terms, namely Reynolds stresses are decomposed into three normal stresses $\tau_{xx} = \rho\overline{u'u'}$, $\tau_{yy} = \rho\overline{v'v'}$ and $\tau_{zz} = \rho\overline{w'w'}$ and into three shear stresses $\tau_{xy} = \tau_{yx} = \rho\overline{u'v'}$, $\tau_{xz} = \tau_{zx} = \rho\overline{u'w'}$ and $\tau_{yz} = \tau_{zy} = \rho\overline{v'w'}$. The continuity equation in tensor notation is as follows [34]:

$$\text{Continuity equation:} \qquad \frac{\partial U_i}{\partial x_i} = 0 \tag{2.46}$$

The momentum conservation in index notation is as follows [34]:

$$\text{Momentum equations:} \quad \frac{\partial U_i}{\partial t} + \frac{\partial U_i U_j}{\partial x_j} = -\frac{1}{\rho_r}\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu \frac{\partial U_i}{\partial x_j}\right) - \frac{\partial \tau_{ij}^{RANS}}{\partial x_j} + g_i \frac{\overline{\rho} - \rho_r}{\rho_r} \tag{2.47}$$

Where $\tau_{ij}^{RANS} = \overline{u_i' uj'}$ are the Reynolds stresses. For dimensional reasons $\rho(\overline{u_i' uj'})$ are the stresses.

Chapter 3

Large-eddy simulation of turbulent channel flow over rough beds

3.1 Introduction

In hydraulic engineering a rough surface is very important because in an open channel all surfaces must be considered rough. Turbulent flow over a rough bed is an active area of research which is less understood than a smooth surface. Through a numerical grid the flow is explicitly resolved and is very useful providing detailed information [34]. In LES well defined roughness elements in Section 3.2.1 are chosen. In this Thesis the simulations rough- and smooth channel are executed in "OpenFOAM-6" on a "Ubuntu 18.04" terminal. The simulation is based on the report of Margalit [22] and has been modified. For creating a smooth channel the section in Section 3.2.1 and in Appendix A 6.10 fvOptions the porosity is omitted. In the following chapters the channel is generated. The computational domain were carried out in a $2\,m$ wide flume with a $4\,m$ long straight reach which is shown in Fig. 3.1. The case is adopted from the channel 395.

3.2 Pre-Processing

In this section the structure of the fluid flow problem is defined. Before executing the solver at first the geometry of domain of interest, roughness elements, boundary and initial conditions and some options have to defined which are discussed in Section 3.2.1 to Section 3.2.5.

3.2.1 Rough Bed generation

At first the geometry of the channel in Section 3.2.2 should be created before generating the rough bed. The roughness elements of the channel is specified in the file boxes.C in

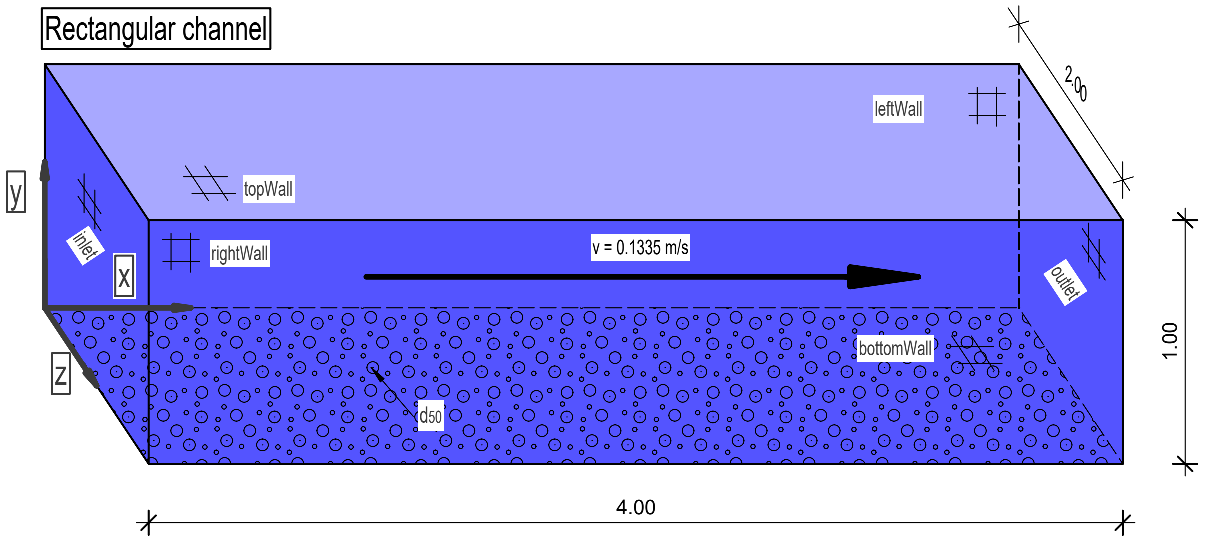


Fig. 3.1: Rectangular channel

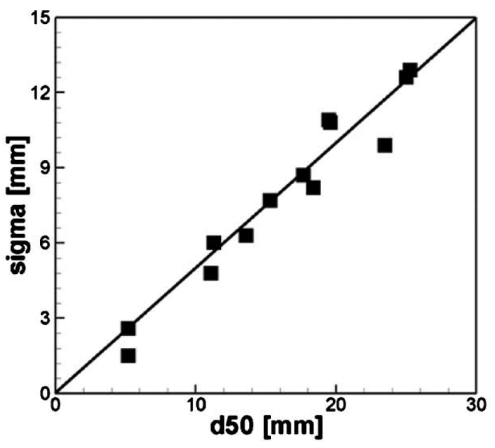


Fig. 3.2: Correlation between standard deviation σ and characteristic diameter d_{50}

Appendix A 6.1 Creation of the roughness elements, boxes.C. The C++ code is adopted from Margalit [22] and it has been modified. At first the user can specify the characteristic sediment size d_{50} which represents the roughness of the channel bed. Where 50% of a samples mass is lower or higher than the d_{50}. The constants x and z defines the streamwise and spanwise domain of the channel which is drawn in Fig. 3.6. This picture has a length of $4\,m$ and width of $2\,m$. A user can specify with scaling factor c the streamwise and spanwise spacing of the roughness elements. The standard deviation σ is defined as $0.5 * d_{50}$ which is adopted from Stoesser [38] and shown in Fig. 3.2. Then the code compute number of boxes in streamwise and spanwise directions. This channel consists of $112 * 56 = 6272$ roughness elements with a length and width of $0.036\,m$. At next the elevation of each element is created considering a normal distribution with the standard deviation of $\sigma = 0.5 * d_{50}$ and $mean = 0$. The height is restricted by $3 * \sigma$ which takes care that 99.7% of all values lie within the deviation. At first the code create positive and negative values around the mean which is 0. In the next step the lowest value will be added to the random values to create a shift that no element has a negative elevation. The shift is graphically shown in Fig. 3.3 on the left hand side. The bright gray blocks are shifted to the top with min. elevation e where dark gray block represents the additional movement. This theoretically model should correspond to the physical model on the right hand side. In this study the roughness height $k = d_{50}$.

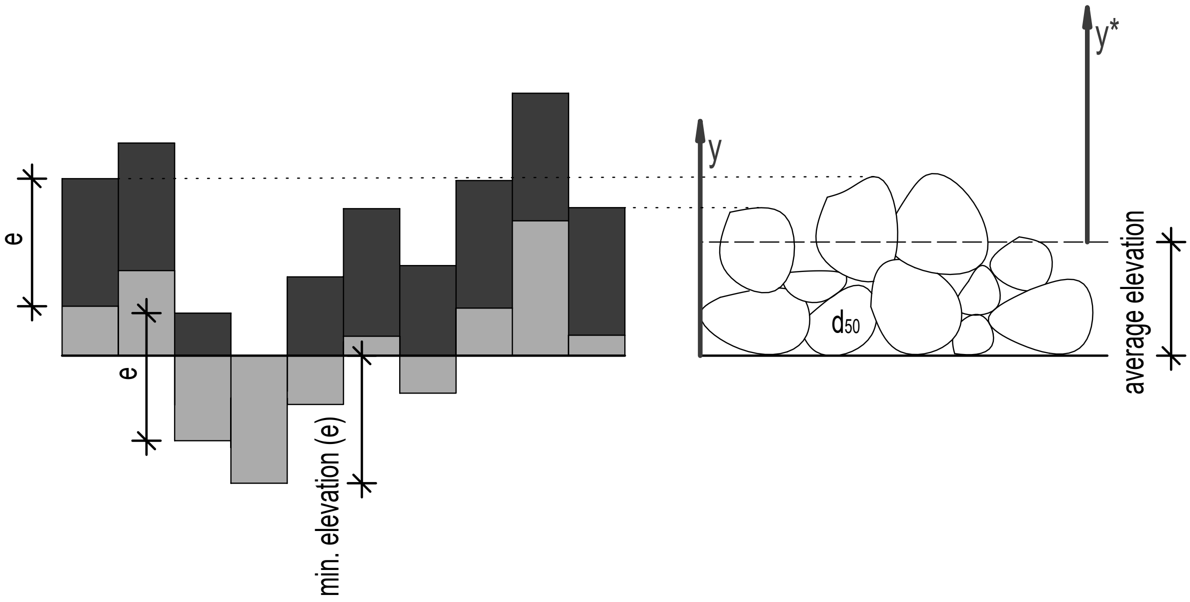


Fig. 3.3: Shift of the random values with rough bed

The code create a boxes.txt file with all roughness elements after executing the following commands:

```
g++ -std=c++11 boxes.C -o boxes
./boxes > log.boxes
```

Each line represents a rough element. The first three values corresponds the left bottom corner and the last three values corresponds the right top corner of a single box. These corners represents the two extreme points of the box. In this channel 6272 boxes are generated. The first and last three entries are shown as follows:

```
(0 0 0)(0.036 0.0345364 0.036)
(0 0 0.036)(0.036 0.0229582 0.072)
(0 0 0.072)(0.036 0.0442115 0.108)
...
(3.996 0 1.908)(4.032 0.0254329 1.944)
(3.996 0 1.944)(4.032 0.0214726 1.98)
(3.996 0 1.98)(4.032 0.0212569 2.016)
```

In the next step these boxes are created by the file topoSetDict which is shown in Appendix A 6.2 topoSetDict. This file creates a cell set into the mesh where the source is the boxes.txt file. The following commands are necessary to add the roughness elements into the domain:

```
topoSet
setsToZones
```

The first command create a new set "bed" and the second one add this set into a zone. In paraview the zone can be visualized by clicking the button "Read zones" to see the bed roughness. The rough elements are shown in Fig. 3.4 The first few roughness elements

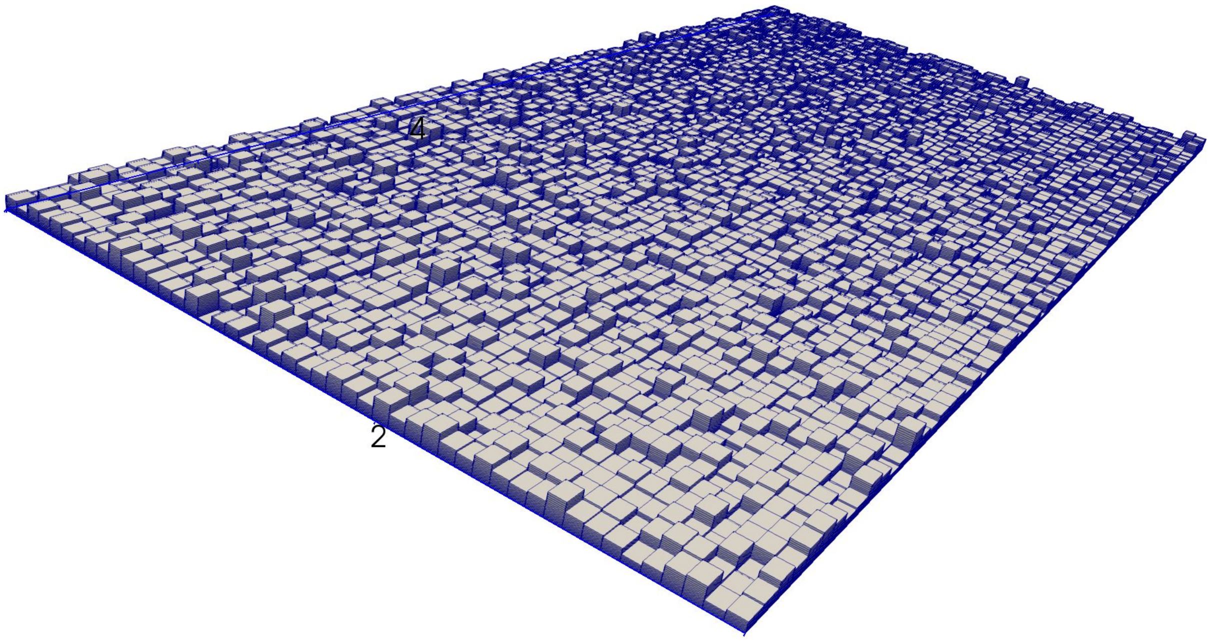


Fig. 3.4: Rough bed

which are located at the coordinate origin are shown in Fig. 3.5. The geometry of the whole domain is shown in Fig. 3.6. In Appendix A 6.3 blockMeshDict the number of cells are defined. The first block at the bottom with a length of $0.2\,m$ has 36 cells in

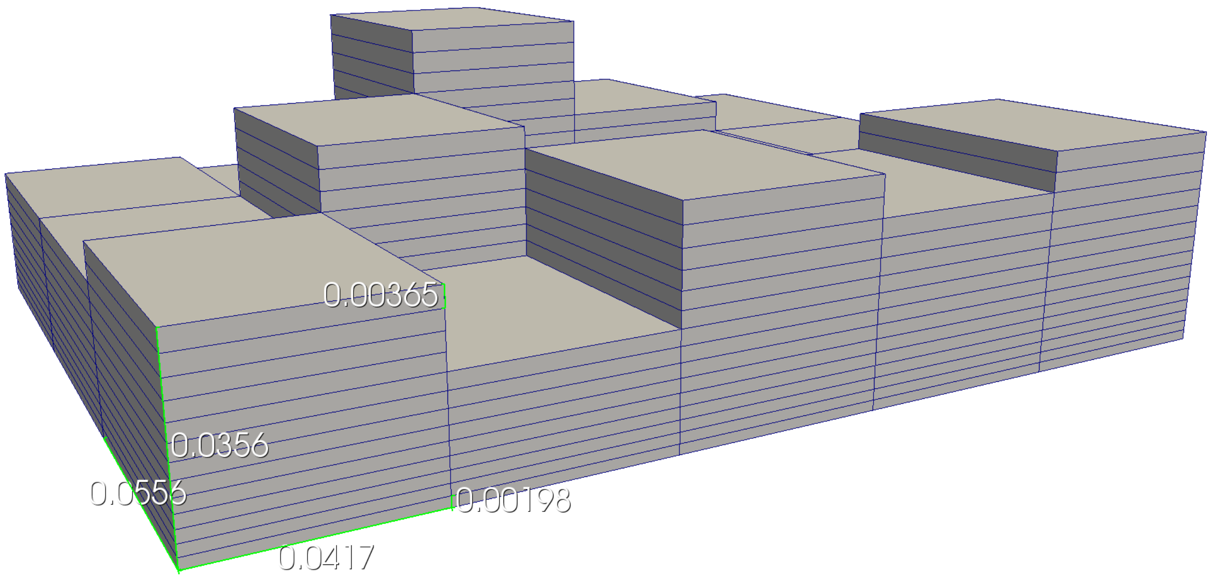


Fig. 3.5: Size of the rough elements

y-direction with an expansion ratio of 6. This grading consists of a cell-to-cell expansion ratio of about 1.0525 where the first cell has a width of about 0.00198 m and the last one has a length of about 0.01186 m which results by calculation the first cell by factor of 6 or 1.0525^{35}. The first roughness element in Fig. 3.5 has a length of 0.0417 m, width of 0.0566 m and height of 0.0356 m. This geometry almost matches with the values in boxes.txt with a length of 0.036 m, width of 0.036 m and height of 0.0345 m. The reason why they do not match perfectly together is that the whole geometry of the channel can only be expressed by a finite number of cells.

3.2.2 Geometry of the Domain

All geometry in OpenFOAM are generated in a three dimensional Cartesian coordinate system. The blockMeshDict entries for this case are shown in Appendix A 6.3 blockMeshDict. The file first specifies coordinates of the block vertices which are converted into meters. The channel domain consists of a cuboid of side length $L = 4.00\,m$, width $B = 2.00\,m$ and height $H = 1.00\,m$. The cuboid is separated into two blocks of height $H_1 = 0.20\,m$ and $H_2 = 0.80\,m$. The block structure is shown in Fig. 3.6 Then the file defines blocks and number of cells within it. This case consists of two blocks which are defined by 8 vertices of each block. The first block at the bottom consists of 72 cells in x-direction, 36 cells in y-direction and 48 cells in z-direction. The second block at the top consists of 72 cells in x-direction, 38 cells in y-direction and 48 cells in z-direction. Therefore the whole domain consists of 255 744 cells. A non uniform mesh is used for both blocks where in y-direction the last cell is 6 times longer than the first one or rather 5 times longer than the first one. The grid structure is shown in Fig. 3.7. The domain

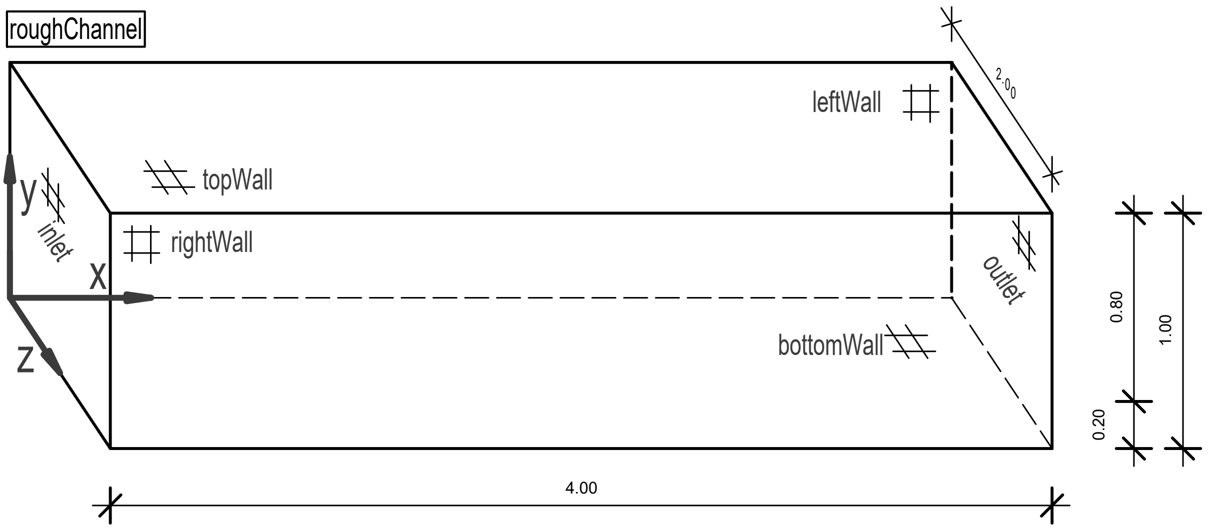


Fig. 3.6: Visualization of the Channel geometry

has 6 boundaries which are named as bottomWall, topWall, leftWall, rightWall, inlet and outlet. the bottomWall is defined by type wall which represent a solid wall at the bottom. The topWall is defined by type patch which contains no geometric or topological information. The remaining boundaries are defined by type cyclic. This type enables two boundaries to be treated as if they are physically connected. With executing the blockMesh command the blocks and numbers of cells withing it are generated.

```
blockMesh
```

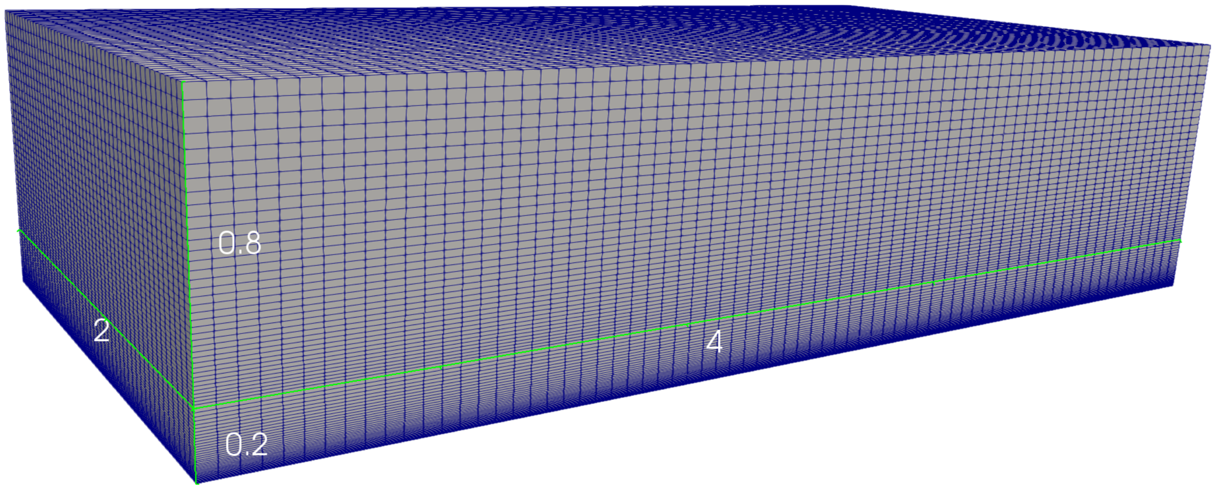


Fig. 3.7: Visualization of the mesh

3.2.3 Boundary and Initial Conditions

3.2.3.1 Velocity, U

Once the mesh generation is done the initial and boundary conditions are generated. The velocity U is set up in Appendix A 6.4 Velocity, U. The internal field is specified to be (0 0 0). In Section 3.2.3.5 the velocity values at time is $0\,s$ will be mapped from channel395. The bottom wall is set up as a noSlip boundary condition which restricts the velocity as a fixed value of (0 0 0) at this patch. At the top wall the velocity is defined as a slip boundary condition. The remaining patches have a cyclic boundary condition. The velocity at the outlet is physically connected to the inlet whereas the left wall is physically connected to the right wall.

3.2.3.2 Kinematic Pressure, p

The kinematic pressure p in $\frac{m^2}{s^2}$ is set up in Appendix A 6.5 Kinematic Pressure, p. The internal field is specified to be 0. In Section 3.2.3.5 the pressure values at time is $0\,s$ will be mapped from channel395. The bottom and top wall consists of a zeroGradient boundary condition. At this patch the normal gradient p is zero over time. The remaining boundary fields have a cyclic boundary condition. The pressure at the outlet is physically connected to the inlet whereas the the left wall is physically connected to the right wall.

3.2.3.3 turbulent kinetic energy, k

The turbulent kinetic energy k in $\frac{m^2}{s^2}$ is set up in Appendix A 6.6 Turbulent Kinetic Energy, k. The internal field is specified to be 0. In Section 3.2.3.5 the turbulent kinetic energy values at time is $0\,s$ will be mapped from channel395. The values at the bottom wall are defined to be zero. The top wall consists of a zeroGradient boundary condition. At this patch the normal gradient p is zero over time. The remaining boundary fields have a cyclic boundary condition. The turbulent kinetic energy at the outlet is physically connected to the inlet whereas the the left wall is physically connected to the right wall.

3.2.3.4 turbulence viscosity, nut

The turbulence viscosity nut in $\frac{m^2}{s}$ is set up in Appendix A 6.7 Turbulence viscosity, nut. The internal field is specified to be 0 initially. In Section 3.2.3.5 the turbulence viscosity values at time is $0\,s$ will be mapped from channel395. The bottom and top wall has a zeroGradient boundary condition which means that the normal gradient from nut is zero over time. The remaining boundary fields have a cyclic boundary condition. The turbulence viscosity at the outlet is connected to the inlet and the left wall is connected to the right wall.

3.2.3.5 Adapting fields from channel395

The file mapFieldsDict is shown in Appendix A 6.8 mapFieldsDict. Inside the mapFieldsDict the patches that coincide can be specified. The patch bottom wall has the same name in channel395 and roughChannel. The fields at the patch sides1_half0 and inout1_half0 are mapped to the patch leftWall and outlet. Where the boundaries inout1_half1 and sides1_half1 are not mapped because the left- and right wall in the rough channel are physically connected. Then the path topWall that cuts the geometry is named. In the picture Fig. 3.8 the domain of channel395 and the rough channel is shown. With the following command the boundary and initial fields from channel395 to roughChannel are mapped.

```
mapFields ../channel395
```

After executing this command the velocity field should look like in Fig. 3.9

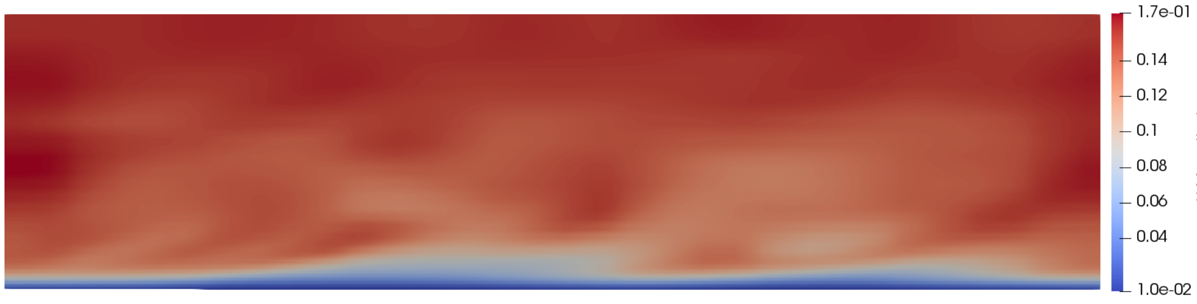


Fig. 3.9: Velocity, U at time 0

3.2.3.6 changeDictionaryDict

With the changeDictionaryDict all fields in the 0 folder can be modified together which is shown in Appendix A 6.9 changeDictionaryDict. To set all internal fields back to 0, a vector is specified by (0 0 0) and a scalar by 0.

