TECHNISCHE UNIVERSITÄT WIEN

DISSERTATION

Three-Dimensional Numerical Modelling of Turbulent River Flow using Polyhedral Finite Volumes

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

O.Univ.Prof. Dipl.-Ing. Dr.techn. Dr.h.c. Dieter Gutknecht
E222
Institut für Wasserbau und Ingenieurhydrologie

eingereicht an der Technischen Universität Wien
Fakultät für Bauingenieurwesen

von

Dipl.-Ing. Michael Tritthart
9315843
Brigittenauer Lände 160-162/3/32
1200 Wien

Wien, im April 2005

Meinen Eltern in Dankbarkeit gewidmet

Kurzfassung

Die vorliegende Arbeit befasst sich mit der numerischen Ermittlung von Strömungskenngrößen und der Lage des Wasserspiegels in turbulenten Fließgewässern. Zum Einsatz gelangt dabei die technisch bewährte und durch Integralformulierung massenerhaltende Finite-Volumen Methode im dreidimensionalen Berechnungsraum. Im Gegensatz zu herkömmlichen Verfahren werden hierbei jedoch Berechnungszellen verwendet, die durch eine beliebige Anzahl an Begrenzungsflächen charakterisiert sind, wodurch das problematische Auftreten von nicht lotrecht auf die Seitenflächen der Zellen gerichteten Massenflüssen reduziert werden kann.

Nach einer Zusammenstellung der Fähigkeiten und Charakteristika heute in Verwendung stehender 3D-Rechenmodelle für den Fließgewässerbereich werden zunächst Algorithmen vorgestellt, mit denen Rechengitter aus vielflächen Zellen erzeugt werden können. Hierauf folgt die mathematische Herleitung der auf diesen Rechenzellen diskretisierten Bestimmungsgleichungen strömender Fluide sowie der grundlegenden Erhaltungsgleichungen turbulenter Vorgänge samt der für eine Anwendung notwendigen Randbedingungen. Diese Gleichungen werden ebenso wie die Algorithmen zur Netzgenerierung und Postprocessingfunktionalität in ein Computermodell mit dem Namen RSim-3D implementiert. Das resultierende numerische Modell wird im Anschluss daran anhand der Messergebnisse von vier verschiedenen Laborexperimenten validiert und letztlich beispielhaft auf einen Abschnitt der österreichischen Donau angewendet. Die Diskussion denkbarer Zukunftsperspektiven bildet den Abschluss der Arbeit.

Abstract

This thesis deals with the numerical computation of flow properties and the position of the free water surface in turbulent water bodies. For this purpose the well-known and mass-conserving Finite Volume method is applied in three spatial dimensions. In contrast to usual techniques, computation cells are being used which are characterised by an arbitrary number of faces. Due to this, the presence of flows not perpendicular to element faces, which are usually the reason for a reduction in accuracy, can be reduced by a reasonable amount.

In a first step, the capabilities and characteristics of 3D models in use in hydraulic research today are evaluated and subject to comparison. Afterwards, algorithms for generating computation grids based on polyhedral cells are presented. Subsequently the governing equations for fluid flow and turbulent properties are discretised on these computation cells, and the required boundary conditions are discussed. The resulting equations and grid generation algorithms, as well as some post processing functionality, are implemented into a computer model called RSim-3D. This model is then validated against measurements of four different laboratory experiments before it is applied to a reach of the river Danube in Austria. The discussion of potential future prospects finally concludes the work.

Preface

This work was developed while I was holding an assistant position at the Institute of Hydraulics, Hydrology and Water Resources Management at Vienna University of Technology. I was supported by many people during this time, and I can honestly say that the present work would not have been possible without their assistance:

First, and most important, I am greatly indebted to my adviser and head of the institute, O.Univ. Prof. Dipl.-Ing. Dr. Dieter Gutknecht. Albeit some people uttered concerns about the feasibility of the work in the beginning, he trusted in me right from the start and gave me the chance to prove the thesis I had proposed. What's even more, he gave me all the time I needed to develop the work from scratch, and invested his own time to provide me with valuable advice. Especially today when university funding is scarce and human resources can hardly be financed other than by projects for third parties, leaving little room for fundamental research work, this cannot be taken for granted at all.

I would also like to express my gratitude to Univ.Prof. Dipl.-Ing. Dr. Peter Rutschmann of the Water Resources Institute at the University of Innsbruck, who acted as my co-adviser and took the time to give me valuable feedback especially during the final phase of the work.

Of course, I am also very grateful to all my colleagues at the institute who have participated in creating an atmosphere that makes it a pleasure to work. Especially I would like to thank Dipl.-Ing. Ulrike Drabek for numerous conversations, both on and also besides the scope of our work, and all the hints of how certain computer application problems could be solved. I'd also like to take the chance to thank Dipl.-Ing. Tim Fischer-Antze for the exchange of professional experience that helped me getting through difficult times during the phase of model development.

This work would never have been written without the ideas of Dr.-Ing. habil. Peter Milbradt from Hannover University who helped me by discussing the subject during the early stages of the thesis and providing me with fundamental literature. I thank him for his support.

Also, I would like to express my gratitude to Dipl.-Ing. Dr. Robert Feurich of the Water Resources Institute at the University of Innsbruck and Dipl.-Ing. Christian Kölbl and Mr. Walter Schmid from the Austrian Federal Waterways Authority for providing me with data.

Last but not least I would like to thank Mag. Michael Herwirsch for proof-reading and giving me some of the energy and motivation needed for this work.

Vienna, April 2005 Michael Tritthart

Table of Contents

List of Tables

List of Figures

Nomenclature

A_i	surface area of face i [m^2]
a_P	discretised diagonal coefficient in the Finite Volume formulation
a_{N_i}	discretised neighbour coefficient in the Finite Volume formulation
b	discretised source term in the Finite Volume formulation
C	Chezy coefficient
$\vec{c}_i$	centroid of face i
C_μ, $C_{1\varepsilon}$, $C_{2\varepsilon}$	empirical constants of the $k-\varepsilon$ model
E_{ij}	mean rate of deformation of a finite fluid volume
f_1	distance-based weighting factor
f_x, f_y, f_z	external forces [N]
h_p	surface elevation of point p obtained by triangle interpolation [m]
h_p^*	surface elevation of point p obtained by kriging [m]
h_r	friction loss [m]
J_e	energy gradient
k	turbulent kinetic energy [m^2/s^2]
k_s	equivalent sand roughness (Nikuradse) [m]
k_{St}	Strickler coefficient [$m^{\frac{1}{3}}/s$]
L	characteristic length of the flow domain [m]
ℓ_m	mixing length
$\dot{m}_f$	mass flux [kg/s]
$\vec{n}_i$	normal vector of face i

$n_{i,x}$, $n_{i,y}$, $n_{i,z}$	component of $\vec{n}_i$ pointing into Cartesian coordinate direction x, y, z
p or p_i	generic point; pressure [N/m^2]; order of the discretisation scheme
Q	discharge [m^3/s]
q or q_i	generic point
q_{jk}	coefficient in the Bivariate Interpolation method
R_h	hydraulic radius [m]
R^ϕ	scaled residual of ϕ
R_u^ϕ	unscaled residual of ϕ
s^2	variance
S_p	linearised source term
S_u	non-linear source term
S_ϕ	source term for transported quantity ϕ
$\Delta\vec{s}$	vector from cell centroid to face centroid
t	time
u or u_1	velocity component in the Cartesian coordinate direction of x or x_1
$\vec{u}$	velocity vector
u^*	shear velocity [m/s]
U_i	perimeter of cross-section i
u_j	short-hand notation for u_1, $u_2$and u_3
$\bar{u}_j$	time-averaged mean velocity component [m/s]
u_t	tangential velocity [m/s]
$\overline{u_i' u_j'}$	Reynolds stress
V	cell volume [m^3]
v or u_2	velocity component in the Cartesian coordinate direction of y or x_2
V_f	control volume centered around the cell boundary [m^3]
v_j	average flow velocity in cross-section j [m/s]
W_i	weighting factor
w or u_3	velocity component in the Cartesian coordinate direction of z or x_3

w_j	water surface elevation in cross section j [m]
x or x_1	Cartesian coordinate direction x
x_j	short-hand notation for x_1, $x_2$and x_3
x_c	x-coordinate of face centroid [m]
y or x_2	Cartesian coordinate direction y
$Y_{E,p}$	estimate of value for Y in point p
Y_i	value of Y in point i
Δy_P	normal wall distance of point P [m]
z or x_3	Cartesian coordinate direction z
z_c	z-coordinate of face centroid [m]
$z_1\,(fv_i)$	bottom elevation of face vertex i [m]
$z_2\,(fv_i)$	top elevation of face vertex i [m]
α_ϕ	relaxation coefficient for property ϕ
Γ	diffusion coefficient
$\gamma\,(d_{ij})$	semivariance between data points i and j
δ_{ij}	Kronecker Delta ($\delta_{ij} = 1$ if $i = j$, $\delta_{ij} = 0$ if $i \neq j$)
δ_{NP}	distance from point N to point P [m]
ϵ	error bound
ε	dissipation [m^2/s^3]
Φ_A	limiting function
ϕ	transported quantity
ϕ_f	face value of ϕ
ϕ_N, ϕ_P	value of ϕ in points N and P
$\nabla\phi$	gradient of ϕ in 3D
κ	von Karman constant
λ	Lagrange multiplier
μ	dynamic viscosity [kg/m.s]
ν	kinematic viscosity [m^2/s]
ν_t	turbulent (eddy) viscosity [m^2/s]

ρ	density [kg/m^3]
σ_k, σ_ε	empirical constants of the $k-\varepsilon$ model
τ_w	wall shear stress [N/m^2]
Θ	rotation angle of the coordinate system
Ω	control volume in general formulation
ω	dissipation per turbulent kinetic energy, inverse time scale [1/s]

1 Introduction

1.1 Thematic Introduction

In recent years a noticeable trend towards the use of numerical modelling can be observed in all engineering disciplines. This development is not surprising as computer models often feature lower cost than comparable physical experiments, are superior in speed and provide complete information of all relevant quantities throughout the domain of interest at once. The wide field of water-related sciences is no exception to this trend: hydrology has a long-standing tradition in rainfall-runoff modelling, groundwater hydraulics uses solute transport computer models for a long time already, and river hydraulics relies heavily on the use of computational fluid dynamics. The present work will focus on the latter of these important topics.

Especially for the investigation of flow conditions and sediment transport in rivers, computational fluid dynamics proves to be a valuable tool. Compared with physical experiments, it allows for a rapid variation in boundary conditions, including surface roughness and discharge, but also the effect of man-made structures can be quantified very quickly using tools for numerical flow analysis. Hence, they are used in the planning stage of proposed structures or modifications in the river or its surrounding areas, in real-time flood forecasting applications, and they also assist experts in forming their opinion on the reasons of incidents that took place in the past. Depending on the spatial modelling detail, the applications are classified into one-, two-, and three-dimensional models. While the use of 1D-models is widespread among engineers, mostly due to their easy application and the little in-depth knowledge required to apply them, 2D-models are not yet used that frequently. Often they are applied by the engineer to simulate spatially confined flow processes that exceed the application limits of one-dimensional models, for instance the flooding of previously dry terrain where two-dimensional effects prevail. Finally, 3D-models are rarely applied in practice; their use seems to be mostly limited to academia. This is not surprising as the use of higher dimensional models usually requires in-depth knowledge about both the underlying physical processes and the corresponding numerics. Furthermore, a much

higher level of detail of the modelled region must be available for a successful application, but very often this data is not at the engineer's disposal, rendering the gain of 3D-models practically useless.

However, if the required data is available, three-dimensional river analysis codes can become extremely valuable tools for investigating phenomena exhibiting 3D flow characteristics. This includes flow through river bends where a secondary motion is induced (*Nguyen* (2000) [51], *Feurich* (2002) [21]), river junctions (*Bradbrook et al.* (2000) [12]), the presence of submerged groynes (*Ouillon & Dartus* (1997) [60], *Miller et al.* (2003) [48]), scour around obstacles in the flow domain (*Premstaller* (2002) [64]), and also the whole region in the vicinity of weirs and other man-made structures. In all these cases statements about specific flow features, like flow direction and magnitude, the position of the water surface, pressure and turbulent kinetic energy can be made, all of which are crucial for an engineer's assessment of the situation. The future value of computational fluid dynamics tools clearly is found in predicting sediment transport on a larger scale – especially since the treatment of sediments will be one of the major challenges of the hydraulic engineer in the 21st century – but also water quality investigations and habitat modelling are applications for the time to come, as soon as the required software will have reached a reasonable level of applicability. It should, however, be noted that the correct prediction of the flow field is of paramount importance for the evaluation of any properties that are transported along with the flow. Therefore research efforts that are directed towards improvement of tools for modelling the flow field are still required and will be the primary subject of this work.

Regardless of the dimension of the model or the discretisation technique employed, the flow domain is always decomposed using a computation grid consisting of a large number of smaller entities denoted cells. The common approach is to use triangular or quadrilateral cells in two spatial dimensions, resulting in wedges, pyramids and hexahedra in 3D. Due to the meshing mechanisms employed for this task, the grid forces the location of the respective cell centroids and the user has little control about the actual points where the flow properties are about to be stored. Besides that, the model operator must take reasonable care to align the computation grid with the streamlines in the flow domain to avoid seeing the result affected by a process called numerical diffusion, which will be subject to a detailed discussion later in this work. However, such an alignment is not always straightforward or even possible if a dominating flow direction cannot be identified.

This thesis proposes a paradigm shift in grid generation that comes at hand for circumventing some of the mentioned problems associated with widely used meshing techniques. It derives and prepares the required algorithms for creating computation grids based on point distributions given

by the model user, enabling the operator to be in full control over the storage locations of the flow properties he is interested in. The grid generator subsequently fulfils the task of creating a mesh using the given point set, applying rules of neighbourship as fundamental base for its workflow. This results in cells featuring an arbitrary number of edges in 2D and associated faces in 3D. These cells are based on logic generation rules and allow for the exchange of mass between a larger number of cells if an appropriate point distribution was chosen, thus reducing the negative effects of flows not perpendicular to cell faces. This can be advantageous in situations where no prevailing flow direction can be identified, as in recirculating flows or in the case of flows in floodplains when multiple streams interact with each other (*Tritthart & Milbradt* (2003) [81]), but also in any other flow situation exhibiting a strong secondary motion, as will be shown later.

1.2 Objectives and Outline

The prime objective of this work is to prove the feasibility of the polyhedral cell methodology in practical situations where turbulent channel or river flow is encountered. To get to this point, several other objectives must be met first. A first step is the design of algorithms for generating a polyhedral mesh and its subsequent software implementation. This is followed by the derivation of the generic discretised equations of flow and turbulence and their implementation in a numerical code, which is to be properly validated against a number of measurements in different flow situations. In the following thesis chapters all required mathematical derivations, as well as the results of validation and application runs are discussed, while the implementation is done in a software model called *RSim-3D*. This name is short for *River Simulation in 3D* and it consists of a pre- and postprocessor written in the Java programming language, hence allowing for a platform-independent usage, and a solver module, coded in GNU compliant C because of speed considerations. Due to all of these objectives, the work employs knowledge in the scientific fields of mathematics, geometry, informatics and hydraulic research alike.

The work is arranged into five core chapters, each representing a distinct step in model development. First of all, chapter 2 reviews a number of commercial and non-commercial 3D models for computational fluid dynamics, listing their numerical capabilities along with usual fields of application and past project references relevant for hydraulic engineering and research. At the end of this chapter, the RSim-3D model is positioned within the framework of these models to allow for a comparison.

In chapter 3, the design and application of polyhedral computation grids is discussed. Algo-

rithms for point distribution and grid generation are the core of this chapter, but it also discusses issues like grid refinement in practical situations and equations for obtaining cell volumes and surface areas in a geometrically complex grid configuration. For such a general grid requires a very general treatment of the governing equations of flow and turbulence, chapter 4 derives the discretised equations in an appropriate way. Furthermore this chapter outlines the boundary conditions of all flow properties required to obtain a solution, before theoretical and practical considerations about numerical issues like stability and convergence conclude that section.

The verification and validation of the model is subject to discussion in chapter 5. Validation is done by applying the model to four different flow cases: a wind-channel duct curved by 90 degrees is computed first, followed by a rectangular laboratory flume with an 180 degree bend, and subsequently a channel exhibiting a 270° bend. In the latter two cases, the used grid type is varied to assess its influence on the results obtained. Finally, an S-shaped trapezoidal channel is investigated to make a first step towards the modelling of realistic real-world flow situations.

The validation work of chapter 5 is followed by an exemplary application of the model to a reach of the river Danube in chapter 6. Finally, a summary and the discussion of possible future perspectives conclude the work.

2 Review of 3D CFD Programs for Hydraulic Engineering

2.1 Introduction

Getting an overview on the capabilities and implementation details of comparable academic and commercial software packages is an important first step towards the development of a new model. Therefore, fourteen different 3D CFD codes that can be applied to general problems in hydraulic engineering have been analyzed to determine their capabilities in this field. As there is constant evolution in the CFD business and new software is developed frequently, it cannot be guaranteed that this list is complete.

The information on the reviewed software packages was gathered from published literature and extensive inquiries on the Internet. Sometimes it was difficult to find precise specifications of the implemented methods and algorithms because this information was not disclosed to the public. Such a non-disclosure policy is found in commercial codes quite frequently. However, the issue that raised the most difficulty was to retrieve comparable price quotes for the different codes, as the pricing policy varies greatly among the companies who author the software packages. For some, an annual license fee applies, others offer perpetual licenses as well, and most of the time discounts apply for academic institutions. In order to solve this problem, *Olsen* (1999) [55] proposes a referencing system relating the software license cost to the price of computer hardware. If the software price is in the same category as a high-end UNIX workstation, it is being referenced as "relatively expensive" according to this scheme, whereas the price category of a regular desktop PC yields a "relatively inexpensive" software price. This referencing system is adopted in this work, using the terms Freeware, High-End and Low-End as classifiers.

Disregarding software packages that were developed for very specific applications, these programs were found to be applicable to tasks within the field of river hydraulics (listed in alphabetic order):

1. *CFX-5* by AEA Technology, UK
2. *Comet* by ICCM, Germany
3. *Delft3D* by WL | Delft Hydraulics, The Netherlands
4. *FEATFLOW* by the University of Dortmund, Germany
5. *FIDAP* by Fluent Inc., USA
6. *Flo++* by Softflo Corp., South Africa
7. *FLOW-3D* by Flow Science Inc., USA
8. *FLUENT* by Fluent Inc., USA
9. *NaSt3DGP* by the University of Bonn, Germany
10. *PHOENICS* by CHAM Ltd., UK
11. *SSIIM* by the Norwegian University of Science and Technology, Norway
12. *STAR-CD* by CD adapco group, UK/USA
13. *SWIFT* by AVL List GmbH, Austria
14. *TELEMAC-3D* by Electricité de France and HR Wallingford, France/UK

While most of the software packages implement different options only applicable to certain flow situations, there is a reasonable number of implementation characteristics common to all programs that can be used as criteria for comparison. The ones used in this review are:

- Operating System the software was written for,
- Method used for spatial discretisation of the partial differential equations (Finite Differences, Finite Elements or Finite Volumes),
- Grid types (structured and unstructured) as well as grid shapes (tetrahedra or hexahedra),
- Numerical scheme used for discretisation of convective terms,
- Numerical methods used for time discretisation,
- Methods to deal with the challenging task of coupling pressure and velocity,
- Implemented turbulence models, and
- Implementation of a free water surface.

2.2 Software Packages

2.2.1 CFX-5

Name of software	CFX-5
Author/company	AEA Technology, UK
Web page	`http://www.software.aeat.com/cfx`
Field of application	Mechanical, biomedical and process engineering; several successful applications to the field of hydraulic engineering are also known
Cost	High-End
Operating System	UNIX, Linux, Windows
Spatial discretisation	Finite Volumes
Grid types	Unstructured grid consisting of tetrahedral, hexahedral, prism and pyramid elements (triangles and quadrilaterals in plan view)
Numerical methods	Either a first order upwind scheme or a so-called "numerical advection corrected scheme" is being used for spatial discretisation along with a specially developed technique for pressure-velocity coupling. For time discretisation, a first order backward Euler scheme is employed.
Turbulence model	Zero-equation model, two kinds of k-ε models (two-equation), k-ω model, Reynolds stress model
Free surface	A fluid mixture model allows for computation of any kind of free surface conditions
Project references	Numerous real-world CFD application references in all branches are listed on the software's website. Among the ones relevant for hydraulic engineering are a scour study for a deep-water terminal jetty in India (by HR Wallingford, UK), the investigation of flow patterns within the vicinity of intakes (Hydroplan, UK), a natural river rehabilitation design study (University of Nottingham, UK), and also an analysis of turbidity currents in a lake (ETH Lausanne, Switzerland, referenced in *Olsen* (1999) [55])
References	*AEA* (2002) [1]
Remarks	The software is equipped with a wealth of different physical models to suit just about any kind of CFD problem in real-world applications. The successful application to a reasonable number of projects in the field of hydraulic engineering proves that the software is capable of solving this kind of problems, as well. However, the fact that the software price is in the high-end region makes it difficult for small or mid-sized businesses to utilize it.

Table 2.1: Software characteristics of CFX-5

2.2.2 Comet

Name of software	Comet
Author/company	Institute of Computational Continuum Mechanics GmbH (member of the CD adapco group), Hamburg, Germany
Web page	`http://www.iccm.de`
Field of application	Mechanical, chemical, environmental and hydraulic engineering
Cost	No recent price quote available
Operating System	UNIX, Linux, Windows
Spatial discretisation	Finite Volumes
Grid types	Unstructured mesh of hexahedra, tetrahedra and prisms
Numerical methods	Spatial discretisation is performed using one of the Upwind, Central, MINMOD or HRIC schemes with the SIMPLE solution method for pressure linkage. In terms of time discretization, fully implicit schemes of first (Euler) or second order are employed in the model.
Turbulence model	Zero-equation model, several types of k-ε models (two-equation), all types of k-ω models, Reynolds stress model
Free surface	Interface-tracking method
Project references	Several references mostly from the industries of mechanical and process engineering are listed on the software's website. The product is also being used at the Potsdam Model Basin (Schiffbau-Versuchsanstalt Potsdam GmbH) and at the Federal Waterways Engineering and Research Institute (BAW) in Hamburg.
References	Except for the manuals that come with the software, no publication related to the internals of the software could be found by the author.
Remarks	Comet is short for "Continuum Mechanics Engineering Tool", a general CFD code with most applications in the field of mechanical engineering. Physics are well represented in the software through numerous different models.

Table 2.2: Software characteristics of Comet

2.2.3 Delft3D

Name of software	Delft3D
Author/company	WL | Delft Hydraulics, Delft, The Netherlands
Web page	`http://www.wldelft.nl/d3d`
Field of application	Hydraulic engineering, in particular wave hydrodynamics, sediment transport and water quality investigations
Cost	High-End. A free evaluation version with limited capabilities is available.
Operating System	Linux, Windows
Spatial discretisation	Finite Differences
Grid types	Orthogonal curvilinear grid
Numerical methods	Alternate Direction Implicit (ADI) method for discretisation of the governing equations including transient terms
Turbulence model	Any choice of k-ε, k-L, algebraic or constant (zero-equation) models
Free surface	Hydrostatic pressure assumption, water surface appears as an unknown in the governing equations and is solved along with all other unknowns
Project references	Recent projects are studies of coastal hydrodynamics related to land reclamation for the new airport in Hongkong, the morphological development of the Dutch coast, and studies of Lake Malawi and Lake Victoria in Africa.
References	Besides the manuals that are supplied with the software, a short discussion of the software's internals is given in the M.Sc. thesis of *Luijendijk* (2001) [44]
Remarks	The software was particularly developed for hydraulic engineering and seems to be well suited for coastal hydrodynamics where it comes with a lot of experience. The hydrostatic pressure assumption, however, is not generally justified and can cause problems for example in river flow computations. Furthermore, a Finite Difference formulation is in general comparably fast but not always stable enough for any kind of problem in hydraulic engineering.

Table 2.3: Software characteristics of Delft3D

2.2.4 FEATFLOW

Name of software	FEATFLOW
Author/company	Department of Applied Mathematics and Numerics, Univ. Dortmund, head of group: Prof. Turek
Web page	`http://www.featflow.de`
Field of application	Unsteady flows of any kind that can be described by the Navier-Stokes equations
Cost	Freeware
Operating System	UNIX, Linux, Windows (Fortran 77 compiler is required because the software is distributed as source code only)
Spatial discretisation	Finite Elements
Grid types	Unstructured grid of tetrahedra and hexahedra (triangles and non-conforming quadrilaterals – predominantly the latter ones – in plan view)
Numerical methods	FEM for spatial discretisation, implicit scheme for time discretisation (choice between Backward Euler, Crank-Nicolson, Fractional Step θ-Method)
Turbulence model	None (implementation planned for future releases)
Free surface	Not implemented
Project references	Numerous academic applications, documented in the Virtual Album of Fluid Motion, available on the CD that is shipped with *Turek* (1999) [82] and on the Web page of the FEATFLOW group; project references in the field of mechanical and chemical engineering
References	*Turek* (1999) [82]
Remarks	FEAT is an abbreviation for "Finite Element Analysis Tools". The software appears to be predominantly suited for scientific purposes in research and teaching; good knowledge of numerics and the basic equations for CFD-computations is assumed for the application of the program.

Table 2.4: Software characteristics of FEATFLOW

2.2.5 FIDAP

Name of software	FIDAP
Author/company	Fluent Inc., Lebanon, New Hampshire, USA
Web page	`http://www.fluent.com`
Field of application	Mechanical, chemical, civil, biomedical engineering – all types of industrial CFD applications
Cost	High-End
Operating System	UNIX, Linux, Windows/NT
Spatial discretisation	Finite Elements
Grid types	Unstructured mesh of tetrahedra, hexahedra, pyramids and wedges (triangles and quadrilaterals in plan view)
Numerical methods	Finite Element Method for spatial discretisation, both explicit (Backward Euler as a first order scheme, trapezoid rule for second order accuracy) and implicit time discretization techniques are available
Turbulence model	Choice between mixing-length model (zero-equation model), four different k-ε models and the k-ω model by Wilcox (two-equation models)
Free surface	Volume of Fluid (VOF) approach available for large deformations and Arbitrary Lagrangian-Eulerian (ALE) method for continuous surface deformations
Project references	More than 50 working examples in all fields of application included with the software release, countless scientific papers on projects accomplished with FIDAP (many in the field of biomedical engineering)
References	*Fluent* (1998) [24]
Remarks	Very well tested software from a company with many years of experience in CFD. Includes numerous options to customize one's research parameters and choose between both different physical and numerical approaches for problem solutions. Extensive software documentation and tutorials. However, the high price tag makes the software unaffordable for small businesses.

Table 2.5: Software characteristics of FIDAP

2.2.6 Flo++

Name of software	Flo++
Author/company	Softflo Corp., Potchefstroom, South Africa
Web page	`http://www.softflo.com`
Field of application	Mechanical, chemical, biomedical and environmental engineering
Cost	Low-End. A free evaluation version with limited capabilities is available.
Operating System	Windows 95/98/NT
Spatial discretisation	Finite Volumes
Grid types	Unstructured mesh of hexahedral or prism cells
Numerical methods	Finite Volume method using the upwind scheme and employing SIMPLE and PISO algorithms for pressure-velocity coupling in spatial discretisation. A fully implicit technique is being used for time discretisation.
Turbulence model	k-ε model for high Reynolds numbers
Free surface	Implemented (technique not specified)
Project references	Several examples for applications of the program are presented on the software's website, however most of them are taken from the field of mechanical engineering.
References	Except for the manuals that come with the software, no publication related to the internals of the software could be found by the author.
Remarks	The software is not that expensive as comparable general purpose CFD codes. Since it is relatively new on the market – compared to other software – it is hard to find references related to experience with the model. The built-in physics, however, look quite promising as far as a successful application to the field of hydraulic engineering is concerned.

Table 2.6: Software characteristics of Flo++

2.2.7 FLOW-3D

Name of software	FLOW-3D
Author/company	Flow Science Inc., Santa Fe, New Mexico, USA
Web page	`http://www.flow3d.com`
Field of application	Free-surface problems in hydraulic and mechanical engineering
Cost	High-End
Operating System	UNIX, Linux, Windows
Spatial discretisation	Finite Differences
Grid types	Structured grid of rectangular shaped elements
Numerical methods	In both space and time, explicit methods are employed by default and implicit methods are available as option (unfortunately, no further detail about the used methods is available)
Turbulence model	Choice between Prandtl mixing length (zero-equation), a one-equation and two types of k-ε two-equation models
Free surface	Volume of Fluid (VOF) method
Project references	Numerous references to hydraulic engineering projects are available as web links on the software's website (e.g. Scribers Creek and Goldenrod Road Bridge by INCA engineers or a snow drifting analysis by the University of Narvik)
References	Several hundred publications of studies performed with FLOW-3D are listed on the software's web site, however, there was no publication found by the author that deals with the software itself
Remarks	When using the software, first a rectangular shaped grid is generated, then the solid boundaries are embedded within that grid. By using this approach, displacements in both the free surface and the (river) bed can be modeled easily. Therefore, the software is well suited for many hydraulic engineering problems (like weir flow, spillways or scour problems in hydraulic engineering).

Table 2.7: Software characteristics of FLOW-3D

2.2.8 FLUENT

Name of software	FLUENT
Author/company	Fluent Inc., Lebanon, New Hampshire, USA
Web page	`http://www.fluent.com`
Field of application	Mostly chemical and mechanical engineering
Cost	High-End
Operating System	UNIX, Linux, Windows
Spatial discretisation	Finite Volumes
Grid types	Unstructured grid consisting of any combination of tetrahedra, hexahedra, pyramids and wedges (triangles and quadrilaterals in plan view)
Numerical methods	Control-Volume spatial discretisation with central-differencing of the diffusion terms and several upwind-schemes (first order, second order, power-law, QUICK) are at the user's disposal. Time discretisation is performed by first and second order explicit and implicit methods upon the users choice.
Turbulence model	Choice between Spalart-Almaras model (one-equation), three different k-ε models and two kinds of k-ω models (two-equation)
Free surface	Volume of Fluid (VOF) approach implemented
Project references	Around 200 different application examples are well documented on the software's website. Of specific interest are the study of flow over a weir and an analysis of currents in drinking water reservoirs.
References	*Fluent* (2003) [25]
Remarks	Even though the software employs the physical Finite Volume approach, its application history seems to be fairly limited to chemical and mechanical engineering. The fact that the software is bundled with other software from Fluent Inc. makes it a good choice for those cases where FIDAP doesn't yield a result in appropriate time. The model and numerical approaches in the program are well tested and there is a reasonable amount of documentation. However, the price tag is too high to be of use for small businesses.

Table 2.8: Software characteristics of FLUENT

2.2.9 NaSt3DGP

Name of software	NaSt3DGP
Author/company	Department of scientific computing and numerical simulation, Univ. Bonn, head of group: Prof. Griebel
Web page	`http://www.wissrech.iam.uni-bonn.de/research/projects/koster/NaSt3DGP`
Field of application	Any type of general scientific 3D CFD problems
Cost	Freeware
Operating System	UNIX, Linux, Windows (C++ compiler is required because the software is distributed as source code only)
Spatial discretisation	Finite Differences
Grid types	Rectangular, non-uniform, staggered mesh
Numerical methods	Higher order upwind scheme, central difference scheme and first order upwind schemes available for spatial discretisation; explicit Adams-Bashford scheme (predictor-corrector method) for time discretisation.
Turbulence model	None (implementation planned for future releases)
Free surface	Not implemented in the standard distribution (level-set approach planned for future releases)
Project references	Three academic applications are documented: driven cavity flow problem, measurement equipment in pharmaceutical applications, odor modeling
References	*Griebel et al.* (1995) [31]
Remarks	NaSt3D is apparently an abbreviation for "Navier-Stokes 3D". The software seems to be best suited for solving academic problems in mechanical and chemical engineering but due to the used techniques it will probably deliver results very fast for just about any type of 3D CFD problem.

Table 2.9: Software characteristics of NaSt3DGP

2.2.10 PHOENICS

Name of software	PHOENICS
Author/company	Concentration Heat & Momentum Ltd, London, UK
Web page	`http://www.cham.co.uk`
Field of application	Mechanical, chemical, civil, environmental and hydraulic engineering
Cost	Low-End. Additionally to an ordinary licensing scheme, an old version is available as inexpensive shareware.
Operating System	UNIX, Linux, Windows
Spatial discretisation	Finite Volumes
Grid types	Structured hexahedral grid
Numerical methods	Spatial discretisation on the Finite Volume grid by linear (QUICK) or non-linear schemes (SMART, OSPRE), employing the SIMPLE solution algorithm
Turbulence model	Several zero-equation models (Prandtl mixing length among the better known), numerous different k-ε models, k-ω model, several other less popular methods
Free surface	Scalar-equation method (position of the free surface deduced from the solution of the conservation equation) and height-of-liquid method available
Project references	Numerous project references and validation cases are referenced on the software's website. Of specific interest are a study of flows in differently shaped drinking water reservoirs, an analysis of currents in a harbor, and the computation of oil spills into the sea.
References	*Spalding* (1986) [76]
Remarks	The name of the software is derived from "Parabolic Hyperbolic Or Elliptic Numerical Integration Code Series" which refers to the types of the underlying equations in general purpose CFD computations. It is on the market since 1981, therefore it can be considered to be very well tested and reliable. Many physical models, especially in turbulence modeling, are included in the software package. However, the structured grid approach is not as flexible as the grid types used by other software authors.

Table 2.10: Software characteristics of PHOENICS

2.2.11 SSIIM

Name of software	SSIIM
Author/company	Norwegian University of Science and Technology, Department of Hydraulic and Environmental Engineering, Assoc. Prof. N.R.B. Olsen
Web page	`http://www.bygg.ntnu.no/~nilsol/ssiimwin`
Field of application	Hydraulic, river and sedimentation engineering
Cost	Freeware
Operating System	OS/2, Windows
Spatial discretisation	Finite Volumes
Grid types	Structured hexahedral grid (version 1.1), unstructured grid of hexahedra and wedges (version 2.0)
Numerical methods	For spatial discretisation, both second-order upwind and power-law schemes can be chosen, pressure-correction is performed by the SIMPLE or the SIMPLEC method. Even though not specified in the software manual, *Olsen et al.* (1999) [59] indicates that time discretisation is performed by making use of an implicit technique.
Turbulence model	k-ε (two-equation) model
Free surface	Transient Free Surface (TFS) algorithm
Project references	In the software manual and on the website, a decent number of references to projects with SSIIM are given, among them are the analysis of secondary currents in a curved channel, a fish farm tank, a study of reservoir trap efficiency, a flood wave hitting a building, a scour in a flume, and several water quality computations for Norwegian lakes. Reservoir flushing studies were done by *Olsen* (2000b) [56] and *Tritthart* (2000) [79]. Recent work with the software is focused on submerged vegetation (*Fischer-Antze et al.*, 2001 [23]) , sediment transport and the evolution of meandering channels (*Olsen*, 2002 [58]).
References	*Olsen* (1999) [55], *Olsen* (2000a) [57]
Remarks	SSIIM stands for "sediment simulation in intakes with multiblock option" and refers to the software's original purpose. Successive improvements have made the software to become a CFD tool for many aspects of hydraulic and sedimentation engineering. Being not too difficult to use and equipped with concepts that are easy to understand, it is also well suited for beginners and students in the field of hydraulic CFD applications. However, as also stated in the manual, there are aspects of the software that are not very well tested and lack stability, sometimes also strange behavior of the graphical pre- and postprocessor may be experienced. But considering that the program is available as freeware, these little problems are more than excusable.

Table 2.11: Software characteristics of SSIIM

2.2.12 STAR-CD

Name of software	STAR-CD
Author/company	CD adapco Group (consisting of Computational Dynamics Ltd, London, UK and adapco, New York, USA)
Web page	`http://www.cd-adapco.com`
Field of application	Mechanical, biomedical, chemical and hydraulic engineering
Cost	High-End
Operating System	UNIX, Linux, Windows
Spatial discretisation	Finite Volumes
Grid types	Fully unstructured grid of tetrahedra and hexahedra (triangles and quadrilaterals in plan view)
Numerical methods	Control-Volume spatial discretisation approach with the SIMPLE method, using an automated technique that either employs central differencing or first order upwind differencing, depending on the level of numerical dissipation. For time discretisation, a fully implicit first order differencing scheme is employed.
Turbulence model	Smagorinsky model (zero-equation) and five different kinds of k-ε models (two-equation) are available
Free surface	Volume of Fluid (VOF) method
Project references	About ten project references for every single of eight different industry categories are made available via the software's website, adding up to almost one hundred references. Among the more interesting ones are the design of artificial reefs (Berlin University of Technology), design studies for weir shapes (University of Hannover), vortex modelling around pillars in rivers, the development of a fish guidance system at Bonneville dam (US Army Corps of Engineers) and a study of reservoir flows (Arup Corp.).
References	Except for the manuals that come with the software, no publication related to the internals of the software could be found by the author.
Remarks	The software is a general purpose CFD code that can also be applied to hydraulic engineering problems. The unstructured grid approach in combination with with free surface and turbulence modelling make it a flexible tool that appears to be quite popular in many industries, especially in mechanical engineering.

Table 2.12: Software characteristics of STAR-CD

2.2.13 SWIFT

Name of software	SWIFT
Author/company	AVL List GmbH, Graz, Austria
Web page	`http://www.avl.com`
Field of application	Mechanical, civil and hydraulic engineering
Cost	No recent price quote available
Operating System	UNIX, Linux, Windows
Spatial discretisation	Finite Volumes
Grid types	Fully unstructured grid of arbitrary cell types
Numerical methods	Control-Volume spatial discretisation approach with a variant of the SIMPLE method. For spatial discretisation, the available options are the first order upwind scheme, central differencing and two third order schemes (MINMOD and AVL-SMART). Regarding time discretisation, fully implicit first and second order differencing schemes are offered.
Turbulence model	The k-ε model (two-equation), the non-linear Reynolds stress model and also a hybrid turbulence model developed by the software authors are available.
Free surface	Volume of Fluid (VOF) method
Project references	About ten project references in different industrial fields can be found on the software's website. In this context, the more relevant ones include avalanche simulations (Federal Office and Research Centre for Forests, Austria) and flooding simulations (VRVis, Austria).
References	*Gouda et al.* (2002) [30]
Remarks	The software is a general purpose CFD code that can also be applied to hydraulic engineering problems. The arbitrary grid approach together with free surface and turbulence modeling make it a very promising tool for the application in complicated flow situations. Unfortunately, the number of project references is still not very high, but this aspect may change over time.

Table 2.13: Software characteristics of SWIFT

2.2.14 TELEMAC-3D

Name of software	TELEMAC-3D
Author/company	Electricité de France (Laboratoire National d'Hydraulique), Clamart, France, and HR Wallingford, Oxfordshire, UK
Web page	`http://www.wallingfordsoftware.com/products/telemac.asp`
Field of application	Hydraulic engineering (hydrodynamics, sediment transport and water quality in the natural environment: river, estuaries, coastal waters)
Cost	High-End
Operating System	UNIX, Windows NT
Spatial discretisation	Finite Elements
Grid types	Unstructured triangular grid (tetrahedra and prisms in 3D)
Numerical methods	Fractional step decomposition (advection step, diffusion step and free surface-continuity-pressure step)
Turbulence model	Prandtl mixing length (zero-equation) and k-ε model (two-equation) available
Free surface	Computation based on the hydrostatic pressure assumption, one separate step in the overall numerical method
Project references	HR Wallingford lists seven different real-life projects that have been done using TELEMAC on its website (mostly marine/coastal applications) plus around a dozen companies and organisations that use the TELEMAC software package.
References	*Hervouet et al.* (1994) [35], *Anderson* (2000) [5]
Remarks	The software was specifically designed for hydraulic engineering and proved its usefulness in this field for many years in numerous applications. Due to this approach, it does not contain so many features which general purpose CFD codes must possess, a fact that makes it even more useful for the hydraulic engineer. Furthermore, it appears to produce very good results in river and coastal engineering, even though some of its assumptions (i.e. the hydrostatic pressure assumption in 3D) are not always the best fit for true physics in nature.

Table 2.14: Software characteristics of TELEMAC-3D

2.3 Summary

Table 2.15 provides an overview of the capabilities of all fourteen reviewed software packages, with the newly developed *RSim-3D* model added to allow a comparison.

More than half of all models operate with the Finite Volume approach, both Finite Element and Finite Difference techniques each make up for a quarter of the total number. Around two-thirds work on unstructured grids, with more than 90 percent using at least hexahedral shaped elements and more than half additionally allowing tetrahedra for spatial decomposition.

The usage of numerical methods and algorithms for both space and time discretisation is highly inhomogeneous and doesn't allow to draw conclusions about preferred techniques. It should be mentioned that two out of the fourteen models operate using a hydrostatic pressure assumption which makes them actually only quasi-3D applications that do not allow for computation of several phenomena.

Two mostly academic products do not account for turbulence at all, a fact that restricts their application to laminar flows. The other codes implement at least one two-equation turbulence model, with the k-ε model being by far the favourite technique. More than half of all products additionally allow usage of zero-equation models, and still almost 50 percent come equipped with other techniques that are mostly based on higher dimensional stress formulations.

More than 80 percent of all models come with the ability to model free surface flows, again mainly the academic codes do not have this feature built in. Techniques for free-surface implementation vary greatly, with the VOF (Volume of Fluid) being the favourite.

	CFX-5	Comet	Delft3D	FEATFLOW	FIDAP	Flo++	FLOW-3D	FLUENT	NaSt3DGP	PHOENICS	RSim-3D	SSIIM	STAR-CD	SWIFT	TELEMAC-3D
Software author															
Academic / Commercial	C	A/C	C	A	C	C	C	C	A	C	A	A	C	C	C
Operating System															
UNIX	x	x		x	x		x	x	x	x	x		x	x	x
Linux	x	x	x	x	x		x	x	x	x	x		x	x	
Windows	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
OS/2												x			
Spatial discretisation															
FDM / FEM / FVM	V	V	D	E	E	V	D	V	D	V	V	V	V	V	E
Grid types															
Structured / Unstructured	U	U	S	U	U	U	S	U	S	S	U	U	U	U	U
Grid shapes															
Tetrahedra	x	x		x	x			x			x		x	x	x
Hexahedra	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
Numerical methods - space															
Central Differences		x						x	x				x	x	
Upwind first order	x	x				x		x	x				x	x	
Upwind second order								x	x		x	x			
QUICK scheme								x		x					
Other	x	x	x	x	x		x	x		x		x		x	x
Numerical methods - time															
Explicit first order					x		x	x							
Implicit first order	x	x		x	x	x	x	x				x	x	x	
Implicit second order		x		x	x		x	x						x	
Other (or no implementation)			x	x					x	x	x				x
Pressure-velocity coupling															
SIMPLE		x				x				x	x	x	x	x	
Other	x			x	x	x	x	x	x			x			
Hydrostatic pressure assump.			x												x
Turbulence models															
zero-equation	x	x	x		x		x			x			x		x
k-ε	x	x	x		x	x	x	x		x	x	x	x	x	x
k-ω	x	x			x			x		x					
Other	x	x	x				x	x		x				x	
Free surface															
Implemented	x	x	x		x	x	x	x		x	x	x	x	x	x

Table 2.15: Comparison of model characteristics

3 Polyhedral Computation Grids

3.1 Background and Fundamentals

3.1.1 Conventional Computation Grids

In typical three-dimensional hydrodynamic simulations, grids based on tetrahedra or hexahedra are employed (see chapter 2). Very often these computation grids are assembled by meshing the domain in 2D and subdividing the resulting cell piles into several smaller entities. Since this approach is also followed in the present work, this chapter will first discuss the process of two-dimensional domain meshing before moving on to three-dimensional grids.

The conventional way of meshing multidimensional domains in 2D is to use triangles (fig. 3.1) and quadrilaterals (fig. 3.2). As a rule of thumb it can be stated that quadrilaterals are frequently used in Finite Volume codes, while triangles are the shape of choice in software packages based on the Finite Element formulation – even though there are some exceptions to this. There is no common standard as to how quadrilateral cells are formed, except for the fact that the longitudinal sides are aligned with the expected main flow direction to avoid solutions being spoiled by false diffusion, which is further discussed in section 5.3. Hence, grids based on quadrilateral cells typically look like the one depicted in figure 3.2 which constitutes a stretch of the river Danube east of Vienna.

Triangular cells, on the other hand, are usually generated by a procedure denoted *Delaunay triangulation* (*Wilhelm* (2000) [91]). Implementation and algorithmic details of this method are discussed in *Shewchuk* (1996) [73]. Figure 3.1 shows a detail of a computation grid that was created using this triangulation method; the grid was employed to analyse the August 2002 flood events in Lower Austria (*Tritthart & Milbradt* (2003) [81]). The principle of the Delaunay triangulation is that exactly one triangular element results when a circle is drawn through three points out of a set of base points while no other point lies within the same perimeter. Therefore

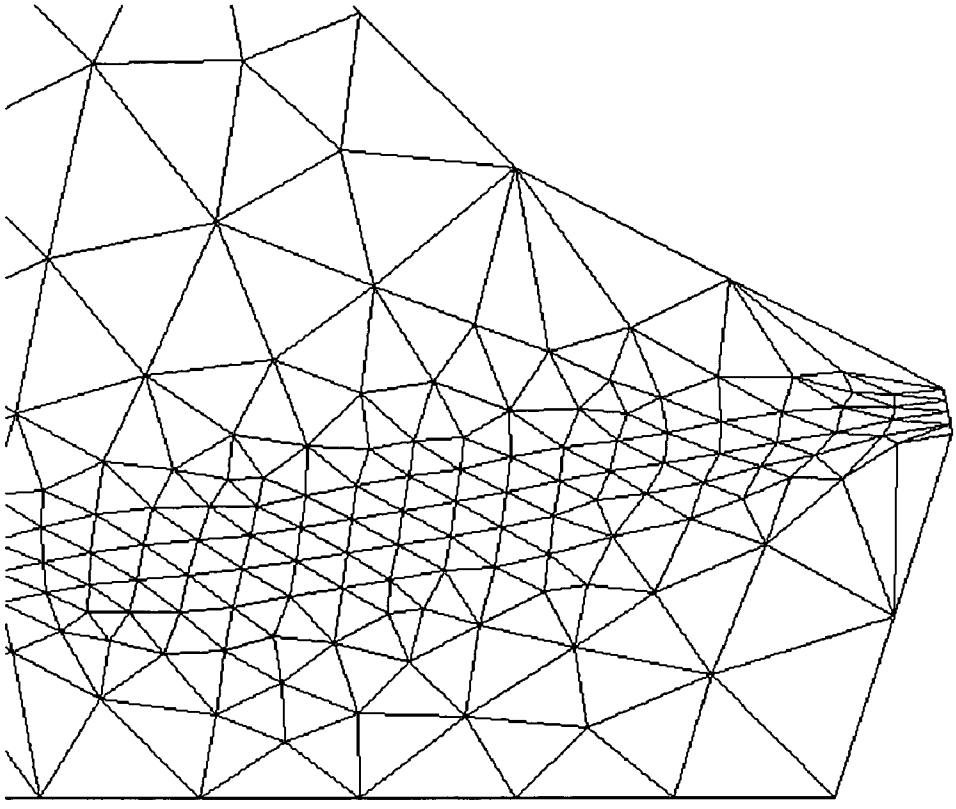

Figure 3.1: Example of a grid using triangular cells

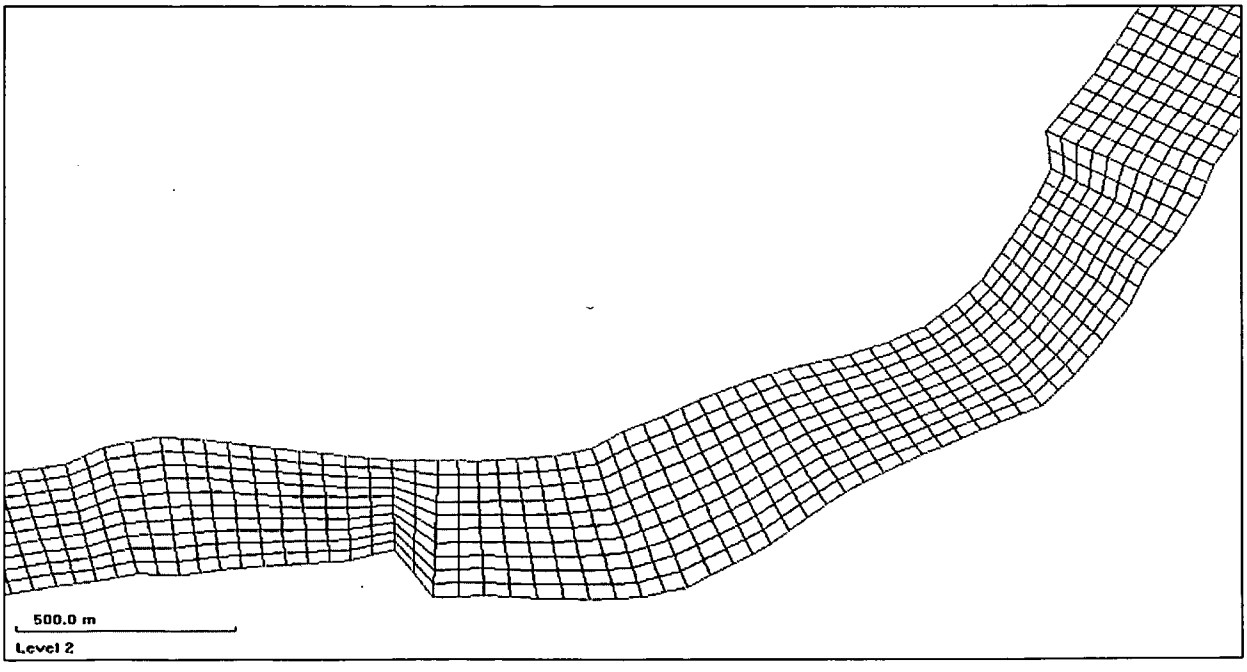

Figure 3.2: Example of a grid using quadrilateral cells

always the nearest neighbours of a point make up a triangle. Further details of this method will not be discussed here since they are out of the scope of the present work.

As opposed to grids based on triangles or quadrilaterals, the application of grids with cells characterised by more than four edges is rarely found, even though some commercially available numerical codes (e.g. *Fluent* (2003) [25]) are capable of dealing with cells of such shape. *Standingford & Forth* (2003) [77] have described the use of polygonal bounded cells in two spatial dimensions in an aerospace application, using the CFD code FLITE3D. *Creswell & Croaker* (2003) [14] presented a study using polygonal bounded Finite Volumes in three spatial dimensions (i.e. *polyhedral* cells) to deal with different length scales within a computational domain, computing internal air flow in a large warehouse with small windows which are represented by several faces of a larger control volume, thus avoiding the use of an impractically high number of elements to solve the problem. However, in the field of river hydrodynamics, no use has been made of this technique so far (*Tritthart* (2004) [80]).

3.1.2 Voronoi Decomposition

As long as the unknowns of a numerical simulation are stored in cell centroids, the user normally has little control over the exact location of these variables. A remedy is to store the unknowns in the vertices of a computational grid, but there are some problems involved with this approach as soon as the cells become more complex in overall shape, as will be discussed in chapter 4. Another approach is to define the location of the cell centroids in a first step and construct the grid around them afterwards. This results in cells of complex shape, being polygonally bounded in 2D and polyhedral in 3D. It is, however, a paradigm shift compared to the common way of grid generation, giving the user full control over the location of the conservation quantities within the computational domain.

It improves the overall behaviour of the numerical solution process if a cell's boundary line lies exactly in the middle of a line connecting two neighbouring cell centroids. This results from the fact that the face values of the conservation quantities can be obtained without the necessity of *weighted* interpolation, as it is shown in chapter 4. The numerical behaviour is obviously further improved when the boundary line is exactly perpendicular to the connection line between cell centroids since it avoids the need to transform both convective and diffusive fluxes. In contrast, if there is a severe non-orthogonality between the connection line and cell boundary lines, non-orthogonal terms must be introduced into the discretised equations, which is further discussed in *Davidson* (1996) [16] and also in the subsequent chapter of the present work. Hence, a spatial

discretisation or meshing algorithm should take care to avoid non-orthogonal cells (i.e. thin stretched triangles) but produce cells conforming to the orthogonality constraint instead.

This constraint is automatically preserved if quadrilateral cells with approximately parallel edges (i.e. cells of rectangular shape) are being used. However, alignment with the main flow direction is mandatory for such a grid setup. It is therefore not a feasible approach in all those cases where a main flow direction is not easy to determine a priori. A remedy to this problem is to introduce a grid based on a *Voronoi decomposition* of the computational domain. This graphical method was first published by G. *Voronoi* [88] in 1908 and is today frequently used in other sciences: hydrology uses the method to obtain the Thiessen polygons surrounding rain gauges; in geography the method is applied to find the region of influence of municipalities.

Graph theory defines the Voronoi decomposition as the *dual graph* to the Delaunay triangulation (*Frank* (2002) [28]). In other words, there exists a unique relation that allows to construct each graph as soon as the other one is known. Hence, it is sufficient to store only the elements composing one of these graphs while the other one can be computed on the fly with very little computational effort. This makes it possible to use the grid based on the Voronoi decomposition to perform the numerical computations while the dual grid is used for interpolation of terrain and water surface elevations – an approach that was used in the present work.

A technical definition of the Voronoi decomposition is given by *Milbradt* (2001) [47]. According to this definition, the Voronoi decomposition is the segmentation of the entire domain based on *neighbourship* of a given set of base points p. Neighbourship is defined by a distance function relating two points; usually the Euclidean norm is employed for this purpose. The nearest neighbour of a given point x is then the reference point p where the distance function becomes a minimum within the full set of points. Several different points x will therefore possess a common nearest neighbour p; the set of points given by this criterion is denoted *region*. For every region a boundary and subsequently neighbouring regions can be defined. In the present work, the regions are the two-dimensional representations of control volumes in the Finite Volume method while their boundaries are the cell edges. The term *region* will therefore be used throughout this work to refer to the 2D projection of cells on the x-y plane. Figure 3.3 illustrates this on the basis of four points that have been arranged in such a way that hexagonal Voronoi regions emerge (one of which has been marked in blue); the figure was plotted under the assumption of a boundary constraint following its contour. Superimposed on the Voronoi regions are the Delaunay triangles (one was coloured in red). It can be seen that the base points always represent the centre of the regions while being the vertices of the Delaunay triangles at the same time. Edges of Voronoi regions and Delaunay triangles are always perpendicular to each other.

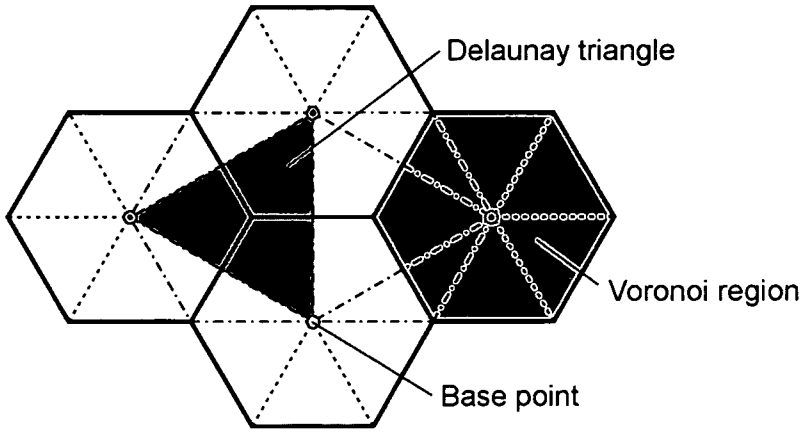

Figure 3.3: Voronoi regions and Delaunay triangles

After the fundamentals of the Voronoi decomposition have been discussed, it is now possible to assess this method with regard to the criteria a computational grid should match for optimum performance:

- *Orthogonality*: The edges of computational cells are always perpendicular to the connection line between two cell centroids; hence, non-orthogonal terms in the discretised equations can be dropped.
- *Unweighted interpolation functions*: The edges of computational cells are always located right in the middle of the connection line of the cell centroids. Therefore the use of weighted interpolation functions becomes unnecessary.
- *Absence of numerical diffusion*: If the computation points are distributed in a deliberate way, cells possessing a larger number of edges result (see chapter 3.2). Since these cells allow for fluxes in more than two main directions, numerical diffusion is reduced by a fair amount.