3.2.4 Options

3.2.4.1 fvOptions

With the file fvOptions any physics can be represented as sources or constraints on the governing equations e.g. porous media MRF (multiple reference frame) and body forces [31]. In this case the rough bed is given a porosity and the the flow is driven by a pressure gradient which is shown in Appendix A 6.10 fvOptions. For this case the mean velocity is defined as 0.1335 $\frac{m}{s}$. The porous media is is given by very high values that the cells become practically impermeable, where d is the Darcy coefficient in $[\frac{1}{m^2}]$ and f is the

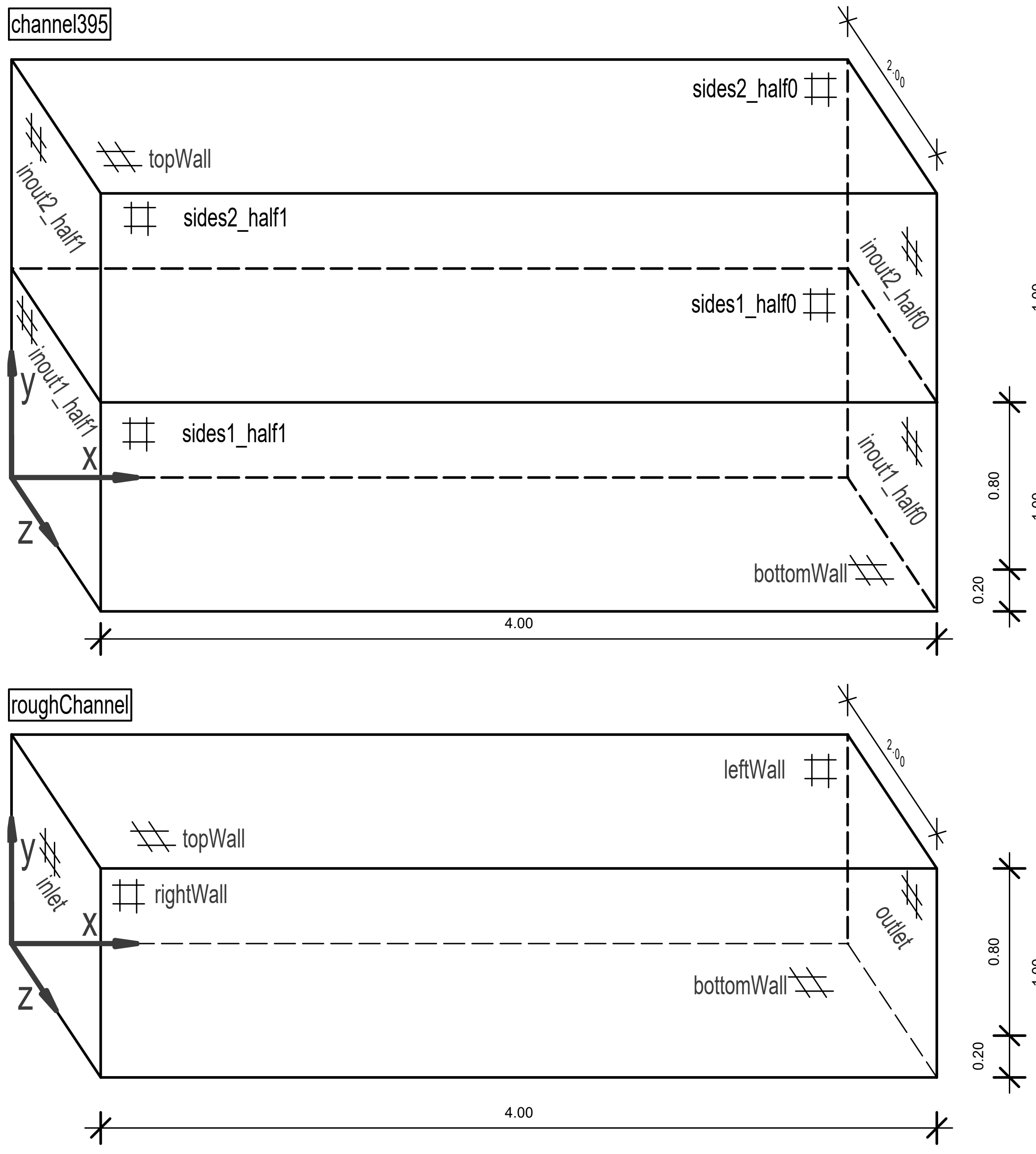


Fig. 3.8: Mapping channel395 to roughChannel

Forchheimer coefficient in $\frac{1}{m}$ [30]. This components are specified by a coordinate system where e1 and e2 sets the local orientation of the coefficients.

3.2.4.2 transportProperties

The kinematic viscosity is specified by $1.9 * 10^{-5} \frac{m^2}{s}$ and the average velocity is defined by $0.1335 \frac{m}{s}$ which is shown in Appendix A 6.11 transportProperties.

3.2.4.3 turbulenceProperties

Obviously the LES model is used for the simulation which can be shown in Appendix A 6.12 turbulenceProperties. Alternatively the RASModel or laminar can be used for the computation. In this case the k-Equation model is used, which uses one k equation to model turbulence in the sub-grid scale. For the spatial filter delta the cube root of each cell is used because the bed cells are not regarded as a wall.

3.2.4.4 fvSchemes (Numerical Schemes)

The discretization schemes which are used for this case are shown in Appendix A 6.13 fvSchemes. In the fvSchemes dictionary the numerical schemes can be specified. The categories are subdivided as follows:

- **ddtSchemes:** Defines a time scheme. $\frac{\partial}{\partial t}, \frac{\partial^2}{\partial^2 t}$
- **gradSchemes:** Defines a gradient scheme. ∇
- **divSchemes:** Defines a divergence scheme. $\nabla *$
- **laplacianSchemes:** Defines a laplacian scheme. ∇^2
- **interpolationSchemes:** Defines the cell to face interpolations of values.
- **snGradSchemes:** Defines the component of gradient normal to a cell face.
- **wallDist:** Defines the distance to wall.

For the time derivatives, a steadyState (derivatives are 0), Euler (first order implicit, transient, bounded), backward (second order implicit, transient, potentially bounded), CrankNicolson (second order implicit, transient, bounded, requires $\psi = 0.9$) and localEuler (first order implicit, pseudo transient) scheme can be specified.

For the gradient the Gauss linear scheme is primarily used, which interpolate for the finite volume discretization the values linearly from cell centers to face centres.

If the default is set to none in the divergence schemes then the Gauss linear scheme is used. The Gauss integration uses a flux phi and by one of the schemes the advected field which is interpolated to the cell faces can be selected. For the divergence, a linear (second order, unbounded), linearUpwind (second order, upwind-biased, unbounded), LUST (blended 75% linear and 25% linearUpwind scheme), limitedLinear (linear scheme that limits in direction of upwind in regions of rapidly changing gradient) and upwind (first-order, bounded) scheme can be used. For more details of these numerical schemes see in [13].

3.2.4.5 fvSolution

In the fvSolution dictionary the numerical solver and tolerances of each field can be specified, where this file is adopted from channel395. It is shown in Appendix A 6.14 fvSolution. For the PIMPLE solver it is necessary to specify a reference point for the pressure. The coordinates for this point is at (0 1 0) specified which is at the top of the channel. For more details of the solution and algorithm control see in [14].

3.2.4.6 controlDict

In Appendix A 6.15 controlDict the configuration of the simulation can be specified. When executing the solver pimpleFoam the computation starts at time 0, write interval at every 500 seconds of the simulation and ends at $20\,000\,s$. The Courant number for this case is set by 0.7 which is defined for one cell as follows:

$$\text{Courant number:} \qquad Co = |\mathbf{U}|\frac{\delta t}{\delta x} \tag{3.1}$$

$|\mathbf{U}|$ is the magnitude of the velocity through the cell, δt is the time step and δx is the cell size in direction of the velocity. (Christopher J. Greenshields [9]) Furthermore time-averaged quantities like the velocity U and the pressure p are of interest. At the bottom in controlDict the fields which have to be averaged during runtime are specified.

3.2.4.7 decomposeParDict

The dictionary for decomposing the mesh and fields of the domain to increase the computation power is shown in Appendix A 6.16 decomposeParDict. At first the number of sub domains can be specified which is dependent on the numbers of cores on the computer. This simulation is done on 4 cores of 2.80 GHz. In addition the simple method is used which splits the geometry into pieces by direction.

3.2.5 Folder tree

Finally after specifying all files and executing some commands the folder tree should look as follow:

```
.
├── 0
│   ├── U
│   ├── k
│   ├── nut
│   └── p
├── constant
│   ├── fvOptions
│   ├── polyMesh
│   │   ├── boundary
│   │   ├── cellZones
│   │   ├── faceZones
│   │   ├── faces
│   │   ├── neighbour
│   │   ├── owner
│   │   ├── pointZones
│   │   ├── points
│   │   └── sets
│   │       └── bed
│   ├── postChannelDict
│   ├── transportProperties
│   └── turbulenceProperties
└── system
    ├── blockMeshDict
    ├── boxes
    ├── boxes.C
    ├── boxes.txt
    ├── changeDictionaryDict
    ├── controlDict
    ├── decomposeParDict
    ├── fvSchemes
    ├── fvSolution
    ├── log.boxes
    ├── mapFieldsDict
    └── topoSetDict
```

Before executing the solver, the geometry has to split into 4 sub domains with the following command:

```
decomposePar
```

This will create 4 folders as follows:

```
.
├── processor0
│   ├── 0
│   │   ├── U
│   │   ├── k
│   │   ├── nut
│   │   └── p
│   └── constant
│       └── polyMesh
│           ├── boundary
│           ├── boundaryProcAddressing
│           ├── cellProcAddressing
│           ├── cellZones
│           ├── faceProcAddressing
│           ├── faces
│           ├── neighbour
│           ├── owner
│           ├── pointProcAddressing
│           ├── points
│           └── sets
│               └── bed
├── processor1
│   ├── 0
│   │   ├── U
.   .   .
├── processor2
│   ├── 0
│   │   ├── U
.   .   .
├── processor3
│   ├── 0
│   │   ├── U
.   .   .
```

3.3 Solver

To solve the momentum and continuity equations the pimpleFoam solver is used. It is a incompressible transient solver for turbulent flow of Newtonian fluids, with optional mesh motion and mesh topology changes [9]. This solver is a transient solver for incompressible, turbulent flow of Newtonian fluids on a moving mesh. The solver uses the PIMPLE (merged PISO-SIMPLE) algorithm to solve the continuity equation and momentum equation. For this solver the velocity- and kinematic pressure field and additional turbulence fields are required[1]. Where the SIMPLE (Semi-Implicit Method for Pressure Linked Equation method) guesses the pressure field and advance the velocity field at the first place, then for a pressure correction variable the Poisson equation will be solved, then with a pressure correction variable the pressure will be corrected and the velocity uses the corrected pressure and in the last step the velocity will be checked for continuity [34]. The parallel simulation starts by the following command:

```
mpirun -np 4 pimpleFoam -parallel » log.run &
```

To follow the simulation the following command is used:

```
tail -f log.run
```

For instance for the rough channel the simulation stopped at $20\,000\,s$. Where in the log.run file every time step has been recorded. At a velocity of $0.1335\,\frac{m}{s}$, channel length of $4\,m$ and time of $20\,000\,s$ the flow has been passed 667.50 times. The following command is used to reconstruct the domain:

```
reconstructPar
```

For more details of the solver see in [8].

3.4 Post-Processing and results

The first example concerns a open channel with a smooth bed and the second and third one concerns a rough bed. At a Reynolds number based on the friction velocity which is shown in Eq. 3.2 and Reynolds number based on the bulk velocity which is shown in Eq. 3.3 is listed in Tab. 3.1 [34]

$$\text{Reynolds number based on friction velocity:} \qquad Re_\tau = \frac{u_* h}{\nu} \tag{3.2}$$

$$\text{Reynolds number based on bulk velocity:} \qquad Re_h = \frac{u_{bulk} h}{\nu} \tag{3.3}$$

The water depth h is $1\,m$ and the kinematic viscosity ν is $1.9 * 10^{-5}\,\frac{m^2}{s}$. The flow is developed by a periodic boundary conditions in streamwise and spanwise directions. The dimensionless roughness height $ks^+$is defined by Eq. 3.6.

Tab. 3.1: Dimensionless parameters obtained by simulations

	$u_{bulk}\ \frac{m}{s}$	Re_τ	Re_h	ks^+
Smooth wall	0.1335	425	700	0
Rough wall	0.1335	480	7000	11.53
Rough wall	0.2500	910	13000	21.35

3.4.1 postChannelDict

For post-processing the file postChannelDict is used which is shown in Appendix A 6.17 postChannelDict. This code can specify the direction of depth and whether if the domain is symmetric or not. This case has a solid bottom and an open top which indicates that the domain is not symmetric. To create spatial averaged quantities in streamwise and spanwise directions the following command is used:

```
postChannel
```

This utility create the folder graphs with the following content:

```
.
├── Uf.xy
├── k.xy
├── pPrime2Mean.xy
├── u.xy
├── uv.xy
├── v.xy
└── w.xy
```

3.4.2 Graphs for rough and smooth channel

In the log.run file the pressure gradient in each time step is listed. With the following command the pressure gradient over time will be scanned and write into the file gradP.txt:

```
cat log.run | grep ’pressure gradient’ | cut -d’ ’ -f11 | tr -d ’,’ > gradP.txt
```

With the following command each time step will be scanned and write into the file time.txt.

```
cat log.run | grep -w ’Time’ | cut -d’ ’ -f3 | tr -d ’,’ > time.txt
```

In Fig. 3.10 the pressure gradient over time for the rough channel is shown. The first $1000\,s$ are cut away which make sure that the values have reached a certain stability. The average pressure gradient has a value of $8.6489 * 10^{-5}$. The values vary from $7.119 * 10^{-5}$ to $10.4356 * 10^{-5}$

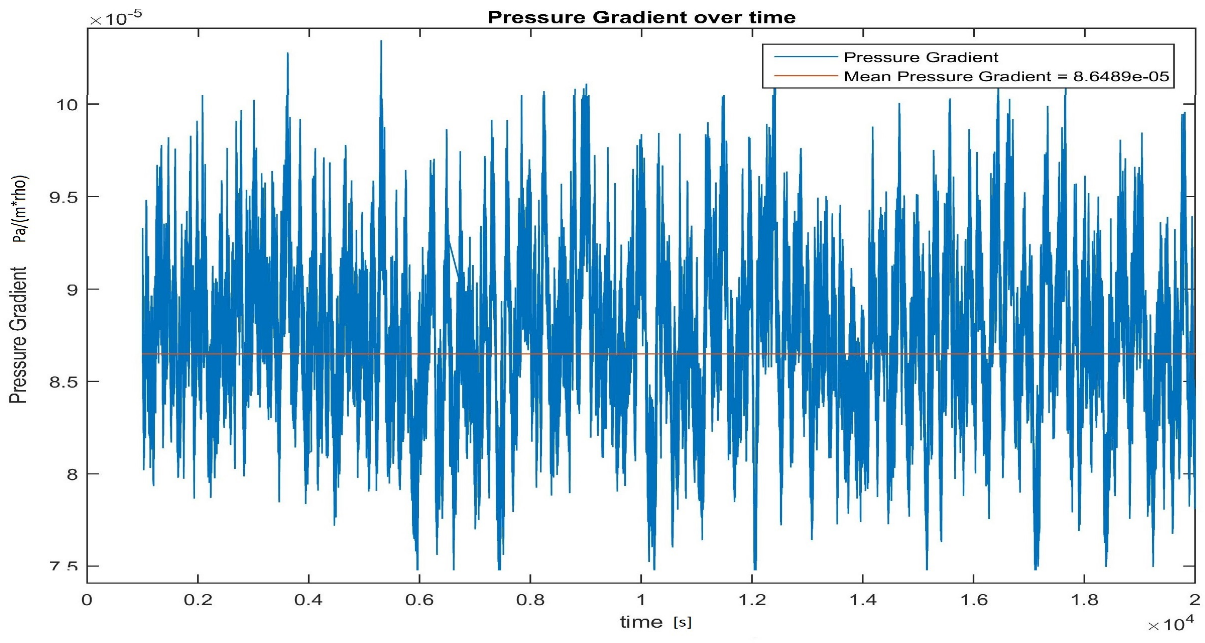


Fig. 3.10: Pressure gradient for rough channel $ks^{+} = 11.53$

In Fig. 3.11 the pressure gradient over time for the rough channel with an increased velocity of $0.25\,\frac{m}{s}$ is shown. The first $1000\,s$ are cut away which make sure that the values have reached a certain stability. The average pressure gradient has a value of $29.633{*}10^{-5}$. The values vary from $36.339 * 10^{-5}$ to $25.3425 * 10^{-5}$

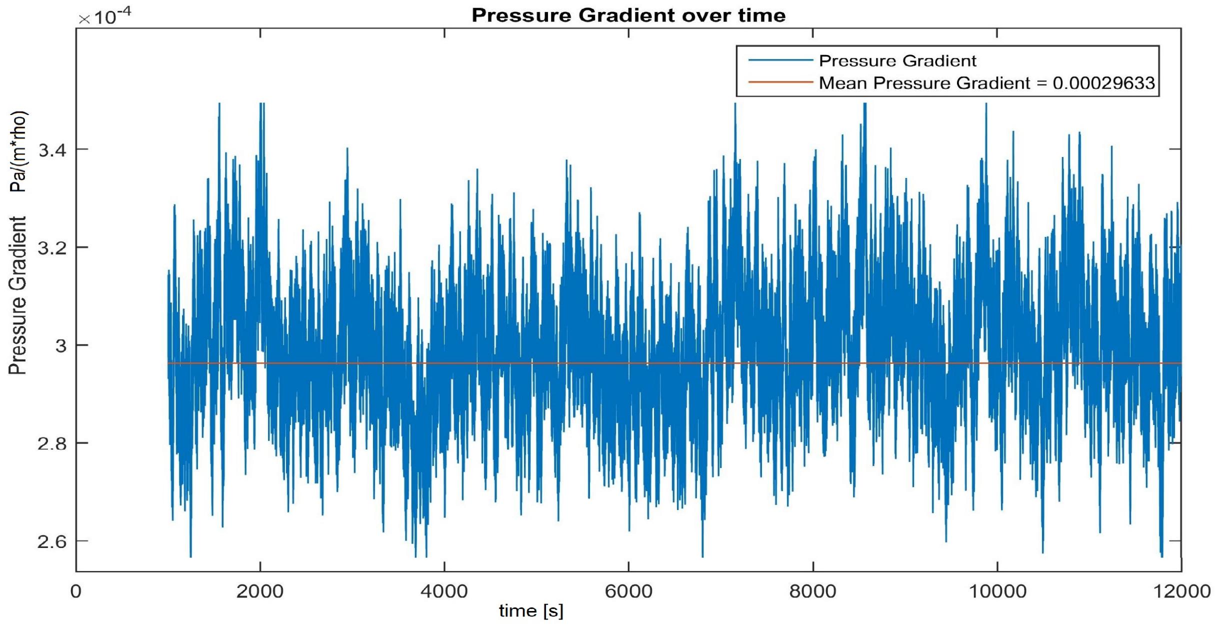


Fig. 3.11: Pressure gradient for rough channel with increased velocity $ks^+ = 21.35$

In Fig. 3.12 the pressure gradient over time for the smooth channel is shown. The first 9500 s are cut away and the simulation ended at 28500 s. The average pressure gradient has a value of $6.5005 * 10^{-5}$. The values vary from $5.457 * 10^{-5}$ to $7.649 * 10^{-5}$

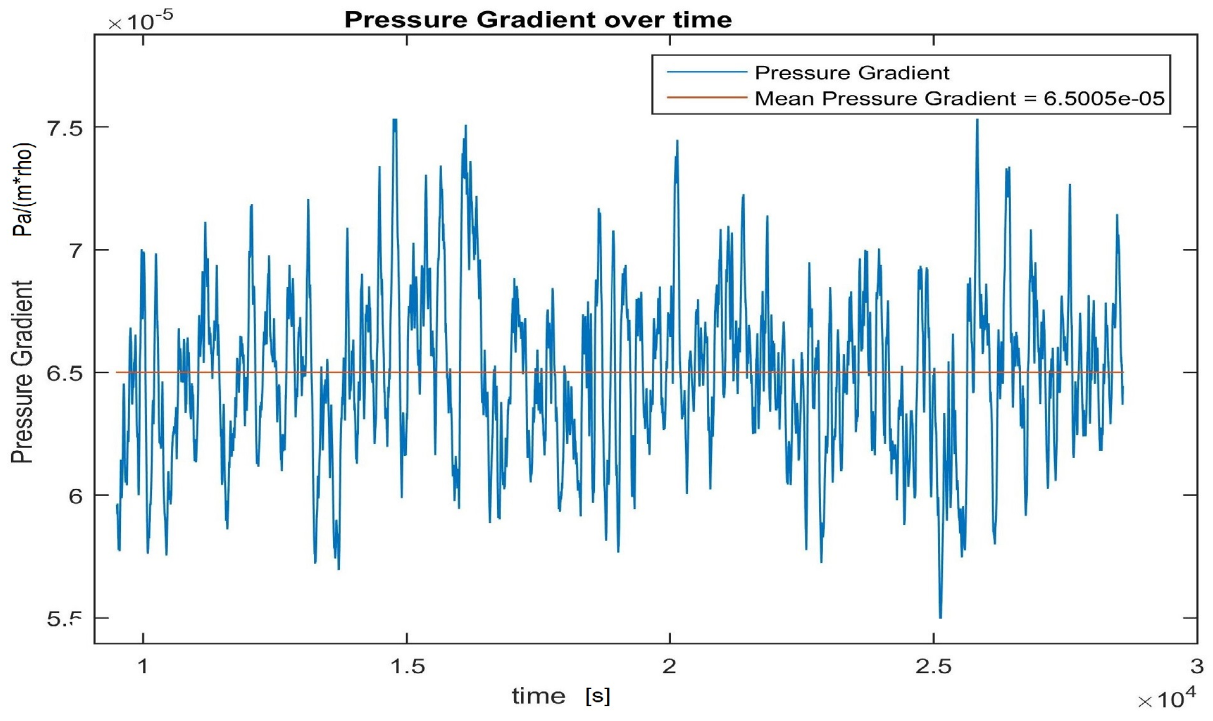


Fig. 3.12: Pressure gradient for smooth channel

For creating the velocity profiles, the smooth channel, rough channel and rough channel velocity increased, the time step $28\,500\,s$, $20\,000\,s$ and $12\,000\,s$ is used. In the file Uf.xy the water depth at the left column and the velocity on the right column is shown. The cells in the x-direction and z-direction are averaged. In y-direction at the first column 74 entries for the smooth and rough channel are shown. For the rough channel with an increased velocity 111 entries are specified. This amount derives by the number of cells in z-direction. It can be shown in the blockMeshDict in Appendix A 6.3 blockMeshDict. The wall shear stress is obtained by the following equation [38]:

$$\text{wall shear stress:} \qquad \tau_0 = \frac{\partial p}{\partial x} * h \tag{3.4}$$

Where $\frac{\partial p}{\partial x}$ is the pressure gradient and h the water depth. For the rough channel velocity plot the height h is subtracted by the min. elevation e which is shown in Section 3.2.1. Therefore the y-coordinate is shifted to the y*-coordinate. The shear velocity is obtained by the following equation [38]:

$$\text{shear velocity:} \qquad u_* = \sqrt{\frac{\tau_0}{\rho}} \tag{3.5}$$

In OpenFOAM the pressure gradient is already divided by the density. For calculating the shear velocity as shown in Eq. 3.5, ρ is omitted. The roughness height is obtained by following equation [38].

$$\text{dimensionless roughness heigth:} \qquad ks^+ = \frac{u_* * k_s}{\nu} \tag{3.6}$$

Where u_* is the shear velocity as in Eq. 3.5, k_s roughness height and ν the kinematic viscosity. The roughness height is chosen to be the characteristic sediment size d_{50}. The kinematic viscosity is defined by $1.9 * 10^{-}5\,\frac{m^2}{s}$ which is shown in Appendix A 6.11 transportProperties. The dimensionless wall distance is defined by following equation [12]:

$$\text{dimensionless wall distance:} \qquad y^+ = \frac{u_* * y}{\nu} \tag{3.7}$$

Where y is the wall distance. The dimensionless velocity is defined by the following equation [11]:

$$\text{dimensionless velocity:} \qquad u^+ = \frac{u}{u_*} \tag{3.8}$$

Where u is the local velocity. To compare the results from the rough channel in OpenFOAM in the log law region, a additional equation for the dimensionless velocity is used.

The logarithmic law of the wall for a rough bed can be written as follows [22]:

$$\text{logarithmic law of the wall rough bed:} \qquad u^+ = \frac{1}{\kappa} * ln\Big(\frac{30 * y}{k_s}\Big) \tag{3.9}$$

Where κ is the von Karman constant which is approximately 0.41. To compare the results from the smooth channel in OpenFOAM in the log law region the following equation is used [41]:

$$\text{logarithmic law of the wall smooth bed:} \qquad u^+ = \frac{1}{\kappa} * ln(y^+) + B \tag{3.10}$$

In equation Eq. 3.10 , the additive constant B is choosen by 5. The velocity profiles are shown in Fig. 3.13 and Fig. 3.14, where the dimensionless velocity profile is plotted on a semilogarithmic graph. The mean velocity is defined by $0.1335\,m/s$. It is shown that the dimensionless velocity profile of the rough channel has a parallel down shift in the outer layer due to the roughness elements. Numerous experimental findings show that the greater the dimensionless roughness height, the bigger the shift [22]. The log law equations Eq. 3.9 and Eq. 3.10 also confirms the log regions.