Literature discusses many different approaches to construct Voronoi decompositions or Voronoi diagrams in general. The most common ones are:

- *Plane intersect method*: This is the straightforward way to constructing a Voronoi diagram. For each point in the total set the bisection line with every other point is computed. This results in a number of half-planes which must be merged. The process must be repeated for each and every point site in the plane (*Viermetz* (2001) [85]). This method is very

inefficient since it is easy to figure out that it operates on the order of $O(n^2)$, where n denotes the number of points.

- *Plane sweep method*: This algorithm is an enhanced version of the previous method for it introduces the concept of incremental generation to minimise redundancy. The set of points is sorted along the positive x-axis in a first step; afterwards always the next point site in the list is inserted into the diagram. This approach minimises the number of required cutting operations and yields an average complexity of $O(n \log n)$, but of course the worst case complexity remains at $O(n^2)$.

- *Divide and conquer algorithm*: This method is well known and widely used. It enhances the plane sweep method by not only inserting point after point into the existing diagram, but rather several point sites already merged into a Voronoi diagram (*Viermetz* (2001) [85]). The algorithm consists of two steps: in the dividing step, the sites in the plane are divided into two halves along a successively evolving bisection line; this procedure is subject to recursion for every subset until only two or less elements are left. As a matter of fact, the result of this step is a binary tree containing very simple Voronoi diagrams in its leaves. These diagrams are then merged in the conquering step to yield the entire Voronoi diagram. The algorithm has a complexity of $O(n \log n)$.

- *Fortune's algorithm*: This algorithm was proposed by *Fortune* [26] in 1986 and is the most efficient of all algorithms as it guarantees a worst-case performance of the order $O(n \log n)$, i.e. in general situations it will operate faster than that. It is the algorithm of choice for the present work, hence it will be explained in more detail in the following section.

3.1.3 Fortune's Algorithm

The algorithm's underlying idea is to interpret the task of constructing Voronoi diagrams as the two-dimensional projection of a three-dimensional procedure. First, a cone with an apex angle of 45 degrees is constructed on each point site in the x-y plane (fig. 3.4). Afterwards, a plane π slanted at 45 degrees is moved along the y-axis of the coordinate system. The intersection line of this plane with each individual cone yields a parabola curve, if projected onto the x-y plane. However, the intersection of two parabola curves is identical to a point of the Voronoi line, defining the boundary between two regions. While the plane π is now dragged through the domain, complete Voronoi lines result that can be stored in an appropriate data structure.

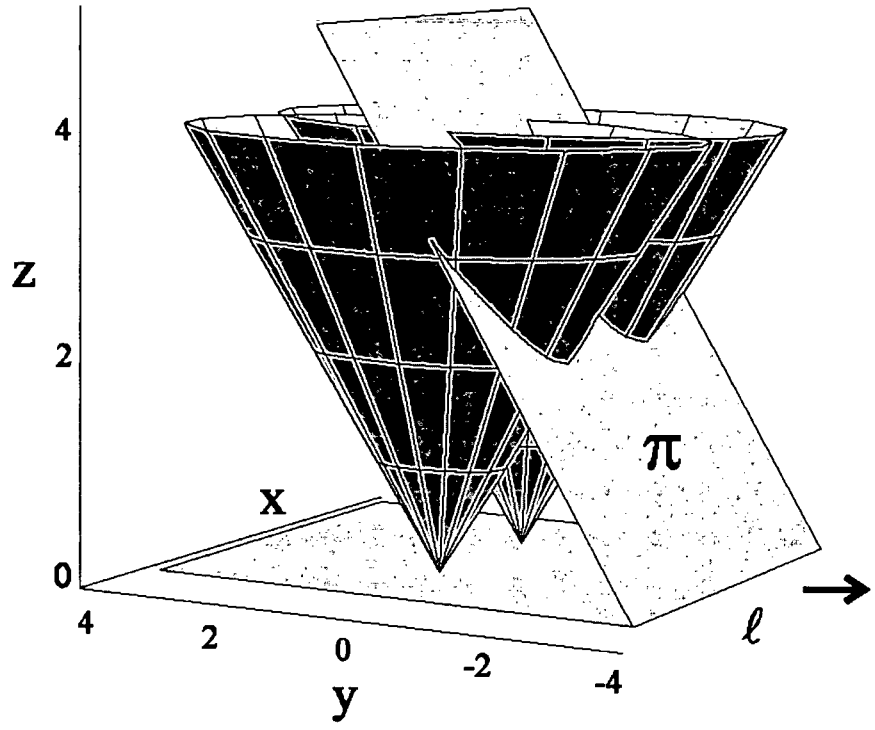

Figure 3.4: Three-dimensional interpretation of Fortune's algorithm (*Cuk* (1999) [15])

The base line of the slanted plane is denoted *sweep line*, whereas the intersection curve of the cones with the plane π is referred to as *parabolic front* or *beach line* (*Wilhelm* (2000) [91]). While the sweep line moves along the x-y plane, the beach line is subject to constant modification. However, there are two sorts of distinctive events that can arise during this procedure: point events and circle events.

Point Event

A point event is encountered when the sweep line has hit a new point p in the plane. Resulting from this, a new parabolic arc appears on the beach line. This is illustrated in figure 3.5: in (a), the sweep line has not yet encountered the new point; in (b), the sweep line is exactly at the site of the new point and the parabolic segment is inserted, even though it degenerates to a straight line at this moment; in (c), the sweep line has passed the point and the beach line is in regular shape again, containing the new parabolic arc. For the joint point of two arcs in the beach line defines a Voronoi line, the point event inserts a new vertex into the Voronoi diagram.

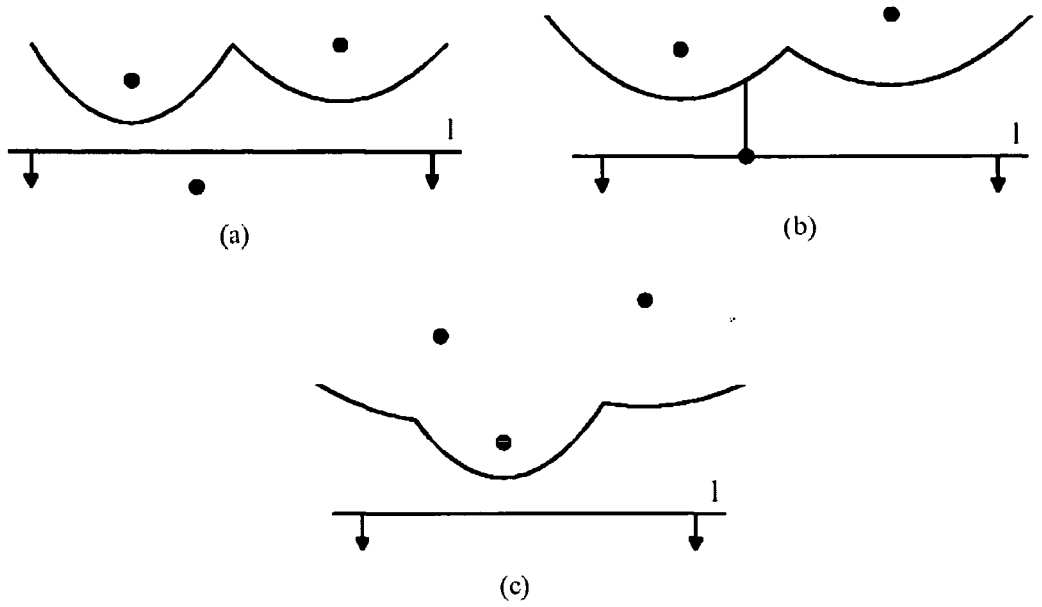

Figure 3.5: Changes in the beach line upon encountering a point event (*Wilhelm* (2000) [91])

Circle Event

A circle event takes place when a parabolic arc shrinks to a point and disappears from the beach line. The condition for the occurrence of this event is that three parabolic arcs – defined by three base points p_i, p_j, p_k – intersect each other in a single point q, as illustrated in figure 3.6. This happens when q has the same distance to the sweep line as to the three base points. In that case, all base points lie on an empty circle with centre point q and the sweep line is tangent to that circle, hence the name of this event. As a consequence, point q is a vertex in the Voronoi diagram where two Voronoi lines intersect.

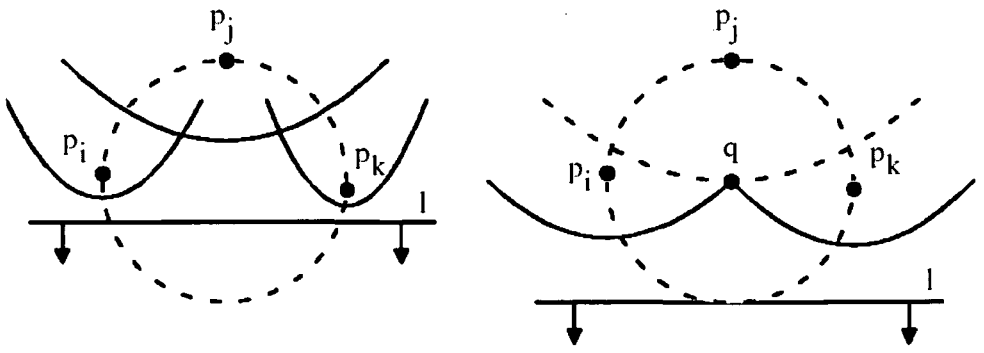

Figure 3.6: Changes in the beach line upon encountering a circle event (*Wilhelm* (2000) [91])

Implementation

By testing for point and circle events while the sweep line moves through the entire domain, it is possible to construct the complete Voronoi diagram in a very efficient way. Actually, *Fortune* (1986) [26] proves that this algorithm is optimal, i.e. the task of computing the Voronoi diagram cannot be done with a better performance. The actual implementation, however, is a challenging task since all elements involved in the generation of the Voronoi diagram using Fortune's method must be stored in appropriate memory structures that need to be dynamically allocated.

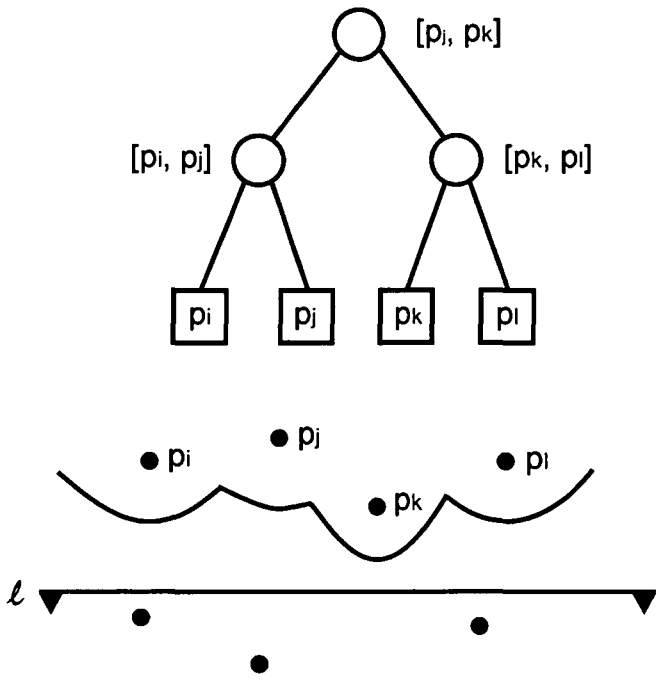

Figure 3.7: Binary tree to represent the beach line

Implementation details are discussed in *Cuk* (1999) [15], *Fortune* (1992) [27], *Münch* (1998) [49] and *Wilhelm* (2000) [91]. There is consensus that three data structures are required: one for storing the Voronoi diagram and two others for the sweepline process, i.e. point/circle events and the parabolic front. The data structure for the Voronoi diagram, modified to suit the needs arising in the present work, is discussed in chapter 3.3. As far as the parabolic front is concerned, a binary tree is best suited for storing its contents. Such a structure is a very natural way to represent data in an object-oriented programming approach, furthermore it allows for fast updates of its contents (i.e. when new elements are inserted or old ones removed). The binary tree structure applied to the parabolic front is illustrated in figure 3.7. Finally, upcoming events are stored in an event queue where the different events are stored by the point sites they refer to.

It should be noted that a usual Voronoi diagram contains half-edges, denoting lines that have a start point but no end point. The unmodified algorithm of Fortune naturally returns such elements as parts of the solution. However, the computational domain in hydrodynamic problems is bounded by definition. Hence, Fortune's algorithm has to be modified to compute the intersection points of half-edges with the domain boundary and include these, as well as the segments of the bounding polygon, into the solution. The resulting domain decomposition is therefore no longer an actual Voronoi diagram but represents a *constraint Voronoi* decomposition.

3.2 Modular System

3.2.1 Background

Up to now, we only dealt with the generation of a Voronoi grid and treated the set of base points as already known. As this is not the case in reality, a mechanism of point generation must be found. For this purpose, the aim is the development of an automated distribution algorithm. However, there are a number of constraints that must be accounted for:

- The grid resulting from the Voronoi decomposition with regard to the distributed set of points must be as regular as possible, without large differences in size among single cells,
- The grid must honour the boundary line of the computational domain, following its course and allowing for a finer discretisation in this region,
- The grid must honour structure lines (for instance levee crests) to avoid wrong terrain interpolation (i.e. "breaches") in regions where structures must be preserved.

It is possible to construct a grid that conforms to all these constraints when a system of three modules is employed for the distribution of base points:

- a *base module*, where points are distributed in a general pattern, forcing specific cell shapes,
- a *boundary module*, where the base points follow the course of the boundary line, hence avoiding the occurrence of irregular cells at the border of the domain and allowing for a finer discretisation there,

- a *structure line module*, where the base points follow the course of a structure line in such a way that the structure line itself is represented by cell edges (or cell faces in a three-dimensional situation).

The properties of these modules will be discussed in the next sections.

3.2.2 Base Module

This module is responsible for the distribution of base points in regions far away from a domain boundary or structure line. As illustrated in figure 3.8, points are distributed starting at the origin of the coordinate system that was rotated by an angle Θ. It is possible to force specific cell shapes by certain point distribution patterns. The RSim-3D model employs only quadrilateral and hexagonal patterns, but in general it is possible to construct cells with a larger number of faces, as well. The quadrilateral pattern is given by two spatial distances, Δx and Δy (fig. 3.8, left), from which an equidistant distribution is obtained. However, in the hexagonal pattern (fig. 3.8, right), the distance Δy is no longer subject to arbitrary choice, but it is derived from the equation

$$\Delta y = \Delta x \cdot \frac{\sqrt{3}}{2} \tag{3.1}$$

which defines an equilateral hexagon. The hexagonal cell shape is finally obtained after applying an offset of $\Delta x/2$ to the lateral distance in every second row of points.

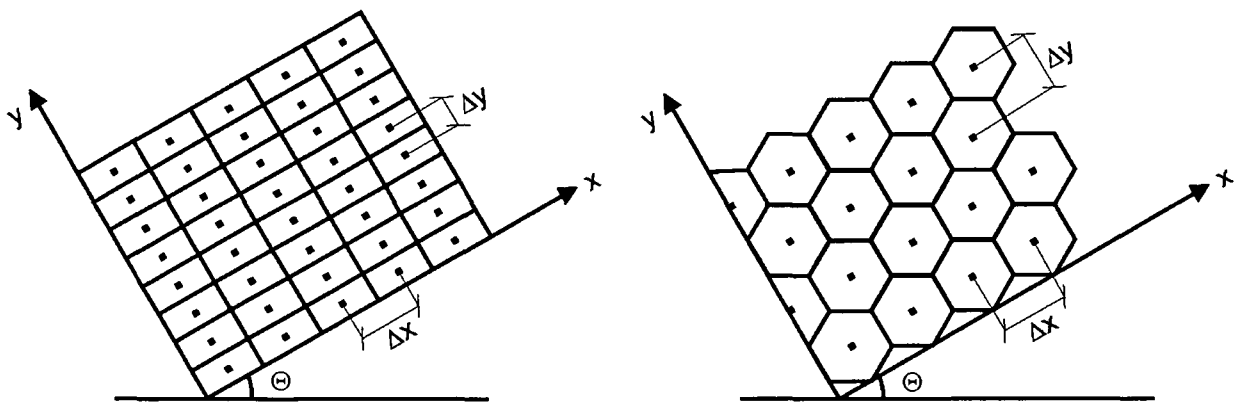

Figure 3.8: Quadrilateral and hexagonal base modules

3.2.3 Boundary Module

The boundary module is used to allow for generating a grid that follows the boundary of the computational domain in its course. This is of high importance in practical situations since it avoids irregularly shaped grid cells near the boundary by ensuring that all cells have the same distance to the border and that cell edges intersect the boundary at an angle of 90 degrees – at least in a quadrilateral configuration. Furthermore, the boundary module makes it possible to apply a finer spatial discretisation in that region, which will almost always be desired. Finally, it is also possible to create a body-fitted grid in the whole computational domain by making use of this module only. The generation process is simple: the boundary polygon is offset by a distance $\Delta y \cdot \left(i - \frac{1}{2}\right)$ where i denotes the row number, and points are distributed along that line. This ensures that the base points of the first row are always located at half the grid spacing distance. Care is taken to compute the correct end points of line segments in "corners" of the flow domain, i.e. where the angle between line segments is not 180 degrees. To illustrate this, figure 3.9 shows an exemplary grid in a circular domain, constructed from a hexagonal base pattern and four rows of hexagonal boundary elements.

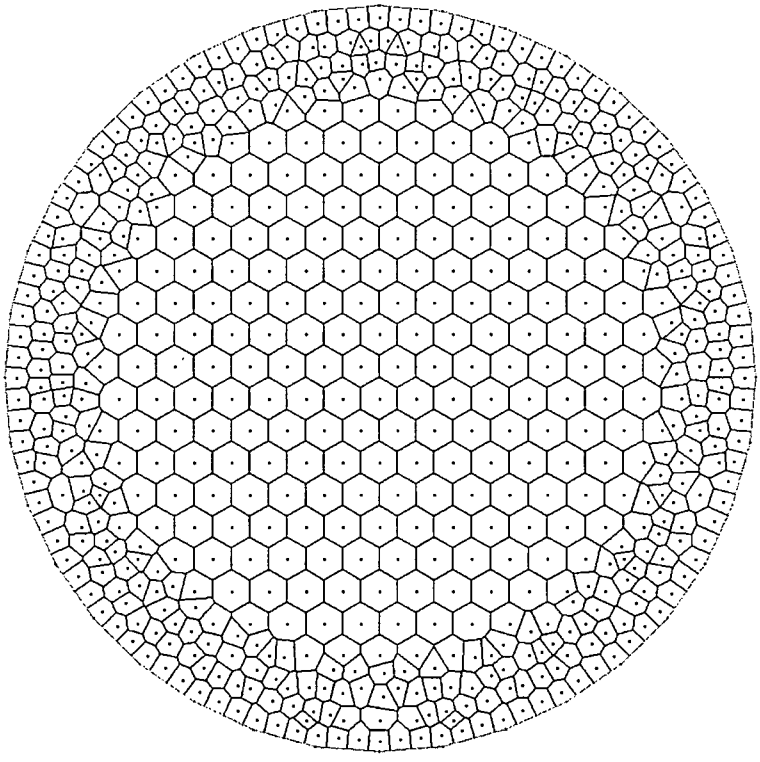

Figure 3.9: Grid composed of hexagonal base elements and four rows of boundary elements

However, care must be taken in the choice of grid spacings and distribution patterns to avoid the creation of distorted elements. Figure 3.10 shows an exemplary grid in a box, based on four rows of the boundary module in a quadrilateral configuration while the centre of the box is filled with regions created by the base module. This grid would not be used for actual hydrodynamic simulations since there is a small number of cells at the transition zone from one module to the other that exhibit computation points which are not close to the cell centre, hence impairing convergence.

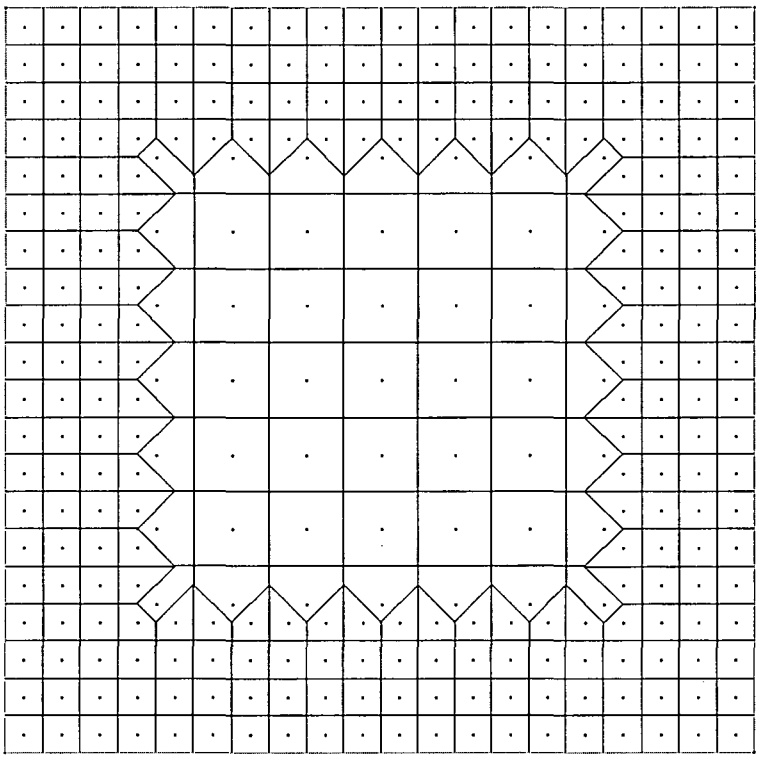

Figure 3.10: Grid in a box, composed of quadrilateral boundary and base elements

3.2.4 Structure Line Module

Structure lines are applied when the meshing algorithm must preserve specific edges, including them as part of the grid. For instance, this is desired when man-made structures (e.g. levees) are to be represented in a numerical simulation. If these structures are not well preserved, the solution of the simulation may turn out numerically correct, but technically wrong (i.e. "breaches" in dams, resulting from wrong interpolation, leading to flooding of terrain which would otherwise

not have been flooded). Fortunately, structure lines can be preserved in the same way as boundaries by the boundary module: at the border, the bounding polygon is offset into the domain by a certain distance, and the points are distributed along that line (section 3.2.3). However, a structure line polygon must be offset to *both* sides at the same distance when distributing points. As Voronoi edges are always located half-way between two base points, the structure lines will be automatically preserved following this approach. This allows for an interpretation of the structure line module simply as a boundary module being applied twice, at either side of the dividing polygon line.

3.3 Data Structure

After the base points have been distributed in the computational domain and the two-dimensional grid lines have been created, the resulting data must be stored in an appropriate data structure. This structure must be designed to follow two major criteria:

- *flexibility*: the data structure must be capable of dealing with regions of all shapes, regardless of the number of edges,
- *no redundancy*: the entire structure must allow for quick access to all data, but at the same time minimise redundancies to allow for fast and correct updates of elements if needed.

Data to be stored can be categorised in four different groups (fig. 3.11):

- *regions*: these are the actual two-dimensional projections of the 3D grid onto the x-y plane.
- *base points*: these 3D points define the computational centre of the regions; the third component stores the terrain surface.
- *grid lines*: these lines represent the border between regions.
- *vertices*: these are points that define start and end coordinates of grid lines.

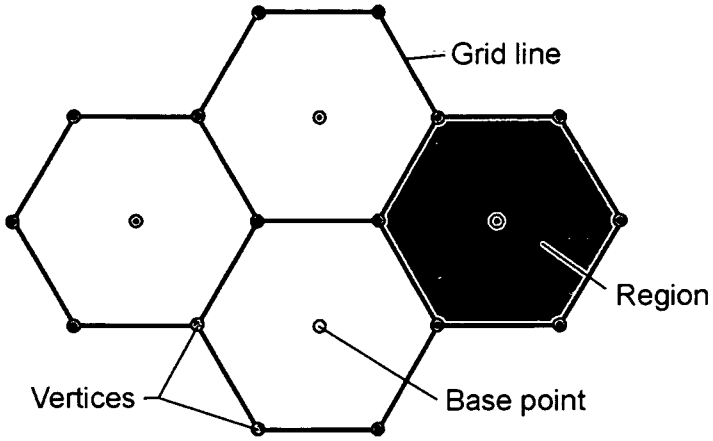

Figure 3.11: Elements of a two-dimensional grid

Figure 3.12 illustrates how the data elements can be arranged into a data structure containing the complete two-dimensional projection of the computation grid. It also lists the data types, using these abbreviations:

- *boolean*: a binary data type that holds an argument of either 0 or 1
- *int*: an integer value
- *float*: a floating-point number in unspecified precision
- *Point3D*: an object-oriented data type, consisting of three *float* elements x, y and z
- *Vector | type*: an array containing an unspecified number of *type* elements, i.e. its size may change at any time if needed

There are a number of issues concerning this data structure that should be noted:

- The 3D coordinates of the *base points* contain the exact location of a point at the terrain surface (e.g. river bed); refer to section 3.4 for details about obtaining the vertical elevation.
- The vast majority of *vertices* does not contain a third 3D coordinate argument, i.e. it is zero or unused. However, interpolation at the boundary and along structure lines is only possible if the vertices in these locations are assigned terrain elevations. Hence, the overall data type must be a point in 3D space, even though the third argument is only used for a small number of vertices.

- Both *boundary* and *structure lines* are identified by a separate boolean value. This is necessary since due to the preprocessing of data, two polygon lines with different meaning may lie on top of each other. However, for the actual treatment of these elements this distinction does not make a difference.

- In a usual implementation, the *index-numbers* of all types are not encoded in an extra field; in order to save memory, the elements are addressed only by their position in memory. This is legitimate since the storage sizes are known and the offset of any given element can be computed easily on the fly.

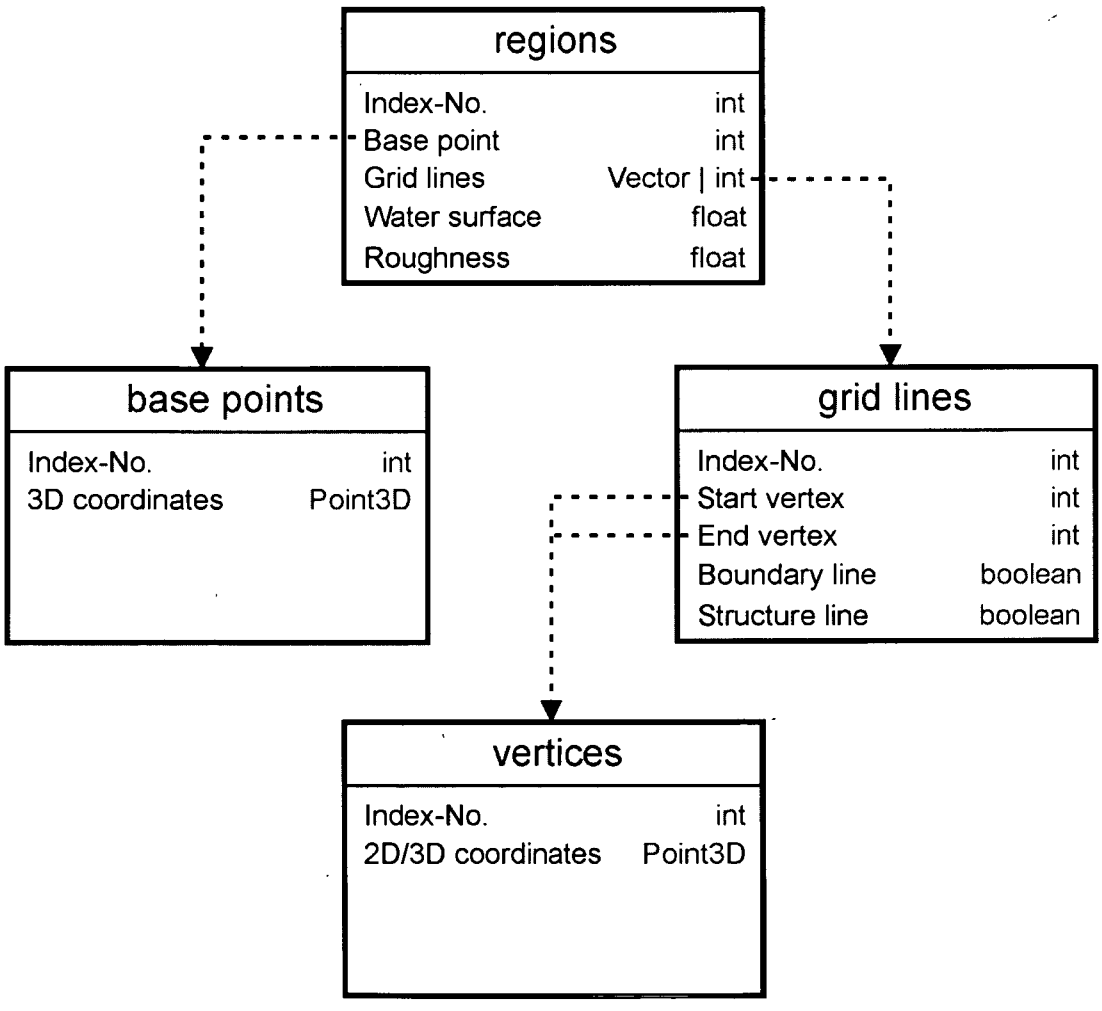

Figure 3.12: Data structure of a two-dimensional grid

3.4 Terrain Elevation

3.4.1 Background

Usually, the geodetic height of the 2D computation points is not known a priori, but either depth measurements along cross section lines of a river bed or a digital terrain model of regularly spaced data are available. Since these spatial data points rarely coincide with the actual base points of the computation grid, a technique of spatial interpolation is required. Both the Bivariate Interpolation Method of *Akima* (1978a, 1978b) [2, 3] and a Kriging approach were evaluated for that purpose and are subject to discussion in the following two subsections of this work. Following this approach, surface elevations can be derived for all computational points and those grid line vertices that are part of the boundary polygon or a structural polygon within the computational domain (*Tritthart* (2004) [80]).

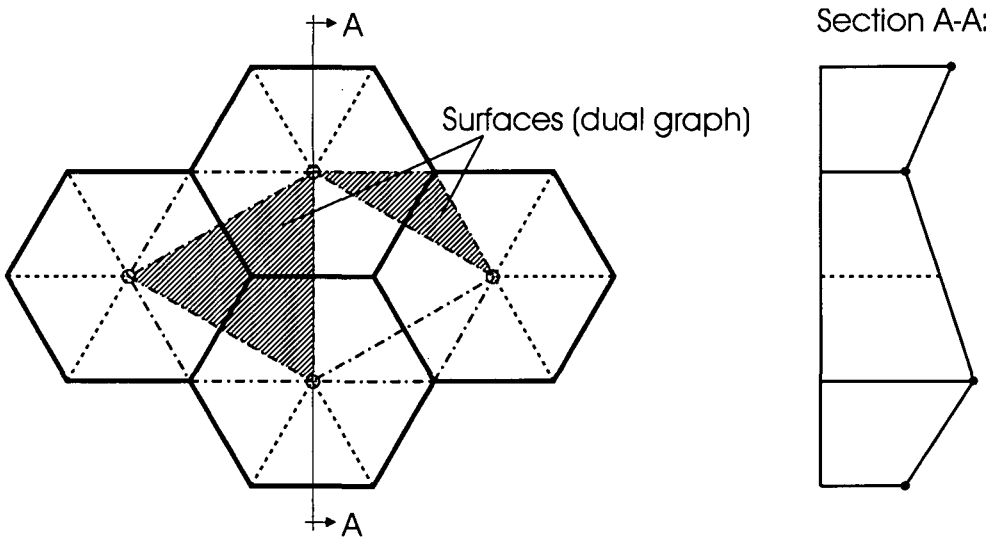

Figure 3.13: Voronoi grid and Delaunay triangles with a section view

Figure 3.13 once again depicts the plan view of a domain represented by four hexagonal cells; here it is presented along with a cross section. Let's assume that the cells' base points have been set a vertical elevation following one of the approaches just mentioned. Now the problem arises to make the surface (actually both the terrain and water surfaces) spatially consistent between neighbouring cells. A possible solution would be to introduce complex surface functions of higher order for every single cell. However, this approach is not feasible in practice since the discretised governing equations of fluid motion (chapter 4) rely on the existence of planar

surfaces that can be defined by face centroids and surface area vectors. Furthermore, the computation of cell volumes and surface areas would become significantly complex tasks. Therefore only a triangulation can be the adequate way to deal with this problem. As already mentioned, the Voronoi decomposition is the dual graph of the Delaunay triangulation. Hence, with the Voronoi grid given, a consistent triangulation is already available without the need for additional time-consuming triangulation procedures. Upon applying the duality property base points turn into vertices in the Delaunay triangulation and edges are always perpendicular to each other. The boundary constraint is honoured in the Delaunay graph as well. For the Delaunay triangles are defined by the base points of the grid, every surface elevation within the grid can be derived by means of simple triangle interpolation. Furthermore, as soon as the variables that were solved for in the discretised equations are available, they can be interpolated and plotted using the same mechanism of triangle interpolation.

3.4.2 Bivariate Interpolation Method

According to *Akima* (1978b) [3] the bivariate interpolation method is a smooth surface fitting technique developed for z values given at points irregularly distributed in the $x-y$ plane. It uses a fifth-degree polynomial in x and y as interpolating function defined in each triangular cell which has projections of three surface data points as its vertices. Triangulation is performed on the surface data points according to a max-min angle criterion described in further detail in *Akima* (1978b) [3]. The interpolation function for any given point (x, y) within each triangle then reads

$$z(x,y) = \sum_{j=0}^{5} \sum_{k=0}^{5-j} q_{jk} x^j y^k \tag{3.2}$$

which results in the need to determine 21 coefficients q_{jk}. These coefficients are found by the assumption that the values of the function, as well as its partial derivatives of first and second order, are given at all vertices of the triangle. In combination with the presumption that the partial derivative of the function differentiated in the direction perpendicular to each edge of the triangle is a third-degree polynomial, 21 conditions are obtained to determine all coefficients. Due to the polynomial functions used, smoothness of the interpolated surface both within each triangle and at its edges results from this process as proven in *Akima* (1978b) [3].

The bivariate interpolation method works very well as long as the distribution of terrain points follows a pattern that does not deviate too much from a regular distribution. Especially when a terrain grid (i.e. a digital terrain model, DTM) is used as basis, the interpolation method

yields reasonable interior values. However, as soon as measured river cross sections are used as terrain data basis, the interpolated terrain elevations exhibit strong and irregular extreme values (see appendix A for an illustrative example). This can be explained easily: the triangulation procedure is performed on the terrain data points, as these are the only locations where actual information is readily available without the need for interpolation. As a matter of fact, cross sections of natural rivers come with a high resolution within each section but comparably large distance between the profiles: for instance, measurements at the river Danube contain between 250 and 1000 data points per cross section, which is typically around 250m in distance, resulting in a resolution of one point per 0.25m to 1.0 m; on the other hand, cross sections are fathomed in a typical distance of 50m to 100m. This yields a ratio of longitudinal to transversal resolution between 1:50 and 1:400. Hence, a triangulation of this data will inevitably result in triangles of the same ratio, no matter how good the triangulation algorithm. It is easy to see that a polynomial constructed on top of such a triangle will exhibit undesired maxima and minima in the interior only to satisfy the first and second derivatives of the surface function at its edges. Therefore we can conclude that this method can be of use when more or less regularly gridded terrain data is available, but not in situations where cross sections of rivers are the only measurements available.

3.4.3 Kriging

This method was first published by *D.G. Krige* (1951) [38]. It is frequently used in geostatistics to determine unknown values using known values and a semivariogram. There exist several different types of kriging methods, but only the procedure denoted *point kriging* was evaluated for the present work. This approach relies on the assumption that an estimate of an unknown value $Y_{E,p}$ at a point p can be found by using a weighted average of the surrounding known values Y_i,

$$Y_{E,p} = \sum W_i Y_i \tag{3.3}$$

where W_i are the respective weights. The estimated value is said to be unbiased when the weights sum to unity. Hence, the weights applied to solving a certain problem must obey the relation:

$$\sum W_i = 1 \tag{3.4}$$

Optimal weights must not only satisfy the condition of producing an unbiased solution; they are also required to have a minimum estimation variance, i.e. the scatter of the estimates $Y_{E,p}$ about the actual value Y_p must be minimised. This criterion can be enforced by introducing a

set of simultaneous equations, complemented by a variable called the Lagrange multiplier λ. To illustrate this procedure, we make the assumption that four known values Y_1 (at point "1") through Y_4 (at point "4") are used to estimate an unknown value $Y_{E,p}$ at point p. Combining this with equation 3.4, we can write the following equation set,

$$\begin{aligned} W_1\gamma(d_{11}) + W_2\gamma(d_{12}) + W_3\gamma(d_{13}) + W_4\gamma(d_{14}) + \lambda &= \gamma(d_{1p}) \\ W_1\gamma(d_{21}) + W_2\gamma(d_{22}) + W_3\gamma(d_{23}) + W_4\gamma(d_{24}) + \lambda &= \gamma(d_{2p}) \\ W_1\gamma(d_{31}) + W_2\gamma(d_{32}) + W_3\gamma(d_{33}) + W_4\gamma(d_{34}) + \lambda &= \gamma(d_{3p}) \\ W_1\gamma(d_{41}) + W_2\gamma(d_{42}) + W_3\gamma(d_{43}) + W_4\gamma(d_{44}) + \lambda &= \gamma(d_{4p}) \\ W_1 + W_2 + W_3 + W_4 &= 1 \end{aligned} \tag{3.5}$$

where $\gamma(d_{ij})$ is the semivariance between data points i and j. In the present work, this semivariance was set equal to the distance between the points, which is also the most common approach. Equation set 3.5 can now be rearranged in matrix form,

$$\begin{bmatrix} \gamma(d_{11}) & \gamma(d_{12}) & \gamma(d_{13}) & \gamma(d_{14}) & 1 \\ \gamma(d_{21}) & \gamma(d_{22}) & \gamma(d_{23}) & \gamma(d_{24}) & 1 \\ \gamma(d_{31}) & \gamma(d_{32}) & \gamma(d_{33}) & \gamma(d_{34}) & 1 \\ \gamma(d_{41}) & \gamma(d_{42}) & \gamma(d_{43}) & \gamma(d_{44}) & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} \bullet \begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \\ \lambda \end{bmatrix} = \begin{bmatrix} \gamma(d_{1p}) \\ \gamma(d_{2p}) \\ \gamma(d_{3p}) \\ \gamma(d_{4p}) \\ 1 \end{bmatrix} \tag{3.6}$$

which makes it possible to solve for the weights using common techniques for solving a set of linear equations. Since the equation $d_{ij} = d_{ji}$ holds true for distances, the left-hand matrix is symmetrical. The main diagonal is filled with zeroes because d_{ii} is obviously nil. After the weights have been determined, the unknown value $Y_{E,p}$ can be estimated by:

$$Y_{E,p} = W_1Y_1 + W_2Y_2 + W_3Y_3 + W_4Y_4 \tag{3.7}$$

The Lagrange multiplier λ is not needed to obtain an estimate of the unknown value $Y_{E,p}$, but its presence ensures that the minimum possible estimation error is obtained. The estimation variance s^2 can now be calculated by

$$s^2 = W_1\gamma(d_{1p}) + W_1\gamma(d_{2p}) + W_1\gamma(d_{3p}) + W_4\gamma(d_{1p}) + \lambda \tag{3.8}$$

which is a great advantage since it allows for a quantification of the error made in the estimation of $Y_{E,p}$.

The method itself does not come with restrictions as to how many points can be used to estimate an unknown terrain elevation at another location; however, in a real-world situation there are constraints like memory requirements in storing the matrix of equation 3.6, or computation time to solve the linear equation system. Due to these practical considerations, the following methodology was adopted in the present work (see appendix A for an example):

- A circle with radius $R = \sqrt{a^2 + b^2}/2$ is constructed on the location of every point with unknown terrain elevation, where a and b are the dimensions of the available terrain information in the directions of x and y within the computational domain.
- The circle is partitioned into four quadrants of equal size, and the terrain data available in each quadrant is sorted according to its distance to the circle's centre point.
- A maximum of eight terrain data points (those with the smallest distance to the point of interest) is selected within each quadrant, summing up to a maximum of 32 terrain data points available for a single kriging operation.

Undoubtedly the kriging method is computationally expensive. However, there are no problems involved in using measured river cross sections as input data as the method is usually not subject to exhibiting irregular maxima or minima of the estimated values. Hence, the kriging approach was selected as method of choice for interpolating terrain elevations at the locations of computation points and grid line vertices that are part of the boundary polygon or a structural polygon.

3.5 Grid Refinement

After the computation grid was created by applying Fortune's method to a set of points distributed following the procedures outlined in section 3.2, the next step is to refine the grid. Mesh refinement in the vicinity of obstacles within the flow domain or at domain boundaries is generally possible by making use of boundary and structure line modules, which have already been discussed. Additionally, there will usually be a desire for further refinement in regions with a steeper surface slope. In order to meet this desire, a criterion of maximum absolute height error is adopted in the present work: by means of kriging, it is possible to derive surface elevations

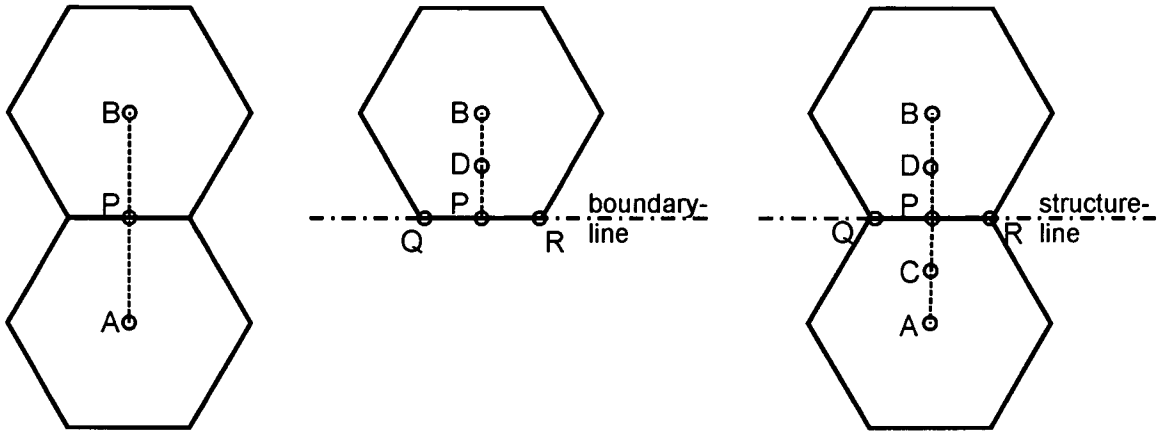

Figure 3.14: Grid refinement procedure for ordinary cells (left), boundary cells (centre) and cells adjacent to structure lines (right)

for arbitrary points within the computational domain; these elevations can be compared to the ones obtained by interpolation on the triangular Delaunay grid (fig. 3.3), inserting new points if a certain error bound is exceeded. This procedure is illustrated in figure 3.14 for three different situations:

- *Left*: Two *ordinary cells* adjacent to each other are subject to refinement if the surface elevation of the midpoint P in the line connecting cell centroids A and B meets the criterion

$$|h_P - h_P^*| > \epsilon \tag{3.9}$$

where h_P is obtained by linear averaging of the surface elevations in points A and B, h_A and h_B,

$$h_P = \frac{h_A + h_B}{2} \tag{3.10}$$

and h_P^* is derived from surrounding terrain data points by means of kriging. ϵ is an error bound given by the user. If the refinement criterion is met, a new basepoint P with elevation h_P^* is inserted into the complete set of basepoints and the grid is generated again.

- *Centre*: A cell at the *domain boundary* is subject to refinement if the midpoint D of the line connecting the cell centroid B with the boundary line in perpendicular direction meets the criterion:

$$|h_D - h_D^*| > \epsilon \tag{3.11}$$

In this case, the geodetic height of the start and end vertices of the boundary line (points Q and R in figure 3.14) is known. Furthermore, we know from figure 3.3 that segments

of boundary lines always make up triangles with the centroid of the adjacent cell. Hence, the elevation of point P can be obtained by distance-weighted interpolation on the line connecting points Q and R. Using this information, the height of point D, h_D, can finally be derived,

$$h_D = \frac{h_B + h_P}{2} \tag{3.12}$$

and a comparison with elevation h_D^*, obtained by kriging, becomes possible. If refinement is necessary, point D is inserted and the grid must be regenerated.

- *Right*: Special care must be taken to test for refinement in cells adjacent to a *structure line* as this line must be preserved even after grid refinement has taken place. Therefore the approach outlined for domain boundaries is adopted here: the surface elevation of points Q and R is known, which makes it possible to derive the height of point P. Knowing this elevation, *both* heights of points C and D, h_C and h_D, obtained from equation 3.12, must be compared with the respective values h_C^* and h_D^*, derived from kriging:

$$\begin{aligned} |h_C - h_C^*| &> \epsilon \\ |h_D - h_D^*| &> \epsilon \end{aligned} \tag{3.13}$$

If only *one* of the criteria set forth in equation 3.13 is met, *both* points C and D must be inserted into the set of basepoints and the grid will be generated again. This is necessary to preserve the structure line at its current location.

It should be noted that the grid refinement approach presented here is actually not very difficult to carry out; still, the procedure may take a while on grids with a large number of base points because the vertical coordinates of every connection line between two cell centroids must be derived from the digital terrain model, and if new points are inserted the grid and its data structure must be recreated.

An example of a computation grid refined using the approach described above is given in figure 3.15. It shows a detail of a grid which was created for a reach of the river Danube east of Vienna, Austria. Based on a hexagonal cell pattern with a horizontal centre point distance of 40m and two cell rows with half that spacing along the boundary, the grid was refined seven times, leading to an absolute height error of less than 20cm in every computational cell within the domain (*Tritthart* (2004) [80]). Contour lines allow for an interpretation of the surface gradient at the left side of fig. 3.15, while the right side shows the dual Delaunay grid used for interpolation. Terrain elevations were available in river cross sections (blue lines) and also along the river banks.

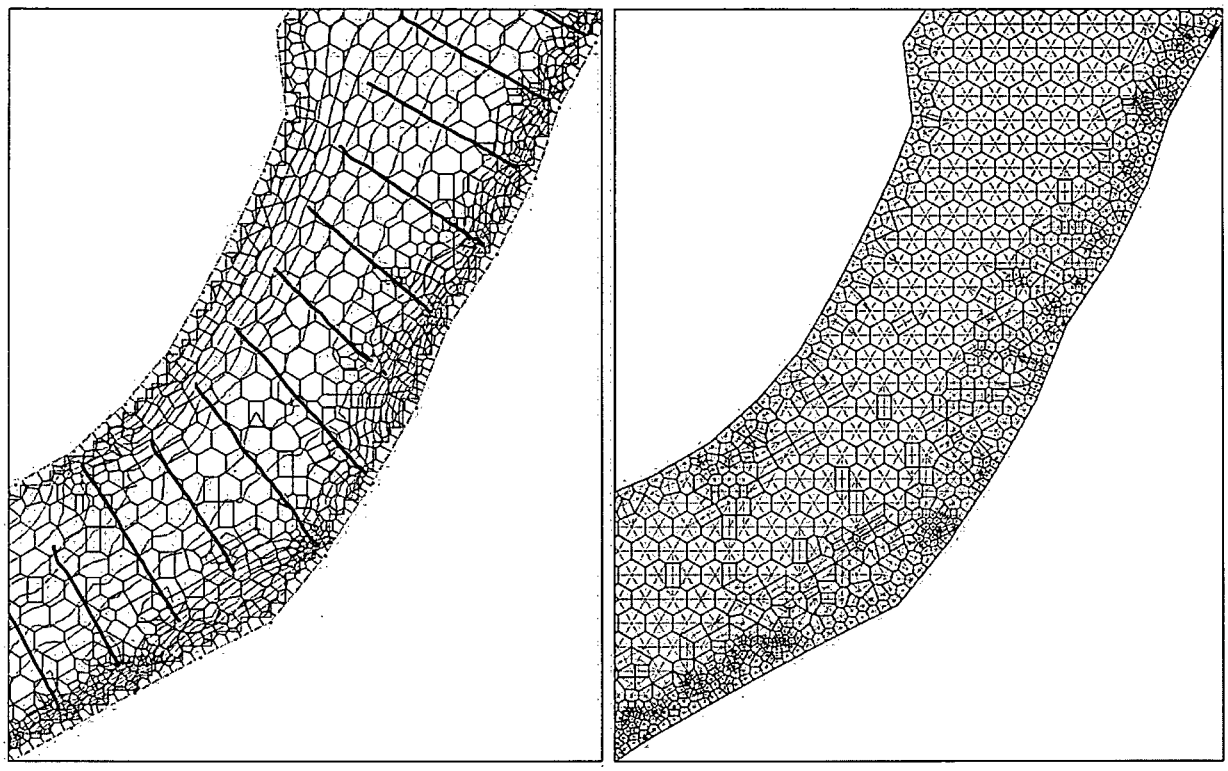

Figure 3.15: Plan view of the grid for a reach of the river Danube east of Vienna with contour and section lines (left) and the dual triangular grid (right)

3.6 Estimation of Water Surface Elevations

Before the actual three-dimensional grid can be created from the two-dimensional grid discussed so far, an estimation of the water surface elevation must be performed for every single grid region. The 3D grid will then be generated only in those regions where the water surface elevation lies above the terrain surface. However, as the water surface changes during the solution of the flow equations, it is possible that previously wet regions suddenly become dry and formerly dry regions turn wet; hence, the 3D grid may significantly change during the computations, but it is nonetheless important to provide a good estimate of the water surface elevation to the model so that a reasonable initial grid may be created.

In order to come up with an estimate for the initial water surface elevation, the grid generator needs to know the flow boundaries, i.e. those grid lines where water enters or exits the computational domain. As these are user-provided, we can treat them as known. The next challenge is then to find the flow path – the polygon lines that connect inflow and outflow boundaries – known as thalweg in rivers. In theory it is possible to obtain flow paths by means of gradient analysis:

starting at a grid region adjacent to an inflow boundary, a polygon line is constructed that always follows the steepest slope. While this procedure works well on hillslopes, it performs significantly worse in actual rivers, especially when applied to rivers with large cross sections. There is a number of reasons for this; among the most important are the very flat slopes encountered in such rivers, leading to irregular flow paths, and the presence of dunes and bars which may cause an automated algorithm to crash. Therefore – even when it may add a little inconvenience to the model application in practical situations – it is best to leave the task of defining flow paths to the user by providing the model with appropriate polygon lines.

A complex flow situation, but not uncommon in reality, is the presence of several such polygon lines, each defining a separate river or channel that disembogues into another river, resulting in a network of rivers. To perform an estimation of the water surface elevation in such a constellation, all available flow paths must be sorted first. It is best to do so by assigning the river comprehending the downstream boundary condition an ordinal number of 1, channels discharging into such a stream receive a classification of 2, and so forth (see fig. 3.16). This ordering system benefits an automatic computer-aided calculation of water surface positions, since the elevations of streams with lower ordinal numbers are computed first and can subsequently be used as boundary conditions for those with higher ordinal numbers.

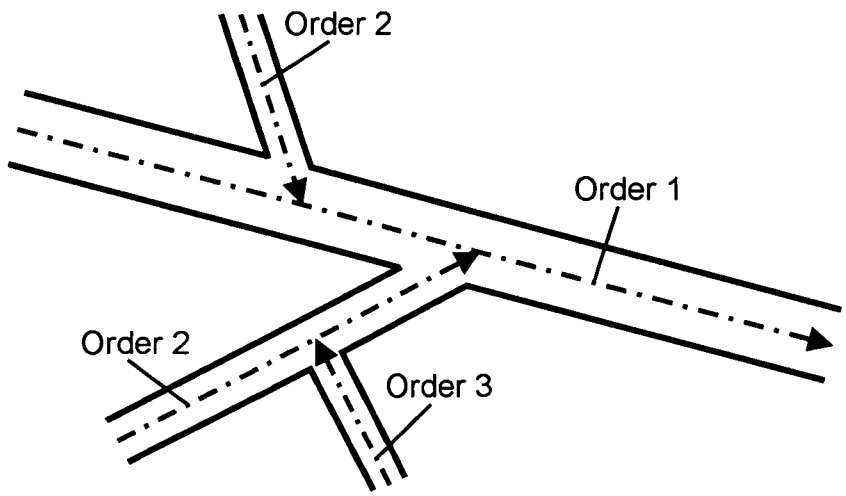

Figure 3.16: Schematic view of a river network with several confluences

As soon as the boundary conditions of all streams within the computational domain are known, the detailed determination of the water surface elevations can be done. In the present work, four different methods are available for performing this task:

- *Constant water surface elevation*: Being the simplest of all methods, a constant water surface elevation in the entire domain is useful for some validation cases (e.g. laboratory flumes) that do not exhibit any bed slope, hence the surface slope is very small as well. It can also be used for real-world situations when the region of interest is small-sized and only local phenomena are being investigated.

- *Linear interpolation*: In this method, both upstream and downstream water levels are prescribed by the user; the model performs linear interpolation along all flow path polygon lines available. Usually, this technique gives a very good initial water surface estimate and will suffice for most flow situations.

- *Constant flow depth*: This method is quite complex, as it requires the slicing of the three-dimensional domain into a number of cross-sections perpendicular to the given flow path. For each section, the minimum terrain elevation is determined, and after adding the flow depth given by the user, the water surface elevation results. Actually this method works very well for channels with a simple cross-section shape or rivers with very little variability in bed forms. When unfiltered terrain data is used, the technique may yield a water surface exhibiting the same irregularities as the bed, possibly leading to problems in the numerical simulation thereafter.

- *1D backwater computation*: Undoubtedly, this method returns the most realistic initial guess for the water surface elevation within the flow domain, but it is only worth the computational effort if the terrain data has been very carefully checked for errors; otherwise unrealistic results may be obtained, impairing convergence in the numerical simulation. The first step of this method is the same as in the case of constant flow depth: the 3D domain is sliced into a number of cross-sections perpendicular to the flow path. Considering two consecutive cross-sections, j and $j+1$, we can use the extended Bernoulli equation to write (*Gutknecht* (2004) [32])

$$w_{j+1} + \alpha_{j+1} \cdot \frac{v_{j+1}^2}{2g} = w_j + \alpha_j \cdot \frac{v_j^2}{2g} + h_r \tag{3.14}$$

where w_j denotes the water surface elevation above sea level (or a reference surface), v_j is the average velocity in a cross-section and the coefficient α_j is equal to unity. h_r is the friction loss due to the influence of roughness,

$$h_r = J_e \cdot \Delta x \tag{3.15}$$

where J_e denotes the energy gradient and Δx is the horizontal distance between cross-sections j and $j+1$. The energy gradient J_e can be quantified by making use of the Gauckler-Manning-Strickler equation to yield

$$J_e = \frac{1}{k_{St}^2 R_{h,m}^{\frac{4}{3}}} \cdot v_m^2 \tag{3.16}$$

where k_{St} is the Strickler coefficient, and v_m and $R_{h,m}$ are mean average velocity and mean hydraulic radius of the two consecutive cross-sections. Since the discharge is known, both of the latter values can be expressed as functions of the geometric properties A_m and U_m, denoting the mean cross-section area and perimeter, respectively. These properties can be obtained by linear averaging from the values at both cross-sections. As the values for A_{j+1} and U_{j+1} are dependent on the unknown water surface elevation w_{j+1}, the computation must be done iteratively. Finally, after a (usually) small number of iterations, w_{j+1} is obtained from equation 3.14, and the procedure is repeated for the next cross-sections until the inflow section is reached.

It is important to mention that the methods of linear interpolation, constant flow depth and 1D backwater computation are only capable of computing water surface elevations along previously defined polygon paths. The model, however, needs water surface elevations for the whole domain, in every single grid region. Hence, the results of these computations are extended into 3D by repeatedly specifying the one-dimensional surface elevations in all grid regions of an area delimited by each two consecutive cross-sections in a user-supplied distance (a typical value would be 10m for a river). This completes the initial guess of water surface elevations and allows for generating the three-dimensional grid.

3.7 3D Grid Geometry

The three-dimensional grid is obtained by partitioning the cell piles, defined by grid regions together with terrain and water surface elevations, into a number of finite cell volumes. Every region is subdivided into the same number of cells, hence the grid is *vertically structured*. Figure 3.17 illustrates this: the vertically structured grid is composed of extruded grid regions in combination with a triangulated surface on top and bottom of the domain, which adds some geometric complexity. Actually, every grid line bordering a grid region within the domain (i.e. not at the domain boundary) is intersected by *at least one* edge of the dual Delaunay triangulation making

up for the terrain interpolation (see fig. 3.3). Therefore the 3D equivalent of the grid lines in 2D – denoted *faces* – actually become polygonal bounded surfaces as the grid becomes polyhedral.