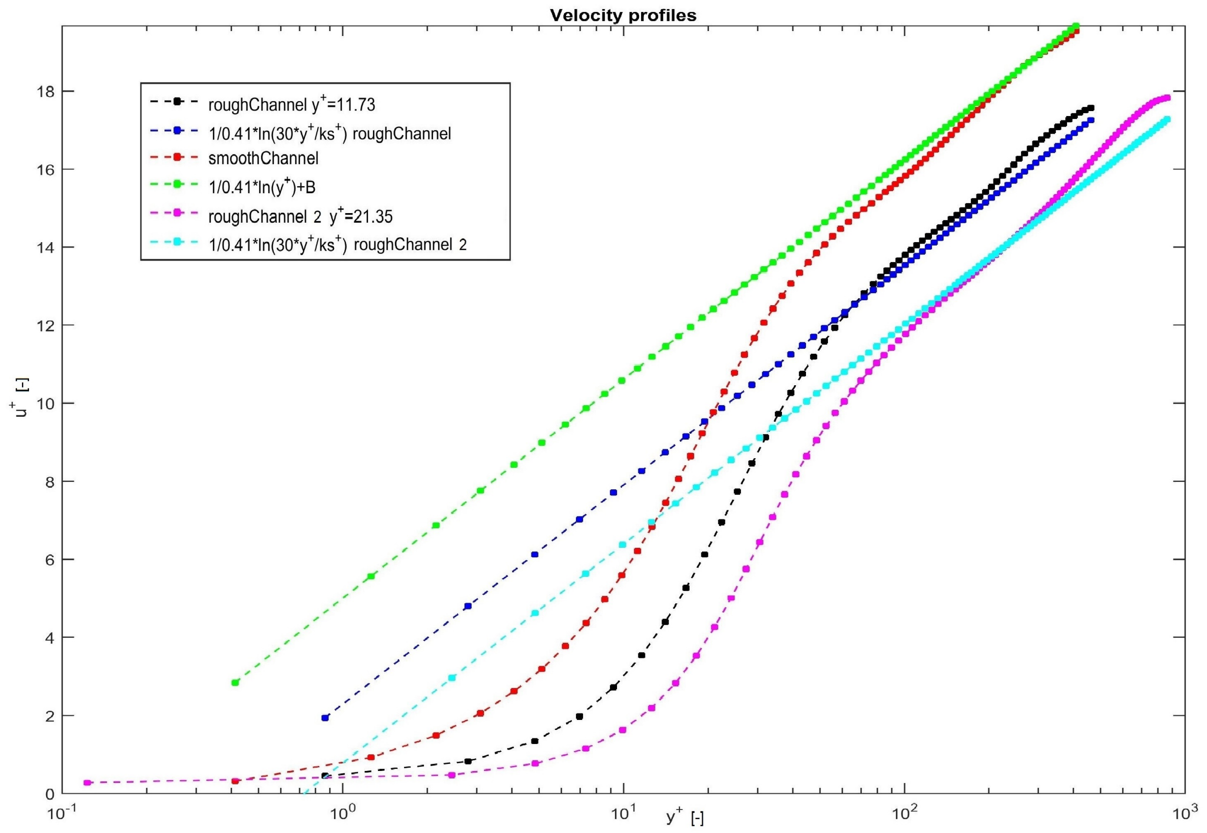


Fig. 3.13: Dimensionless velocity profiles

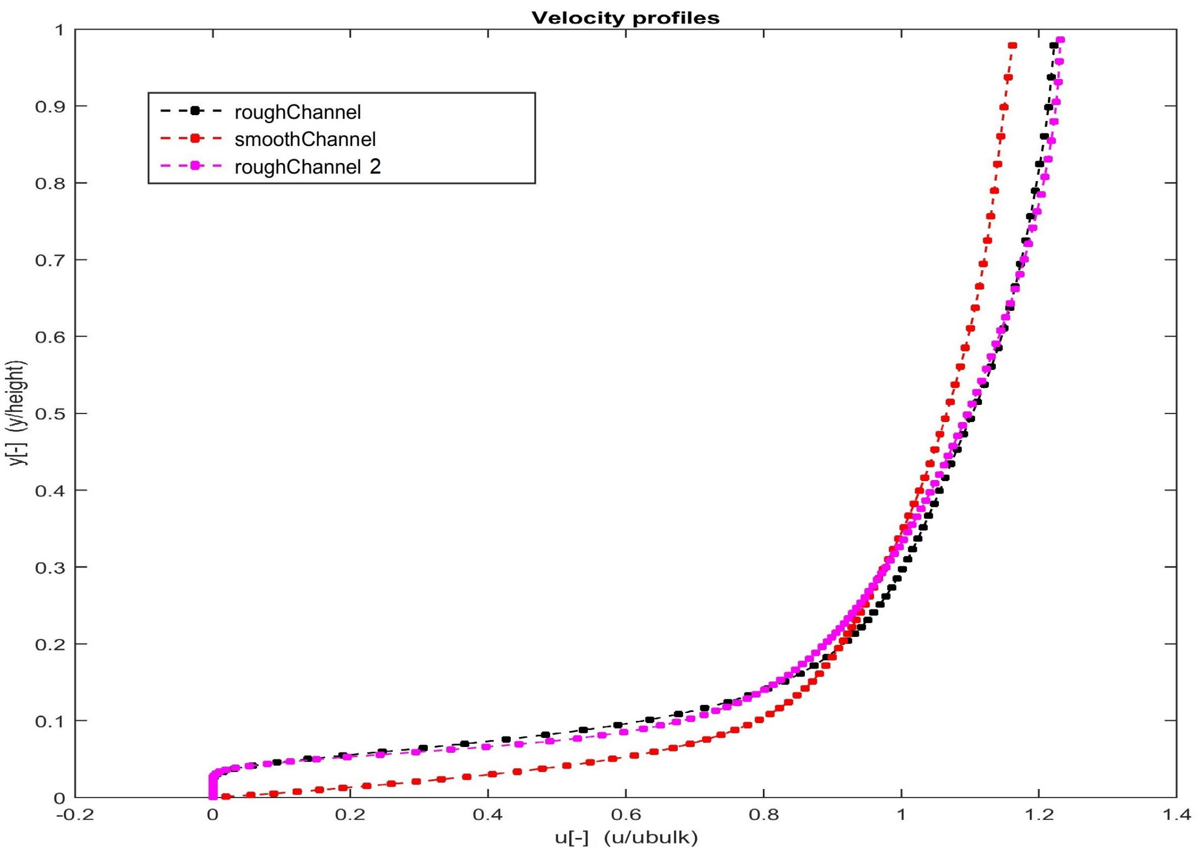


Fig. 3.14: Velocity profiles compared to the bulk velocity

3.4.3 Plots for the rough channel with an increased velocity

In the Fig. 3.6, the velocity Ux,Uy,Uz, magnitude U, UMean and pressure p for the rough channel with an increased velocity of 0.25 $\frac{m}{s}$ is shown.

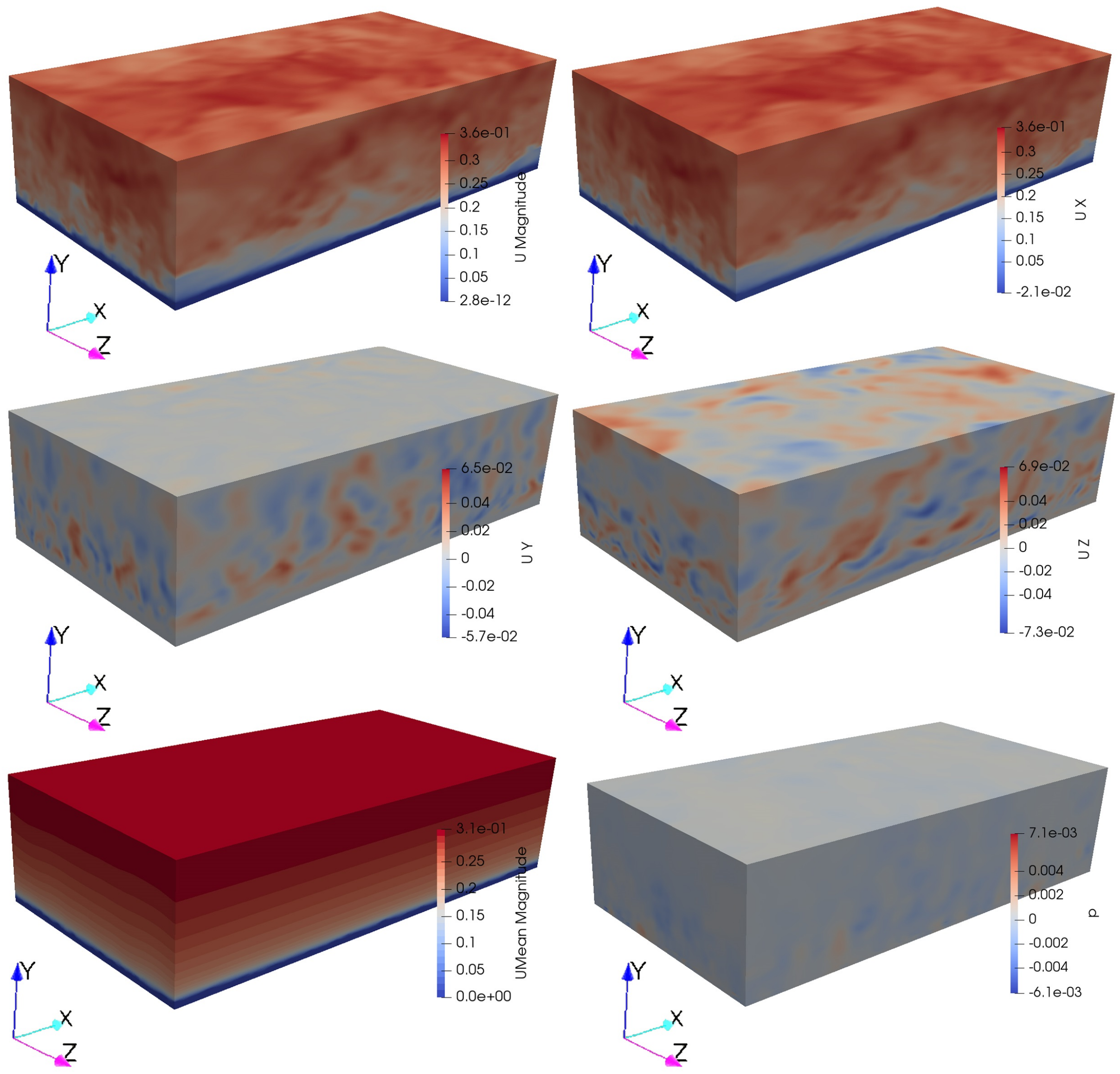


Fig. 3.15: Plots for rough channel with increased velocity

Chapter 4

Trapezoidal Channel

4.1 Introduction

At present the multi-cellular secondary currents in straight shallow rivers is quite unknown. River engineers pointed out that in straight wide rivers the sediment concentration varied periodically in the spanwise direction which is may caused by cellular secondary currents [40]. Matthes [23] suggested that there may exist a strong upward current near the river bed which was observed as a swollen water surface. This upward current developed as a so called boil. Kinoshita [18] found that in a spacing of twice the flow depth h a low speed boil zones were formed. In this region he suggested that two counter rotating streamwise vortices exist. Where the vortex has a diameter of about the water depth h which is shown in Fig. 4.1.

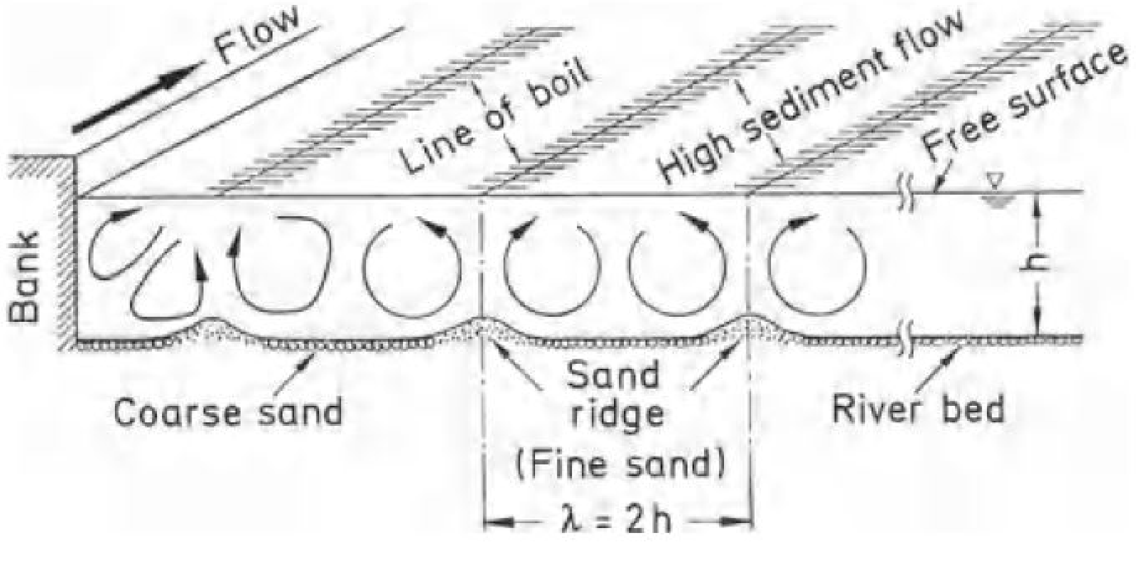


Fig. 4.1: Multi-cellular secondary currents [5]

Karcz [17] and Culbertson [10] found after the flood longitudinal ridges which indicates an existence of cellular secondary currents in a straight river. In hydraulic engineering an investigation of the secondary currents are very important because they form the river beds and might cause three dimensional sediment transport [5]. In a straight open channel the authors (1982 [19] [25]) measured secondary currents with the X-type hot-films (DISA 55R61) In Fig. 4.2 the channel width $B = 30\,cm$, flow depth $h = 4.0\,cm$, bulk velocity $U_m = 32\,\frac{cm}{s}$, $Re = \frac{hU_m}{\nu} = 13\,000$ and $Fr = \frac{U_m}{\sqrt{g\,h}} = 0.51$ is shown. Longitudinal ridge elements in spanwise spacing of $\lambda = 2h = 8\,cm$, $8\,m$ long, $5\,mm$ thick, $20\,mm$ wide and with a 45° trapezoid cross section are attached to the smooth channel bed. They obtained secondary currents experimentally which can be shown through the mean velocities V and W. Two counter rotating streamwise vortices are seen clearly. The circular center is roughly to $\frac{y}{h}$ and the diameter of the vortex motion is equal to h. The velocity of the circular motion is within only 5% of the maximum mainstream velocity [5].

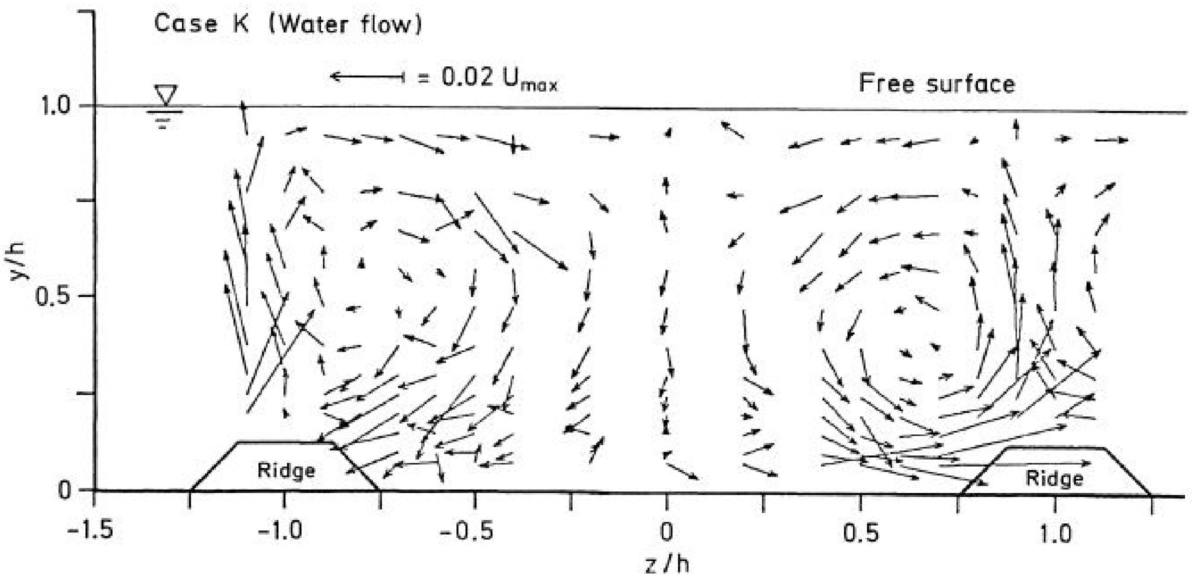


Fig. 4.2: Cellular secondary currents in a straight open channel [5]

Tominaga et al. [39] pointed out the importance of secondary currents in open channels. The vortices affect the primary mean flow, three-dimensional bed formation such as sand ribbons and three dimensional transport. The secondary currents are induced by the anisotropy of turbulence and the velocity is about 2 to 3 % of the primary mean flow. Tominaga et al. [39] conducted all experiments in a tilting flume with a cross section of 40 x 40 cm and length of 12.5 m, see in Fig. 4.3.

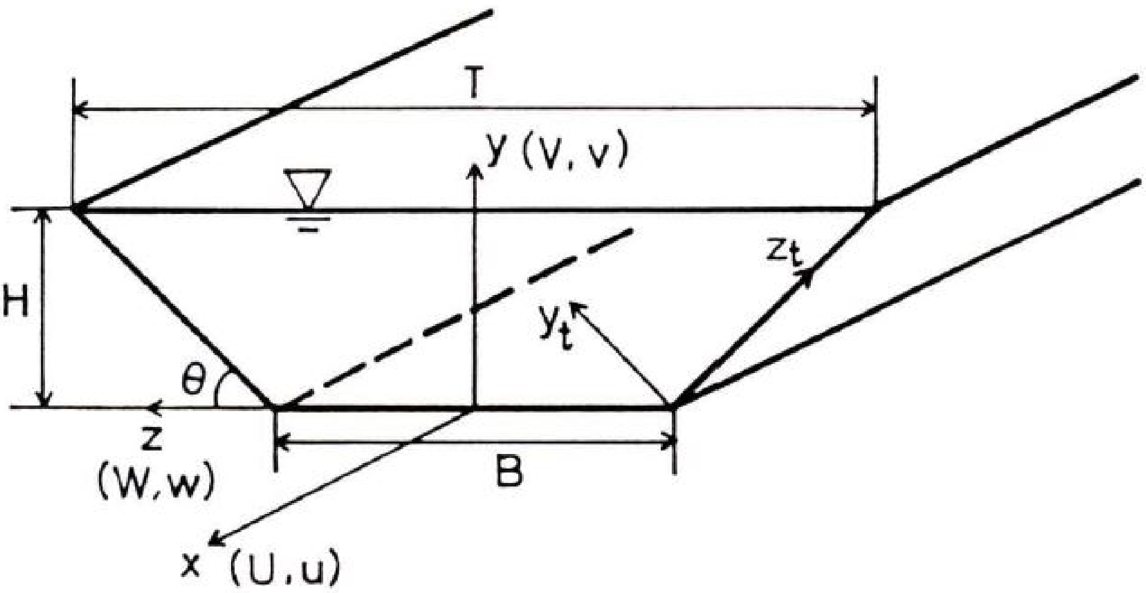


Fig. 4.3: Trapezoidal Channel [39]

The experiments are carried out with the use of a hot-film anemometer. They investigated experimentally and numerically the effects of the channel shape, three-dimensional turbulent structures, boundary roughness and the free surface. The main findings of the turbulent structures are listed as follows:

- In an open channel flow the secondary current structure is quite different from a closed channel flow (top left in Fig. 4.4). In an open channel the structure is composed into a bottom and free surface vortex at about $y/h = 0.6$. Where the top vortex decelerate the primary mean velocity near the surface.

- In a rectangular channel (top right in Fig. 4.4) the secondary current structure is different from the trapezoidal channel (bottom in Fig. 4.10).

- The basic structure of the secondary currents does not change much while varying the boundary roughness, but if the side-wall shear to bottom wall shear ratio increases, the spanwise vortex scale becomes larger. It is possible that multi- cellular secondary currents occur over the rough wall.

- The secondary currents affect the turbulence intensities, mean velocity and boundary shear stress.

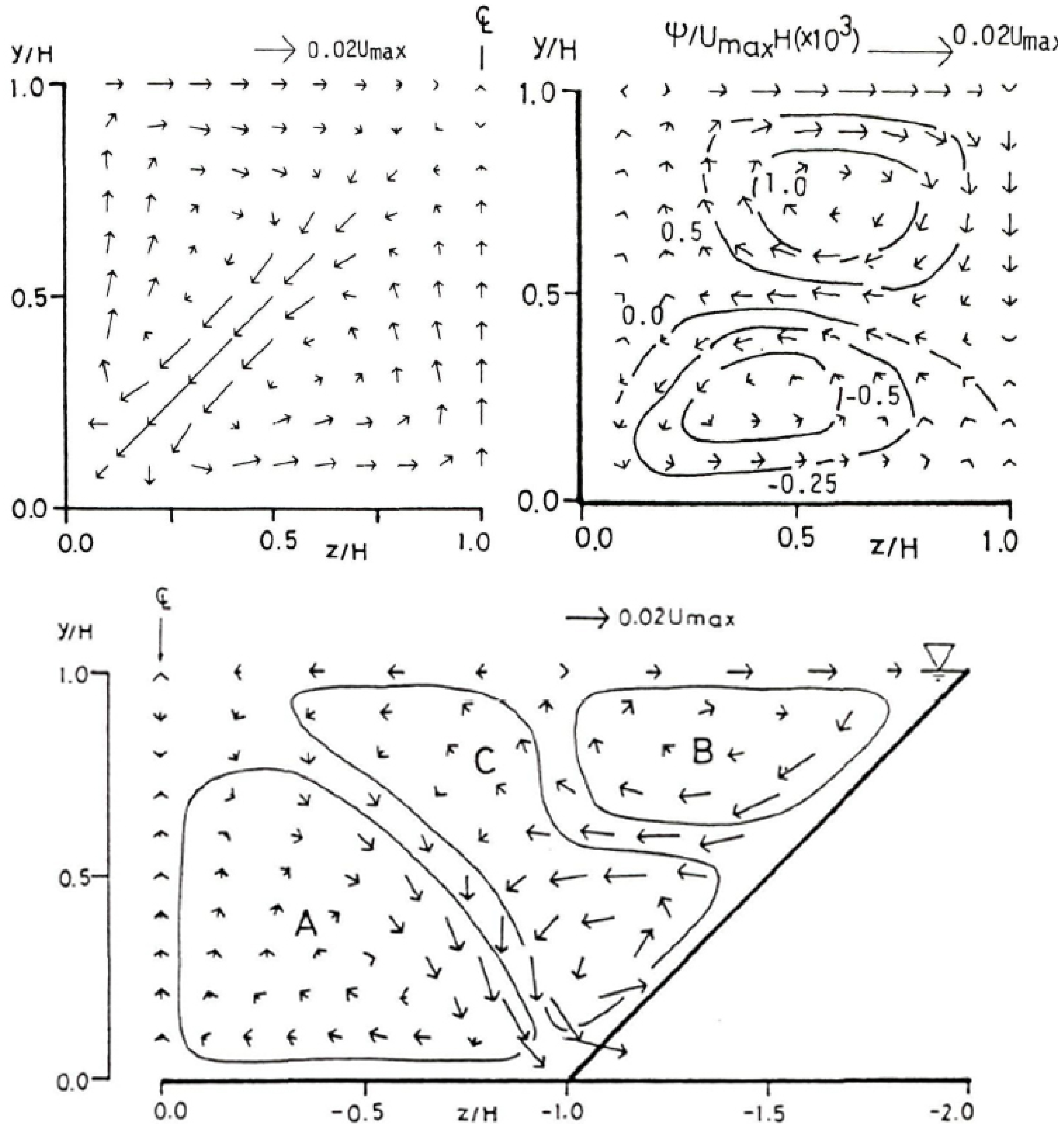


Fig. 4.4: Closed Channel (top left), Open Channel (top right) and Trapezoidal Channel (bottom) [39]

4.2 Rough and Smooth Trapezoidal Channel

For the rough and smooth trapezoidal channel the experimental case "Exp. F16_45_30" by Blanckaert, Duarte, and Schleiss [4], with a decreased velocity is computed. The bulk velocity is reduced from $0.44\,\frac{m}{s}$ to $0.1\,\frac{m}{s}$ because of computational effort which will be discussed in Section 4.2.1.1. They analyzed experiments in straight open channel flows. They investigated a wide range of shallowness, roughness of the bed, banks and bank inclination, as well as combinations of these three parameters. They found secondary currents over the entire width that scale with the flow depth in all experiments. A widespread theory in straight open channel flows is that these currents are generated by the bank and die out at a distance of 2.5 times the flow depth from the bank. These findings contradict to the widespread theory. The secondary currents reduce and the bed shear stress in the vicinity of the bank and cause transverse variability for the streamwise velocity and for the bed shear stress. The resistance of the flow, sediment transport and the conveyance capacity is characterized by the boundary shear stress. For the design of stable channels and mitigation of hazards a accurate prediction of the boundary shear stress is very important. [4]. Detailed information of the geometric and flow conditions of the Experiment is given in Tab. 4.1. Where the bulk velocity U is defined in Eq. 4.1 and the flow discharge is Q, the average width at the bed and water surface is B and H is the water depth.

$$\text{bulk velocity:} \qquad U = \frac{Q}{B\,H} \tag{4.1}$$

The shear velocity is based on the following equation:

$$\text{shear velocity:} \qquad u_* = \sqrt{g\,R_h\,E_s} \tag{4.2}$$

Where R_h is the hydraulic radius defined in Eq. 4.3 and E_s is the energy slope.

$$\text{hydraulic radius:} \qquad R_h = \frac{cross\,sectional\,area}{wetted\,perimeter} \tag{4.3}$$

The Reynolds particle number is based on the following equation:

$$\text{Reynolds particle number:} \qquad Re_* = \frac{u_*\,d}{\nu} \tag{4.4}$$

Tab. 4.1: Detailed information of the hydraulic and geometric conditions of the Exp. F16_45_30 [4]

B	$\frac{B}{H}$	H	$Re = \frac{U\,H}{\nu}$	$Fr = \frac{U}{\sqrt{g\,H}}$
$1.22\,[m]$	$7.6\,[-]$	$0.16\,[m]$	$70\,[10^3]$	$0.35\,[-]$
ν	d_{bank}	d_{bed}	U	θ_{bank}
$1\,[10^{-6}\frac{m^2}{s}]$	$30\,[mm]$	$2\,[mm]$	$0.44\,[\frac{m}{s}]$	$45\,[^\circ]$
E_s	$\frac{d_{bed}}{H}$	$\frac{d_{bank}}{H}$	$Re_* = \frac{u_* d_{bed}}{\nu}$	$u_* = \sqrt{g\,R_h\,E_s}$
$8.7\,[10^{-4}]$	$0.013\,[-]$	$0.188\,[-]$	$66\,[-]$	$0.033\,[\frac{m}{s}]$
$Re_\tau = \frac{u_* H}{\nu}$	R_h	Q	$perimeter$	$cross\,section$
$5.3\,[10^3]$	$0.128\,[m]$	$8.6\,[\frac{l}{s}]$	$1.53\,[m]$	$0.195\,[m^2]$
$\tau_0 = u_*^2\,\rho$				
$1.09\,\frac{N}{m^2}$				

4.2.1 Computation requirement for trapezoidal channel

4.2.1.1 Experiment F16_45_30

The Reynolds number in the experiment F16_45_30 is about 70 000 which is way too high for simulating this case in LES with an average computer. Chapmen (1979) [7], Georgiadis, Rizzetta, and Fureby [15] and other authors found out that $\Delta x^+ \approx 50 - 150$ (streamwise), $\Delta y^+ \approx 15 - 40$ (spanwise) and $\Delta z_1^+ \approx 1$ (wall-normal) is necessary as a minimum resolution for flow over a flat plate [34]. The numerical mesh cells near the wall is shown in Fig. 4.5. In Tab. 4.1 the friction velocity is $0.033\,\frac{m}{s}$ and the kinematic viscosity is $10^{-6}\,\frac{m^2}{s}$. In the following equations $\Delta x\,(\,\Delta x^+ = 150)$, $\Delta y\,(\,\Delta y^+ = 40)$ and $\Delta z_1\,(\,\Delta z_1^+ = 1)$ are calculated.

$$\text{minimum grid size required for } \Delta \text{ x:} \qquad \Delta x^+ = \frac{\Delta x\,u_*}{\nu} \rightarrow \Delta x \approx 4.54 * 10^{-3}\,m \qquad (4.5)$$

$$\text{minimum grid size required for } \Delta \text{ y:} \qquad \Delta y^+ = \frac{\Delta y\,u_*}{\nu} \rightarrow \Delta y \approx 1.21 * 10^{-3}\,m \qquad (4.6)$$

$$\text{minimum grid size required for } \Delta z_1\text{:} \quad \Delta z_1^+ = \frac{\Delta z_1 \, u_*}{\nu} \rightarrow \Delta z_1 \approx 3.03 * 10^{-5} \, m \tag{4.7}$$

For a total expansion ratio of 8 and height of $0.16\,m$ the number of cells in z-direction is 416 which is shown in Fig. 4.6. For the streamwise direction at least 440 ($2/4.54 * 10^{-3}$) cells and for spanwise at least 1072 ($1.30/1.21 * 10^{-3}$) cells are required. This give a number of $196 * 10^6$ cells.