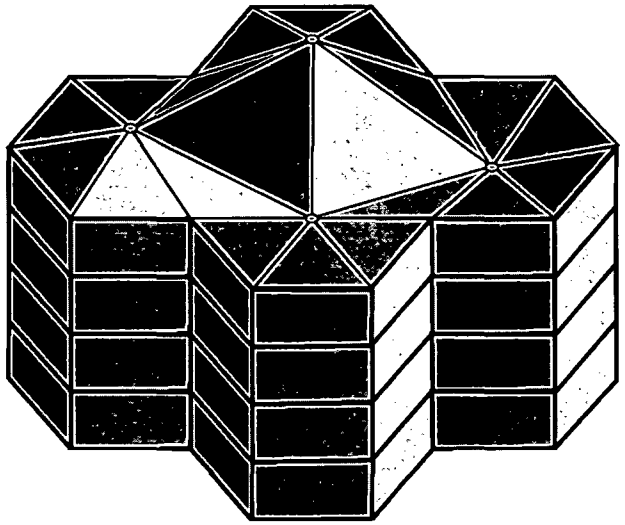

Figure 3.17: Three-dimensional grid composed of extruded grid regions and a triangulated surface

It is obvious that the calculation of cell volumes and face areas, both of which are needed for the numerical solution of the discretised flow equations, becomes a complex and computationally intensive task for the grid presented in this work. Fortunately, it is not necessary to recompute these properties too often; instead, it is sufficient to calculate them only after updates of the water surface have taken place. However, even then the two-dimensional intersection points between the polygonal grid regions and the triangular surface representation stay the same. Therefore it is adequate to compute these intersection points only once, after the grid generation has been completed, and modify the vertical elevations (bottom and top) after every surface update, which can be done reasonably fast. A new data type, denoted *face vertex*, is introduced: it stores four floating point numbers, two of which represent the x and y coordinates of the intersection points between grid regions and Delaunay triangles, the other two store the elevations z_1 and z_2 of bottom and top. Start and end points of grid lines in 2D of course transform into face vertices in 3D, as well. Figure 3.18 shows an exemplary face composed of four face vertices, further illustrating this.

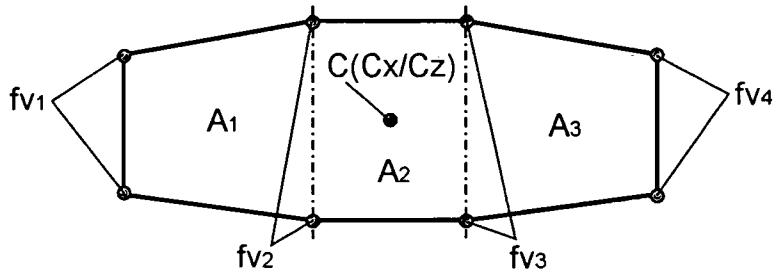

Figure 3.18: Cell face composed of four face vertices and three areas; C denotes the face centroid

Face areas

Face areas are computed by summing up all partial areas defining one face. By definition, the area spanned by two face vertices is always a trapezoid. Hence, we can write for the area of an entire vertical face given by n face vertices fv,

$$A_f = \sum_{i=2}^{n} x_{i-1} \frac{z_2\left(fv_{i-1}\right) + z_2\left(fv_i\right) - z_1\left(fv_{i-1}\right) - z_1\left(fv_i\right)}{2} \tag{3.17}$$

where x denotes the distance of one face vertex to his neighbour in the $x-y$ plane, $z_2\left(fv\right)$ is the face vertex' top elevation and $z_1\left(fv\right)$ its bottom elevation.

Top and bottom areas of a cell always consist of a number of triangles, each of which is defined by two face vertices and a base point with a top and a bottom elevation assigned. As triangles can be interpreted as degenerate trapezoids, the same equations for face areas or centroids can be applied; therefore top and bottom areas of cells will not be subject to further discussion in this chapter.

Face centroids

The exact location of each face centroid must be known in order to compute the volume of the cell enclosed by a number of faces. We start with the equations for the coordinates C_x and C_z of a compound section in two-dimensional space,

$$\begin{aligned} C_x &= \frac{\sum_i A_i x_{c,i}}{\sum_i A_i} \\ C_z &= \frac{\sum_i A_i z_{c,i}}{\sum_i A_i} \end{aligned} \tag{3.18}$$

where A_i stands for the partial area i (in the present work spanned by two neighbouring face

vertices), and $x_{c,i}$ and $z_{c,i}$ are the centroid coordinates of every partial area. The denominator of both relations in eq. 3.18 is, of course, equal to the face area as given by equation 3.17. After introducing three geometric relations a, b and c in accordance with the definitions made for eq. 3.17,

$$\begin{aligned} a &= z_2(fv_{i-1}) - z_1(fv_{i-1}) \\ b &= z_2(fv_i) - z_1(fv_i) \\ c &= z_1(fv_{i-1}) - z_1(fv_i) \end{aligned} \tag{3.19}$$

we can use the formulae available for the centroid coordinates of trapezoids,

$$\begin{aligned} x'_{c,i} &= \frac{x_{i-1}(2a+b)}{3(a+b)} \\ z'_{c,i} &= \frac{2ac + a^2 + cb + ab + b^2}{3(a+b)} \end{aligned} \tag{3.20}$$

to compute the centroid coordinates $x'_{c,i}$ and $z'_{c,i}$ in a local coordinate system with origin in the bottom point of face vertex fv_i. Figure 3.19 illustrates this procedure. Subsequently these local coordinates must be shifted to a fixed point – in the present work the bottom point of the last face vertex fv_n was used – so that they can be used in eq. 3.18 to yield the coordinates of the face centroids. These coordinates are finally transformed into $x/y/z$ coordinate triples in the global coordinate system to be of use for the computation of cell volumes.

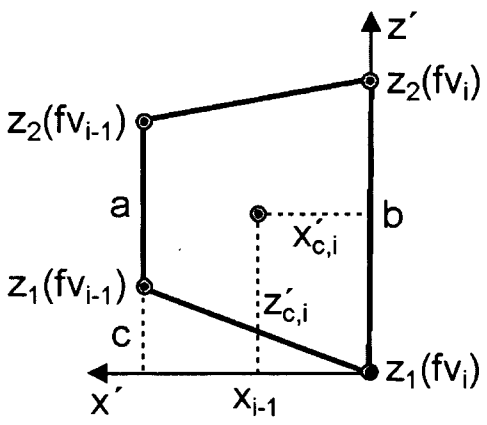

Figure 3.19: Coordinate system and nomenclature for the calculation of centroid coordinates

Face normals

Face normal vectors are required for the calculation of cell volumes and are also used excessively throughout the solution process of the discretised flow equations (chapter 4). The normals of vertical faces are obtained by rotating the two-dimensional grid lines by 90 degrees and inserting a zero for the third coordinate. Normal vectors of triangle faces at the top and the bottom of cells can be calculated by computing the cross product of two vectors that span the triangle surface. All vectors are finally normalised to become unit vectors.

By definition, face normal vectors always point out of the cell they belong to. However, vectors are stored as properties of faces, not cells. Since a face uniquely separates two cells, the vector will always point outwards for one of the cells and inwards for the other. This means that any algorithm must be able to determine into which direction the vector is actually pointing, to invert it if needed. In the present work this problem was solved by storing the numbers of the cells adjacent to each face along with the data for that face, defining that the surface vector always points outwards for the first cell in this table. It is then possible for an algorithm to compare the cell number it is working on with the ones in this table, and thus determine whether inversion of the vector is necessary. The underlying cell numbering scheme is exemplified in appendix A.

Cell volumes

Calculating cell volumes of polyhedra by partitioning them into several small geometric elements is a fairly complex and time-consuming task. However, it is possible to compute cell volumes also in a different way, making use of Gauss's Divergence Theorem (eq. 4.6, chapter 4) to replace volume integrals by surface integrals. In other words, the cell volume can be computed by summing over all its bounding faces. *Ferziger* (2002) [20] gives the exact equation for this; adapted to the notation used in the present work, we can write

$$V = \frac{1}{3} \sum_{i=1}^{n} (A_i \vec{n}_i) \cdot \vec{c}_i \tag{3.21}$$

where V denotes the volume of a cell bounded by n faces with the respective face areas A_i, the three-dimensional face normal vectors $\vec{n}_i$ and the 3D coordinates of the face centroids, $\vec{c}_i$.

Partially dry cells

Cells are declared dry when the cell centroid lies below the terrain surface. However, it is easily possible that this condition is not met, but still some face vertices lie below the surface. In this case a *partially dry* cell is encountered. Such a cell geometry causes problems in the calculation

of cell face areas, centroids and volumes, especially since the conservative formulation of eq. 3.21 does no longer yield the expected result for such a situation. It would be possible to deal with partially dry cells by treating them as a set of smaller geometric entities, but this adds a fair amount of complexity to the solution process. Therefore a workaround was chosen for the work discussed here: if certain face vertices exhibit terrain elevations that lie above the water surface, the terrain elevation is set equal to the water level. Indeed, this procedure introduces a small geometric error into the whole solution process, but especially when measured terrain data is used, this additional error is small compared to the errors inherent in the terrain data itself.

Data structure

The three-dimensional grid data is stored in an appropriate data structure, just as the one discussed in section 3.3 for the two-dimensional data. The structure used in the present work is illustrated in fig. 3.20. The two main data types of the 3D grid are *cells* and *faces*; additionally *face vertices* are required to save computation time in computing certain geometric properties of the main types.

The nomenclature defined in section 3.3 is extended by these data type definitions:

- *Vect3D*: an object-oriented data type, consisting of three *float* elements x, y and z; this is the same definition as for *Point3D*, only the name is different to make clear that a vector and not a point is stored there.
- *type[size]*: an array of *size* elements of the data type *type* (i.e. a fixed-size array, not one varying in size, as in *Vector*).

It should be noted that the *cells* data structure does not only store geometric data but also data required for the actual flow simulation. This includes

- the discretised diagonal coefficient a_P of all six governing equations,
- the right-hand side of the six discretised governing equations, usually filled in by source and sink terms,
- the conservation quantities (u, v, w, p, k, ε)
- the gradient of the conservation quantities in all three cartesian coordinate directions,
- and the isotropic eddy viscosity,

all of which are subject to discussion in the following chapter.

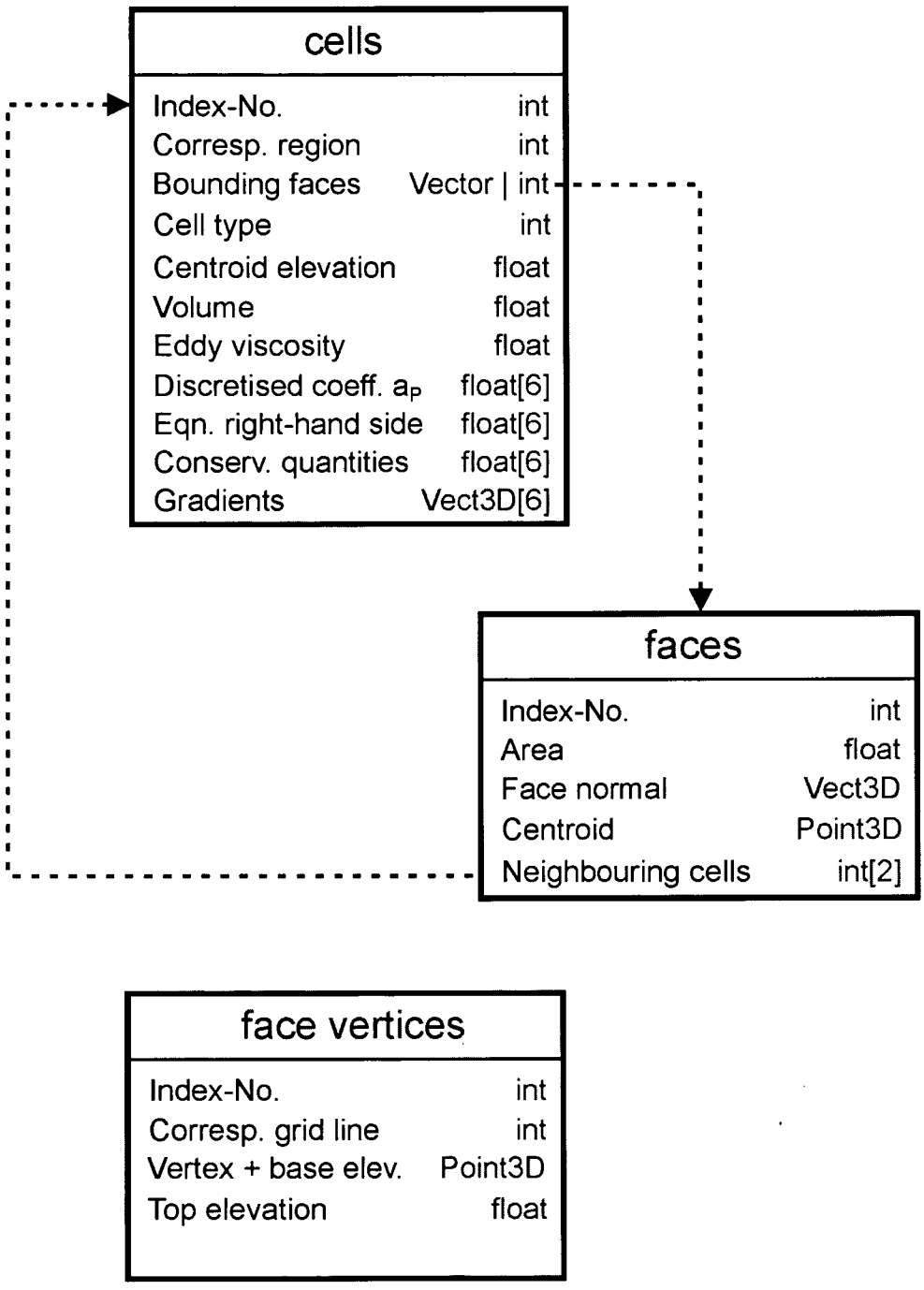

Figure 3.20: Grid data structure for 3D flow simulations

4 Governing Equations and Discretisation

4.1 Momentum Equations

4.1.1 Equations

The motion of moving fluids in three spatial dimensions is governed by the Navier-Stokes equations, a set of three nonlinear partial differential equations (PDEs). Using a complete notation for the incompressible case, they can be written as follows:

$$\begin{aligned}
\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} &= -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\frac{\partial^2 u}{\partial x^2} + \nu\frac{\partial^2 u}{\partial y^2} + \nu\frac{\partial^2 u}{\partial z^2} + f_x \\
\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} &= -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\frac{\partial^2 v}{\partial x^2} + \nu\frac{\partial^2 v}{\partial y^2} + \nu\frac{\partial^2 v}{\partial z^2} + f_y \\
\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z} &= -\frac{1}{\rho}\frac{\partial p}{\partial z} + \nu\frac{\partial^2 w}{\partial x^2} + \nu\frac{\partial^2 w}{\partial y^2} + \nu\frac{\partial^2 w}{\partial z^2} + f_z
\end{aligned} \tag{4.1}$$

In equation 4.1 u, v and w denote the velocities in the three spatial dimensions x, y and z. The density of the fluid ρ and the kinematic viscosity ν are the two fluid properties that are being used in this equation set. Pressure is denoted by p, and the terms f_x, f_y and f_z are external forces acting on the fluid, with gravity or the Coriolis force being the most prominent examples. The temporal dimension t enters the equation through an additional transient term.

In addition to the Navier-Stokes equations, a moving fluid must satisfy the continuity equation, which in three spatial dimensions is given by:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{4.2}$$

These equations all have a similar structure and can therefore be written in a single form of a generic transport equation using tensor notation (*Ferziger* (2002) [20]):

$$\underbrace{\frac{\partial(\rho\phi)}{\partial t}}_{(I)} + \underbrace{\frac{\partial(\rho u_j \phi)}{\partial x_j}}_{(II)} = \underbrace{\frac{\partial}{\partial x_j}\left(\Gamma \frac{\partial \phi}{\partial x_j}\right)}_{(III)} + \underbrace{S_\phi}_{(IV)} \tag{4.3}$$

This generic conservation equation is presented for general (i.e. compressible and incompressible) fluids and uses the symbol ϕ for the quantity that is going to be transported through the computational domain. With $\phi = u, v, w$ (or $\phi = u_j = u_1, u_2, u_3$) the Navier-Stokes equations for the three coordinate directions $x_j = x, y, z = x_1, x_2, x_3$ can be obtained when the diffusion coefficient Γ is set equal to $\nu \cdot \rho$. Source terms are denoted by S_ϕ. In the absence of such source terms, the continuity equation is obtained by using $\phi = 1$ in equation 4.3.

The generic transport equation therefore consists of four main terms:

- a transient term (I),
- a convective term (II),
- a diffusive term (III),
- and a source term (IV).

These terms must be treated differently in an implementation since they describe different physical phenomena and – from a mathematical point of view – are members of different types of underlying PDEs.

For the discretisation of this generic conservation equation using the Finite Volume Method on grids with arbitrary cell shapes, it is an advantage to present equation 4.3 in a coordinate-free vector form using the divergence and gradient operators:

$$\frac{\partial(\rho\phi)}{\partial t} + \mathrm{div}\,(\rho\phi\vec{u}) = \mathrm{div}\,(\Gamma \mathrm{grad}\phi) + S_\phi \tag{4.4}$$

This notation takes into account the vectorial nature of velocity, denoted by $\vec{u} = (u, v, w) = (u_1, u_2, u_3)$.

Starting point of the discretisation is the integral form of the generic conservation equation in vector form (eq. 4.4) which has been integrated over a control volume Ω:

$$\int_\Omega \frac{\partial(\rho\phi)}{\partial t} \mathrm{d}\Omega + \int_\Omega \mathrm{div}(\rho\phi\vec{u})\,\mathrm{d}\Omega = \int_\Omega \mathrm{div}(\Gamma \mathrm{grad}\phi)\,\mathrm{d}\Omega + \int_\Omega S_\phi \mathrm{d}\Omega \tag{4.5}$$

Using Gauss's Divergence Theorem

$$\int_\Omega \mathrm{div}\vec{a}\mathrm{d}\Omega = \int_A \vec{n}\cdot\vec{a}\mathrm{d}A \tag{4.6}$$

where $\vec{a}$ is a generic vector, we can substitute the volume integral by an integral over the volume's surrounding surface A, with $\vec{n}$ denoting the surface normal vector of A. As this work will not deal with unsteady flows, we can drop the transient term, and after introducing the constraint of incompressibility, we finally obtain:

$$\int_A \vec{n}\cdot(\phi\vec{u})\,\mathrm{d}A = \int_A \vec{n}\cdot\left(\frac{\Gamma}{\rho}\mathrm{grad}\phi\right)\mathrm{d}A + \frac{1}{\rho}\int_\Omega S_\phi \mathrm{d}\Omega \tag{4.7}$$

It should be noted that the pressure term of equation 4.1 is contained in the source term in this notation, which is the usual procedure in the derivation of the discretised momentum equations. It will be dealt with later in this chapter.

4.1.2 Diffusive Term

The diffusive term of equation 4.7,

$$\int_A \vec{n}\cdot\left(\frac{\Gamma}{\rho}\mathrm{grad}\phi\right)\mathrm{d}A \tag{4.8}$$

contains a diffusion coefficient Γ which was already found to be equal to $\nu\cdot\rho$ in equation 4.3. Since the kinematic viscosity ν can be considered constant in the flow regimes dealt with in the present work, this can be discretised as

$$\nu\sum_{i=1}^{n}\vec{n_i}\cdot\mathrm{grad}\phi\cdot A_i \tag{4.9}$$

for a finite control volume confined by n faces with the respective areas A_i and face normal

vectors $\vec{n_i}$. After breaking up the gradient operator, the diffusive term can be written as

$$\nu \sum_{i=1}^{n} A_i \left[n_{i,x} \left(\frac{\partial \phi}{\partial x} \right)_f + n_{i,y} \left(\frac{\partial \phi}{\partial y} \right)_f + n_{i,z} \left(\frac{\partial \phi}{\partial z} \right)_f \right] \tag{4.10}$$

using the subscript f to denote that the partial differential is to be evaluated at the face instead of the cell centre. $n_{i,x}$, $n_{i,y}$ and $n_{i,z}$ are the components of $\vec{n_i}$ in the three Cartesian coordinate directions. *Davidson & Stolcis* (1995) [18] use Green's Formula to express the bracket term in equation 4.10 in a notation that contains the values of ϕ at two discrete locations:

$$n_{i,x} \left(\frac{\partial \phi}{\partial x} \right)_f + n_{i,y} \left(\frac{\partial \phi}{\partial y} \right)_f + n_{i,z} \left(\frac{\partial \phi}{\partial z} \right)_f = \frac{1}{V_f} \cdot A_f \cdot (\phi_N - \phi_P) + \text{NOD} \tag{4.11}$$

A_f is used for the area of the face common to the two neighbouring cells N and P with the respective cell centre values ϕ_N and ϕ_P. V_f denotes a control volume from one cell centre to the other, passing through the neighbouring face (see fig. 4.1, where V_f is bordered by the dashed line).

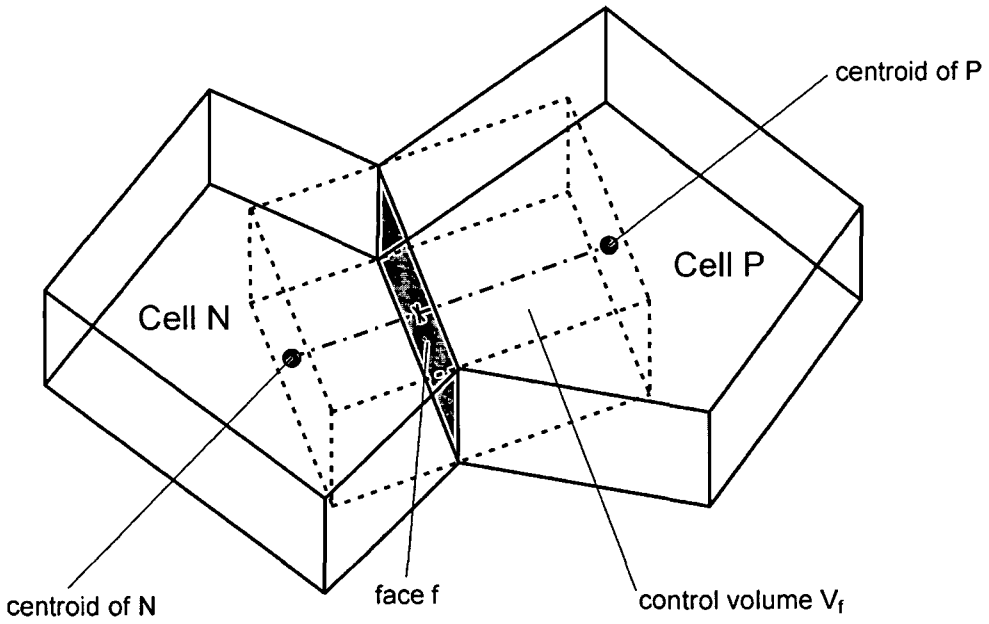

Figure 4.1: Cells and control volume for face gradient computation

The final term of equation 4.11 – denoted NOD – describes the phenomenon of non-orthogonal diffusion. According to *Davidson* (1996) [16] this term equals zero in an orthogonal cell setup. Per definition of the Voronoi diagram, however, the connection line between two cell centres

is always orthogonal on the dividing face. This allows us to neglect this term for all faces vertically separating two adjacent cells. As for the top and bottom cell faces, a slight non-orthogonality may result due to different gradients in surface and bottom of the flow domain. This non-orthogonality, however, is hardly severe, and will therefore be neglected as well. From figure 4.1 we can construct the relationship

$$\frac{A_f}{V_f} = \frac{1}{\delta_{NP}} \tag{4.12}$$

using $\delta_{NP} = \delta_{PN}$ to identify the spatial distance between cell centre points N and P. This allows us to write the complete diffusive term as follows:

$$\nu \sum_{i=1}^{n} A_i \frac{\phi_{N_i} - \phi_P}{\delta_{N_iP}} \tag{4.13}$$

Here, P denotes the cell centre point of the cell the discretised equations are written for and N_i are the respective neighbouring cell centre points (see fig. 4.2) with the spatial distance $\delta_{N_iP} = \delta_{PN_i}$ to P.

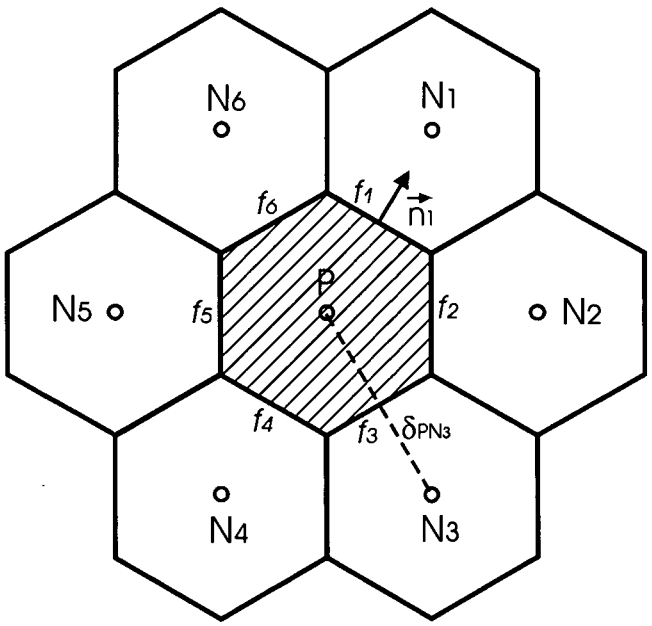

Figure 4.2: Definition of control volumes and cell centroids

4.1.3 Convective Term

The convective term of equation 4.7,

$$\int_A \vec{n} \cdot (\phi \vec{u}) \, \mathrm{d}A \tag{4.14}$$

can be discretised as

$$\sum_{i=1}^{n} \vec{n_i} \cdot (\phi \vec{u})_f \cdot A_i \tag{4.15}$$

for a finite control volume confined by n faces with the respective areas A_i and face normal vectors $\vec{n_i}$. The operator $(.)_f$ denotes the face values of its argument, in this case the product of ϕ_f and $\vec{u_f}$.

Central Differencing Scheme

Since all conservation quantities are stored in the cell centres in the style of a colocated arrangement (*Ferziger* (2002) [20]), obtaining interpolated face values is a challenging task. A straightforward way would be to use linear interpolation, resulting in the *central differencing scheme*:

$$\phi_f = f_1 \phi_N + (1 - f_1) \phi_P \tag{4.16}$$

In this equation, f_1 is a weighting factor based on the spatial distance between nodes N and P and the separating face f, respectively. Unfortunately, this technique has severe restrictions on the boundedness of the solution, based on the cell Peclet number, which can only be satisfied if the velocity is small, hence in diffusion-dominated low Reynolds number flows, or if the grid spacing is small (*Versteeg & Malalasekera* (2001) [84]). Therefore discretisation schemes with more favourable properties need to be employed.

Upwind Differencing Scheme

The basic idea of the *upwind differencing scheme* is that the value of a conservation quantity at a given cell centre point contains all the information needed at the cell face. Therefore, after identifying the flow direction, the cell face value is simply set equal to the cell centre value of the upstream cell. For a cell setup as in fig. 4.3 with a western cell W, a center-cell P and an eastern cell E, one can write for a flow in the positive coordinate direction:

$$\phi_w = \phi_W \text{ and } \phi_e = \phi_E \tag{4.17}$$

While the boundedness property is not violated by this scheme, it must be noted that it produces erroneous results when the flow is not aligned with the grid lines (*Versteeg & Malalasekera* (2001) [84]), resulting in a smeared distribution of the transported quantities. For its appearance is similar to diffusion effects, it is usually referred to as false diffusion. This undesired effect reduces the usability of the scheme in many cases and therefore it is not being used in this work.

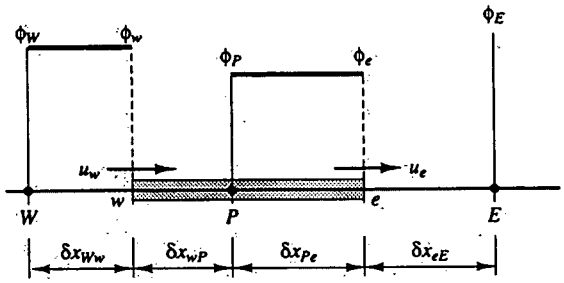

Figure 4.3: Cell setup and nodal values for the upwind differencing scheme (*Versteeg & Malalasekera* (2001) [84])

QUICK Scheme

In the *QUICK* (Quadratic Interpolation for Convective Kinematics) scheme of *Leonard* (1979) [41], the face values are interpolated from the nodal values using a quadratic interpolation function which results in a scheme of higher order, but without the problems involved with the central differencing scheme. This scheme involves using a larger number of neighbouring cells. Unfortunately, the scheme can not be generalised for arbitrary cell shapes as employed in the present work since it uses not only the upstream cell in the discretised equation but also the cell which is upstream to the upstream cell. When using arbitrary cell shapes, such a cell cannot be defined, a fact which is also stated by *Fluent* (2003) [25]. For this reason, the scheme could not be used in the present work.

Second Order Upwind Scheme

In a general formulation for arbitrary cell shapes, according to *Fluent* (2003) [25] the *second order upwind scheme* is given by

$$\phi_f = \phi + \nabla\phi \cdot \Delta\vec{s} \tag{4.18}$$

with ϕ_f denoting the face value of the transported property ϕ. $\nabla\phi$ is the gradient of ϕ in the upstream cell, and $\Delta\vec{s}$ is a vector from the upstream cell centroid to the face centroid. By employing this scheme it is ensured that the boundedness property is not violated but at the same

time reasonable accuracy of the result is preserved. However, this scheme requires two challenging tasks:

- computing the gradient of ϕ,
- and selecting the appropriate upstream cells for arbitrary cell shapes.

Gradient Computation

The appropriate way to evaluate gradients of transported properties in cells of arbitrary shape is given in *Barth & Jespersen* (1989) [10]. By making use of the divergence theorem (eq. 4.6), we can write:

$$\nabla\phi = \Phi_A \cdot \frac{1}{V}\sum_{i=1}^{n} \tilde{\phi}_i \cdot A_i \cdot \vec{n_i} \tag{4.19}$$

The value for $\tilde{\phi}_i$ is obtained by distance-weighted interpolation between two neighbouring cell centres, making use of equation 4.16; V denotes the cell volume. Φ_A is a limiting function which ensures that no new maxima or minima are introduced upon evaluation of the gradient. It is defined by

$$\Phi_A = \min \bar{\Phi}_{A_i} \tag{4.20}$$

with $\bar{\Phi}_{A_i}$ given by

$$\bar{\Phi}_{A_i} = \begin{cases} \min\left(1, \frac{\phi_A^{max}-\phi_A}{\tilde{\phi}_i-\phi_A}\right), & \text{if } \tilde{\phi}_i - \phi_A > 0 \\ \min\left(1, \frac{\phi_A^{min}-\phi_A}{\tilde{\phi}_i-\phi_A}\right), & \text{if } \tilde{\phi}_i - \phi_A < 0 \\ 1 & \text{if } \tilde{\phi}_i - \phi_A = 0 \end{cases} \tag{4.21}$$

and ϕ_A equal to the value in the centroid of the cell where the gradient is to be evaluated. ϕ_A^{max} is the maximum of the property ϕ among the cell and all its neighbour cells, and ϕ_A^{min} its minimum.

Selection of Upstream Cell

Determining the upstream cell for use in the second order upwind scheme is not a straightforward task, especially in arbitrary geometries. According to *Patankar* (1980) [61] the expression used to evaluate cell face values must be spatially consistent. Following this requirement, we can write equation 4.22 for two cells with centre points P and N:

$$\phi_f = f_1\left(\phi_N + \nabla\phi_N \Delta\vec{s}_{Nf}\right) + (1 - f_1)\left(\phi_P + \nabla\phi_P \Delta\vec{s}_{Pf}\right) \tag{4.22}$$

$\Delta\vec{s}_{Nf}$ and $\Delta\vec{s}_{Pf}$ are the vectors from the cell centroids of N and P to the face centroid of f. Using the inverse distance weighting approach by f_1 as given in equation 4.16 this method allows a spatially consistent evaluation of ϕ_f. In order to determine whether cell N has actually any convective influence on cell P, i.e. is an upstream cell or not, the entire convective term as derived in equation 4.15 can be evaluated as follows:

$$\sum_{i=1}^{n} \max\left(-A_i\vec{n}_i \cdot \vec{u}_f, 0\right) \cdot (\phi_P - \phi_{N_i}) \tag{4.23}$$

The face velocity vector $\vec{u}_f$ is evaluated by applying formula 4.22 to its components u, v and w. Using the scalar product between this face velocity vector and the face normal vector, which is pointing outwards of the cell by definition, yields the projection of the velocity onto the face normal. The multiplication with the face area returns the convective mass flux through the cell face. Since the influence of the neighbouring cell is positive when the mass flux points into the current cell, the direction of the mass flux must be inverted, which is done by applying the minus sign in equation 4.23. The maximum function ensures that any convective influence of the neighbouring cell is disregarded as soon as the mass flux points out of the current cell.

4.1.4 Pressure Term

For the sake of convenience the pressure term of equation 4.1 was included into the source term in the derivation and discretisation of a generic transport equation. We shall now explicitly deal with this term starting with its integrated form

$$\frac{1}{\rho}\int_{\Omega}\frac{\partial p}{\partial x_j}\mathrm{d}\Omega \tag{4.24}$$

which is to be included in the momentum equations for $x_1 = u$, $x_2 = v$ and $x_3 = w$. Once again using the divergence theorem, we obtain

$$\frac{1}{\rho}\int_{A} p n_{x_j}\mathrm{d}A \tag{4.25}$$

where n_{x_j} denotes the component of the face normal vector of the surrounding surface A which

points into the direction of x_j. This can be discretised as

$$S_u = \frac{1}{\rho} \sum_{i=1}^{n} n_{x_j,i} \cdot A_i \cdot p_{f,i} \tag{4.26}$$

with $p_{f,i}$ meaning the pressure on each face. S_u, the result of the discretisation, is a non-linearised source term (see section 4.1.5). The face pressure value can be obtained by weighted interpolation

$$p_f = f_1 p_N + (1 - f_1) p_P \tag{4.27}$$

where f_1 is based on the ratio of the normal distances from both cell centroids to the dividing face.

4.1.5 Source Terms

Other source terms enter the momentum equations via the last term of equation 4.7:

$$\frac{1}{\rho} \int_\Omega S_\phi \mathrm{d}\Omega \tag{4.28}$$

Discretisation of this integral for a cell with volume V yields:

$$\frac{1}{\rho} S_\phi V \tag{4.29}$$

Usually, S_ϕ is linearised, i.e. split into a term $S_{\phi 1}$ dependent of ϕ and a term $S_{\phi 2}$ independent of ϕ. For cell P we can therefore write the complete source term as

$$S_u - S_p \phi_P \tag{4.30}$$

using the relations

$$S_u = \frac{V}{\rho} S_{\phi 1} \text{ and } S_p = -\frac{V}{\rho} S_{\phi 2} \tag{4.31}$$

4.1.6 Complete Discretised Equation

After all terms of the momentum equation have been discretised they can be assembled to yield the complete discretised set of three momentum equations for $j = 1, 2, 3$:

$$\sum_{i=1}^{n} \left[\nu A_i \frac{u_{j,N_i} - u_{j,P}}{\delta_{N_i P}} + \max\left(-A_i \vec{n_i} \cdot \vec{u}_{f_i}, 0\right) \cdot \left(u_{j,N_i} - u_{j,P}\right) \right.$$
$$\left. -\frac{1}{\rho} n_{x_j,i} \cdot A_i \cdot \left(f_1 p_{N_i} + (1 - f_1) p_P\right) \right] + S_u - S_p u_{j,P} = 0 \tag{4.32}$$

In a next step all terms containing the transported properties for cell P are arranged on the left hand side of the equation and all neighbour properties and nonlinear source terms on the right hand side in order to receive an equation of the type

$$a_P u_{j,P} = \sum_{i=1}^{n} a_{N_i} u_{j,N_i} + b \tag{4.33}$$

which is the typical form of a discretised equation when using the Finite Volume method. In the problem discussed here, the coefficients a_P and a_{N_i} become

$$a_P = \sum_{i=1}^{n} \left[\frac{\nu A_i}{\delta_{N_i P}} + \max\left(-A_i \vec{n_i} \cdot \vec{u}_{f_i}, 0\right) \right] + S_p \quad \text{and}$$
$$a_{N_i} = \frac{\nu A_i}{\delta_{N_i P}} + \max\left(-A_i \vec{n_i} \cdot \vec{u}_{f_i}, 0\right) \tag{4.34}$$

The source term b finally becomes

$$b = \sum_{i=1}^{n} \left[-\frac{1}{\rho} n_{x_j,i} \cdot A_i \cdot \left(f_1 p_{N_i} + (1 - f_1) p_P\right) \right] + S_u \tag{4.35}$$

4.1.7 Basic rules of the Finite Volume Method

Patankar (1980) [61] states the four basic rules which must not be violated in the discretisation of the governing equations:

1. *Consistency at control-volume faces*: fluxes across faces of adjacent cells must be represented by the same expression in the discretisation equations,

2. *Positive coefficients*: all coefficients a_P and a_{N_i} must always be positive,
3. *Negative-slope linearisation of the source term*: in order to avoid a violation of rule 2, S_p must always be positive,
4. *Sum of the neighbour coefficients*: the coefficient a_P must always equal the sum of the neighbour coefficients in the absence of S_p.

In section 4.1.3 a consistent formula (eq. 4.22) was developed to evaluate transported properties at cell faces. By applying this formula to all occurrences of $\vec{u}_{f_i}$, the first rule is observed. Rule 2 is satisfied in the presented discretisation since the values of ν, A_i and δ_{N_iP} are physical and geometrical properties which cannot become negative, and the maximum-function approach for the convective terms ensures that this term is always positive as well. Since no linearised source term occurs in the momentum equations, the third rule is not violated. Finally, from equation 4.34 it is clearly visible that a_P equals the sum of all neighbour coefficients, thus satisfying the fourth rule.

4.2 Pressure Correction Equation

The pressure appears in all three momentum equations where it usually represents the main momentum source term. It is therefore highly important to obtain a valid pressure field. This is, however, a very challenging task since there is no governing equation for pressure. A common approach in incompressible flows is to use the continuity equation to couple pressure and velocity, introducing a constraint on the solution such that if the correct pressure field is applied in the momentum equations, the resulting velocity field satisfies continuity (*Versteeg & Malalasekera* (2001) [84]). This is achieved using an iterative procedure called *SIMPLE algorithm* – which is an acronym for Semi-Implicit Method for Pressure-Linked Equations – first presented in *Patankar & Spalding* (1972) [62]. The idea behind it is to use a guessed pressure field to solve the momentum equations, and to use the resulting velocity field in a so-called pressure correction equation, which is derived from the continuity equation, in order to obtain a pressure correction field. This field again is used to correct velocity and pressure distributions before the next iteration cycle is entered. Figure 4.4 illustrates this procedure; a detailed flow-chart of its implementation in RSim-3D is provided in appendix A.

One of the major challenges in establishing a valid pressure field is illustrated in figure 4.5. In a regularly spaced grid arrangement a decoupling of the pressure term (eq. 4.26) between neigh-

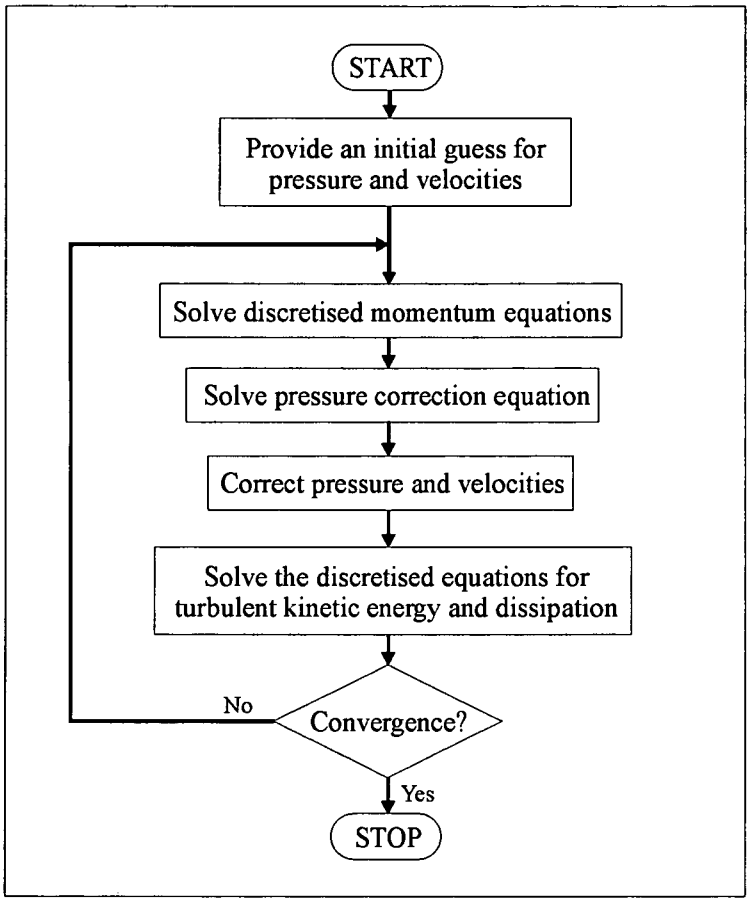

Figure 4.4: Flowchart for the SIMPLE algorithm

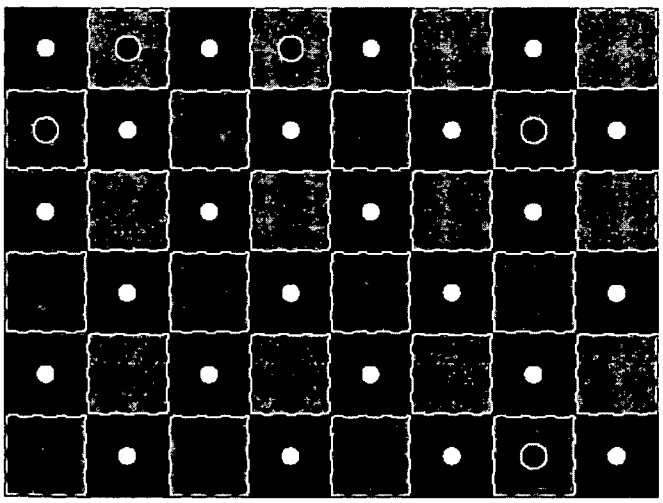

Figure 4.5: Checker-board pressure field (*Steinrück* (2002) [78])

bouring cells may result, leading to a so-called *checker-board pressure field* (*Patankar* (1980) [61], *Steinrück* (2002) [78]). While such a pressure distribution perfectly satisfies the momentum equations it impairs overall convergence and yields oscillatory solutions. In the polyhedral grid arrangement adopted in this work, there is no reason to expect oscillations in the horizontal projection of the pressure field since the cell shapes do not permit such solutions to occur. In the vertical direction, however, the grid is structured and therefore at a high risk of producing checker-board solutions. A usual remedy to this problem is to store pressure and velocities in different locations within the grid, which results in a staggered grid arrangement (see fig. 4.6). In the polyhedral grid setup this would result in large additional complexities as far as storage requirements and memory structures are concerned, which renders the approach not feasible. Instead, an approach presented by *Rhie & Chow* (1983) [66] is used where oscillations are prevented by introducing a third derivative term in pressure into the expression for the mass flux over cell faces. This term is only added when the continuity defect in the pressure correction equation has to be resolved and is not applied to the convection terms in the momentum equations.

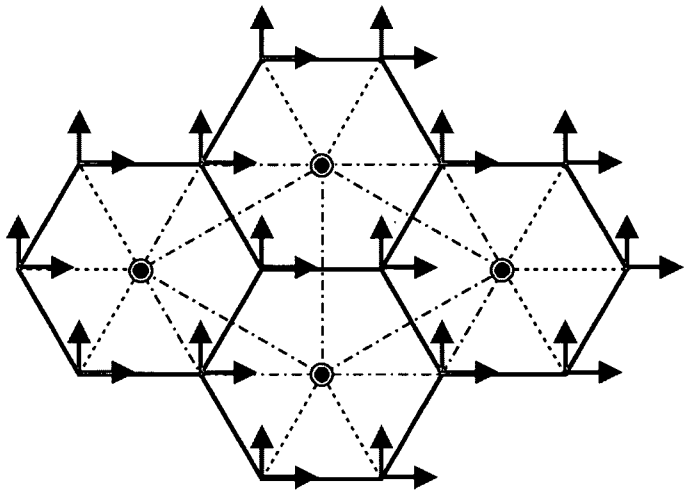

Figure 4.6: Complex staggered variable placement (location of velocities: red vectors; location of pressure: blue circles)

Following *Davidson* (1996) [16] the derivation starts with the evaluation of the pressure gradient vector at the *centroids* of two adjacent cells P and N_i and projecting it on the normal vector of the dividing face:

$$\left\{\frac{\partial p}{\partial x_j}\right\}_P \vec{n_i} \quad \text{and} \quad \left\{\frac{\partial p}{\partial x_j}\right\}_{N_i} \vec{n_i} \tag{4.36}$$

The normal vector projection of the pressure gradient at the *cell face* can be written as

$$\left\{\frac{\partial p}{\partial x_j}\right\}_f \vec{n_i} = \frac{p_{N_i} - p_P}{\delta_{PN_i}} \tag{4.37}$$

with δ_{PN_i} denoting the spatial distance between the cell centroids. Defining the third derivative term as difference between two first derivative terms, the first evaluated at the cell centroids (eq. 4.36), the second at the cell face (eq. 4.37), we obtain

$$\frac{p_{N_i} - p_P}{\delta_{PN_i}} - \left(f_1 \left\{\frac{\partial p}{\partial x_j}\right\}_{N_i} + (1 - f_1)\left\{\frac{\partial p}{\partial x_j}\right\}_P\right)\vec{n_i} \tag{4.38}$$

using the interpolation technique presented in equation 4.27 with f_1 implemented as distance-weighted coefficient.

The mass flux at the cell faces can now be written as (*Davidson* (1996) [16])

$$\dot{m}_{f_i} = \rho A_i \vec{u}_{f_i}\vec{n}_i - \left(\frac{\rho V A}{a_P}\right)_f \left[\frac{p_{N_i} - p_P}{\delta_{PN_i}} - \left(f_1 \left\{\frac{\partial p}{\partial x_j}\right\}_{N_i} + (1 - f_1)\left\{\frac{\partial p}{\partial x_j}\right\}_P\right)\vec{n_i}\right] \tag{4.39}$$

once again using the operator $(.)_f$ to denote that its argument is to be evaluated at the face. With a_P as the discretised diagonal coefficient in the momentum equations (eq. 4.34), we can write

$$\left(\frac{\rho V A}{a_P}\right)_f = V_f A_i \left[\frac{f_1}{a_{P,N_i}} + \frac{1 - f_1}{a_{P,P}}\right] \tag{4.40}$$

with V_f denoting the face control volume defined in figure 4.1. The density ρ is dropped from the right hand side of the equation since the discretised diagonal coefficients a_{P,N_i} and $a_{P,P}$ were already divided by the density during the derivation of the momentum equations. The face control volume V_f can now be approximated as

$$V_f = A_i \delta_{PN_i} \tag{4.41}$$

which allows us to rewrite equation 4.39 as follows:

$$\begin{aligned} \dot{m}_{f_i} &= \rho A_i \vec{u}_{f_i}\vec{n}_i \\ &\quad - A_i^2 \left[\frac{f_1}{a_{P,N_i}} + \frac{1 - f_1}{a_{P,P}}\right]\left[(p_{N_i} - p_P) - \delta_{PN_i}\left(f_1 \left\{\frac{\partial p}{\partial x_j}\right\}_{N_i} + (1 - f_1)\left\{\frac{\partial p}{\partial x_j}\right\}_P\right)\vec{n_i}\right] \end{aligned} \tag{4.42}$$

The face velocity $\vec{u}_{f_i}$ can be evaluated by applying equation 4.22 to its components. Finally, both pressure gradients at cell centroids are obtained through equation 4.19.

In a next step, we express the continuity equation for a discretised cell with n faces by using the mass flux:

$$\sum_{i=1}^{n} \dot{m}_{f_i} = 0 \tag{4.43}$$

However, during an iterative procedure, this equation will not be satisfied if the mass flux was derived using the pressure and velocity fields from a previous iteration step. Therefore the mass fluxes $\dot{m}_{f_i}$ are split into an old value $\dot{m}^*_{f_i}$ and a correction $\dot{m}'_{f_i}$:

$$\dot{m}_{f_i} = \dot{m}^*_{f_i} + \dot{m}'_{f_i} \tag{4.44}$$

In a converged solution, the pressure correction must be zero. Hence, using the equation for the mass flux over cell faces (eq. 4.42) without the stabilising third derivative term, we obtain the following relation

$$\dot{m}'_{f_i} = \rho A_i \vec{u}_{f_i} \vec{n}_i = A_i^2 \left(\frac{f_1}{a_{P,N_i}} + \frac{1-f_1}{a_{P,P}} \right) \left(p'_{N_i} - p'_P \right) \tag{4.45}$$

where p'_{N_i} and p'_P are the respective pressure correction values for cell centroids N_i and P. Inserting this expression into equation 4.44, the continuity equation (eq. 4.43) yields:

$$\sum_{i=1}^{n} A_i^2 \left(\frac{f_1}{a_{P,N_i}} + \frac{1-f_1}{a_{P,P}} \right) \left(p'_{N_i} - p'_P \right) + \sum_{i=1}^{n} \dot{m}^*_{f_i} = 0 \tag{4.46}$$

After arranging all terms containing the pressure correction for P on the left hand side and those for the neighbours N_i on the right hand side we obtain an equation similar to eq. 4.33 in the usual way of a Finite Volume discretisation:

$$a_P p'_P = \sum_{i=1}^{n} a_{N_i} p'_{N_i} + b \tag{4.47}$$

The coefficients a_P, a_{N_i} and b subsequently take the following values:

$$a_P = \sum_{i=1}^{n} A_i^2 \left(\frac{f_1}{a_{P,N_i}} + \frac{1-f_1}{a_{P,P}} \right)$$

$$
\begin{aligned}
a_{N_i} &= A_i^2 \left(\frac{f_1}{a_{P,N_i}} + \frac{1-f_1}{a_{P,P}} \right) \\
b = \sum_{i=1}^{n} \dot{m}_{f_i}^* &= \sum_{i=1}^{n} \left\{ \rho A_i \vec{u}_{f_i}^* \vec{n}_i - A_i^2 \left[\frac{f_1}{a_{P,N_i}} + \frac{1-f_1}{a_{P,P}} \right] \left[\left(p_{N_i}^* - p_P^* \right) \right. \right. \\
&\quad \left. \left. -\delta_{PN_i} \left(f_1 \nabla p_{N_i} + (1-f_1) \nabla p_P \right) \vec{n}_i \right] \right\}
\end{aligned}
\tag{4.48}
$$

p^* and $\vec{u}_f^*$ are pressure and velocity fields from a previous iteration step. ∇p was used as a short-hand notation for the pressure gradient evaluated at cell centroids.

Assessment of the discretised pressure correction equation in regard to the four basic rules of the Finite Volume Method (section 4.1.7) gives:

1. Consistent functions were used to obtain velocities at cell faces and weighted diagonal coefficients of the momentum equation.
2. The coefficients a_P and a_{N_i} are dependent on the face areas and the diagonal coefficients of the momentum equation only, which both take positive values at any time.
3. There is no linearised source term that could become negative.
4. a_P is equal to the sum of all a_{N_i} for a given cell.

Hence the discretised pressure correction equation obeys all basic rules of the Finite Volume Method.

It is important to point out that the result of the solution of the pressure correction equation is a pressure *correction* field for the whole computational domain. Thus the pressure field itself is never obtained explicitly but only after applying the pressure correction everywhere in a final correction step which also includes updating the velocity field:

$$
\begin{aligned}
p &= p^* + p' \\
u_j &= u_j^* - \frac{V}{\rho a_{P,P}} \frac{\partial p'}{\partial x_j}
\end{aligned}
\tag{4.49}
$$

V denotes the cell volume while $a_{P,P}$ is the discretised diagonal coefficient in the momentum equation which must be multiplied with the density ρ since the momentum equations were derived without density terms. The gradient of the pressure correction in all three coordinate direc-

tions can be obtained by making use of the divergence theorem (eq. 4.6) once more,

$$\frac{\partial p'}{\partial x_j} = \frac{1}{V}\sum_{i=1}^{n} p'_{f_i} \cdot A_i \cdot n_{i,j} \tag{4.50}$$

where the face values of the pressure correction p'_{f_i} can easily be obtained using linear interpolation.

4.3 Turbulence Modelling

4.3.1 Turbulence Models

The first step towards modelling of turbulent flows is the splitting of velocity components and pressure into a mean and a fluctuating component:

$$\begin{aligned} u_j &= \bar{u}_j + u'_j \\ p &= \bar{p} + p' \end{aligned} \tag{4.51}$$

Introducing this concept into the Navier-Stokes equations the Reynolds equations for turbulent flow can be written. In analogy to equation 4.1, they are written in a complete notation:

$$\begin{aligned}
\frac{\partial \bar{u}}{\partial t} + \bar{u}\frac{\partial \bar{u}}{\partial x} + \bar{v}\frac{\partial \bar{u}}{\partial y} + \bar{w}\frac{\partial \bar{u}}{\partial z} &= -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x} + \nu\frac{\partial^2 \bar{u}}{\partial x^2} + \nu\frac{\partial^2 \bar{u}}{\partial y^2} + \nu\frac{\partial^2 \bar{u}}{\partial z^2} - \frac{\partial \overline{u'^2}}{\partial x} - \frac{\partial \overline{u'v'}}{\partial y} - \frac{\partial \overline{u'w'}}{\partial z} + f_x \\
\frac{\partial \bar{v}}{\partial t} + \bar{u}\frac{\partial \bar{v}}{\partial x} + \bar{v}\frac{\partial \bar{v}}{\partial y} + \bar{w}\frac{\partial \bar{v}}{\partial z} &= -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial y} + \nu\frac{\partial^2 \bar{v}}{\partial x^2} + \nu\frac{\partial^2 \bar{v}}{\partial y^2} + \nu\frac{\partial^2 \bar{v}}{\partial z^2} - \frac{\partial \overline{u'v'}}{\partial x} - \frac{\partial \overline{v'^2}}{\partial y} - \frac{\partial \overline{v'w'}}{\partial z} + f_y \\
\frac{\partial \bar{w}}{\partial t} + \bar{u}\frac{\partial \bar{w}}{\partial x} + \bar{v}\frac{\partial \bar{w}}{\partial y} + \bar{w}\frac{\partial \bar{w}}{\partial z} &= -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial z} + \nu\frac{\partial^2 \bar{w}}{\partial x^2} + \nu\frac{\partial^2 \bar{w}}{\partial y^2} + \nu\frac{\partial^2 \bar{w}}{\partial z^2} - \frac{\partial \overline{u'w'}}{\partial x} - \frac{\partial \overline{v'w'}}{\partial y} - \frac{\partial \overline{w'^2}}{\partial z} + f_z
\end{aligned} \tag{4.52}$$

Each momentum equation contains three additional terms which involve products of fluctuating velocities. These additional terms act as turbulent stresses (Reynolds stresses) on the mean velocity components (*Versteeg & Malalasekera* (2001) [84]). In tensor notation we can write the Reynolds equations in short:

$$\overline{u_i}\frac{\partial \overline{u_j}}{\partial x_i} = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} - \frac{\partial \overline{u'_i u'_j}}{\partial x_j} + \nu\frac{\partial^2 \overline{u_i}}{\partial x_j^2} + S \tag{4.53}$$

In general it is not possible to derive governing equations for the Reynolds stresses. Several turbulence models are available to deal with this *closure problem*, the most common methods are described in the following sections.