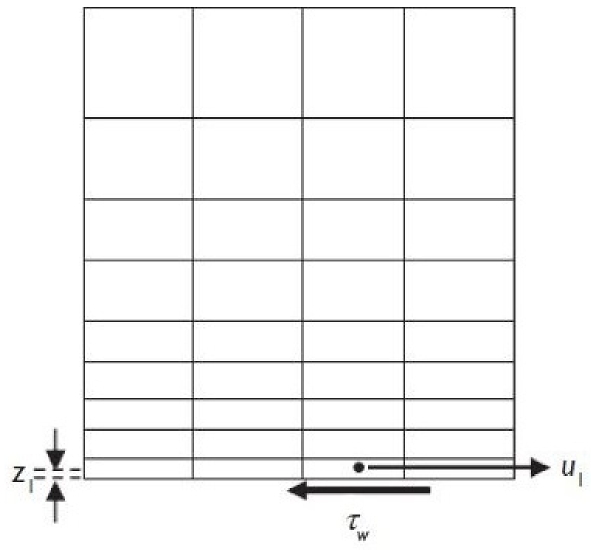
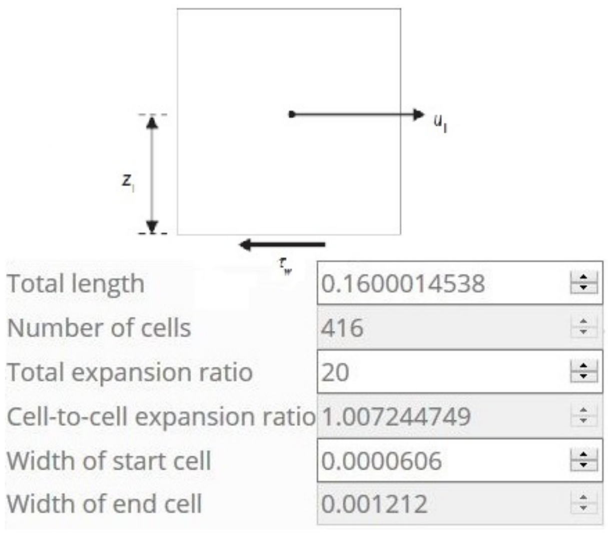


Fig. 4.5: grid size for wall resolving LES [34]



Total length	0.1600014538
Number of cells	416
Total expansion ratio	20
Cell-to-cell expansion ratio	1.007244749
Width of start cell	0.0000606
Width of end cell	0.001212

Fig. 4.6: blockMesh grading calculation [36]

4.2.1.2 Experiment F16_45_30 decreased velocity

Where this thesis broaden the experimental case Exp. $F16\_45\_30$ with a decreased velocity. The computational domain is carried out in a $1.3\,m$ wide flume with $0.16\,m$ flow depth, a $2\,m$ long straight reach and bulk velocity of $0.1\,\frac{m}{s}$ which is shown in Fig. 4.7. Detailed information of the geometric flow conditions of the reduced velocity is shown in Tab. 4.2. The friction velocity for the decreased computational case is a priory not known. Therefore the mesh size has to be assumed at first and then the friction velocity can be calculated. Then the mesh size is obtained by Eq. 4.10 to Eq. 4.12. This is a iterative process. However the final wall shear stress and friction velocity after the procedure is shown in Tab. 4.2 which are defined by following equations:

$$\text{wall shear stress:} \qquad \tau_0 = \mu \left(\frac{\partial u}{\partial y} \right) \tag{4.8}$$

$$\text{friction velocity:} \qquad u_* = \sqrt{\frac{\tau_0}{\rho}} = \sqrt{\nu \left(\frac{\partial u}{\partial y} \right)} \tag{4.9}$$

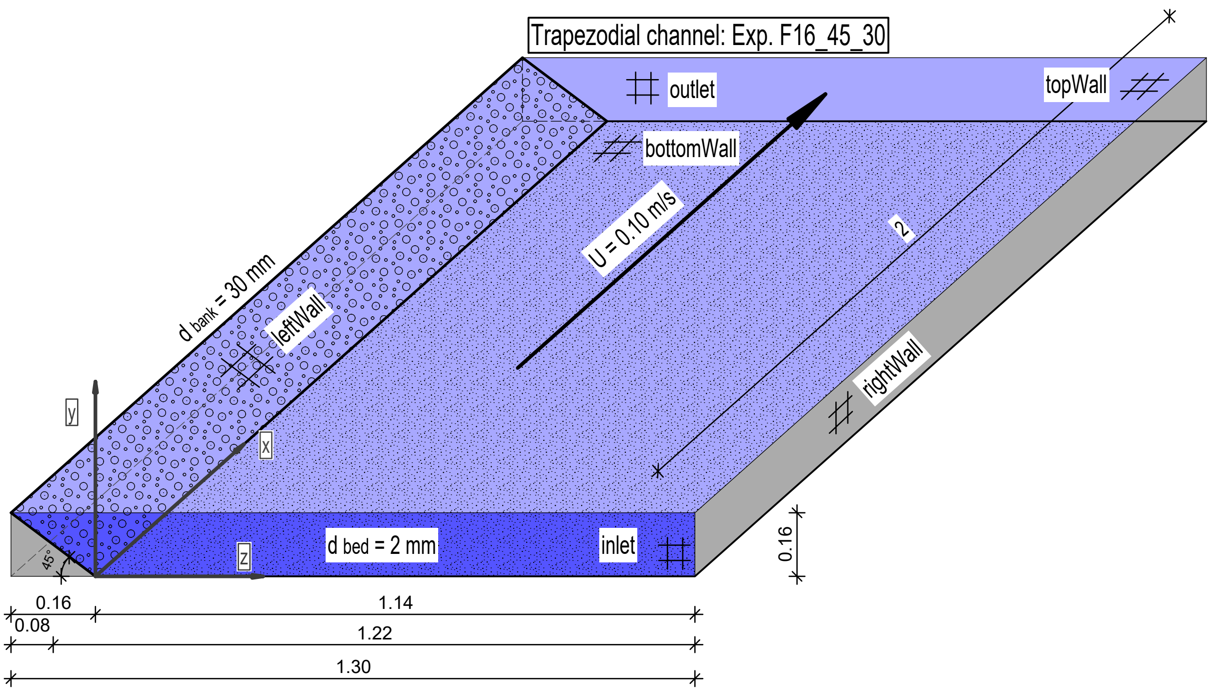


Fig. 4.7: Trapezodial channel

Tab. 4.2: Detailed information of the Exp. F16_45_30 with a decreased velocity [4]

B	$\frac{B}{H}$	H	$Re = \frac{U\,H}{\nu}$	$Fr = \frac{U}{\sqrt{g\,H}}$
$1.22\,[m]$	$7.6\,[-]$	$0.16\,[m]$	$16\,[10^3]$	$0.08\,[-]$
ν	d_{bank}	d_{bed}	U	θ_{bank}
$1\,[10^{-6}\frac{m^2}{s}]$	$30\,[mm]$	$2\,[mm]$	$0.1\,[\frac{m}{s}]$	$45\,[^\circ]$
$E_s = \frac{u_*^2}{R_h\,g}$	$\frac{d_{bed}}{H}$	$\frac{d_{bank}}{H}$	$Re_* = \frac{u_* d_{bed}}{\nu}$	$u_* = \sqrt{\nu\left(\frac{\partial u}{\partial y}\right)}$
$1.1\,[10^{-5}]$	$0.013\,[-]$	$0.188\,[-]$	$7\,[-]$	$0.0037\,[\frac{m}{s}]$
$Re_\tau = \frac{u_* H}{\nu}$	R_h	Q	$perimeter$	$cross\ section$
590	$0.128\,[m]$	$8.6\,[\frac{l}{s}]$	$1.53\,[m]$	$0.195\,[m^2]$
$\tau_0 = u_*^2\,\rho$				
$0.0136\,\frac{N}{m^2}$				

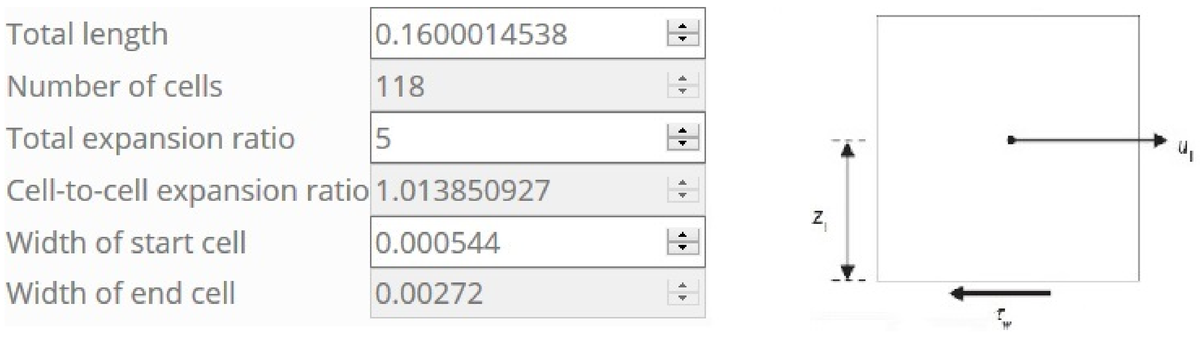
In Tab. 4.2 the friction velocity is $0.0037\,\frac{m}{s}$ and the kinematic viscosity is $10^{-6}\,\frac{m^2}{s}$. In the following equations $\Delta x\,(\,\Delta x^+ = 70)$, $\Delta y\,(\,\Delta y^+ = 30)$ and $\Delta z_1\,(\,\Delta z_1^+ = 1)$ are calculated.

$$\text{minimum grid size required for } \Delta \text{ x:} \quad \Delta x^+ = \frac{\Delta x\, u_*}{\nu} \rightarrow \Delta x \approx 1.89 * 10^{-2}\, m \tag{4.10}$$

$$\text{minimum grid size required for } \Delta \text{ y:} \quad \Delta y^+ = \frac{\Delta y\, u_*}{\nu} \rightarrow \Delta y \approx 8.11 * 10^{-3}\, m \tag{4.11}$$

$$\text{minimum grid size required for } \Delta z_1\text{:} \quad \Delta z_1^+ = \frac{\Delta z_1\, u_*}{\nu} \rightarrow \Delta z_1 \approx 2.72 * 10^{-4}\, m \tag{4.12}$$

For a simple grading of 5 and height of $0.16\,m$ the number of cells in z-direction is 118 which is shown in Fig. 4.8. For x-direction at least 105 ($2/1.89 * 10^{-2}$) cells and for y-direction at least 160 ($1.30/8.11 * 10^{-3}$) cells are required. This give a number of $2 * 10^6$ cells.



Total length	0.1600014538
Number of cells	118
Total expansion ratio	5
Cell-to-cell expansion ratio	1.013850927
Width of start cell	0.000544
Width of end cell	0.00272

Fig. 4.8: blockMesh grading calculation

4.3 Pre-Processing

4.3.1 Geometry of the Domain

The blockMeshDict entries for this case are shown in Appendix B 7.1 blockMeshDict. The laboratory experiment by Blanckaert, Duarte, and Schleiss [4] is carried out on a $9\,m$ long straight reach. Where the trapezoidal channel in OpenFOAM considered a length of $L = 2.00\,m$ to save computation power. The trapezoid is separated into seven hexahedral blocks where the length and height is specified as a variable. The block structure is shown in Fig. 4.9. The left wedge is separated into 3 smaller blocks for more accuracy and quality of the mesh. If the skewness ratio is too large a further refinement is required. This case has a skewness ratio of 2.49648. With the following command the mesh can be checked:

```
checkMesh
```

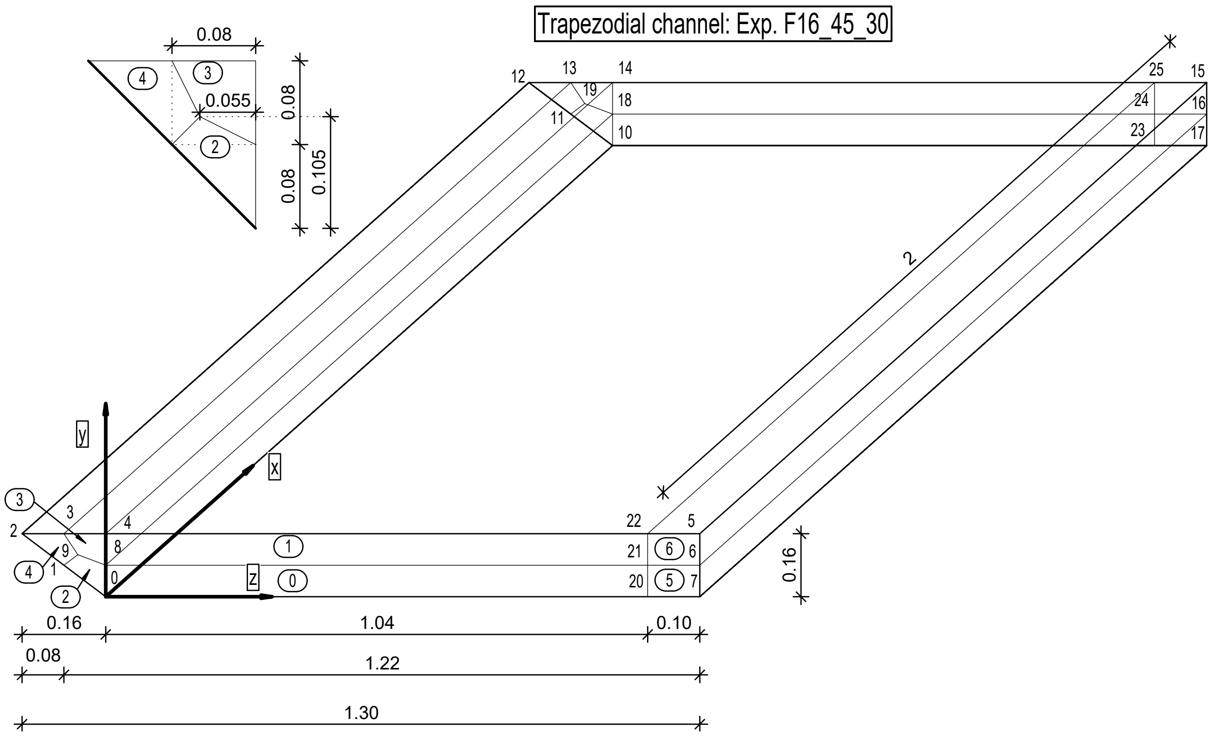


Fig. 4.9: Visualization of the trapezoidal channel geometry

The whole domain consists of 3 039 525 points and 2 967 296 cells. A non uniform mesh is used for the blocks near the bed and bank. The grid structure is shown in Fig. 4.10. The domain has 6 boundaries which are named as bottomWall, topWall, leftWall, rightWall, inlet and outlet. the bottomWall, leftWall and rightWall is defined by type wall which represent a solid wall. The topWall is defined by type patch which contains no geometric or topological information. The inlet and outlet are defined by type cyclic. This type enables two boundaries to be treated as if they are physically connected. With executing the blockMesh command the blocks and numbers of cells withing it are generated.

```
blockMesh
```

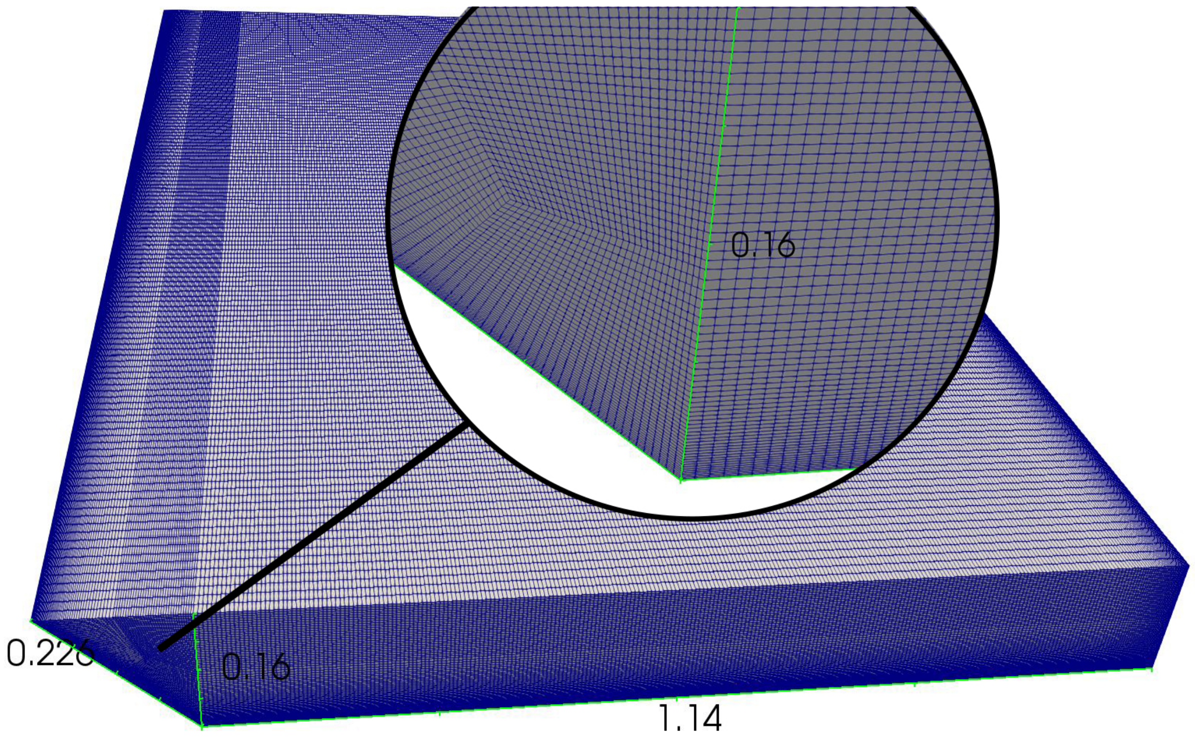


Fig. 4.10: Visualization of the mesh

4.3.2 Rough Bed generation

For creating the roughness elements on the bank and on the bed the file boxes.C in Appendix A 6.1 Creation of the roughness elements, boxes.C has been modified. The modification for the bank and bed is shown in Appendix B 7.3 bed.C and Appendix B 7.2 bank.C. The characteristic sediment size d_{50} is changed to $30\,mm$ for the bank and $2\,mm$ for the bed. Additionally the streamwise and spanwise length is changed to $2\,m$ and $1.14\,m$ for the bed and $2\,m$ and $0.227\,m$ for the bank respectively. In Fig. 4.11 on the right hand side the model for creating the roughness elements for the bed and bank is shown. The height for the blocks varies at max. 3 times sigma around the characteristic sediment size. For the bank and bed sigma is estimated to $0.1 * d_{50}$.

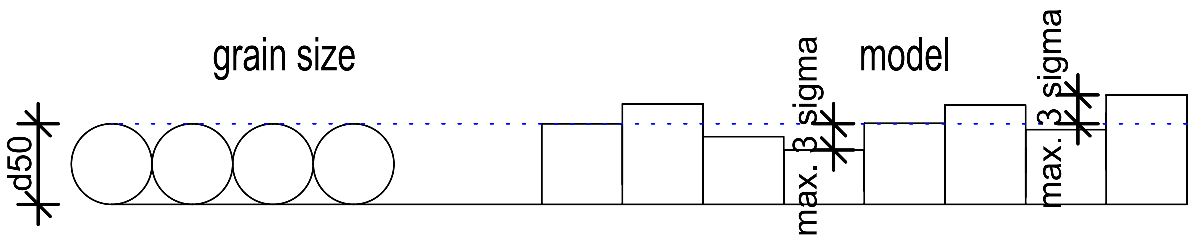


Fig. 4.11: d_{50} for bed and bank roughness

For creating the file bank.txt and bank.txt the following commands should be executed:

```
g++ -std=c++11 bed.C -o bed
g++ -std=c++11 bank.C -o bank
./bed > log.bed
./bank > log.bank
```

This will create the coordinates for the roughness elements. Where the first coordinate corresponds the left bottom corner and the second coordinate correspond to the right top corner of a single box. For the file bed.txt the first three elements are shown as follows:

```
(0 0 0)(0.01 0.00192682 0.01)
(0 0 0.01)(0.01 0.00134791 0.02)
(0 0 0.02)(0.01 0.00241057 0.03)
```

For the file bank.txt the first three elements are shown as follows:

```
(0 0 0)(0.03 0.0289023 0.03)
(0 0 0.03)(0.03 0.0202186 0.06)
(0 0 0.06)(0.03 0.0361586 0.09)
```

To create the roughness elements in OpenFOAM, the topoSet utility is used. In Appendix B 7.4 topoSetBank, Appendix B 7.5 topoSetDelete and Appendix B 7.6 topoSetBed the topoSet dictionaries are shown. It is important that these roughness blocks are defined as cellZones for the fvOptions. For this case at first a bank_set and bed_set and then the bank_zone and bed_zone are generated. To create the bank roughness, the domain has to be rotated and translated at first. This procedure can be done through the following command:

```
transformPoints -translate '(0 -0.16 0.16)'
transformPoints -rotate '((0 1 0) (0 1 -1))'
```

After executing these commands the domain should look like in Fig. 4.12 After the rotation and translation the file topoSetBank is renamed to topoSet. Then the topoSet utility can be executed for the bank as follows:

```
topoSet
```

This will create a bank_set and bank_zone as shown in Fig. 4.13 To get rid of the blocks which are generated on the right hand side in Fig. 4.13, the topoSetDelete utility is used which has to be renamed to topoSet to get use of it again. Then thee domain is rotated and translated back to its origin as follows:

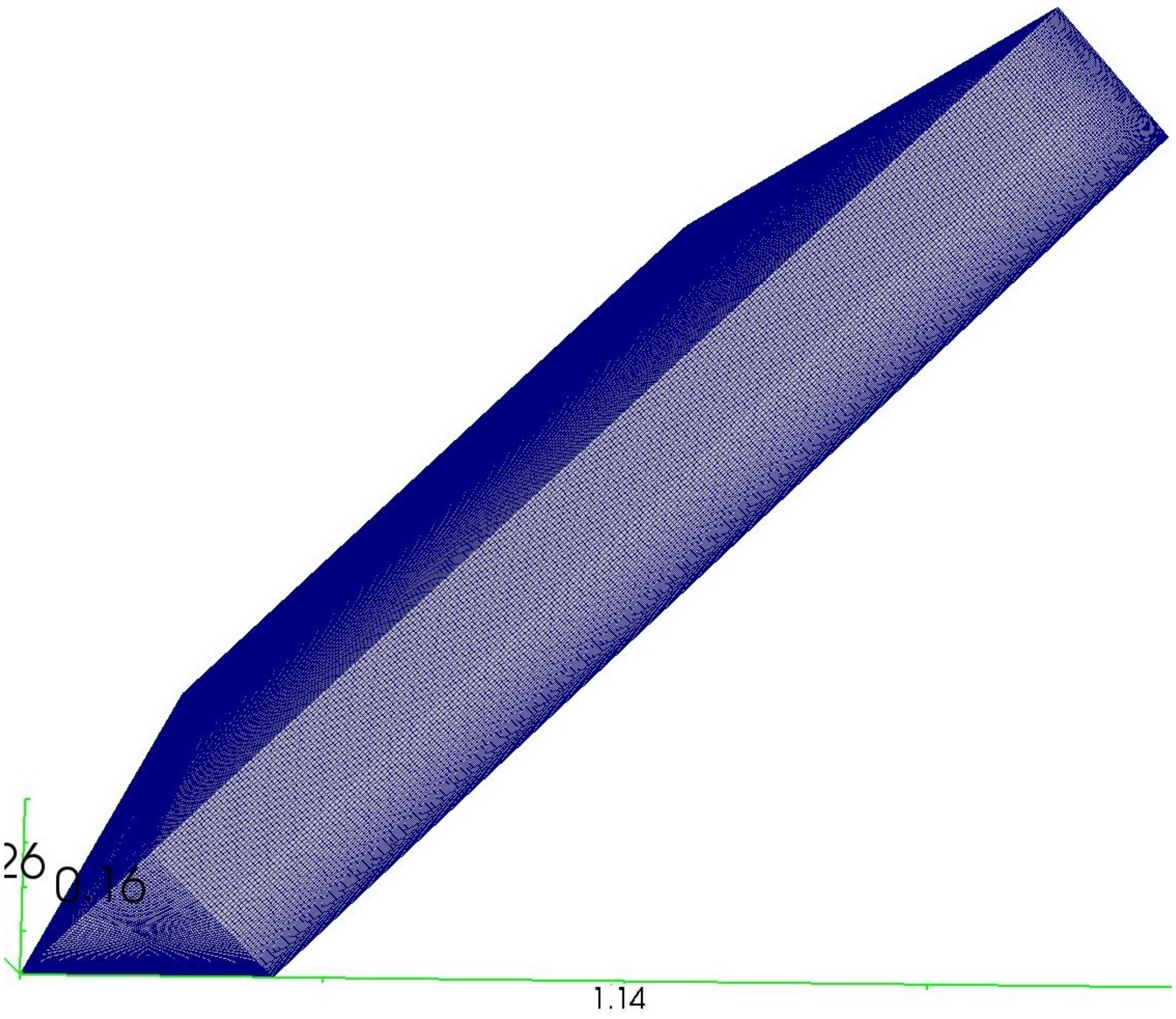


Fig. 4.12: translated and rotated domain

```
transformPoints -rotate '((0 1 0) (0 1 1))'
transformPoints -translate '(0 0.16 -0.16)'
topoSet
```

The result is shown in Fig. 4.14. The last step is to generate the bed roughness blocks with the topoSet utility again. The final result is shown in Fig. 4.15.

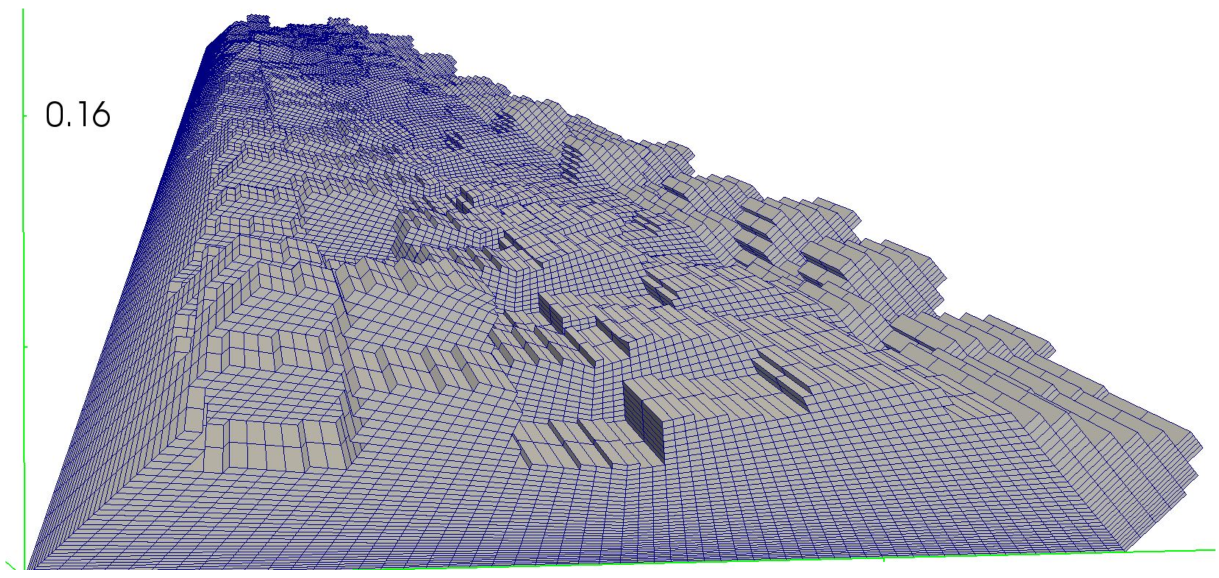


Fig. 4.13: bank roughness elements

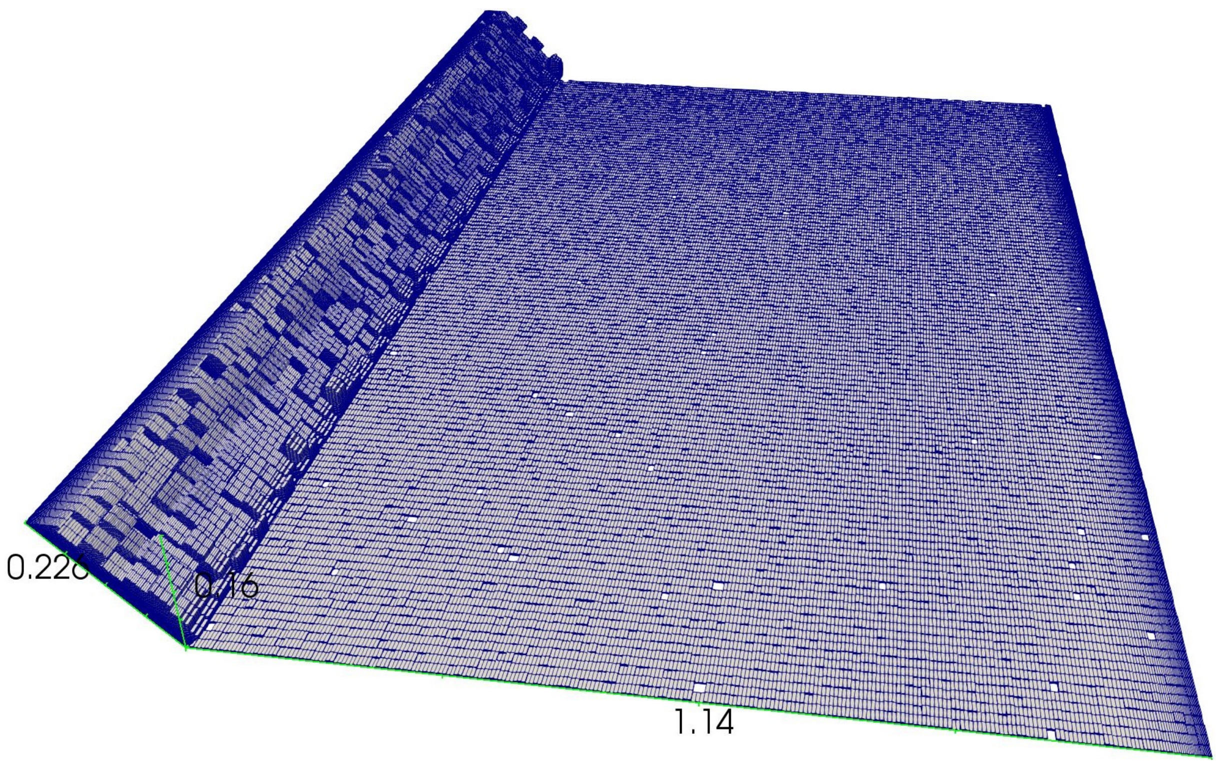


Fig. 4.15: bed and bank roughness

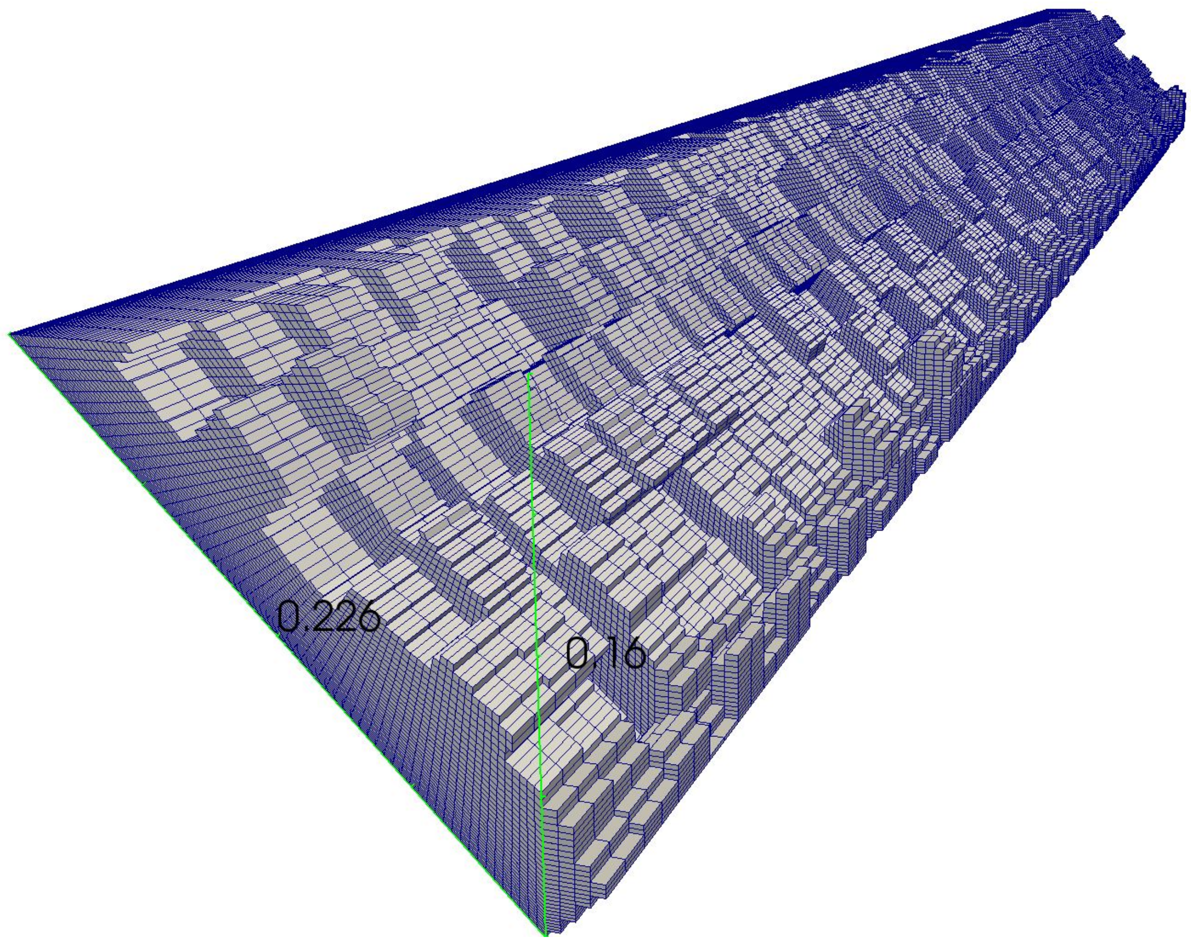


Fig. 4.14: back- rotated and -translated domain

4.3.3 Boundary and Initial Conditions

4.3.3.1 Velocity, U

Once the mesh generation is done the initial and boundary conditions are generated. The velocity U is set up in Appendix B 7.7 Velocity, U. The internal field is specified to be (0.1 0 0). The velocity values at time $0\,s$ for the rough trapezoidal channel will be mapped from the smooth trapezoidal channel similar as in Section 3.2.3.5. Additionally the velocity values at the bank and bed elements will be set to zero to get better initial boundary conditions for the computation. Otherwise the simulation diverges. To set the values at a certain position to zero, the setFields utility is used which is shown in Appendix B 7.8 setFieldsDict. To set the bank roughness elements to zero the translation and rotation utility is used again as follows:

```
transformPoints -translate '(0 -0.16 0.16)'
transformPoints -rotate '( (0 1 0) (0 1 -1) )'
```

Then the setFields utility can be used as follows:

```
setFields
```

To set the bed roughness elements to zero the domain is rotated and translated back to its origin and the setFields utility is used again.

```
transformPoints -rotate '( (0 1 0) (0 1 1) )'
transformPoints -translate '(0 0.16 -0.16)
```

The result is shown in Fig. 4.16.

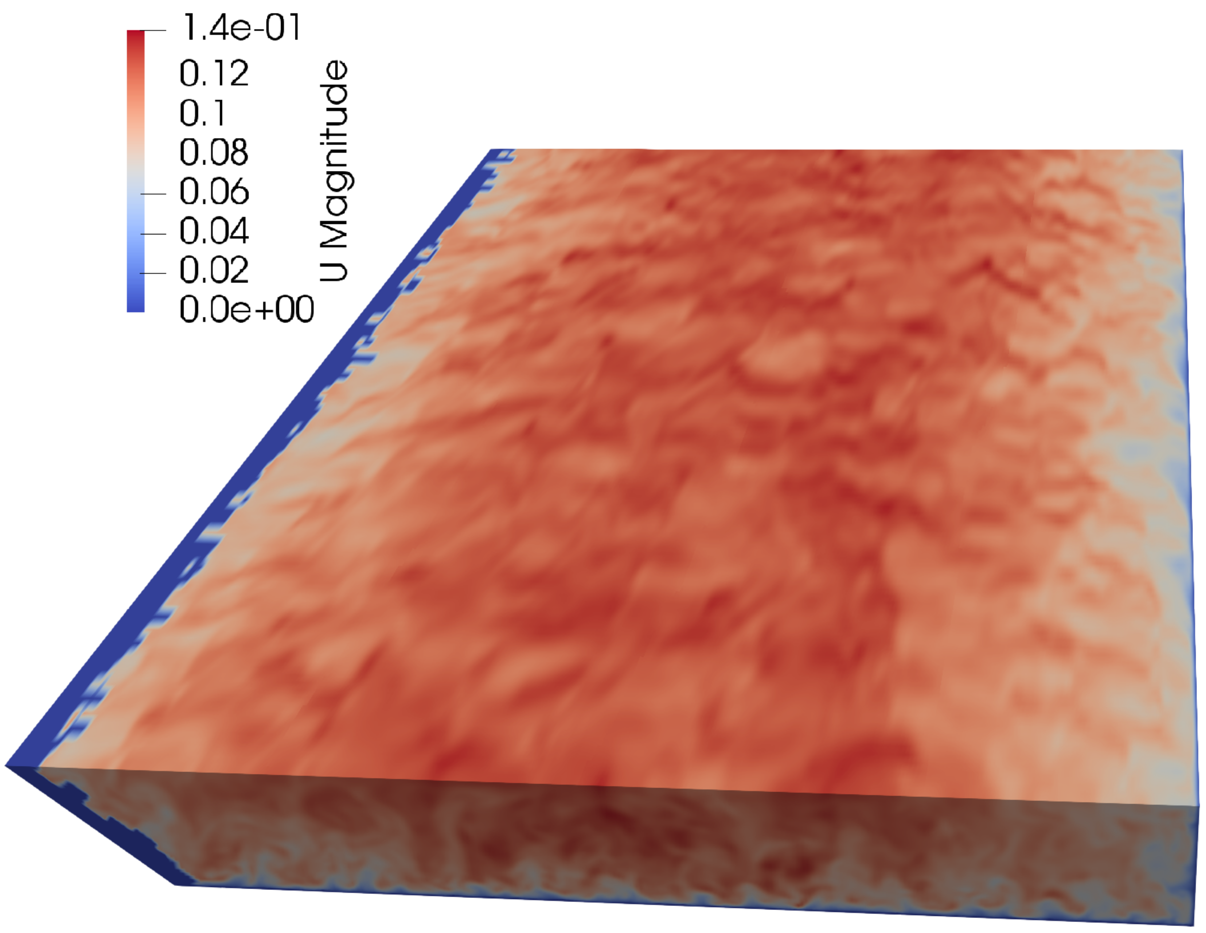


Fig. 4.16: SetFieldsUtility for bank and bed elements

The bottom wall, left wall and right wall are set up as a noSlip boundary condition which restricts the velocity as a fixed value of (0 0 0) at this patch. At the top wall the velocity is defined as a slip boundary condition. The inlet and outlet patches have a cyclic boundary condition. The velocity at the outlet is physically connected to the inlet.

4.3.3.2 Kinematic Pressure, p

The kinematic pressure p in $\frac{m^2}{s^2}$ is set up in Appendix B 7.9 Pressure, P. The internal field is specified to be 0. The pressure quantities for the rough trapezoidal channel will be mapped similar as in Section 3.2.3.5.The bottom, top, left and right wall consists of a zeroGradient boundary condition. At this patch the normal gradient p is zero over time. The inlet and outlet have a cyclic boundary condition. The pressure at the outlet is physically connected to the inlet.

4.3.3.3 turbulent kinetic energy, k

The turbulent kinetic energy k in $\frac{m^2}{s^2}$ is set up in Appendix B 7.10 Turbulent Kinetic Energy, k. The internal field is specified to be 0. The turbulent kinetic energy quantities for the rough trapezoidal channel will be mapped similar as in Section 3.2.3.5. The values at the bottom, left and right wall are defined to be zero. The top wall consists of a zeroGradient boundary condition. At this patch the normal gradient p is zero over time. The inlet and outlet have a cyclic boundary condition. The turbulent kinetic energy at the outlet is physically connected to the inlet.

4.3.3.4 turbulence viscosity, nut

The turbulence viscosity nut in $\frac{m^2}{s}$ is set up in Appendix B 7.11 Turbulence viscosity, nut. The internal field is specified to be 0 initially. The turbulent viscosity quantities for the rough trapezoidal channel will be mapped similar as in Section 3.2.3.5. The bottom, top, left and right wall has a zeroGradient boundary condition which means that the normal gradient from nut is zero over time. The inlet and outlet have a cyclic boundary condition. The turbulence viscosity at the outlet is connected to the inlet.

4.3.4 Options

4.3.4.1 fvOptions

With the file fvOptions any physics can be represented as sources or constraints on the governing equations e.g. porous media MRF (multiple reference frame) and body forces [31]. In this case the rough bed is given a porosity and the the flow is driven by a pressure gradient which is shown in Appendix B 7.12 fvOptions. For this case the mean velocity is defined as 0.1 $\frac{m}{s}$. The porous media is is given by very high values that the cells become practically impermeable, where d is the Darcy coefficient in $[\frac{1}{m^2}]$ and f is the Forchheimer coefficient in $\frac{1}{m}$ [30]. This components are specified by a coordinate system where e1 and e2 sets the local orientation of the coefficients.

4.3.4.2 transportProperties

The kinematic viscosity is specified by $1.0 * 10^{-6} \frac{m^2}{s}$ and the average velocity is defined by $0.1 \frac{m}{s}$ which is shown in Appendix B 7.13 transportProperties.

4.3.4.3 turbulenceProperties

The smooth and rough trapezoidal channel uses the same turbulence properties as in Section 3.2.4.3

4.3.4.4 fvSchemes (Numerical Schemes)

The smooth and rough trapezoidal channel uses the same numerical schemes as in Section 3.2.4.4

4.3.4.5 fvSolution

In the fvSolution dictionary the numerical solver and tolerances of each field can be specified, where this file is uses almost the same settings as in Section 3.2.4.5. The difference is shown in Appendix B 7.14 fvSolution. For the PIMPLE solver it is necessary to specify a reference point for the pressure. The coordinates for this point is at (2 0.16 0.5) specified which is at the top of the channel. For more details of the solution and algorithm control see in [14].

4.3.4.6 controlDict

In the file controlDict the configuration of the simulation can be specified which is similar to Section 3.2.4.6. the difference is shown in Appendix B 7.14 fvSolution.

4.3.4.7 decomposeParDict

The smooth and rough trapezoidal channel is decomposed as in Section 3.2.4.7. Only the method is changed to scotch.

4.4 Solver

To solve the momentum and continuity equations the pimpleFoam solver is used. It is a incompressible transient solver for turbulent flow of Newtonian fluids, with optional mesh motion and mesh topology changes [9].

4.5 Results

In this section the results in a trapezoidal channel from Blanckaert, Duarte, and Schleiss [4] and Tominaga et al. [39] are compared. In hydraulic engineering it is very important to investigate the multi-cellular currents in an open channel because the rotating vortices's affect the three dimensional sediment transport, the three dimensional bed configurations such as sand ribbons and the primary mean flow [39]. With an Acoustic Doppler Velocity Profiler (ADVP) Blanckaert, Duarte, and Schleiss [4] enabled accurately the patterns of the weak secondary currents as shown in Fig. 4.17.

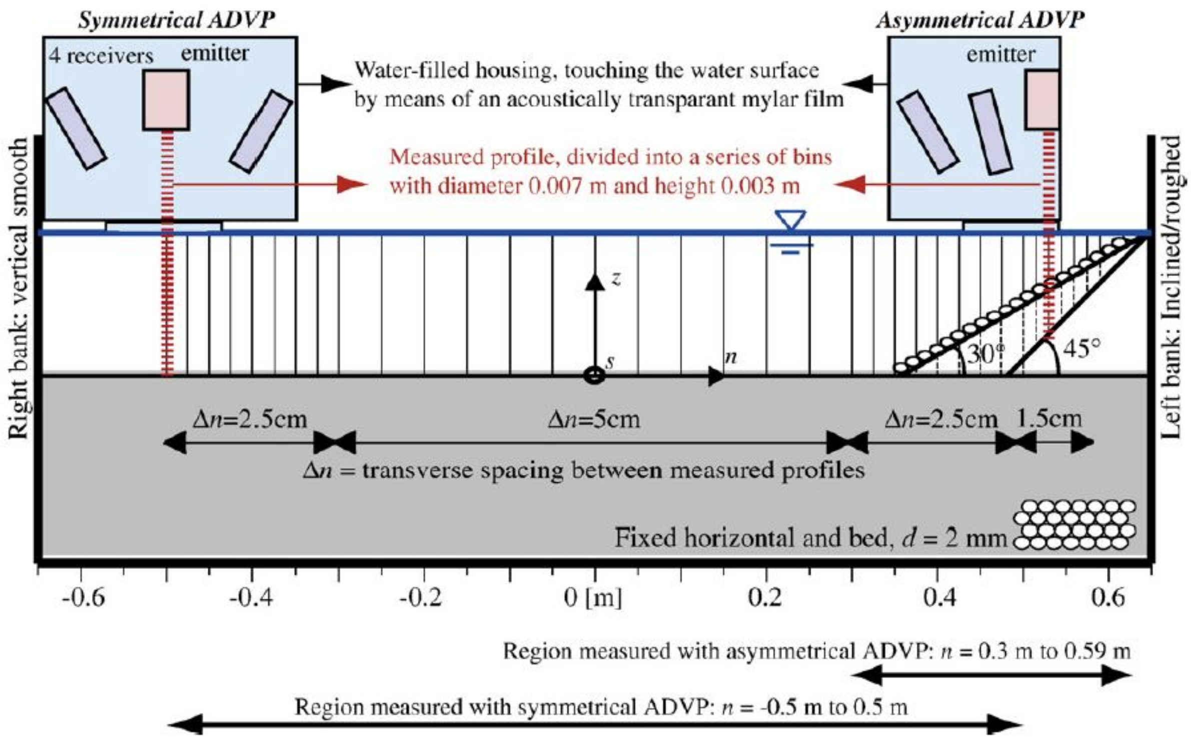

Symmetrical ADVP
4 receivers
emitter
Asymmetrical ADVP
emitter
Water-filled housing, touching the water surface by means of an acoustically transparant mylar film
Measured profile, divided into a series of bins with diameter 0.007 m and height 0.003 m
Right bank: vertical smooth
Left bank: Inclined/roughed
z
s
n
30°
45°
Δn=2.5cm
Δn=5cm
Δn=2.5cm
1.5cm
Δn = transverse spacing between measured profiles
Fixed horizontal and bed, d = 2 mm
-0.6
-0.4
-0.2
0 [m]
0.2
0.4
0.6
Region measured with asymmetrical ADVP: n = 0.3 m to 0.59 m
Region measured with symmetrical ADVP: n = -0.5 m to 0.5 m

Fig. 4.17: Upstream looking view of a laboratory flume with the measurement of an ADVP [4]

Due to the weakness of the cross- stream velocities the quantification of the patterns of the secondary currents are difficult. Blanckaert, Duarte, and Schleiss [4] found the secondary currents over the entire width of the cross section which is shown in Fig. 4.18. Where the top picture in Fig. 4.18 is based on a smooth bank and the bottom picture is based on a rough bank with a grain size of 30 mm. Also in the simulation of the trapezoidal channel the schematic pattern of secondary currents are over the entire width of the cross section and they do not weaken considerably with distance of the bank, which is shown in Fig. 4.20. In the experiment they found that the secondary currents scale with the flow depth which is similar as in the simulation. The influence of the bank inclination is limited to the half width of the cross section which suggests that the influence of the inclined boundary decreases with distance of the bank (shown in Fig. 4.20 and Fig. 4.18). On the smooth bank in Fig. 4.18 Blanckaert, Duarte, and Schleiss [4] found one secondary current where in the simulation in Fig. 4.20 three large and two small currents are shown which is more similar as in Fig. 4.19 (right picture) from Tominaga et al. [39]. In the riprap bank in Fig. 4.18 (bottom) three secondary currents are shown. Whereas in the rough simulation in Fig. 4.21 two secondary currents are created due to the bank roughness. Additionally these two vortices are shifted away from the bank. This is different compared the smooth trapezoidal channel simulation. Whereas on the vertical wall in all four figures (Fig. 4.18, Fig. 4.19, Fig. 4.20) and Fig. 4.21 two secondary currents are shown.

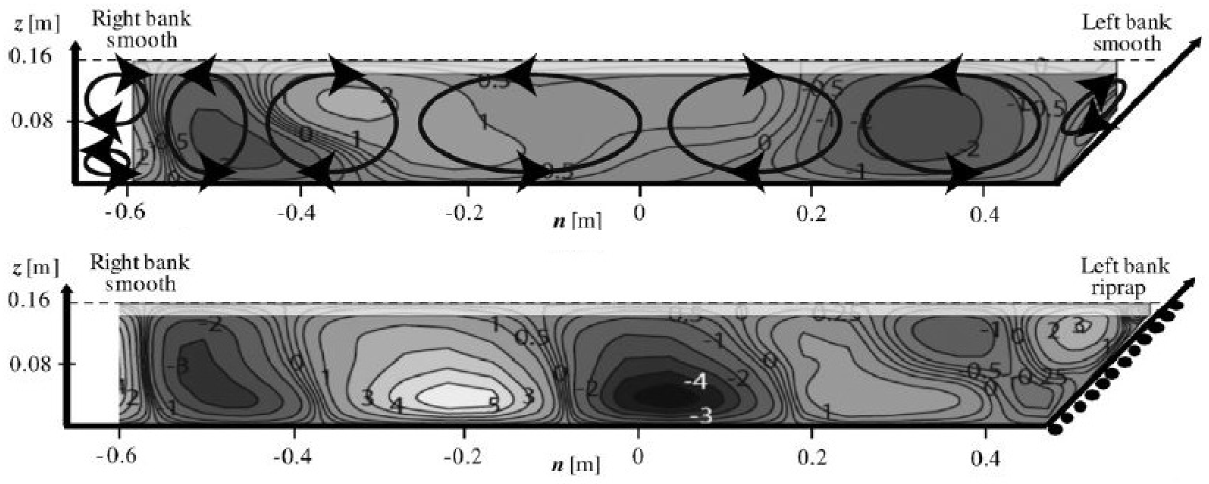


Fig. 4.18: Smooth and Rough Trapezoidal Channel: Patterns of the secondary currents in a smooth (top) and rough (bottom) trapezoidal channel from Blanckaert, Duarte, and Schleiss [4]

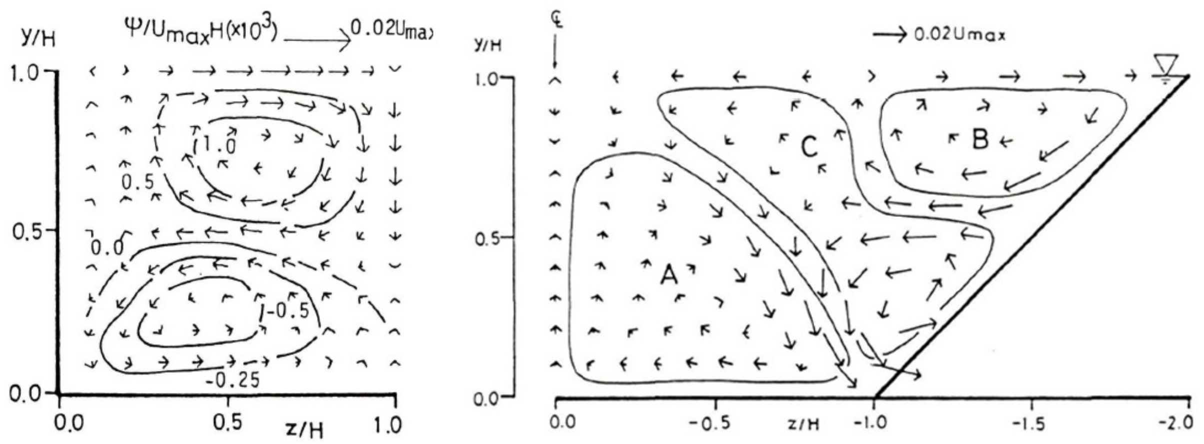


Fig. 4.19: Smooth Trapezoidal Channel: Patterns of the secondary currents in a rectangular (left) and trapezoidal channel (right) from Tominaga et al. [39]