Boussinesq established a relationship between the Reynolds stresses and the mean rates of deformation, resulting in

$$-\overline{u_i' u_j'} = \nu_t \left(\frac{\partial \overline{u_i}}{\partial x_j} + \frac{\partial \overline{u_j}}{\partial x_i} \right) - \frac{2}{3} k \delta_{ij} \tag{4.54}$$

where k is the turbulent kinetic energy

$$k = \frac{1}{2} \left(\overline{u'^2} + \overline{v'^2} + \overline{w'^2} \right) \tag{4.55}$$

and δ_{ij} the Kronecker delta, which takes the value $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ if $i \neq j$. The symbol ν_t is used to denote the turbulent or eddy viscosity for which an assumption must be made. Based on the complexity of this assumption, a distinction in zero-equation-, one-equation-two-equation-, and turbulent stress models is made (*Rodi* (1984) [68]).

Zero-Equation Models

This model type is characterised by the absence of any kind of transport equation for turbulence quantities. The eddy viscosity is either assumed constant or related to the mean velocity distribution. According to *Rodi* (1984) [68] a constant eddy-viscosity assumption has little significance for the calculation of hydrodynamic properties since in flow situations where the turbulence terms are unimportant the model has no influence anyway, and in all other flow situations the model is mostly too coarse to describe this behaviour correctly.

A specific way to relate the eddy viscosity to the mean velocity gradient significant in simple two-dimensional flows was first proposed by Prandtl,

$$\nu_t = \ell_m^2 \left| \frac{\partial \bar{u}}{\partial y} \right| \tag{4.56}$$

which yields much more realistic results. The parameter ℓ_m is the so-called mixing length which can be computed by simple algebraic formulae based on the flow type encountered. A disadvantage of this model is its incapability of describing flows with separation and recirculation (*Versteeg & Malalasekera* (2001) [84]), therefore it is not the best choice for the present work.

One-Equation Models

In this model type the eddy viscosity is related to a velocity scale for which a transport equation can be written. Known as the Kolmogorov-Prandtl expression, it yields

$$\nu_t = c_\mu' \sqrt{k} L \tag{4.57}$$

with $\sqrt{k}$ as the velocity scale related to the turbulent kinetic energy defined in equation 4.55, L a characteristic length of the flow domain and c_μ' an empirical constant. Following *Versteeg & Malalasekera* (2001) [84], the model equation for the turbulent kinetic energy k written in coordinate-free vector form in analogy to equation 4.4 can be written as:

$$\underbrace{\frac{\partial k}{\partial t}}_{(I)} + \underbrace{\mathrm{div}\left(k\vec{\bar{u}}\right)}_{(II)} = \underbrace{\mathrm{div}\left(\frac{\nu_t}{\sigma_k}\mathrm{grad}k\right)}_{(III)} - \underbrace{\overline{u_i' u_j'} \cdot E_{ij}}_{(IV)} - \underbrace{\varepsilon}_{(V)} \tag{4.58}$$

In this equation, σ_k is an empirical constant. E_{ij} is the mean rate of deformation of a finite fluid volume, given by:

$$E_{ij} = \frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \tag{4.59}$$

The symbol ε denotes the viscous dissipation, i.e. the transfer of kinetic energy into internal energy of the fluid, and stands for:

$$\varepsilon = \nu \overline{\frac{\partial u_i'}{\partial x_j}} \overline{\frac{\partial u_i'}{\partial x_j}} \tag{4.60}$$

The transport equation for the turbulent kinetic energy therefore consists of five main terms:

- a transient term, describing the rate of change of k over time (I)
- a term describing the convective transport (II),
- a term giving the diffusive transport (III),
- a term accounting for production of kinetic energy by shear (IV) which is often also denoted P_k,
- and finally a term describing the dissipation of kinetic energy (V).

Following *Rodi* (1984) [68], in a one-equation model the dissipation is usually modelled by the expression

$$\varepsilon = c_D \frac{k^{\frac{3}{2}}}{L} \tag{4.61}$$

where c_D is another empirical constant.

Rodi (1984) [68] assesses the class of one-equation models as superior to the models based on the mixing-length hypothesis since they account for convective and diffusive transport of the turbulent velocity scale. However, one of the main difficulties is the specification of the length scale L in flows that are more complex than shear layers as there is little empirical information available on the length scale distribution. That's the reason why two-equation models, where the length scale is determined from another transport equation, are a better choice for the present work than one-equation models.

Two-Equation Models

The most popular two-equation model today is the $k - \varepsilon$ model by *Launder & Spalding* (1974) [40]. Following the idea that dissipation itself is a process influencing the length scale, a transport equation for ε can be derived

$$\frac{\partial \varepsilon}{\partial t} + \mathrm{div}\left(\varepsilon \vec{u}\right) = \mathrm{div}\left(\frac{\nu_t}{\sigma_\varepsilon}\mathrm{grad}\varepsilon\right) - C_{1\varepsilon}\frac{\varepsilon}{k}\overline{u_i' u_j'} \cdot E_{ij} - C_{2\varepsilon}\frac{\varepsilon^2}{k} \tag{4.62}$$

where the eddy viscosity ν_t is modelled as

$$\nu_t = C_\mu \frac{k^2}{\varepsilon} \tag{4.63}$$

and $C_{1\varepsilon}$, $C_{2\varepsilon}$, σ_ε and C_μ are empirical constants like σ_k in the transport equation for k. The values recommended by *Launder & Spalding* (1974) [40] are:

$$\begin{aligned} C_\mu &= 0.09 \\ \sigma_k &= 1.00 \\ \sigma_\varepsilon &= 1.30 \\ C_{1\varepsilon} &= 1.44 \\ C_{2\varepsilon} &= 1.92 \end{aligned} \tag{4.64}$$

The meaning of the terms in the ε- equation is the same as those in the k-equation with the last two terms accounting for production and destruction of ε.

One of the advantages of the $k-\varepsilon$ model is the validity of the model constants for a large number of flow situations. However, complete universality of the constants cannot and should not be expected (*Rodi* (1984) [68]). Problems where the standard $k-\varepsilon$ model performs poorly have been well documented in literature. These include rotating flows and flows with highly curved boundary layers, fully developed flows in non-circular ducts (*Versteeg & Malalasekera* (2001) [84]) or the prediction of the spreading rate of axisymmetric jets (*Rodi* (1972) [67]). Modifications of the model constants have been proposed to overcome some of these problems. The RNG $k-\varepsilon$ model (*Yakhot & Orszag* (1986) [93]) and the $k-\varepsilon$ model for low Reynolds numbers (*Patel et al.* (1985) [63]) are among the better known ones.

Another two-equation model is the $k-\omega$ model by *Wilcox* (1994) [90]. This model uses an inverse time scale denoted ω – defined as dissipation per turbulent kinetic energy – and employs a transport equation for this quantity which is quite similar to the ε-equation except for a different set of empirical constants. *Wilcox* (1994) [90] states that the model has a high numerical stability and shows better rates of convergence than comparable models. However, its applicability decreases with increasing Reynolds numbers with the model performing best at low to medium Reynolds numbers. Furthermore, the model is quite sensitive as far as the choice of boundary conditions is concerned. To overcome this problem a modified $k-\omega$ model has been proposed by *Menter* (1994) [46].

Because the $k-\varepsilon$ model has been used extensively in the past and is therefore very well documented for its applicability to flows with high Reynolds numbers, it is chosen for this work.

Reynolds Stress Models

Reynolds or turbulent stress models, also called second-moment closure models, account for individual transport of the six Reynolds stresses $\overline{u_i' u_j'}$. This overcomes the limitations introduced by the concept of an isotropic eddy viscosity and allows for an application of the model to flow situations which do not yield satisfactory results when computed using two-equation models. The single most relevant flow phenomenon in river flow applications affected by this limitation is the prediction of turbulence-driven secondary motions (*Rodi* (1984) [68]).

On the other hand, the application of Reynolds stress models is highly complicated and computationally expensive. Six additional transport equations need to be solved and all of them require the interpolation of gradients at cell faces which is a fairly complex task in the case of arbitrary

cell shapes as employed in the present work. However, a requirement for the successful application of Reynolds stress models is the availability of measurements for all transported properties at inlet and outlet of the computational domain, including correlations between velocity fluctuations. If this necessary input data can be provided, this type of turbulence models has the capabilities to account for an improved representation of the physical processes of turbulence. An implementation of a Reynolds stress model into RSim-3D is therefore an important prospect for future improvements of this simulation model.

4.3.2 Discretisation

Transport equation for kinetic energy k

After dropping the transient term from the transport equation for turbulent kinetic energy k (eq. 4.58) it can be presented in an integrated form suitable for a Finite Volume discretisation:

$$\int_\Omega \mathrm{div}\left(k\vec{\bar{u}}\right)\mathrm{d}\Omega = \int_\Omega \mathrm{div}\left(\frac{\nu_t}{\sigma_k}\mathrm{grad}k\right)\mathrm{d}\Omega - \int_\Omega \overline{u_i'u_j'}\cdot E_{ij}\mathrm{d}\Omega - \int_\Omega \varepsilon \mathrm{d}\Omega \tag{4.65}$$

The convective and diffusive terms can be discretised in the same way as in the momentum equations. Following the procedure outlined in equations 4.7 through 4.13, we obtain for the diffusive term

$$\int_\Omega \mathrm{div}\left(\frac{\nu_t}{\sigma_k}\mathrm{grad}k\right)\mathrm{d}\Omega \tag{4.66}$$

the following discretised form

$$\frac{1}{\sigma_k}\sum_{i=1}^{n} A_i \nu_{t,f_i}\frac{k_{N_i}-k_P}{\delta_{N_iP}} \tag{4.67}$$

where ν_{t,f_i} is the eddy viscosity at the cell face f_i, obtained by linear interpolation from the values at neighbouring cell centroids.

In analogy to equations 4.14 through 4.23, the convective term

$$\int_\Omega \mathrm{div}\left(k\vec{\bar{u}}\right)\mathrm{d}\Omega \tag{4.68}$$

can be discretised to become

$$\sum_{i=1}^{n}\max\left(-A_i\vec{n}_i\cdot\vec{\bar{u}}_f, 0\right)\cdot\left(k_P - k_{N_i}\right) \tag{4.69}$$

with $\vec{u}_f$ denoting the face velocity vector defined in the derivation of the momentum equations.

In order to discretise the production term we can employ the Boussinesq approximation (eq. 4.54) in conjunction with equation 4.59 to obtain

$$-\int_\Omega \overline{u_i' u_j'} \cdot E_{ij} \mathrm{d}\Omega = P_k = V \left\{ 2 \left[\left(\frac{\partial \bar{u}}{\partial x} \right)^2 + \left(\frac{\partial \bar{v}}{\partial y} \right)^2 + \left(\frac{\partial \bar{w}}{\partial z} \right)^2 \right] + \left(\frac{\partial \bar{u}}{\partial y} + \frac{\partial \bar{v}}{\partial x} \right)^2 + \left(\frac{\partial \bar{u}}{\partial z} + \frac{\partial \bar{w}}{\partial x} \right)^2 + \left(\frac{\partial \bar{v}}{\partial z} + \frac{\partial \bar{w}}{\partial y} \right)^2 - \frac{2}{3} k_P \left(\frac{\partial \bar{u}}{\partial x} + \frac{\partial \bar{v}}{\partial y} + \frac{\partial \bar{w}}{\partial z} \right) \right\} \tag{4.70}$$

which was presented in a full notation for the sake of clarity. All six different partial differentials are evaluated using equation 4.19. Fortunately, these differentials are also needed in the second order upwind procedure and therefore no additional effort is needed to obtain them. While it may be tempting to implement the part containing the turbulent kinetic energy as a linearised source term, it is not wise to do so since it cannot be guaranteed that this term will always be positive, thus violating the third basic rule of the Finite Volume method. Therefore the complete right hand side of equation 4.70 must be implemented as a nonlinear source term S_u.

Finally, the dissipation term is discretised as:

$$-\int_\Omega \varepsilon \mathrm{d}\Omega = -\varepsilon_P V \tag{4.71}$$

While it was not possible to linearise the production term, it is possible to do so with the dissipation term which will have a stabilising effect on the solution. Neither a negative dissipation nor a negative cell volume are physically possible, and so we can "linearise" the dissipation term without violation of the basic rules of the Finite Volume Method to yield

$$-\int_\Omega \varepsilon \mathrm{d}\Omega = -\frac{\varepsilon_P V}{k_P^*} \cdot k_P \tag{4.72}$$

where k_P^* is the turbulent kinetic energy value after the previous iteration, resulting in the linear source term

$$S_p = \frac{\varepsilon_P V}{k_P^*} \tag{4.73}$$

Putting the terms together and re-arranging them in the same way as in the previous sections for

the momentum and pressure correction equations, we obtain

$$a_P k_P = \sum_{i=1}^{n} a_{N_i} k_{N_i} + b \tag{4.74}$$

with the following values for a_P, a_{N_i} and b:

$$\begin{aligned} a_P &= \sum_{i=1}^{n} \left\{ \frac{A_i}{\sigma_k \delta_{N_i P}} \left[f_1 \nu_{t,N_i} + (1 - f_1)\, \nu_{t,P} \right] + \max\left(-A_i \vec{n_i} \cdot \vec{\bar{u}}_f, 0 \right) \right\} + \frac{\varepsilon_P V}{k_P^*} \\ a_{N_i} &= \frac{A_i}{\sigma_k \delta_{N_i P}} \left[f_1 \nu_{t,N_i} + (1 - f_1)\, \nu_{t,P} \right] + \max\left(-A_i \vec{n_i} \cdot \vec{\bar{u}}_f, 0 \right) \\ b &= V \left\{ 2 \left[\left(\frac{\partial \bar{u}}{\partial x} \right)^2 + \left(\frac{\partial \bar{v}}{\partial y} \right)^2 + \left(\frac{\partial \bar{w}}{\partial z} \right)^2 \right] + \left(\frac{\partial \bar{u}}{\partial y} + \frac{\partial \bar{v}}{\partial x} \right)^2 + \left(\frac{\partial \bar{u}}{\partial z} + \frac{\partial \bar{w}}{\partial x} \right)^2 \right. \\ &\quad \left. + \left(\frac{\partial \bar{v}}{\partial z} + \frac{\partial \bar{w}}{\partial y} \right)^2 - \frac{2}{3} k_P^* \left(\frac{\partial \bar{u}}{\partial x} + \frac{\partial \bar{v}}{\partial y} + \frac{\partial \bar{w}}{\partial z} \right) \right\} \end{aligned} \tag{4.75}$$

It is important to note that upon solution of the discretised k-equation slightly negative values for k may be obtained. This is a phenomenon of numerics which must be avoided since it disequilibrates the whole solution algorithm. Therefore it must be ensured that k is always positive by enforcing $k \geq 0$ during the iteration cycle.

A consideration of the discretised k-equation regarding the basic rules of the Finite Volume method yields that none of these rules are violated since consistent formulations were used at cell faces, all coefficients are always positive, the linearised source term is positive and the diagonal coefficient equals the sum of the neighbour coefficients.

Transport equation for dissipation ε

The transport equation for dissipation ε (eq. 4.62) presented without the transient term and in integrated form reads:

$$\int_\Omega \mathrm{div}\left(\varepsilon \vec{\bar{u}}\right) \mathrm{d}\Omega = \int_\Omega \mathrm{div}\left(\frac{\nu_t}{\sigma_\varepsilon} \mathrm{grad}\varepsilon \right) \mathrm{d}\Omega - \int_\Omega C_{1\varepsilon} \frac{\varepsilon}{k} \overline{u_i' u_j'} \cdot E_{ij} \mathrm{d}\Omega - \int_\Omega C_{2\varepsilon} \frac{\varepsilon^2}{k} \mathrm{d}\Omega \tag{4.76}$$

The diffusive and convective terms are very similar to their counterparts in the k-equation. Therefore we can directly write them in discretised form:

$$\int_\Omega \mathrm{div}\left(\frac{\nu_t}{\sigma_\varepsilon} \mathrm{grad}\varepsilon \right) \mathrm{d}\Omega = \frac{1}{\sigma_\varepsilon} \sum_{i=1}^{n} A_i \nu_{t,f_i} \frac{\varepsilon_{N_i} - \varepsilon_P}{\delta_{N_i P}} \tag{4.77}$$

$$\int_\Omega \operatorname{div}\left(\varepsilon\vec{\bar{u}}\right) \mathrm{d}\Omega = \sum_{i=1}^{n} \max\left(-A_i\vec{n_i}\cdot\vec{\bar{u}}_f, 0\right)\cdot\left(\varepsilon_P - \varepsilon_{N_i}\right) \tag{4.78}$$

Using the definition of P_k from equation 4.70 the production term can be discretised as:

$$-\int_\Omega C_{1\varepsilon}\frac{\varepsilon}{k}\overline{u_i'u_j'}\cdot E_{ij}\mathrm{d}\Omega = VC_{1\varepsilon}\frac{\varepsilon_P}{k_P}P_k \tag{4.79}$$

Again, it is not possible to linearise the production term without violating the third basic rule of the Finite Volume method. But it is possible to improve stability in another way; by using the definition of the eddy viscosity (eq. 4.63), we can write

$$VC_{1\varepsilon}\frac{\varepsilon_P}{k_P}P_k = VC_{1\varepsilon}C_\mu P_k\frac{k_P}{\nu_t} \tag{4.80}$$

which removes any direct reference to ε_P from the source term since ν_t is computed before the equations of turbulence are evaluated. Finally, the destruction term of ε is modelled as:

$$-\int_\Omega C_{2\varepsilon}\frac{\varepsilon^2}{k}\mathrm{d}\Omega = -VC_{2\varepsilon}\frac{\varepsilon_P^2}{k_P} \tag{4.81}$$

Using the same procedure as with the production term, it can be written

$$-VC_{2\varepsilon}\frac{\varepsilon_P^2}{k_P} = -VC_{2\varepsilon}C_\mu\frac{k_P}{\nu_t}\varepsilon_P \tag{4.82}$$

and this term can be expressed as a linearised source term:

$$S_p = VC_{2\varepsilon}C_\mu\frac{k_P}{\nu_t} \tag{4.83}$$

Finally, the terms are put together in the usual way to obtain

$$a_P\varepsilon_P = \sum_{i=1}^{n} a_{N_i}\varepsilon_{N_i} + b \tag{4.84}$$

with the following values for a_P, a_{N_i} and b:

$$a_P = \sum_{i=1}^{n}\left\{\frac{A_i}{\sigma_\varepsilon\delta_{N_iP}}\left[f_1\nu_{t,N_i} + (1-f_1)\,\nu_{t,P}\right] + \max\left(-A_i\vec{n_i}\cdot\vec{\bar{u}}_f, 0\right)\right\} + VC_{2\varepsilon}C_\mu\frac{k_P}{\nu_t}$$

$$a_{N_i} = \frac{A_i}{\sigma_\varepsilon \delta_{N_i P}} \left[f_1 \nu_{t,N_i} + (1 - f_1)\, \nu_{t,P} \right] + \max\left(-A_i \vec{n}_i \cdot \vec{\bar{u}}_f, 0 \right) \tag{4.85}$$

$$b = V C_{1\varepsilon} C_\mu P_k \frac{k_P}{\nu_t}$$

The statement about enforcing the positiveness of k applies in the same way to ε as well and the considerations regarding the basic rules of the Finite Volume method also hold true for the discretised ε-equation.

Modification of momentum equations

Very little change is required to solve the Reynolds equations (eq. 4.52) with the technique already derived for the Navier-Stokes equations (eq. 4.1). The equations have the same form if the molecular viscosity μ is replaced by the effective viscosity

$$\mu_{\text{eff}} = \mu + \mu_t \tag{4.86}$$

as pointed out by *Ferziger* (2002) [20]. For the discretised momentum equations in the present work contain the kinematic viscosity ν, it is therefore sufficient to replace it by $\nu_{\text{eff}} = \nu + \nu_t$. This is the only modification needed in order to solve the Reynolds equations.

4.4 Boundary Conditions

4.4.1 Inlet

Momentum equations

At the inlet of the computational domain, the normal velocities are prescribed. Given the discharge Q, usually it is sufficient to compute the normal velocities u_n by

$$u_n = \frac{Q}{\sum_{i=1}^{m} A_i} \tag{4.87}$$

where m is the number of faces at the inlet. This prescribes a uniform velocity distribution which does not occur in reality; a fact that does not matter if the inlet boundary is placed sufficiently far upstream from the area of interest since the flow will develop a natural distribution very quickly.

If this method appears too inaccurate, it is alternatively possible to prescribe a logarithmic velocity profile at the inlet. For points outside of the viscous wall layer, *Schlichting & Gersten* (1997) [71] give the following *law of the wall*

$$u^+ = \frac{u(y)}{u^*} = \frac{1}{\kappa} \ln\left(\frac{y}{y_k}\right) \tag{4.88}$$

where $u(y)$ is the velocity dependent on the wall distance y, u^* the shear velocity defined as

$$u^* = \sqrt{\frac{\tau_w}{\rho}} \tag{4.89}$$

with τ_w denoting the wall shear stress. κ is the Von Karman constant which takes the value $\kappa = 0.41$. For the fully rough flow domain, the parameter y_k can be computed by

$$y_k = k_s \exp\left(-8.0\kappa\right) \tag{4.90}$$

where k_s is Nikuradse's equivalent sand roughness of the wall. Using this relationship, equation 4.88 can be reformulated to read:

$$u(y) = \frac{u^*}{\kappa} \ln\left(\frac{26.58y}{k_s}\right) \tag{4.91}$$

The velocity distribution given in this formula can subsequently be integrated over the depth of the flow domain h in order to obtain the discharge Q:

$$Q = \int_0^h u(y) \cdot \frac{A}{h} \mathrm{d}y \tag{4.92}$$

The result of this integral is used to compute the shear velocity u^*

$$u^* = \kappa \frac{Q}{A} \frac{1}{2.2802 + \ln\left(\frac{h}{k_s}\right)} \tag{4.93}$$

which finally allows us to use expression 4.91 to prescribe the velocity at the inlet dependent on the vertical layer. It should be noted that the denominator of this equation must not become negative for this approach to work. This sets a natural limit for the ratio between the depth of the flow domain and the equivalent sand roughness of the wall.

While the equivalent sand roughness k_s is used throughout many boundary conditions, often this parameter is not available in engineering calculations, but the Strickler coefficient k_{St} is given instead. These two parameters are related by the empirical relationship:

$$k_s = \left(\frac{26.4}{k_{St}}\right)^6 \tag{4.94}$$

According to *USACE* (1994) [83], this formula holds true for riprapped channels where $k_s = D_{90}$, while for natural sediment where $k_s = D_{50}$, the relationship

$$k_s = \left(\frac{29.4}{k_{St}}\right)^6 \tag{4.95}$$

is appropriate. However, these equations are applicable only to medium-range values for the Strickler parameter, i.e. not to smooth and very rough surfaces. *Naudascher* (1987) [50] assesses their applicability for typical Reynolds numbers in river flow situations to be in the range $20 < 4R/k_s < 1000$ where R denotes the hydraulic radius.

Turbulent kinetic energy

If the inlet boundary is placed sufficiently far away from the region of interest, the prescribed values for k have no significant influence on the turbulence field further downstream as long as the $k-\varepsilon$ model is employed. This conclusion can be drawn when comparing the suggestions given by different authors as far as this quantity is concerned. *Ferziger* (2002) [20] suggests to use a small value for k and gives $k = 10^{-4}\bar{u}^2$ as example, while *Versteeg & Malalasekera* (2001) [84] propose to estimate k from the turbulence intensity T_i which is typically between 1% and 6%: $k = \frac{3}{2}\left(\bar{u}T_i\right)^2$. *Davidson & Nielsen* (1995) [17] document the use of $k = 10^{-2}\bar{u}^2$ and *Olsen* (1999) [55] finally relates the turbulent kinetic energy to the wall shear stress τ at the inlet and obtains

$$k_B = \frac{\tau}{\rho\sqrt{C_\mu}} \tag{4.96}$$

for the inlet bed with a linear decrease towards the free surface. This approach is adopted here.

Dissipation

The dissipation ε must return the correct scale, thus it is useful to choose its inlet value in accordance with equation 4.61 which can be reformulated (*Versteeg & Malalasekera* (2001) [84])

as

$$\varepsilon = C_{\mu}^{\frac{3}{4}} \frac{k^{\frac{3}{2}}}{0.07L} \tag{4.97}$$

where L is a characteristic length of the flow domain that can be approximated by $L \approx h$ where h is the water depth. This procedure is also confirmed in a paper by *Davidson & Nielsen* (1995) [17] where an inlet value of $\varepsilon = \frac{0.16k^{1.5}}{0.1L}$ is being used; the only difference is the choice of $0.1L$ as mixing length in the denominator owing to the different type of flow problem modelled.

4.4.2 Outlet

Ferziger (2002) [20] notes that we usually know little about the flow at the outlet and for this reason, these boundaries should be as far downstream of the region of interest as possible. In the present work a zero gradient boundary condition in the streamwise direction is applied to all quantities:

$$\frac{\partial u_j}{\partial n} = \frac{\partial p}{\partial n} = \frac{\partial k}{\partial n} = \frac{\partial \varepsilon}{\partial n} = 0 \tag{4.98}$$

Of course, for this assumption to hold true, the outlet boundary must be placed perpendicular to the flow direction. Additionally it must be noted that the pressure correction equation does only yield relative pressures, therefore it is necessary and common practice to fix the pressure at one outlet node and let the pressure field evolve from there.

As stated in *Versteeg & Malalasekera* (2001) [84] mass conservation over the whole computational domain is not guaranteed during the iterative solution process. Therefore it can be advantageous to sum up the mass flux M_{out} going out of the domain after an iteration cycle and then use the relation

$$u_n = u_n^* \frac{M_{\text{in}}}{M_{\text{out}}} \tag{4.99}$$

to correct the normal velocities u_n at the outlet for the next iteration cycle. This procedure has also been adopted in the present work.

4.4.3 Solid Walls

Momentum equations

Directly at the wall all velocities are zero (no-slip condition) and no convective fluxes take place through the wall. The momentum equations, however, receive sink terms based on the shear

stress at the wall. In order to derive these terms, first the velocity in tangential direction to the wall u_t needs to be computed. This is done by calculating the tangent vector $\vec{t}$ in a first step by using the relations derived by *Ferziger* (2002) [20]

$$\begin{aligned} \vec{u}_p &= \vec{u} - (\vec{u} \cdot \vec{n})\,\vec{n} \\ \vec{t} &= \frac{\vec{u}_p}{|\vec{u}_p|} \end{aligned} \tag{4.100}$$

where $\vec{u}$ is the three-dimensional velocity vector at the nearest wall node and $\vec{n}$ the normal vector of the cell face at the wall, pointing outwards as usual. The tangential velocity is then obtained by projecting the velocity vector on $\vec{t}$:

$$u_t = \vec{u} \cdot \vec{t} \tag{4.101}$$

In a next step it must be evaluated whether the cell centroid nearest to the wall lies within the viscous sublayer or the turbulent outer region. This evaluation is performed by first computing

$$y^+ = \frac{\Delta y_P}{\nu}\sqrt{\frac{\tau_w}{\rho}} \tag{4.102}$$

where Δy_P is the normal distance from the cell centroid to the wall. Assuming the near wall node is in the turbulent outer region – which is almost always the case – the wall shear stress τ_w can be estimated as given by *Versteeg & Malalasekera* (2001) [84],

$$\tau_w = \rho C_\mu^{\frac{1}{4}} k_P^{\frac{1}{2}} \frac{u_t}{u^+} \tag{4.103}$$

where k_P denotes the turbulent kinetic energy at the near-wall node and u^+ is derived from equation 4.88. The limit of the viscous sublayer is given by $y^+ < 11.63$. If y^+ is found to be less than this value, the assumption that the cell centroid was in the turbulent region does not hold, and the wall shear stress must be computed by:

$$\tau_w = \mu \frac{u_t}{\Delta y_P} \tag{4.104}$$

After the wall shear stress has been obtained, the sink term for the momentum equations $j = 1, 2, 3$ reads

$$S_{u,j} = -\frac{\tau_w}{\rho} A_f \cdot t_j \tag{4.105}$$

where t_j is the row of vector $\vec{t}$ that points into the direction of j and A_f denotes the surface area the cell shares with the wall.