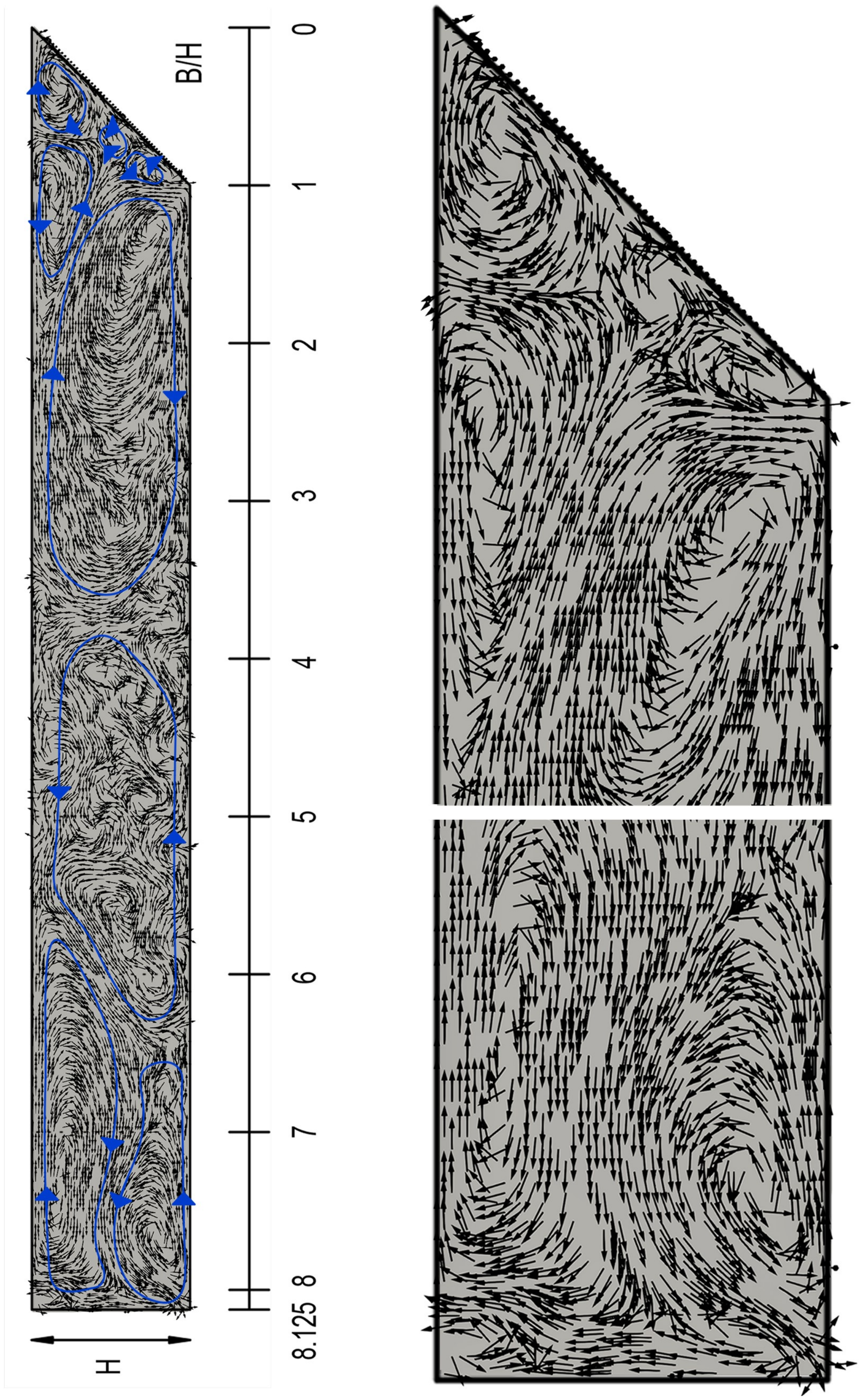


Fig. 4.20: Smooth Trapezoidal Channel: Pattern of the secondary currents from the simulation

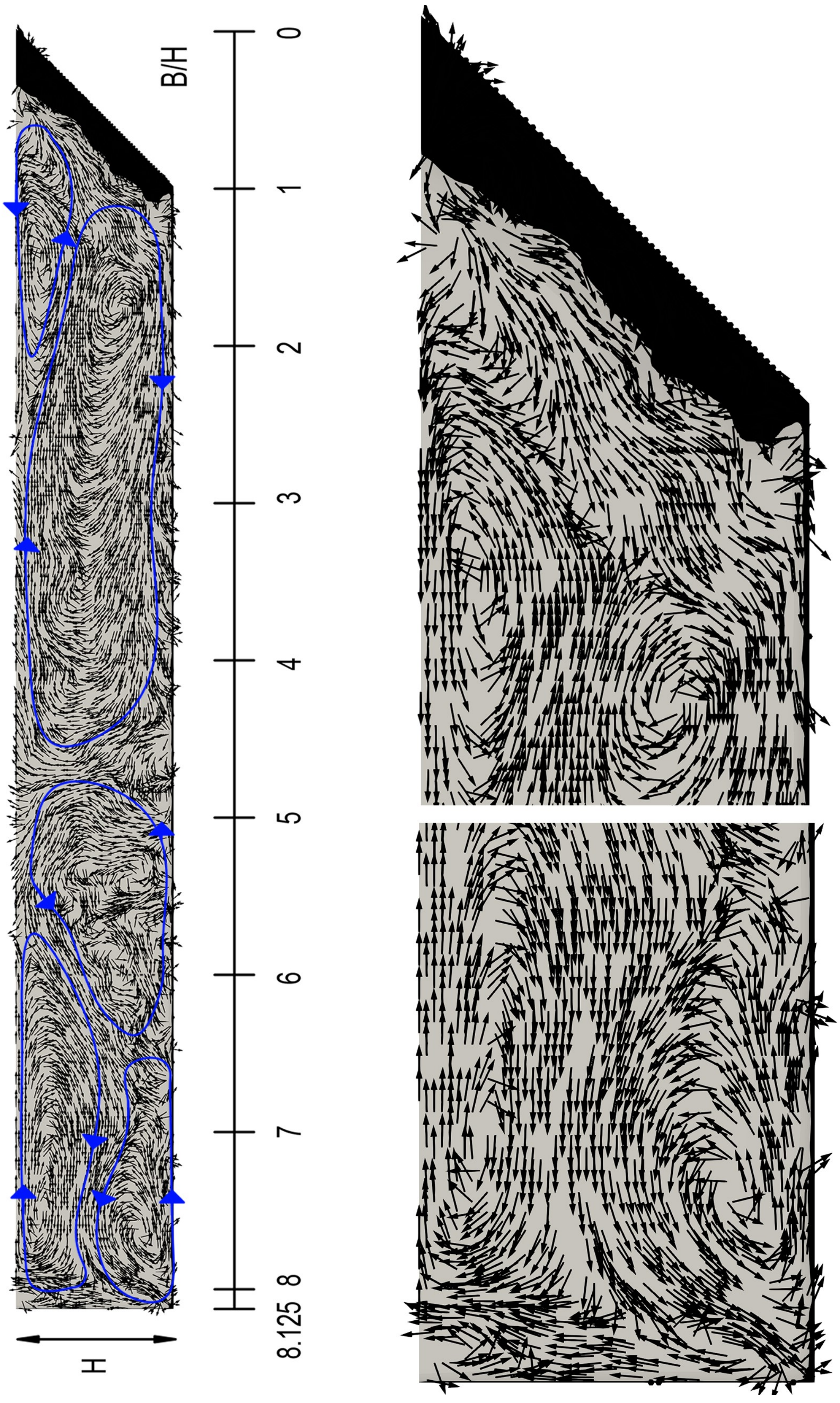


Fig. 4.21: Rough Trapezoidal Channel: Pattern of the secondary currents from the simulation

The turbulent intensities u', v' and w' by Tominaga et al. [39] are shown in Fig. 4.22 and for the simulation the turbulence intensities are shown in Fig. 4.23 and Fig. 4.24. Both cases shows the most noticeable feature on the vertical turbulent intensity v', which is rapidly damped near the surface. Whereas u' and w' show a similar distribution. Most of the region in u' shows a similar distribution as in U in Fig. 4.25. Whereas the isolines from u' near the side wall bulge further as compared of those in U. For the rough simulation the turbulent intensities increased near the rough bank and bed as shown in Fig. 4.24.

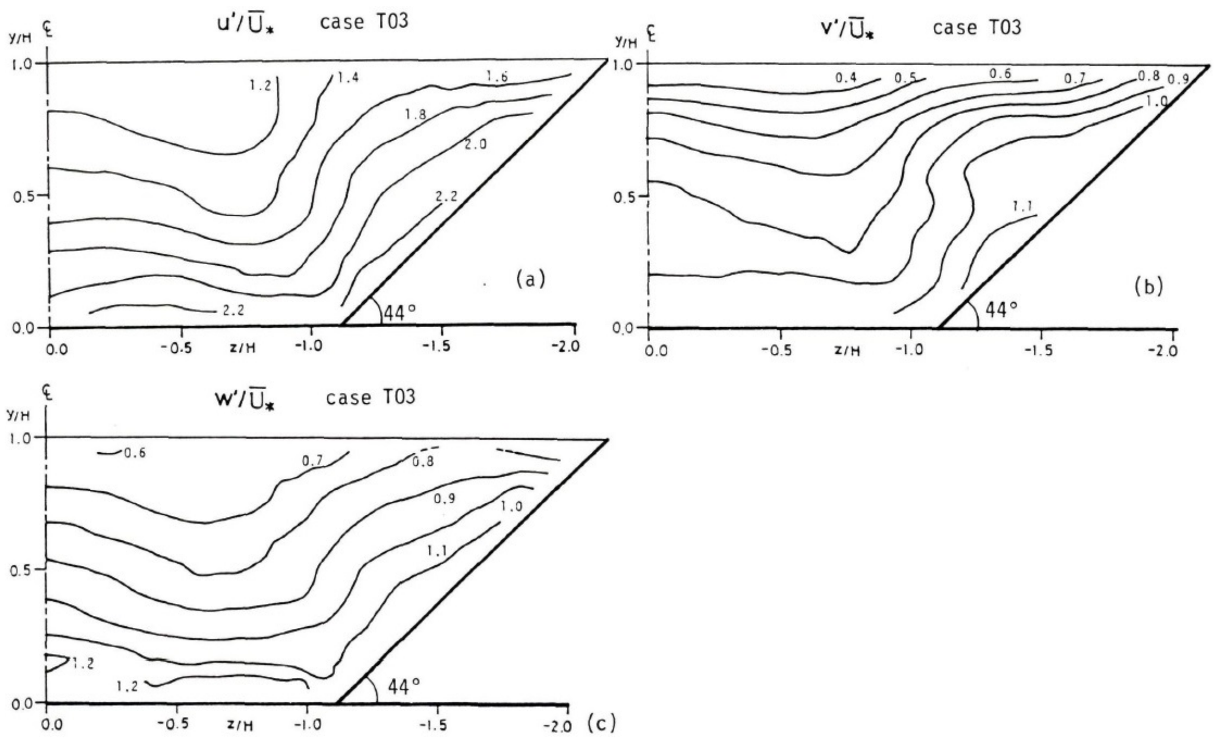


Fig. 4.22: Smooth Trapezoidal Channel: Isolines of turbulent intensities u', v' and w' in a trapezoidal channel by Tominaga et al. [39]

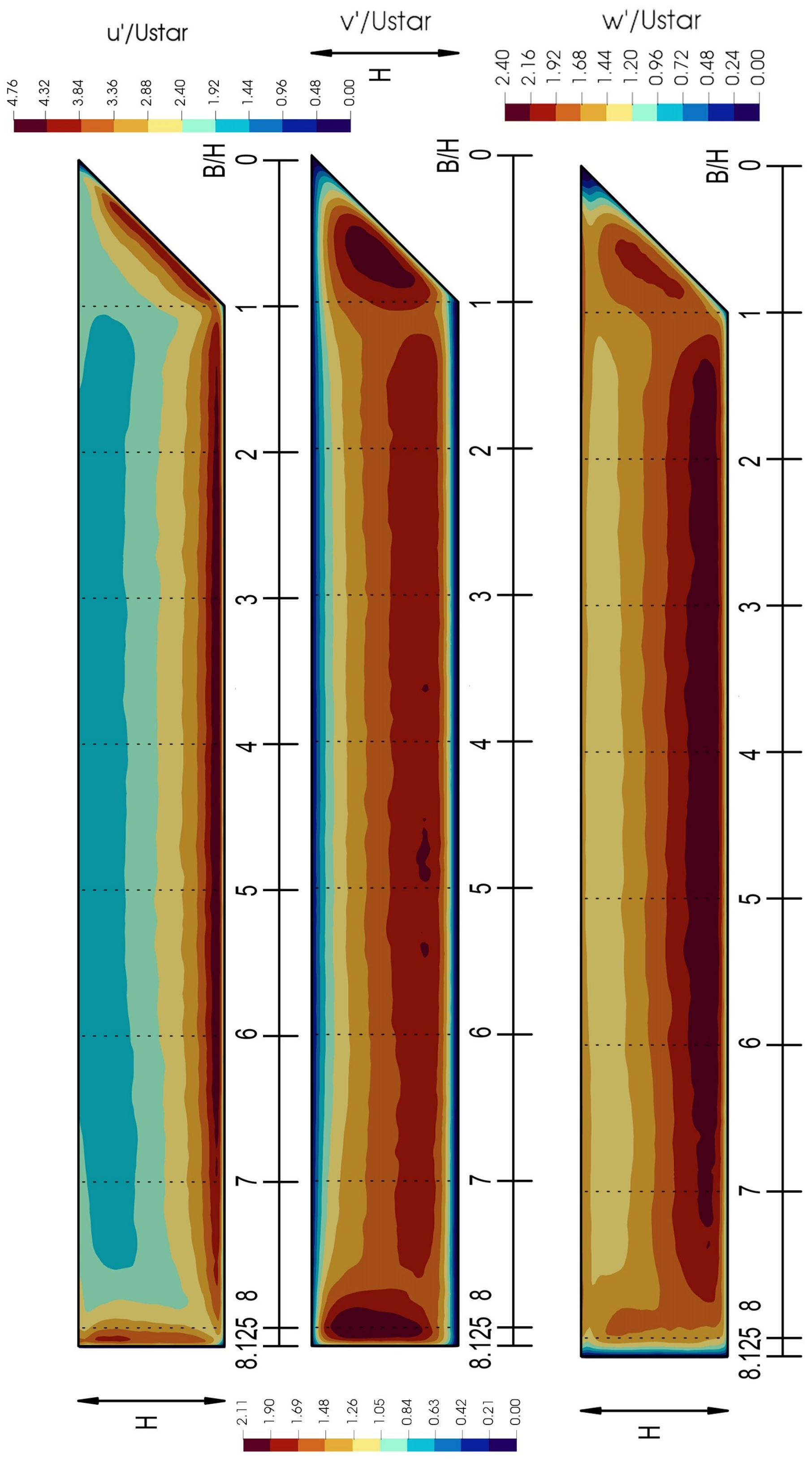


Fig. 4.23: Smooth Trapezoidal Channel: Isolines of turbulent intensities u', v' and w' in a trapezoidal channel by simulation

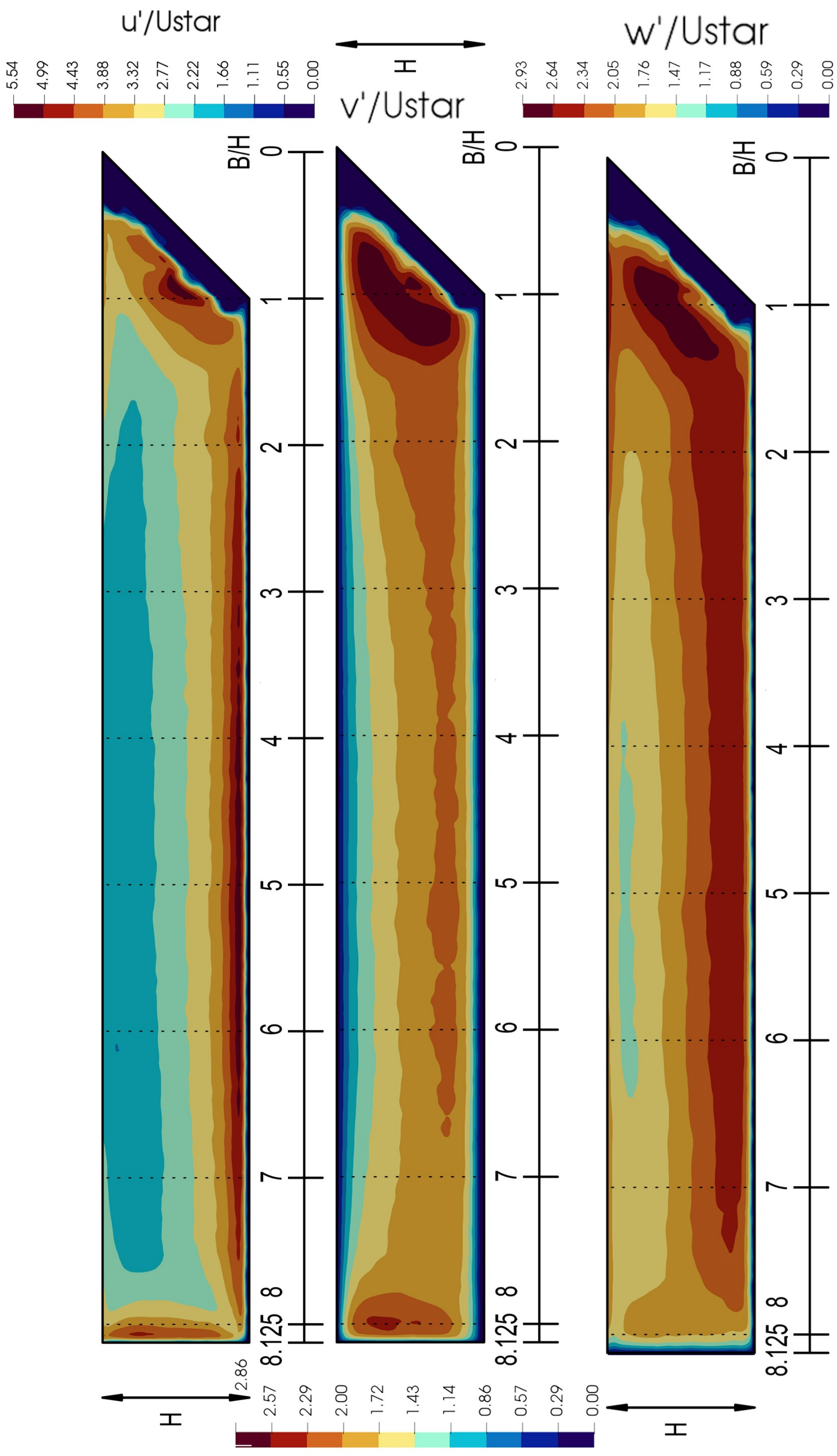


Fig. 4.24: Rough Trapezoidal Channel: Isolines of turbulent intensities u', v' and w' in a trapezoidal channel by simulation

In Fig. 4.26 the normalized streamwise velocity U/U_{bulk} from Blanckaert, Duarte, and Schleiss [4], in Fig. 4.25 the normalized streamwise velocity U/U_{max} from Tominaga et al. [39] is shown. In Fig. 4.28 the normalized streamwise velocity U/U_{bulk} and in Fig. 4.27 the normalized streamwise velocity U/U_{max} for the smooth and rough simulation is shown.

- In Fig. 4.25 the maximum velocity appears in about $0.2H$ below the surface (aspect ratio $B/H = 8$ and 99% of U_{max}) whereas in Fig. 4.27 the maximum velocity appears about in $0.15H$ (aspect ratio $B/H = 8$ and 99% of U_{max}) below the surface.

- Furthermore the decelerated region is in about $z/H = 2.5$ near the surface in the rectangular channel and $z/H = -0.75$ in the trapezoidal channel which is shown in Fig. 4.25 (99% of U_{max}). Whereas the decelerated region is in about $B/H = 6.2$ near the surface on the left side and $B/H = 2$ on the right side in the smooth trapezoidal channel (95% of U_{max}) which is shown in Fig. 4.27 on the left picture. The decelerated region from the rough channel in Fig. 4.27 on the right picture is in about $B/H = 6.6$ near the surface on the left side and is $B/H = 3.3$ on the right side (95% of U_{max}). The roughness elements on the right wall shift the maximal velocity distribution away from the bank.

- Moreover the bulk velocity from the simulation appears in about $0.8\,H$ below the surface which is shown in Fig. 4.28. The bulk velocity by Blanckaert, Duarte, and Schleiss [4] also appears in about $0.8\,H$ below the surface which is shown in Fig. 4.26. The roughness on the bed and at the bank increases the normalized velocity near the surface from 1.25 in Fig. 4.28 left picture to 1.4 in Fig. 4.28 right picture. The roughness on the bed and at the bank increases the normalized velocity near the surface from 1.3 to 1.4 in Fig. 4.26 (top and bottom). The normalized velocity distribution compared with the bulk velocity form Blanckaert, Duarte, and Schleiss [4] is similar as in the numerical simulations. Except on the left side at -0.5 in Fig. 4.26 there is a bulge shown which is different from the numerical simulation.

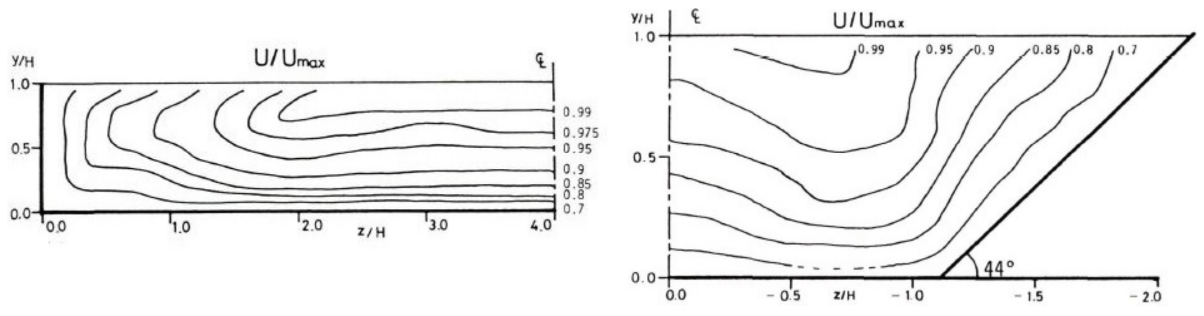


Fig. 4.25: Smooth Trapezoidal Channel: Patterns of the normalized streamwise velocity U/U_{max} in a rectangular (left) and trapezoidal channel (right) from Tominaga et al. [39]

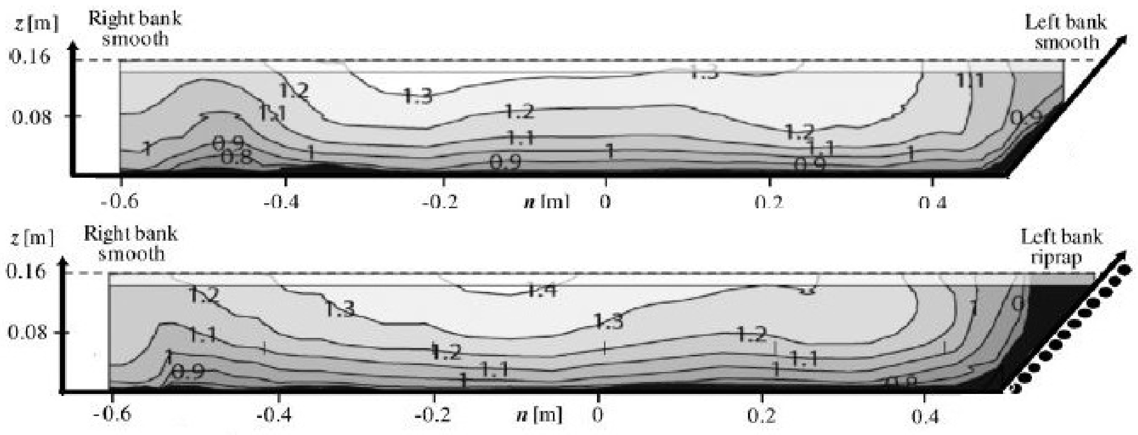


Fig. 4.26: Smooth and Rough Trapezoidal Channel: Patterns of the normalized streamwise velocity U/U_{bulk} in a smooth (top) and rough (bottom) trapezoidal channel from Blanckaert, Duarte, and Schleiss [4]

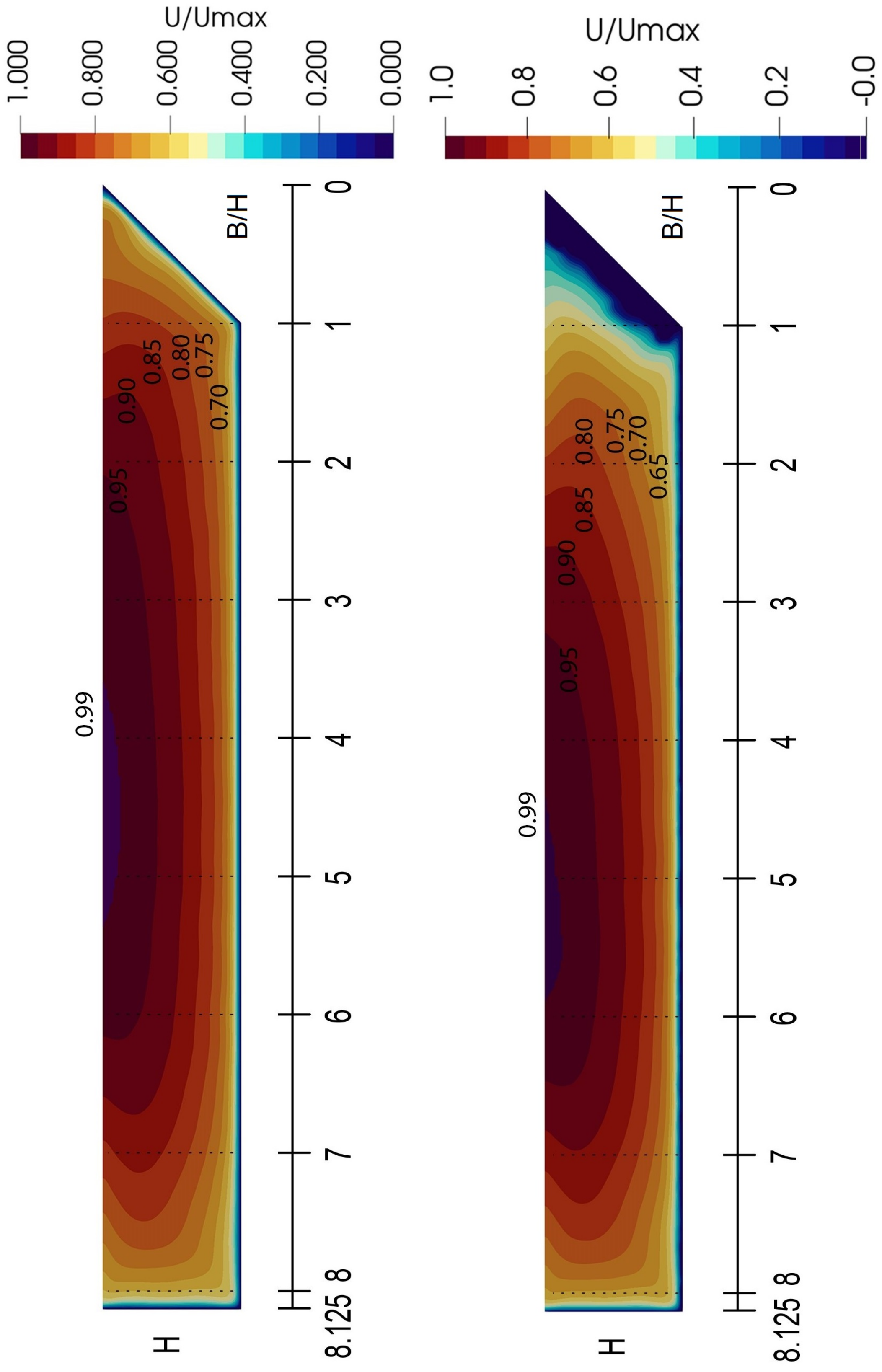


Fig. 4.27: Rough (right picture) and Smooth (left picture) Trapezoidal Channel: Patterns of the normalized streamwise velocity U/U_{max} in a trapezoidal channel from the simulation

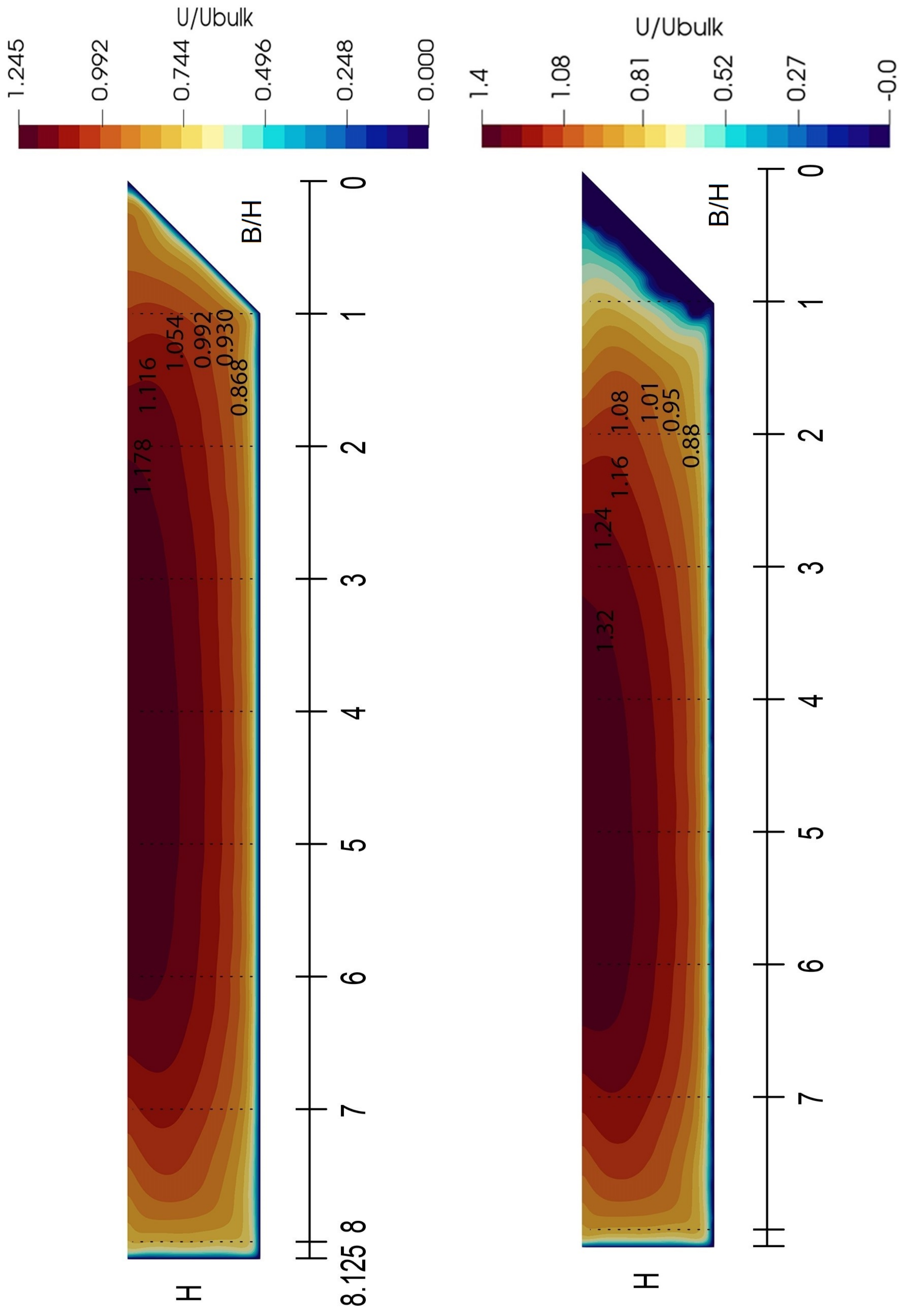


Fig. 4.28: Rough (right picture) and Smooth (left picture) Trapezoidal Channel: Patterns of the normalized streamwise velocity U/U_{bulk} in a trapezoidal channel from the simulation

The instantaneous velocity for the smooth and rough simulation is shown in a plan view and in a cross section in Fig. 4.29 and Fig. 4.30. Three horizontal cuts are at H, $2H/3$ and $H/3$. In the flow structure at the surface the eddies are much larger than in $2H/3$. Whereas in $H/3$ the eddies are getting even smaller. The whole turbulent structure has been visualized. The roughness is not accounted in Fig. 4.29. Whereas the simulation with roughness elements is shown in Fig. 4.30. The turbulent structure on the bank for the rough and smooth channel are different. On the rough bank there is a larger deceleration region (blue and cyan) than on the smooth bank which is shown in Fig. 4.29 and Fig. 4.30. Furthermore the effect of roughness at the bank shifts the instantaneous high velocity region away from the bank. Moreover this shift increases the normalized maximal velocity from 1.43 to 1.57. The deceleration region on the rough bed is similar as on the smooth bed because of the small roughness height.

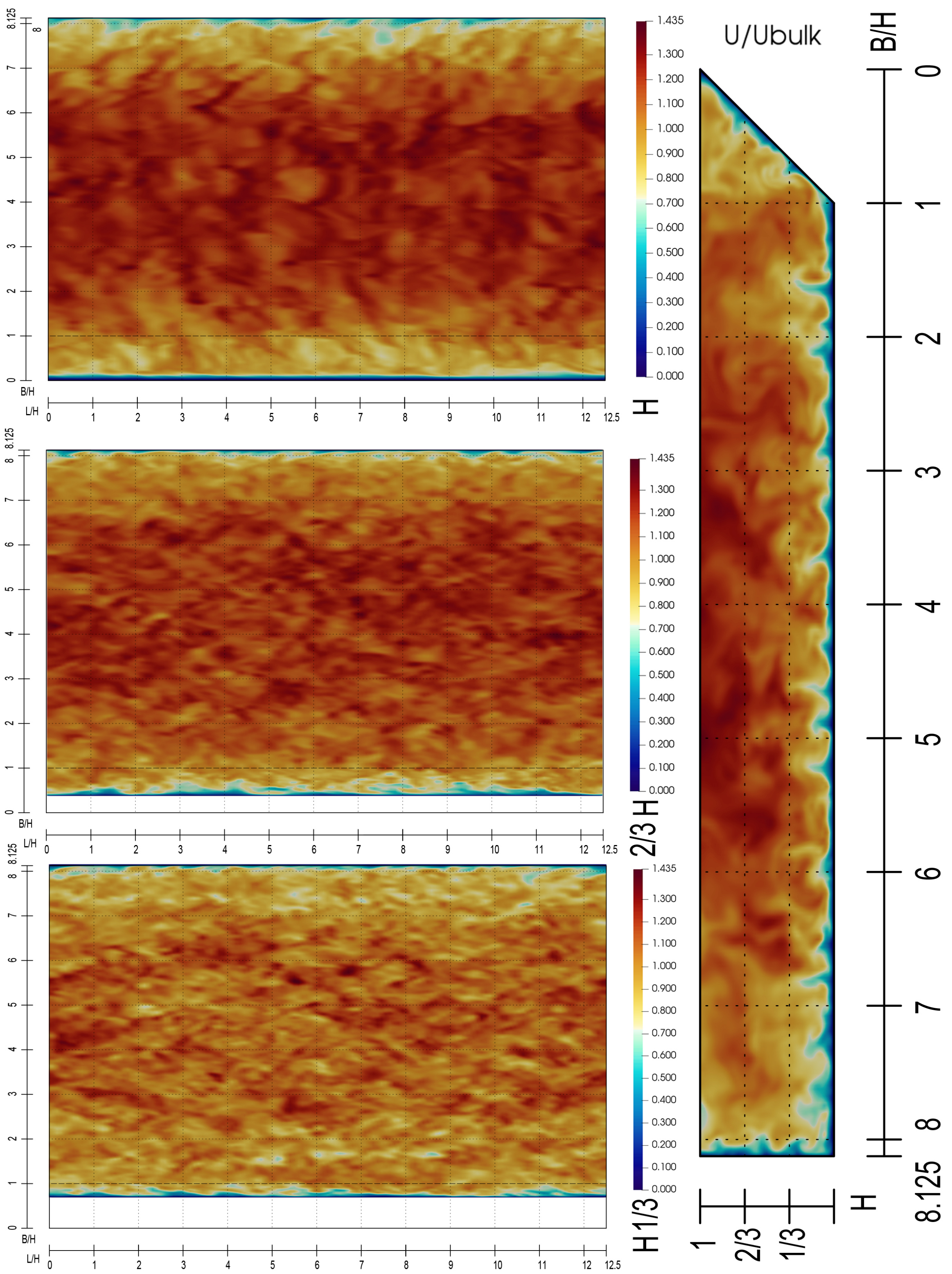


Fig. 4.29: Smooth Trapezoidal Channel: Turbulent structure of the domain

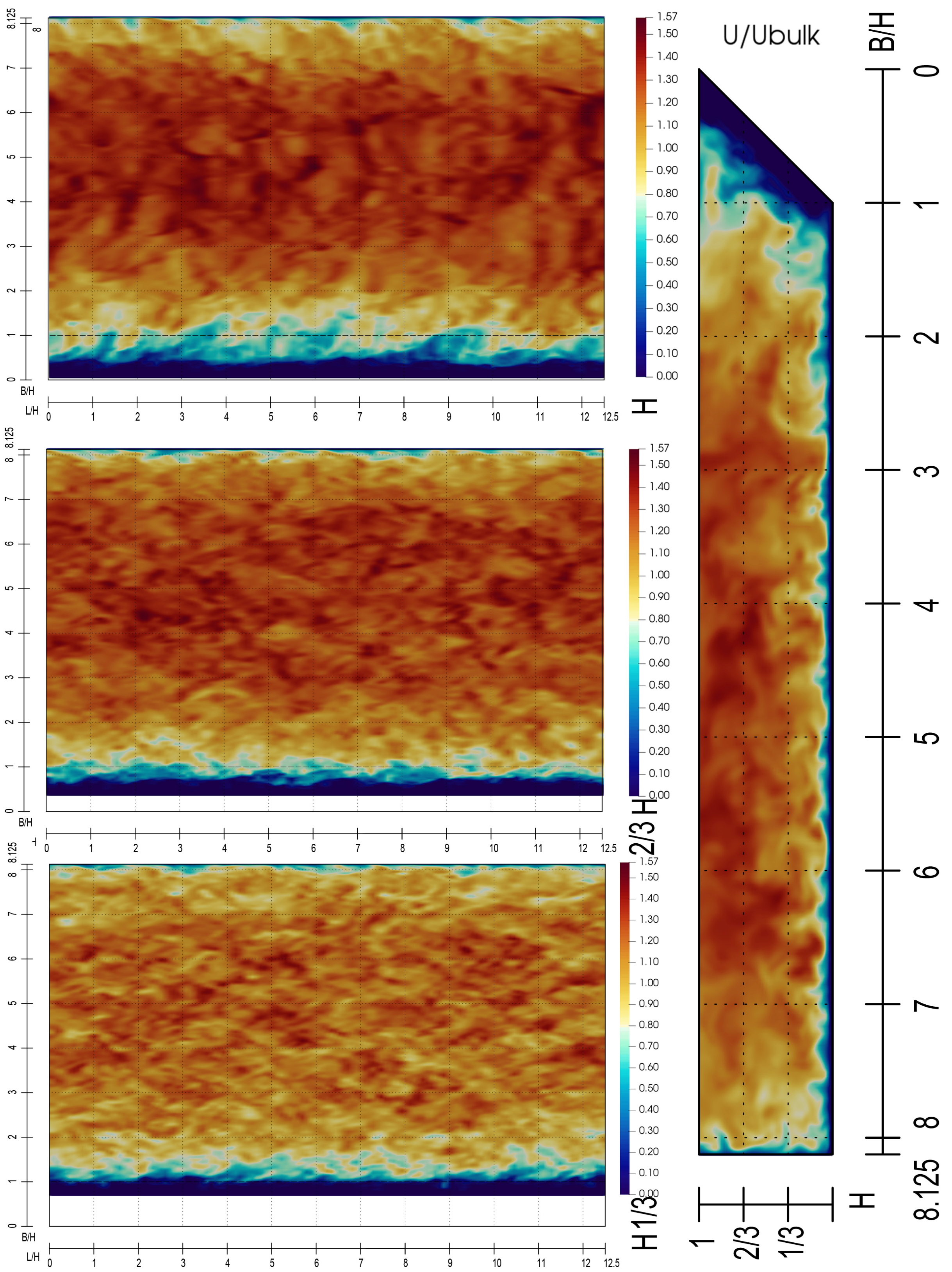


Fig. 4.30: Rough Trapezoidal Channel: Turbulent structure of the domain

Chapter 5

Summary and Conclusions

The first section shows the importance of the computational fluid dynamics in general and in hydraulic engineering. Additional a short literature review of different roughness approaches are listed. Then the objective of the work and the methodology is described briefly. The second chapter gives an overview of the theoretical background of the governing equations, turbulence, direct numerical simulation (DNS), Large-eddy simulation (LES) and Reynolds averaged Navier-Stokes (RANS) simulation. The third chapter shows the creation of roughness elements on a surface by using a the porous medium approach in detail. Additional three cases for the simulation with boundary conditions, initial conditions, options etc. are described. Whereas the first simulation does not consider roughness at all. The second simulation considers roughness and the third one uses the same rough elements but with an increased velocity. In conclusion a velocity profile of the three cases are compared. In the fourth chapter a trapezoidal channel with bed and bank roughness from laboratory experiments from Tominaga et al. [39] and Blanckaert, Duarte, and Schleiss [4] are compared with the simulation. Whereas the velocity in this simulation is decreased because of too high computational power for an average computer at present. In the last chapter the summary and conclusions are presented. Attached to this thesis the files for generating the roughness elements, the domain, options, boundary and initial conditions are shown in Appendix A for chapter three and Appendix B for chapter four. The conclusions from the cases in this thesis can be summarized as follows:

- The discretization of the roughness elements on a surface is shown in the third chapter. Whereas the roughness is restricted on a rectangular horizontal surface. For the simulation in chapter three the channel 395 from OpenFOAM is adopted. In this case the roughness generation is relatively simple. For the trapezoidal case in the fourth chapter the roughness generation on the bank is more difficult and time intensive. The domain of the channel has to be translated and rotated at a horizontal rectangular surface to implement the roughness. In conclusion it is possible to generate the elements on an inclined surface but it is very time consuming. Whereas the roughness generation on a curved surface is impossible. Maybe in future a utility in OpenFOAM would automatically generate the roughness elements on the required patches.

- The classical turbulent channel flow is modeled for $Re_\tau = 395$ with smooth wall. This is done to evaluate numerical settings and the grid resolution. The velocity profile is compared with the theoretical values. It is found that the computed values are in good agreement with the law of the wall. Furthermore, two additional rough cases are carried out with a characteristic sediments size of $0.024\,m$. Whereas the second rough channel have an increased velocity of $0.25\,m/s$. Also these two simulations confirms the law of the wall. Additional the results shows that the greater the dimensionless roughness height, the bigger the shift of the log law region.

- In the final chapter the affect of the bank roughness on the secondary flow in a trapezoidal channel is carried out. The simulation used the same method as in chapter three for considering roughness. The numerical model represents the experiments which are carried out by Blanckaert, Duarte, and Schleiss [4]. Due to the large computational demand of the experimental model the velocity has been reduced from $0.44\,m/s$ to $0.1\,m/s$. The simulation shows a pattern of the multicellular currents at the domain. Additionally a difference between a smooth and rough surface on the bank is clearly shown. Also it is shown that the presence of the roughness elements at the bank breaks up the two secondary current cells which exist in the case with the smooth bank. Finally, the rough wall at the bank generates a larger decelerated region as the smooth wall. Moreover, the effect of roughness at the bank shifts the high velocity region away from the bank and increases the normalized maximal velocity because of the reduced cross section.

Outlook

In hydraulic engineering the river bank or bed is often very rough. This study shows a technique of generating roughness elements on a rectangular horizontal surface. The idea is adopted from Stoesser [38] and Margalit [22]. Maybe in near future the porous artificially generated rough elements can be automatically computed with a few simple steps in OpenFOAM or in other computational fluid dynamic codes. Furthermore, the roughness could also be generated on a more complex geometry. This would model the practical flow problem more realistically. Additionally with the roughness generation, the secondary flow pattern in a channel flow could be investigated more in detail.

Chapter 6

Appendix A

6.1 Creation of the roughness elements, boxes.C

```
%// C++ script that generates a file 'boxes.txt' that contains
%// the coordinates to 3D bars, of which heights
%// follow a normal distribution

#include <iostream>
#include <fstream>
#include <random>
#include <math.h>
using namespace std;

int main()
{
    const double d50=0.024;     %// mean grain diameter
    const double x=4;           %// streamwise domain size
    const double z=2;           %// spanwise domain size
    const double c=3;           %// scaling factor

    const double sigma=0.5*d50; %// standard deviation
    double dx=c*sigma;          %// streamwise spacing
    double dz=c*sigma;          %// spanwise spacing
    const int nx=ceil(x/dx);    %// number of boxes streamwise
    const int nz=ceil(z/dz);    %// number of boxes spanwise

    %// outputs to terminal
    cout << "Mean grain diameter: " << d50 << endl;
    cout << "Number of boxes in streamwise direction: " << nx
       << endl;
```

```
cout << "Number of boxes in spanwise direction: " << nz <<
   endl;
cout << "Streamwise bar size: " << dx << endl;
cout << "Spanwise bar size: " << dz << endl;

%// generate coordinates with the depthwise being random
default_random_engine generator;
normal_distribution<double> distribution(0,sigma); %//
   distribution(mean,sigma)

double X[nx+1];
double Z[nz+1];
double number[nx][nz];
double small = number[0][0];

for (int i=0; i<nx; i++)
{
    for (int j=0; j<nz; j++)
    {
        %//generate random values
        number[i][j] = distribution(generator);
        %// max. 3*sigma of the random values
        %// three-sigma rule of thumb
        if (number[i][j]>3*sigma)
            number[i][j] = 3*sigma;
        if (number[i][j]<-3*sigma)
            number[i][j] = -3*sigma;
    }
}

%// find the min. coordinate and add this value to the
%// random values
for (int i=0; i<nx; i++)
{
    for (int j=0; j<nz; j++)
    {
        if(small>number[i][j])
            small = number[i][j];
    }
}
cout << "Mean 0 bed located " << abs(small) << " above
   bottomWall" << endl;
```

```
    for (int i=0; i<nx; i++)
    {
        for (int j=0; j<nz; j++)
        {
            number[i][j] += abs(small) ;
        }
    }
    %// create output file boxes.txt
    for (int i=0; i<nx+1; i++)
    {
        X[i] = i*dx;
    }
    for (int i=0; i<nz+1; i++)
    {
        Z[i] = i*dz;
    }

    ofstream myfile;
    myfile.open ("boxes.txt");
    for (int i=0; i<nx; i++)
    {
        for (int j=0; j<nz; j++)
        {
            myfile << "(" << X[i] << " 0 " << Z[j] << ")(" << X
               [i+1] << " " <<
            number[i][j] << " " << Z[j+1] << ")\n";
        }
    }
    myfile.close();
    return 0;
}
```

Listing 6.1: boxes.C

6.2 topoSetDict

```
/*--------------------------*- C++ -*--------------------------*\
=========                |
\\      /  F ield        | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration    | Website:  https://openfoam.org
  \\  /    A nd          | Version:  6
   \\/     M anipulation|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      topoSetDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

actions
(
    {
        name    bed;
        type    cellSet;
        action  new;
        source  boxToCell;
        sourceInfo
        {
            boxes
                (
                    #include "boxes.txt"
                );
        }
    }
);

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

Listing 6.2: topoSetDict

6.3 blockMeshDict

```
/*--------------------------*- C++ -*--------------------------*\
=========                 |
\\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration     | Website:  https://openfoam.org
  \\  /    A nd           | Version:  6
   \\/     M anipulation|
\*---------------------------------------------------------------------------*/
FoamFile
```

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

convertToMeters 1;

vertices
(
    (0 0 0)
    (4 0 0)
    (0 0.2 0)
    (4 0.2 0)
    (0 1 0)
    (4 1 0)
    (0 0 2)
    (4 0 2)
    (0 0.2 2)
    (4 0.2 2)
    (0 1 2)
    (4 1 2)
);

blocks
(
    hex (0 1 3 2 6 7 9 8) (72 36 48) simpleGrading (1 6 1)
    hex (2 3 5 4 8 9 11 10) (72 38 48) simpleGrading (1 5 1)
);

edges
(
);

boundary
(
    bottomWall
    {
        type            wall;
        faces           (
                        (0 1 7 6)
                    );
    }
    topWall
    {
        type            patch;
        faces           (
                            (4 10 11 5)
                        );
    }
    leftWall
    {
        type            cyclic;
        neighbourPatch  rightWall;
        faces           (
                            (0 2 3 1)
                            (2 4 5 3)
                        );
    }
    rightWall
    {
        type            cyclic;
        neighbourPatch  leftWall;
        faces                   (
                            (6 7 9 8)
                            (8 9 11 10)
                        );
    }
    inlet
    {
        type            cyclic;
        neighbourPatch  outlet;
        faces           (
                            (0 6 8 2)
                            (2 8 10 4)
                        );
    }
    outlet
    {
        type            cyclic;
        neighbourPatch  inlet;
        faces           (
                            (1 3 9 7)
                            (3 5 11 9)
                        );
    }
);

mergePatchPairs
(
);

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

Listing 6.3: blockMeshDict

6.4 Velocity, U

```
/*--------------------------*- C++ -*------------------------*\
=========                |
\\      /  F ield        | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration    | Website:  https://openfoam.org
  \\  /    A nd          | Version:  6
   \\/     M anipulation|
\*-------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform ( 0 0 0 );

boundaryField
{
    bottomWall
    {
        type            noSlip;
    }
    topWall
    {
        type            slip;
    }
    leftWall
    {
        type            cyclic;
    }
    rightWall
    {
        type            cyclic;
    }
    inlet
    {
        type            cyclic;
    }
    outlet
    {
        type            cyclic;
    }
}

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

Listing 6.4: U

6.5 Kinematic Pressure, p

```
/*--------------------------*- C++ -*------------------------*\
=========                |
\\      /  F ield        | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration    | Website:  https://openfoam.org
  \\  /    A nd          | Version:  6
   \\/     M anipulation|
\*-------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0;

boundaryField
{
    bottomWall
    {
        type            zeroGradient;
    }
        topWall
    {
        type            zeroGradient;
    }
        leftWall
    {
        type            cyclic;
    }
        rightWall
    {
        type            cyclic;
    }
        inlet
    {
        type            cyclic;
    }
        outlet
    {
        type            cyclic;
    }
}

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

Listing 6.5: p

6.6 Turbulent Kinetic Energy, k

```
/*--------------------------*- C++ -*-------------------------*\
=========                |
\\      /  F ield        | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration    | Website:  https://openfoam.org
  \\  /    A nd          | Version:  6
   \\/     M anipulation|
\*-------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0;

boundaryField
{
    bottomWall
    {
        type            fixedValue;
        value           uniform 0;
    }
        topWall
    {
        type            zeroGradient;
    }
        leftWall
    {
        type            cyclic;
    }
        rightWall
    {
        type            cyclic;
    }
        inlet
    {
        type            cyclic;
    }
        outlet
    {
        type            cyclic;
    }
}


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

Listing 6.6: k

6.7 Turbulence viscosity, nut

```
/*--------------------------*- C++ -*------------------------*\
=========               |
\\      /  F ield       | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration   | Website:  https://openfoam.org
  \\  /    A nd         | Version:  6
   \\/     M anipulation|
\*-----------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0;

boundaryField
{
    bottomWall
    {
        type            zeroGradient;
    }
```

```
        topWall
    {
        type            zeroGradient;
    }
        leftWall
    {
        type            cyclic;
    }
        rightWall
    {
        type            cyclic;
    }
        inlet
    {
        type            cyclic;
    }
        outlet
    {
        type            cyclic;
    }
}



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

Listing 6.7: nut

6.8 mapFieldsDict

```
/*-------------------------*- C++ -*------------------------*\
=========                 |
\\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration     | Website:  https://openfoam.org
  \\  /    A nd           | Version:  6
   \\/     M anipulation|
\*---------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      mapFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

patchMap
(
    bottomWall bottomWall
    sides1_half0 leftWall
    inout1_half0 outlet
);

cuttingPatches
(
    topWall
);


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

Listing 6.8: mapFieldsDict

6.9 changeDictionaryDict

```
/*-------------------------*- C++ -*------------------------*\
=========                 |
\\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration     | Website:  https://openfoam.org
  \\  /    A nd           | Version:  6
   \\/     M anipulation|
\*----------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      changeDictionaryDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

U
{
    internalField   uniform (0 0 0);
}

p
{
    internalField   uniform 0;
}

nut
{
    internalField   uniform 0;
}

k
{
    internalField   uniform 0;
}


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

Listing 6.9: changeDictionaryDict

6.10 fvOptions

```
/*-------------------------*- C++ -*------------------------*\
=========                 |
\\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration     | Website:  https://openfoam.org
  \\  /    A nd           | Version:  6
   \\/     M anipulation|
\*-------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      fvOptions;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * *  //

momentumSource
```

```
{
    type            meanVelocityForce;
    selectionMode   all;
    fields          (U);
    Ubar            (0.1335 0 0);
}

porosity1
{
    type     explicitPorositySource;
    explicitPorositySourceCoeffs
    {
        selectionMode   cellZone;
        cellZone        bed;
        type            DarcyForchheimer;
        d (1e12 1e12 1e12);
        f (1e12 1e12 1e12);
        coordinateSystem
        {
            type    cartesian;
            origin  (0 0 0);
            coordinateRotation
            {
                type axesRotation;
                e1 (1 0 0);
                e2 (0 1 0);
            }
        }
    }
}

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

Listing 6.10: fvOptions

6.11 transportProperties

```
/*--------------------------*- C++ -*------------------------*\
=========                 |
\\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration     | Website:  https://openfoam.org
  \\  /    A nd           | Version:  6
```

```
    \\/     M anipulation|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

Ubar            [0 1 -1 0 0 0 0] (0.1335 0 0);

transportModel  Newtonian;

nu              [0 2 -1 0 0 0 0] 1.9e-05;

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

Listing 6.11: transportProperties

6.12 turbulenceProperties

```
/*--------------------------------*- C++ -*----------------------------------*\
=========                 |
\\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration     | Website:  https://openfoam.org
  \\  /    A nd           | Version:  6
   \\/     M anipulation|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType LES;
```

```
LES
{
turbulence      on;

LESModel        kEqn;
delta           cubeRootVol;
}

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

Listing 6.12: turbulenceProperties

6.13 fvSchemes

```
/*--------------------------*- C++ -*-------------------------*\
=========               |
\\      /  F ield       | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration   | Website:  https://openfoam.org
  \\  /    A nd         | Version:  6
   \\/     M anipulation|
\*-------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         backward;
}

gradSchemes
{
    default         Gauss linear;
}
```

```
divSchemes
{
    default         none;
    div(phi,U)      Gauss linear;
    div(phi,k)      Gauss limitedLinear 0.1;
    div(phi,B)      Gauss limitedLinear 1;
    div(B)          Gauss linear;
    div(phi,nuTilda) Gauss limitedLinear 1;
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

wallDist
{
    method meshWave;
}

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

Listing 6.13: fvSchemes

6.14 fvSolution

```
/*--------------------------------*- C++ -*----------------------------------*\
=========                 |
\\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration     | Website:  https://openfoam.org
  \\  /    A nd           | Version:  6
```

```
   \\/     M anipulation|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       0;
        relTol          0.1;
        smoother        GaussSeidel;
    }

    pFinal
    {
        $p;
        smoother        DICGaussSeidel;
        tolerance       1e-06;
        relTol          0;
    }

    "(U|k|nuTilda)"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-05;
        relTol          0.1;
    }

    "(U|k|nuTilda)Final"
    {
        $U;
        tolerance       1e-05;
        relTol          0;
    }
}

PIMPLE
{
    nOuterCorrectors 1;
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefPoint         (0 1 0);
    pRefValue       0;
}

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

Listing 6.14: fvSolution

6.15 controlDict

```
/*-------------------------*- C++ -*-----------------------*\
=========                 |
\\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration     | Website:  https://openfoam.org
  \\  /    A nd           | Version:  6
   \\/     M anipulation|
\*---------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * *  //

application     pimpleFoam_mod;

startFrom       latestTime;

startTime       0;
```

```
stopAt          endTime;

endTime         20000;

deltaT          0.2;

writeControl    adjustableRunTime;

writeInterval   500;

purgeWrite      0;

writeFormat     ascii;

writePrecision  6;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

adjustTimeStep  yes;

maxCo   0.7;

functions
{
    fieldAverage1
    {
        type            fieldAverage;
        libs            ("libfieldFunctionObjects.so");
        writeControl    writeTime;

        fields
        (
            U
            {
                mean        on;
                prime2Mean  on;
                base         time;
                window       125;
            }

            p
            {
                mean         on;
                prime2Mean   on;
                base         time;
                window       125;
            }

            nuGradU
            {
                mean         on;
                prime2Mean   on;
                base         time;
                window       125;
            }
            B
            {
                mean         on;
                prime2Mean   on;
                base         time;
                window       125;
            }
        );
    }
}

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

Listing 6.15: controlDict

6.16 decomposeParDict

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
  \*---------------------------------------------------------------*/
  FoamFile
  {
      version     2.0;
      format      ascii;
      class       dictionary;
      note        "mesh decomposition control dictionary";
      object      decomposeParDict;
  }
  // * * * * * * * * * * * * * * * * * * * * * * * * * * //

  numberOfSubdomains  4;

  method          simple;

  multiLevelCoeffs
  {

      level0
      {
          numberOfSubdomains  64;
          method scotch;
      }
      level1
      {
          numberOfSubdomains  4;
          method scotch;
      }
  }

  simpleCoeffs
  {
      n           (1 2 2);
      delta       0.001;
  }

  hierarchicalCoeffs
  {
      n           (1 2 1);
      delta       0.001;
      order       xyz;
  }
```

```
metisCoeffs
{
}

scotchCoeffs
{
}

manualCoeffs
{
    dataFile    "decompositionData";
}

structuredCoeffs
{
    patches     (bottomPatch);
}


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

Listing 6.16: decomposeParDict

6.17 postChannelDict

```
/*-------------------------*- C++ -*-------------------------*\
=========                |
\\      /  F ield        | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration    | Website:  https://openfoam.org
  \\  /    A nd          | Version:  6
   \\/     M anipulation|
\*-----------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      postChannelDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * *  //

// Seed patches to start layering from
patches          ( bottomWall );

// Direction in which the layers are
component        y;

symmetric        false;

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

Listing 6.17: postChannelDict

Chapter 7

Appendix B

7.1 blockMeshDict

```
/*--------------------------*- C++ -*------------------------*\
=========                |
\\      /  F ield        | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration    | Website:  https://openfoam.org
  \\  /    A nd          | Version:  6
   \\/     M anipulation|
\*-----------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

convertToMeters 1;

L #calc "2";
H #calc "0.16";

Sw #calc "134"; //streamwise x-direction
Sp #calc "44"; //bed and bank in y-direction
Sa #calc "26"; //block 1 and 3 and 5 y-direction
Sb #calc "228"; //block 0 and 1 in z-direction
Sc #calc "32"; //block 2 and 3 in z-direction
```

```
vertices
(
    (0   0 0          ) //0
    (0   0.08 -0.08   ) //1
    (0   $H     -0.16 ) //2
    (0   $H     -0.08 ) //3
    (0   $H     0     ) //4
    (0   $H     1.14  ) //5
    (0   0.08   1.14  ) //6
    (0   0      1.14  ) //7
    (0   0.08   0     ) //8
    (0   0.105  -0.055) //9
    ($L 0      0      ) //10
    ($L 0.08   -0.08  ) //11
    ($L $H     -0.16  ) //12
    ($L $H     -0.08  ) //13
    ($L $H     0      ) //14
    ($L $H     1.14   ) //15
    ($L 0.08   1.14   ) //16
    ($L 0      1.14   ) //17
    ($L 0.08   0      ) //18
    ($L 0.105  -0.055) //19
    (0   0      1.04  ) //20
    (0   0.08   1.04  ) //21
    (0   $H     1.04  ) //22
    ($L 0      1.04   ) //23
    ($L 0.08   1.04   ) //24
    ($L $H     1.04   ) //25
);

blocks
(
    hex (0 20 23 10 8 21 24 18) ($Sb $Sw $Sp) simpleGrading (1
       1 5) //0
    hex (8 21 24 18 4 22 25 14) ($Sb $Sw $Sa) simpleGrading (1
       1 1) //1
    hex (1 0 10 11 9 8 18 19)   ($Sc $Sw $Sp) simpleGrading (1
       1 5) //2
    hex (9 8 18 19 3 4 14 13)   ($Sc $Sw $Sa) simpleGrading (1
       1 1) //3
    hex (2 1 11 12 3 9 19 13)   ($Sa $Sw $Sp) simpleGrading (1
       1 5) //4
    hex (20 7 17 23 21 6 16 24) (40  $Sw $Sp) simpleGrading
       (0.2 1 5) //5
    hex (21 6 16 24 22 5 15 25) (40  $Sw $Sa) simpleGrading
       (0.2 1 1) //6
);

edges
(
);

boundary
(
    bottomWall
    {
        type                wall;
        faces               (
                                (0 10 23 20)
                                (20 23 17 7)
                            );
    }
    topWall
    {
        type                patch;
        faces               (
                                (2 3 13 12)
                                (3 4 14 13)
                                (4 22 25 14)
                                (22 5 15 25)
                            );
    }

    leftWall
    {
        type                wall;
        faces               (
                                (1 2 12 11)
                                (0 1 11 10)
                            );
    }
    rightWall
    {
        type                wall;
        faces           (
                            (5 6 16 15)
                            (6 7 17 16)
                        );
    }

    inlet
    {
        type            cyclic;
        neighbourPatch  outlet;
        faces           (
                            (0 20 21 8)
                            (4 8 21 22)
                            (0 8 9 1)
                            (3 9 8 4)
                            (1 9 3 2)
                            (6 21 20 7)
                            (5 22 21 6)
                        );
    }

    outlet
    {
        type            cyclic;
        neighbourPatch  inlet;
        faces           (
                            (10 18 24 23)
                            (14 25 24 18)
                            (10 11 19 18)
                            (13 14 18 19)
                            (11 12 13 19)
                            (16 17 23 24)
                            (15 16 24 25)
                        );
    }
);

mergePatchPairs
(
);

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

Listing 7.1: blockMeshDict

7.2 bank.C

```
const double d50=0.03;        %// mean grain diameter
const double c=3;             %// scaling factor
const double x=2;             %// streamwise domain size
const double z=0.227;         %// spanwise domain size
const double mean=d50;        %// mean
const double sigma=0.3*d50;   %// standard deviation
double dx=d50;                %// streamwise spacing
double dz=d50;                %// spanwise spacing
...
for (int i=0; i<nx; i++)
{
    for (int j=0; j<nz; j++)
    {
        number[i][j] += abs(small)-abs(small);
    }
}
...
myfile.open ("bank.txt");
```

Listing 7.2: bank.C

7.3 bed.C

```
const double d50=0.002;        %// mean grain diameter
const double c=5;             %// scaling factor
const double x=2;             %// streamwise domain size
const double z=1.14;         %// spanwise domain size
const double mean=d50;        %// mean
const double sigma=0.3*d50;   %// standard deviation
double dx=d50*c;                %// streamwise spacing
double dz=d50*c;                %// spanwise spacing
...
for (int i=0; i<nx; i++)
{
for (int j=0; j<nz; j++)
{
number[i][j] += abs(small)-abs(small);
}
}
...
myfile.open ("bed.txt");
```

Listing 7.3: bed.C

7.4 topoSetBank

```
/*--------------------------*- C++ -*-------------------------*\
=========                 |
\\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration     | Website:  https://openfoam.org
  \\  /    A nd           | Version:  6
   \\/     M anipulation|
\*-------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      topoSetDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * //

actions
(
    {
        name    bank_set;
        type    cellSet;
        action  new;
        source  boxToCell;
        sourceInfo
        {
            boxes
            (
                #include "bank.txt"
            );
        }
    }

    {
        name    bank_zone;
        type    cellZoneSet;
        action  new;
        source  setToCellZone;
        sourceInfo
        {
            set bank_set;
        }
    }
);

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

Listing 7.4: topoSetBank

7.5 topoSetDelete

```
/*--------------------------*- C++ -*------------------------*\
=========                |
\\      /  F ield        | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration    | Website:  https://openfoam.org
  \\  /    A nd          | Version:  6
   \\/     M anipulation|
\*-----------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      topoSetDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * //

actions
(
    {
        name    bank_set;
        type    cellSet;
        action  delete;
        source  boxToCell;
        sourceInfo
        {
            boxes
            (
                (0 0 0.22627) (2 1 2)
            );
        }
    }

    {
        name    bank_zone;
        type    cellZoneSet;
        action  new;
        source  setToCellZone;
        sourceInfo
        {
            set bank_set;
        }
    }
);

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

Listing 7.5: topoSetDelete

7.6 topoSetBed

```
/*-------------------------*- C++ -*------------------------*\
=========                |
\\      /  F ield        | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration    | Website:  https://openfoam.org
  \\  /    A nd          | Version:  6
   \\/     M anipulation|
\*-----------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      topoSetDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * //

actions
(
    {
        name    bed_set;
        type    cellSet;
        action  new;
        source  boxToCell;
        sourceInfo
        {
            boxes
            (
                #include "bed.txt"
            );
        }
    }

    {
        name    bed_zone;
        type    cellZoneSet;
        action  new;
        source  setToCellZone;
        sourceInfo
        {
            set bed_set;
        }
    }
);

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

Listing 7.6: topoSetBed

7.7 Velocity, U

```
/*--------------------------*- C++ -*-------------------------*\
=========                |
\\      /  F ield        | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration    | Website:  https://openfoam.org
```

```
  \\  /    A nd           | Version:  6
   \\/     M anipulation|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform ( 0.1 0 0 );

boundaryField
{
    bottomWall
    {
        type            noSlip;
    }
        topWall
    {
        type            slip;
    }
        leftWall
    {
        type            noSlip;
    }
        rightWall
    {
        type            noSlip;
    }
        inlet
    {
        type            cyclic;
    }
        outlet
    {
        type            cyclic;
    }
}

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

Listing 7.7: U

7.8 setFieldsDict

```
/*-------------------------*- C++ -*------------------------*\
=========                |
\\      /  F ield        | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration    | Website:  https://openfoam.org
  \\  /    A nd          | Version:  6
   \\/     M anipulation|
\*---------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      setFieldsDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * *  //

defaultFieldValues
(
);

regions
(
    boxToCell
    {
        boxes
        (
            #include "bank.txt" %// or #include "bed.txt"
        );

        fieldValues
        (
            volVectorFieldValue U (0 0 0)
        );
    }
);

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

Listing 7.8: setFieldsDict

7.9 Pressure, P

```
/*--------------------------------*- C++ -*----------------------------------*\
=========                 |
\\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration     | Website:  https://openfoam.org
  \\  /    A nd           | Version:  6
   \\/     M anipulation|
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0;

boundaryField
{
    bottomWall
    {
        type            zeroGradient;
    }
    topWall
    {
        type            zeroGradient;
    }
    leftWall
    {
        type            zeroGradient;
    }
    rightWall
    {
        type            zeroGradient;
    }
    inlet
    {
        type            cyclic;
    }
    outlet
    {
        type            cyclic;
    }
}

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

Listing 7.9: P

7.10 Turbulent Kinetic Energy, k

```
/*--------------------------*- C++ -*-------------------------*\
=========                 |
\\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration     | Website:  https://openfoam.org
  \\  /    A nd           | Version:  6
   \\/     M anipulation|
\*-------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0;
```

```
boundaryField
{
    bottomWall
    {
        type            fixedValue;
        value           uniform 0;
    }
    topWall
    {
        type            zeroGradient;
    }
    leftWall
    {
        type            fixedValue;
        value           uniform 0;
    }
    rightWall
    {
        type            fixedValue;
        value           uniform 0;
    }
    inlet
    {
        type            cyclic;
    }
    outlet
    {
        type            cyclic;
    }
}

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

Listing 7.10: k

7.11 Turbulence viscosity, nut

```
/*--------------------------*- C++ -*-------------------------*\
=========                 |
\\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration     | Website:  https://openfoam.org
```

```
  \\  /    A nd           | Version:  6
   \\/     M anipulation|
\*---------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0;

boundaryField
{
    bottomWall
    {
         type             zeroGradient;
    }
    topWall
    {
         type             zeroGradient;
    }
    leftWall
    {
        type             zeroGradient;
    }
    rightWall
    {
        type             zeroGradient;
    }
    inlet
    {
        type             cyclic;
    }
        outlet
    {
        type             cyclic;
    }
}

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

Listing 7.11: nut

7.12 fvOptions

```
/*-------------------------*- C++ -*------------------------*\
=========                |
\\      /  F ield        | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration    | Website:  https://openfoam.org
  \\  /    A nd          | Version:  6
   \\/     M anipulation|
\*----------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      fvOptions;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * //

momentumSource
{
    type            meanVelocityForce;

    meanVelocityForceCoeffs
    {
    selectionMode   all;
    fields          (U);
    Ubar            (0.1 0 0);
    relaxation      1.0;
    }
}

porosity1
{
    type         explicitPorositySource;
    explicitPorositySourceCoeffs
    {
        selectionMode   cellZone;
        cellZone        bed_zone;

        type            DarcyForchheimer;

        DarcyForchheimerCoeffs
        {
            d (1e12 1e12 1e12);
            f (1e12 1e12 1e12);

            coordinateSystem
            {
                type    cartesian;
                origin  (0 0 0);
                coordinateRotation
                {
                    type axesRotation;
                    e1 (1 0 0);
                    e2 (0 1 0);
                }
            }
        }
    }
}

porosity2
{
    type          explicitPorositySource;

    explicitPorositySourceCoeffs
    {
        selectionMode   cellZone;
        cellZone        bank_zone;

        type            DarcyForchheimer;

        DarcyForchheimerCoeffs
        {
            d (1e12 1e12 1e12);
            f (1e12 1e12 1e12);

            coordinateSystem
            {
                type    cartesian;
                origin  (0 0 0);
                coordinateRotation
                {
                    type axesRotation;
                    e1 (1 0 0);
                    e2 (0 1 0);
                }
            }
        }
    }
}

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

Listing 7.12: fvOptions

7.13 transportProperties

```
/*--------------------------*- C++ -*-------------------------*\
=========                 |
\\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
 \\    /   O peration     | Website:  https://openfoam.org
  \\  /    A nd           | Version:  6
   \\/     M anipulation|
\*-------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

Ubar            [0 1 -1 0 0 0 0] (0.1 0 0);
```

```
transportModel  Newtonian;

nu              [0 2 -1 0 0 0 0] 1e-06;

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

Listing 7.13: transportProperties

7.14 fvSolution

```

PIMPLE
{
    nOuterCorrectors 1;
    nCorrectors     1;
    nNonOrthogonalCorrectors 1;
    pRefPoint        (2 0.16 0.5);
    pRefValue       0;
}
```

Listing 7.14: fvSolution

7.15 controlDict

```

functions
{
    Q1
    {
        type            Q;
        libs            ("libfieldFunctionObjects.so");
        writeControl    writeTime;
    }
    vorticity1
    {
        type            vorticity;
        libs            ("libfieldFunctionObjects.so");
        writeControl    writeTime;
    }
    yPlus
    {
        type            yPlus;
        libs            ("libfieldFunctionObjects.so");
        writeControl    writeTime;
    }
    fieldAverage1
    {
        type            fieldAverage;
        libs            ("libfieldFunctionObjects.so");
        writeControl    writeTime;
        timeStart       100;

        fields
        (
            U
            {
                mean        on;
                prime2Mean  on;
                base        time;
            }

            p
            {
                mean        on;
                prime2Mean  on;
                base        time;
            }
        );
    }
}
```

Listing 7.15: controlDict

Bibliography

[1] Doxygen 1.8.14. *pimpleFoam.C.* `https://www.openfoam.com/documentation/cpp-guide/html/pimpleFoam_8C.html`. [Online; accessed 12-December-2018]. 2018.

[2] Paul D Bates, Stuart N Lane, and Robert I Ferguson. *Computational fluid dynamics: applications in environmental hydraulics.* John Wiley & Sons, 2005.

[3] Kiran Bhaganagar, John Kim, and Gary Coleman. "Effect of roughness on wall-bounded turbulence". In: *Flow, turbulence and combustion* 72.2-4 (2004), pp. 463–492.

[4] Koen Blanckaert, A Duarte, and Anton J Schleiss. "Influence of shallowness, bank inclination and bank roughness on the variability of flow patterns and boundary shear stress due to secondary currents in straight open-channels". In: *Advances in Water Resources* 33.9 (2010), pp. 1062–1074.

[5] Leslie JS Bradbury et al. *Turbulent Shear Flows 4: Selected Papers from the Fourth International Symposium on Turbulent Shear Flows, University of Karlsruhe, Karlsruhe, FRG, September 12–14, 1983.* Springer Science & Business Media, 2012.

[6] Ronald J Calhoun and Robert L Street. "Turbulent flow over a wavy surface: Neutral case". In: *Journal of Geophysical Research: Oceans* 106.C5 (2001), pp. 9277–9293.

[7] Haecheon Choi and Parviz Moin. "Grid-point requirements for large eddy simulation: Chapman's estimates revisited". In: *Physics of fluids* 24.1 (2012), p. 011702.

[8] CFD Direct Ltd. Christopher J. Greenshields. *OpenFOAM Programmer's Guide.* `http://foam.sourceforge.net/docs/Guides-a4/ProgrammersGuide.pdf`. [Online; accessed 05-January-2019]. 2019.

[9] CFD Direct Ltd. Christopher J. Greenshields. *OpenFOAM User Guide version 6.* `http://foam.sourceforge.net/docs/Guides-a4/OpenFOAMUserGuide-A4.pdf`. [Online; accessed 04-January-2019]. 2019.

[10] JAMES K Culbertson. "Evidence of secondary circulation in an alluvial channel". In: *US Geological Survey Professional Paper* (1967), pp. D214–D216.

[11] *Dimensionless velocity.* `https://www.cfd-online.com/Wiki/Dimensionless_velocity`. [Online; accessed 09-January-2019]. 2019.

[12] *Dimensionless wall distance (y plus).* `https://www.cfd-online.com/Wiki/Dimensionless_wall_distance_(y_plus)`. [Online; accessed 09-January-2019]. 2019.

[13] CFD Direct. *OpenFOAM v6 User Guide: 4.4 Numerical schemes.* `https://cfd.direct/openfoam/user-guide/v6-fvschemes/`. [Online; accessed 04-January-2019]. 2019.

[14] CFD Direct. *OpenFOAM v6 User Guide: 4.5 Solution and algorithm control.* `https://cfd.direct/openfoam/user-guide/v6-fvsolution/`. [Online; accessed 05-January-2019]. 2019.

[15] Nicholas J Georgiadis, Donald P Rizzetta, and Christer Fureby. "Large-eddy simulation: current capabilities, recommended practices, and future research". In: *AIAA journal* 48.8 (2010), pp. 1772–1784.

[16] Robert J Houghtalen, A Osman Akan, and Ned HC Hwang. *Fundamentals of hydraulic engineering systems.* Prentice Hall New York, 2016.

[17] I Karcz. "Reflections on the origin of source small-scale longitudinal streambed scours". In: *Fluvial geomorphology* (1973), pp. 149–173.

[18] Ryosaku Kinoshita. "An analysis of the movement of flood waters by aerial photography". In: *Journal of the Japan society of photogrammetry* 6.1 (1967), pp. 1–17.

[19] DW Knight et al. "Boundary shear stress distributions in open channel and closed conduit flows". In: *Proc. Euromech 156–Mechanics of Sediment Transport, Istanbul, Turkey* (1982), pp. 33–40.

[20] S Leonardi et al. "Direct numerical simulations of turbulent channel flow with transverse square bars on one wall". In: *Journal of Fluid Mechanics* 491 (2003), pp. 229–238.

[21] DA Lyn. "Turbulence measurements in open-channel flows over artificial bed forms". In: *Journal of Hydraulic Engineering* 119.3 (1993), pp. 306–326.

[22] Jonatan Margalit. *Modeling of bed roughness using a geometry function and forcing terms in the momentum equation.* `http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2014/JonatanMargalit/report.pdf`. [Online; accessed 12-December-2018]. 2018.

[23] Gerard H Matthes. "Macroturbulence in natural stream flow". In: *Eos, Transactions American Geophysical Union* 28.2 (1947), pp. 255–265.

[24] Parviz Moin and Krishnan Mahesh. "Direct numerical simulation: a tool in turbulence research". In: *Annual review of fluid mechanics* 30.1 (1998), pp. 539–578.

[25] H Nakagawa, I Nezu, and Y Ooishi. "Experimental study on cellular secondary currents in open-channel flow". In: *Annual Kansai-Branch Meeting of Japan Soc. of Civil Engrs.* 1982.

[26] Hiroji Nakagawa, Iehisa Nezu, and Hiroshi Ueda. "Turbulence of open channel flow over smooth and rough beds". In: *Proceedings of the Japan Society of Civil Engineers*. Vol. 1975. 241. Japan Society of Civil Engineers. 1975, pp. 155–168.

[27] A Nakayama. "Filtering and LES of flow over irregular rough boundary, Center for Turbulence Research". In: *Proceedings of the Summer Program 2004* (2004).

[28] Iehisa Nezu and Wolfgang Rodi. "Open-channel flow measurements with a laser Doppler anemometer". In: *Journal of Hydraulic Engineering* 112.5 (1986), pp. 335–355.

[29] Vladimir Nikora et al. "Spatially averaged open-channel flow over rough bed". In: *Journal of Hydraulic Engineering* 127.2 (2001), pp. 123–133.

[30] OpenFoam. *DarcyForchheimer Class Reference.* `https://www.openfoam.com/documentation/cpp-guide/html/classFoam_1_1porosityModels_1_1DarcyForchheimer.html`. [Online; accessed 03-January-2019]. 2019.

[31] OpenFoam. *OpenFOAM 2.2.0: fvOptions.* `https://openfoam.org/release/2-2-0/fv-options/`. [Online; accessed 03-January-2019]. 2019.

[32] Stephen B Pope and Stephen B Pope. *Turbulent flows.* Cambridge university press, 2000.

[33] Osborne Reynolds. "An experimental investigation of the circumstances which determine whether the motion of water shall bo direct or sinuous". In: *Phil. Trans* 24 (), p. 935.

[34] Wolfgang Rodi, George Constantinescu, and Thorsten Stoesser. *Large-eddy simulation in hydraulics.* Crc Press, 2013.

[35] Alberto Scotti. "Direct numerical simulation of turbulent channel flows with boundary roughened with virtual sandpaper". In: *Physics of Fluids* 18.3 (2006), p. 031701.

[36] *Scripts/blockMesh grading calculation.* `https://openfoamwiki.net/index.php/Scripts/blockMesh_grading_calculation`. [Online; accessed 22-February-2019]. 2019.

[37] KM Singh, ND Sandham, and JJR Williams. "Numerical simulation of flow over a rough bed". In: *Journal of Hydraulic Engineering* 133.4 (2007), pp. 386–398.

[38] Thorsten Stoesser. "Physically realistic roughness closure scheme to simulate turbulent channel flow over rough beds within the framework of LES". In: *Journal of Hydraulic Engineering* 136.10 (2010), pp. 812–819.

[39] Akihiro Tominaga et al. “Three-dimensional turbulent structure in straight open channel flows”. In: *Journal of hydraulic research* 27.1 (1989), pp. 149–173.

[40] Vito A Vanoni. “Transportation of suspended sediment by water”. In: *Trans. of ASCE* 111 (1946), pp. 67–102.

[41] Henk Kaarle Versteeg and Weeratunge Malalasekera. *An introduction to computational fluid dynamics: the finite volume method.* Pearson Education, 2007.

[42] Timothy Wray. “Development of a One-Equation Eddy Viscosity Turbulence Model for Application to Complex Turbulent Flows”. In: (2016).

[43] Wusi Yue, Ching-Long Lin, and Virendra C Patel. “Large-eddy simulation of turbulent flow over a fixed two-dimensional dune”. In: *Journal of Hydraulic Engineering* 132.7 (2006), pp. 643–651.

List of Figures

Listings