Attention must also be paid to the implementation of the pressure term. The pressure p_f at the wall cannot be obtained in the usual way since there is no second cell the linear interpolation could be performed with. *Wesseling* (2001) [89] suggests to extrapolate the needed values from the cell centroid values of the near-wall cell and the cell next to it in the computational domain. The problem with such an approach, however, is that such a second cell is undefined in arbitrary unstructured grid setups – which was also the reason why the QUICK scheme could not be used (section 4.1.3). Some other authors state that their flow problems gave satisfactory results upon using the zero-gradient boundary condition for pressure. Especially in curved channels where the pressure gradients become significant, such an approach is not a feasible solution. Therefore it is proposed to use the pressure gradient ∇p_P obtained via the procedure outlined in equation 4.19 and already used in the pressure correction equation and set

$$p_f = p_P + \nabla p_P \cdot \vec{s} \tag{4.106}$$

where $\vec{s}$ is the vector from the cell centroid to the wall face centroid. However, it should be noted that during the evaluation of the pressure gradient itself assumptions about the value at the face need to be made. Due to these assumptions the pressure gradient within the cell can never be steeper than obtained from all surrounding cells which eventually leads to the same result as in *Wesseling*'s (2001) [89] gradient evaluation.

Turbulent kinetic energy

For the turbulent kinetic energy right at the wall it is appropriate to set $k = 0$ (*Ferziger* (2002) [20]). At the same time, the equations for near-wall cells must receive an explicit source term

$$S_k = \left(\tau_w u_t - \rho C_\mu^{\frac{3}{4}} k_P^{\frac{3}{2}} u^+\right) \frac{V}{\Delta y_P} \tag{4.107}$$

according to *Versteeg & Malalasekera* (2001) [84]. This source term can be linearised, and after introducing k_P^* to denote the value of kinetic energy after the previous iteration and dividing the term by density, we can write:

$$S_p = \frac{C_\mu^{\frac{3}{4}} k_P^{*\frac{1}{2}} u^+ V}{\Delta y_P} \quad \text{and} \quad S_u = \frac{\tau_w u_t V}{\rho \Delta y_P} \tag{4.108}$$

Dissipation

Setting a zero dissipation value at the wall is inappropriate. But since a value at the wall would be needed in order to apply the transport equation for ε there, the usual procedure is to avoid solving the ε-equations in near-wall regions by setting the nodal value of the cell adjacent to a wall to the value given by *Versteeg & Malalasekera* (2001) [84]:

$$\varepsilon_P = \frac{C_\mu^{\frac{3}{4}} k_P^{\frac{3}{2}}}{\kappa \Delta y_P} \tag{4.109}$$

4.4.4 Free Surface

The free surface is implemented as a symmetry boundary which means that no fluxes occur across this boundary. The normal velocity (usually the $u_3 = w$ component) is therefore zero right at the boundary. All other properties take boundary values equal to those encountered in the cell centroid right beneath the free surface. *Ferziger* (2002) [20] notes that a free surface boundary condition of $k \approx 0$ and $\varepsilon \approx 0$ would be appropriate, but on the other hand turbulent structures can be observed exactly at the free surface of a water body. However, as a matter of fact no scalar fluxes take place concerning these quantities. Therefore the condition of zero fluxes of k and ε was used in this work.

The pressure at the free surface is evaluated by using the same gradient approach as described in the previous section. This implies, however, that the pressure is not zero at the surface (except for the single outlet surface cell where it is fixed). The resulting extra pressure head can be directly used to locate and subsequently update the position of the free surface. This approach has been implemented in the present work.

4.5 Solution of Equations

4.5.1 Solver

After the discretised equations have been assembled they need to be solved in an efficient way without consuming too much time or computer memory. Very efficient algorithms are available if the coefficients of the discretised equations can be arranged in diagonally dominant matrix form (*Schmid* (2001) [72]) as it is possible when structured grids are employed. For unstructured

grids with a fixed number of neighbours for each cell it may be possible to use solvers developed in context of the Finite Element method (*Zienkiewicz* (1971) [95]). However, if even the number of neighbouring cells is unknown a priori and varies from cell to cell, none of these techniques is applicable.

The solution methods applicable to problems involving grids with cells that may have an arbitrary number of neighbours include the Gauss-Seidel method (*Press* (1987) [65]), the Conjugate Gradient Squared method (*Barrett et al.* (1994) [9]) or its modification, the Biconjugate Gradient Stabilized method. The latter two exhibit a very good convergence behaviour at the cost of large memory requirements during a cycle of the rather complex solution algorithms. For the work presented here, the Gauss-Seidel method was preferred. It is a slow solution algorithm but there are very few memory requirements for the solver itself and implementation is uncomplicated. Furthermore the algorithm is easy to understand and the reasons for instabilities can be located quickly, which is an advantage during model development and validation.

The Gauss-Seidel method solves the linear system of equations $Ax = b$ where A is a matrix of coefficients, b a vector containing boundary conditions and x the unknown solution vector. In the present work A is composed of the coefficients a_P and a_{N_i}, and b takes the values of the nonlinear source terms in the discretised equations. The algorithm solves the n_{cells} equations one at a time in sequence and can be written as

$$x_i^{(k)} = \frac{b_i - \sum_{j<i} a_{ij} x_j^{(k)} - \sum_{j>i} a_{ij} x_j^{(k-1)}}{a_{ii}} \tag{4.110}$$

where i and j are the cell numbers, thus a_{ij} the coefficients a_{N_i} and a_{ii} the diagonal coefficient a_P for each cell, and k is the inner iteration counter of the algorithm. It is easy to see that results of the solution of previous equations are used as soon as they are available. The solution requires an initial guess $x^{(0)}$ which is provided by using results obtained in the previous outer iteration step. The only exception is the pressure correction equation, where it is useful to start from $p' = 0$ so that the solution for p' does not acquire a large absolute value (*Patankar* (1980) [61]).

The Gauss-Seidel procedure is usually repeated until convergence is obtained. However, convergence is difficult to assess within the solver itself for it is applied to equations with entirely different scales and physical meanings. Furthermore it is not useful to solve the equations until convergence is achieved since the result is only used as an intermediate result in the overall solution (*Ferziger* (2002) [20]). For this reason, the Gauss-Seidel procedure is repeated a predefined number of times only, without extra assessment of convergence after every iteration. Usually, evaluating each equation 10 times is sufficient for the most common flow situations.

4.5.2 Relaxation Scheme

Due to the strong nonlinearity in the governing equations the solution process may rapidly diverge if the solution vector obtained from the solver is used as starting point for the next overall iteration cycle. Especially the pressure correction equation is susceptible to divergence (*Versteeg & Malalasekera* (2001) [84]). It is therefore necessary to employ an under-relaxation scheme, such as

$$\phi^{\text{new}} = \alpha_\phi \phi + (1 - \alpha_\phi)\phi^{\text{old}} \tag{4.111}$$

where the new value ϕ^{new} is obtained from the solution ϕ^{old} after the previous iteration and the solution ϕ given by the solver in the current iteration step by using the relaxation factor α_ϕ which is typically in the range between 0 and 1. The choice of optimum relaxation factors is problem dependent and can be found by test computations or experience. There are, however, typical ranges for the selection of the relaxation factors for the six governing equations of three-dimensional CFD problems:

The velocity relaxation factor α_u is usually in the range 0.5 – 0.8 (*Apsley* (2003) [6]). *Olsen* (2000a) [57] employs a default value of 0.8 but notes that the factor can be set as low as 0.1 for flow situations that are difficult to converge.

For the pressure relaxation factor α_p, *Apsley* (2003) [6] gives the typical range 0.1 – 0.3. This is fully consistent with *Olsen* (1999, 2000a) [55, 57] where a default value of 0.2 is used. The latter work also gives a lower bound of 0.03 for complex flow situations. These observations can be confirmed by the author as well: some flow problems required a pressure relaxation factor less than 0.1 during the first iteration cycles in order to achieve convergence.

Finally, the relaxation factors for the turbulent kinetic energy and the dissipation (α_k and α_ε) default to 0.5 in *Olsen* (2000a) [57] but are said to sometimes require values as low as 0.05 in order to result in a converged solution. RSim-3D uses default values for the relaxation factors according to table 4.1.

RSim-3D uses lower factors for the turbulence properties because the governing equations for these quantities are not the ones that dominate the overall convergence behaviour (see figure 4.8) as the number of iterations required to solve a flow problem is vastly dominated by the pressure correction equation. So, in order to damp the sometimes observed instabilities in the turbulence equations lower default relaxation factors have been chosen. However, for the vast majority of flow situations choosing $\alpha_k = \alpha_\varepsilon = 0.5$ or larger would be sufficient.

Also the update of the free surface position introduces instabilities into the solution process since

Quantity	Symbol	Default value
Velocities	α_u	0.8
Pressure	α_p	0.2
Turb. kinetic energy	α_k	0.3
Dissipation	α_ε	0.3
Free surface position	α_s	0.5

Table 4.1: Default relaxation factors

cell volumes and vertical position of the cell centroids are changed after every update. Depending on the complexity of the flow problem these instabilities may also lead to rapid divergence. Therefore it is advisable to relax the free surface position as well. A value of 0.5 for this value has worked for most flow situations examined. Finally it should be noted that updates of the free surface are not overly useful if the scaled residuals (see section 4.5.3) of all other properties are still large. Therefore these updates are only performed if all residuals are below 0.05.

4.5.3 Residuals

In order to define convergence of a solution an assessment of the solution errors is required. For every property ϕ in each cell the imbalance within the discretised equation can be computed as the difference between right hand side and left hand side. Summing this imbalance over all cells we obtain the unscaled residual R_u^ϕ:

$$R_u^\phi = \sum_{j=1}^{n_{\text{cells}}} \left| \sum_{i=1}^{n} a_{N_i}\phi_{N_i} + b - a_P\phi_P \right| \tag{4.112}$$

In *Fluent* (2003) [25] it is noted that even though R_u^ϕ is a measure of the solution error, it is difficult to judge convergence since no scaling is employed that allows for a generalised assessment in different types of flow problems. A good scaling factor is the left hand side of the discretised equation. Therefore a scaled residual R^ϕ can be introduced

$$R^\phi = \frac{R_u^\phi}{\sum_{j=1}^{n_{\text{cells}}} |a_P\phi_P|} \tag{4.113}$$

that is suitable for judging convergence on any type of flow problem. This formula is used in the present work to determine the residuals of the k and ε equations. For the momentum equations

the denominator term $a_P \phi_P$ is replaced by $a_P \, |\vec{u}_P|$ where $|\vec{u}_P|$ is the magnitude of the velocity at cell P (*Fluent* (2003) [25]).

The unscaled residual for the continuity equation is defined as the absolute value of the source term of the pressure correction equation (eq. 4.48) summed over all cells. This definition is equal to stating that the unscaled continuity residual is the sum of mass creation in all cells:

$$R_u^c = \sum_{j=1}^{n_{\text{cells}}} \left| \sum_{i=1}^{n} \dot{m}_{f_i}^* \right| \tag{4.114}$$

In order to scale the continuity residuals they are divided by the maximum unscaled residual within the first five iterations ($R_{u,5}^c$) resulting in:

$$R^c = \frac{R_u^c}{R_{u,5}^c} \tag{4.115}$$

In the present work, convergence of the solution is obtained when the scaled continuity residual is less than 10^{-4} and all other residuals are less than 10^{-5}.

4.6 Remarks on Numerical Stability and Convergence

Usually, the numerical stability in the iterative procedure increases when lower relaxation factors are being used, even when there is no guarantee that this assumption holds true for every flow situation encountered. On the other hand, lower relaxation factors often result in a significant increase of iterations needed for the solution to converge. *Ferziger* (2002) [20] analysed the number of iterations needed to reduce the residual levels by three orders of magnitude on four different grid setups for the problem of a lid-driven cavity as a function of the velocity relaxation factor α_u. The result is plotted in figure 4.7. From this diagram it is obvious that the optimum relaxation factor α_u for this type of flow problem is around 0.9; higher factors introduce instabilities and oscillations that slow down the overall convergence process while lower factors have a negative influence on the convergence speed as well.

Other sources of instabilities during the solution process can be the improper placement of inlet and outlet boundaries (i.e. not sufficiently far away from the region of interest or not perpendicular to the flow direction), a bad choice of wall boundary conditions or the presence of irregularly shaped cells within the computational domain. These issues can therefore seriously slow down

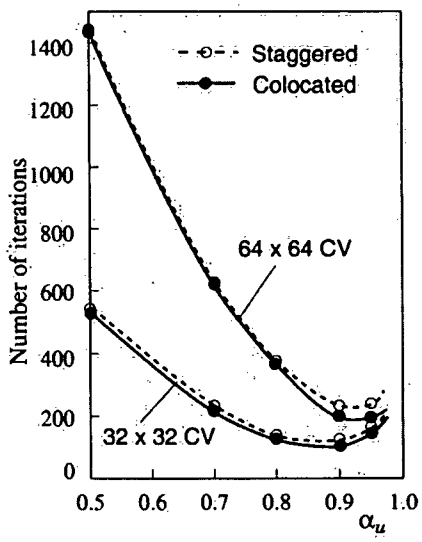

Figure 4.7: Number of iterations for the lid-driven cavity problem (*Ferziger* (2002) [20])

convergence or even cause the solution algorithm to diverge. A natural source of instabilities is the update of the vertical water surface position during the solution process since it leads to the recalculation of cell volumes and the location of the respective cell centroids. But as long as the relaxation factors are not too large, these instabilities are damped very quickly and the residuals reach their original level again soon. This fact is shown in figure 4.8 where the scaled residuals for the simple case of turbulent flow in a straight channel are plotted. The left diagram was created without updates of the water surface; in the right diagram the surface was free to move. The case without moving water surface reached a converged state (continuity residuals have dropped by four orders of magnitude, all other residuals five orders of magnitude) after 730 iterations while the moving water surface problem took 990 iterations, which is a 35% increase in computation time. The instabilities introduced by the surface updates are clearly visible in the convergence norms. It should also be noted that the decrease of the continuity error norm for the fixed surface problem is almost linear in a logarithmic scale.

The time which is needed to obtain a solution for a specific flow problem was discussed to be related to issues of stability and relaxation factors so far. However, these are not the only causes that affect the convergence speed. The technique employed in the solver introduces a relation between the size of the problem (i.e. the number of cells) and convergence speed: the Gauss-Seidel scheme is very efficient in removing local (high-frequency) errors but global (low-frequency) errors are reduced at a rate inversely related to the grid size (*Fluent* (2003) [25]). Hence it is impossible to solve very large problems within an acceptable time span using the

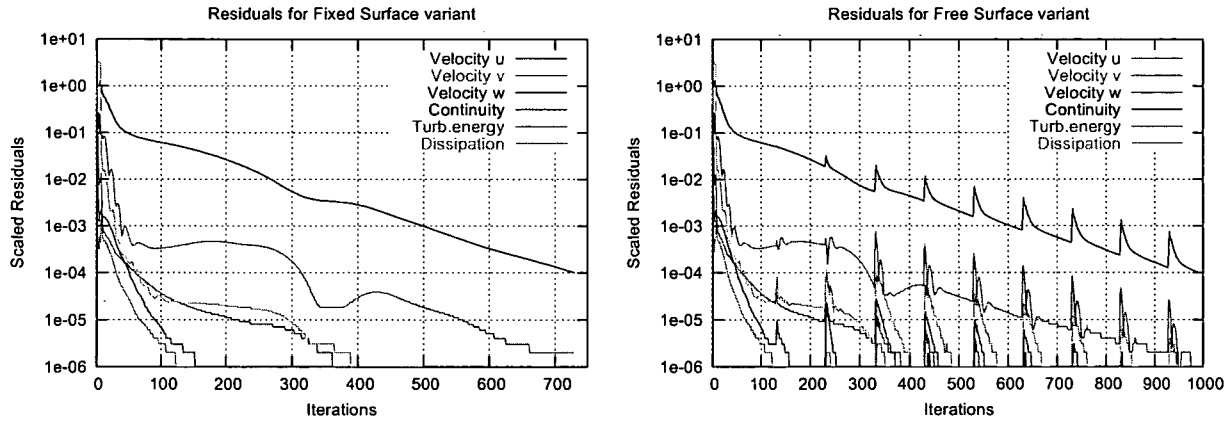

Figure 4.8: Convergence history for turbulent channel flow

Gauss-Seidel solver. But since the solution methods based on the Conjugate Gradient scheme (section 4.5.1) are both complicated to implement and suffer from serious robustness problems, only the so-called *Multigrid schemes* would be capable of delivering robustness and efficiency in removing global errors. The basic concept is to treat these low-frequency errors by a series of successively coarser grids whereas high-frequency errors are reduced by finer meshes. For the present work, however, this is not a feasible solution since the two-dimensional domain gridding process using polyhedral cells is a time-consuming process. Therefore, if a reduction of the vertical solution accuracy can be accepted, it is advisable to decrease the vertical resolution of the flow domain upon an increase of the horizontal resolution to allow for a converged solution to be obtained within an acceptable amount of time using the Gauss-Seidel solver.

Finally, the time until convergence is also affected by hardware characteristics of the computer system used. While CPU architecture and clock rate are straightforward parameters, the relation

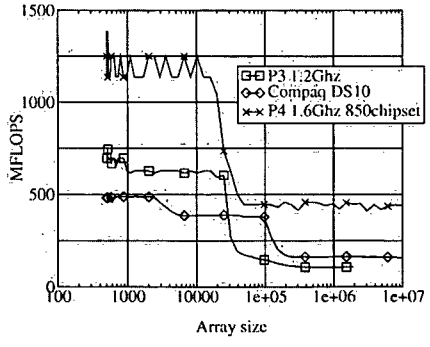

Figure 4.9: Performance for three different computer systems (*Armfield* (2003) [8])

between internal cache size of the CPU (level 2 cache) and number of floating point operations per second dependent on the problem size is not so easy to assess. *Armfield* (2003) [8] demonstrates this relation which is given in figure 4.9. It can be seen that the performance – given as million floating point operations per second (MFLOPS) – rapidly decreases once the problem size exceeds the size of the CPU's level 2 cache and direct RAM access operations become necessary.

5 Verification and Validation Cases

5.1 Verification

The purpose of the verification study is to assess the discretisation error, defined as the difference between the exact solution of the governing equations and the exact solution of the discrete approximation (*Ferziger* (2002) [20]). This error is due to the use of approximations for the various terms in the equations and the boundary conditions (*ibid.*). The quality of these approximations is described by the *order* of the numerical scheme. It is assessed by comparing the solutions obtained on different grids that are distinguished by their grid spacing.

In theory, the numerical scheme employed in this work should be of second order, hence the name *Second Order Upwind* scheme. Analysis to derive the actual order of the scheme is done by a study on three different grids that can be characterised by a unique grid spacing, with the grid point distance being halved from each grid level to the next one. For this study to be successful, it is important that the grid can be actually characterised by a unique length scale because the equation used to assess the order of the scheme (eq. 5.1) was derived using this precondition. Therefore the order assessment uses a hexahedral grid where grid spacing can be defined. Using such a grid with constant grid point distance h at the finest level, the order p of the implemented scheme is obtained by evaluating

$$p = \frac{1}{\ln 2} \ln \frac{\phi_{4h} - \phi_{2h}}{\phi_{2h} - \phi_{1h}} + o(1) \tag{5.1}$$

(*Steinrück* (2002) [78]), where ϕ is the quantity that the underlying governing equation was being derived for. Equation 5.1 is to be evaluated for a grid node that is located sufficiently far away from any boundary, hence allowing for an assessment with as little influence by boundary conditions as possible.

In the present study, a simple rectangular duct is being modelled: 50m in length, 5m in width, 1m in height, without bottom slope. All surrounding walls are assigned an equivalent sand roughness

k_s of 0.00034 m which corresponds to a Strickler coefficient of $k_{St} = 100$. A constant discharge of 5 m^3/s leads to a pressure gradient between inlet and outlet since the surface is not allowed to move.

Three different computation grids are employed:

- a coarse grid using 250 regions in 2D (size: 1.0 x 1.0m) and 11 vertical layers, resulting in 2750 cells,
- a medium sized grid using 1,000 regions in 2D (size: 0.5 x 0.5 m) and 11 vertical layers, resulting in 11,000 cells,
- a fine grid using 4,000 regions in 2D (size: 0.25 x 0.25 m) and 11 vertical layers, resulting in 44,000 cells.

The grid spacing in the vertical direction was kept constant for all three variants, with the top and the lowest cell each accounting for 5% of the computational domain and all other cells for 10%. It can be expected that this fact did not have a significant influence on the result because the flow conditions are essentially one-dimensional.

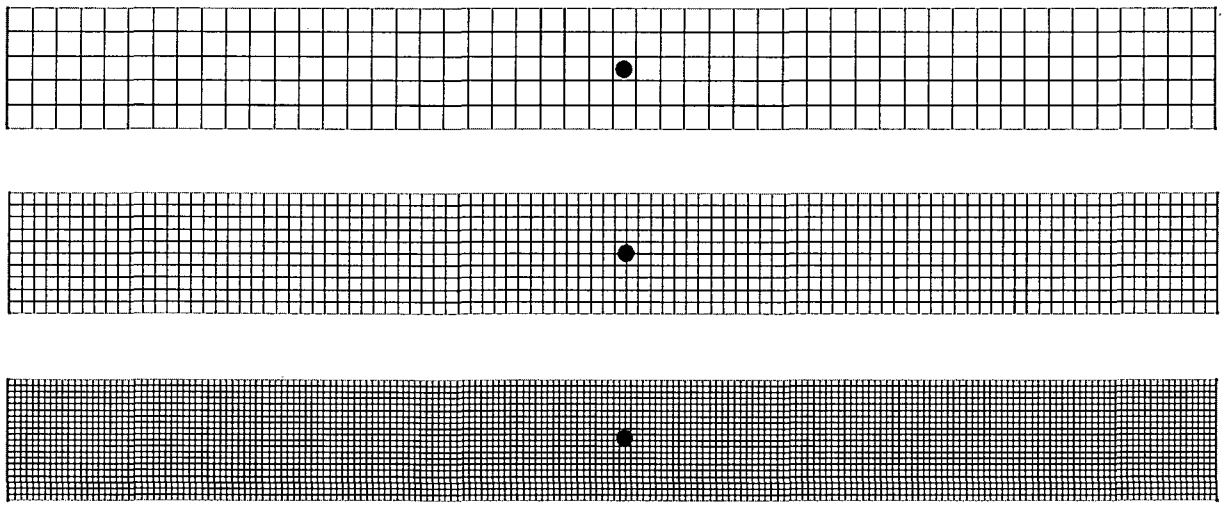

Figure 5.1: Grid levels for verification case: 250 regions (top), 1,000 regions (center), 4,000 regions (bottom), reference location marked in red

To minimise the influence of boundary conditions, a point close to the centre of the domain was selected as reference location: 25.5m from the inlet, 2.5m from the side walls, 0.5m from the bottom. Figure 5.1 depicts the three different grid levels and the reference location.

Since flow along a straight channel is a one-dimensional problem, the quantity to be evaluated at the reference location is u_1, the velocity in the direction of the global coordinate axis x. Its values are given in table 5.1 in high precision for an accurate assessment of the discretisation error and subsequently the order of the numerical scheme.

Case	N regions	Velocity u_1 [m/s]
4h	250	1.15930315
2h	1000	1.154079538
1h	4000	1.152698668

Table 5.1: Velocity u_1 at the reference location for different grid levels

Evaluating equation 5.1 using the values from table 5.1 yields the order of the implemented numerical scheme. It takes the value

$$p = 1.92 \tag{5.2}$$

which is very close to 2.0, the perfect result for a scheme of second order. The slight difference can be explained by the influence of boundary conditions, the missing refinement in the vertical direction and effects with similar impact that cannot be quantified exactly.

Case	N Cells	Iterations	Time [h]
4h	2750	8100	0:35
2h	11000	5900	2:05
1h	44000	5100	7:20

Table 5.2: Number of iterations and computation time for the verification case

In addition, the convergence behaviour of the verification case on the different grid levels is of special interest from an engineer's point of view. Table 5.2 gives the number of iterations needed for the solution to converge on the different grids along with the computation time on a reference computer system. The reference system is a Pentium IV-class machine with 2.8 GHz clock rate and 1 GB RAM installed in a dual-channel setup.

Most notably the number of iterations decreases when the problem size increases. This can be easily explained by the better spatial resolution of the computational domain which causes fewer errors in the solution, hence fewer iterations are required for convergence. Furthermore the actual time spent on the solution of the problem scales sub-linearly with the problem size, i.e. computations on coarse grids converge relatively slower than on refined grids: during every iteration, additional computations must be performed (e.g. evaluation of gradients or the eddy

viscosity for all cells), hence an increase in the number of iterations has an additional adverse effect on the time the model takes to reach a converged state.

Finally, the convergence history for the verification case on the three different grid levels is depicted in figure 5.2. From this figure it is clear that the decrease in the continuity residual is linear in a logarithmic scale. Furthermore, it is possible to see that the equations of the turbulence model take more time to converge than the momentum equations; according to the author's experience this can consistently be seen in a large number of flow problems.

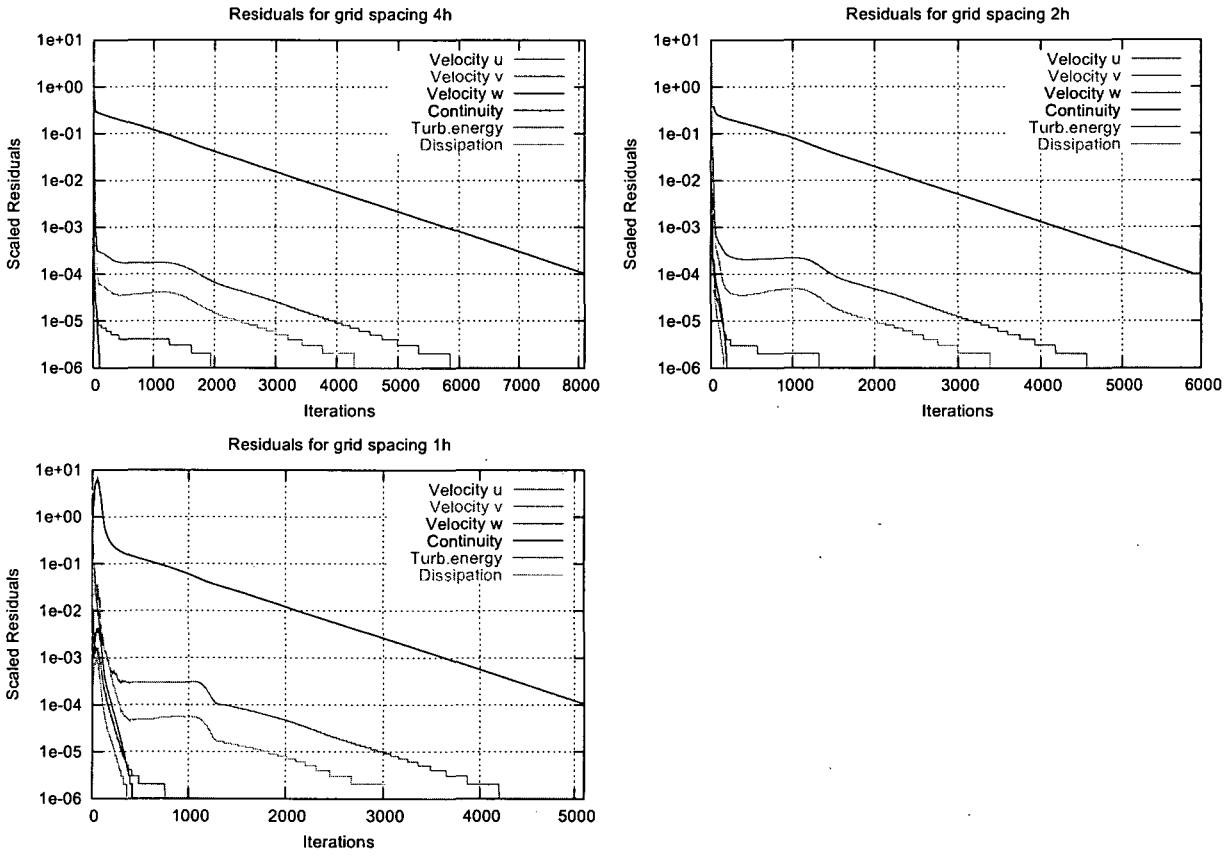

Figure 5.2: Convergence history for verification case

5.2 Developing Flow in a Curved Rectangular Duct

In this section, the numerical model will be validated against measurements in a curved rectangular duct. The experiment is described and selected results are published in *Kim & Patel* (1994) [37]. The full data sets of the results are available on the Classic Data Base of the European Research Community on Flow, Turbulence and Combustion (*ERCOFTAC* (1995) [19]). The measurements of Kim and Patel's experiment were already used to assess the validity of computational codes in the past (e.g. *Nguyen* (2000) [51]).

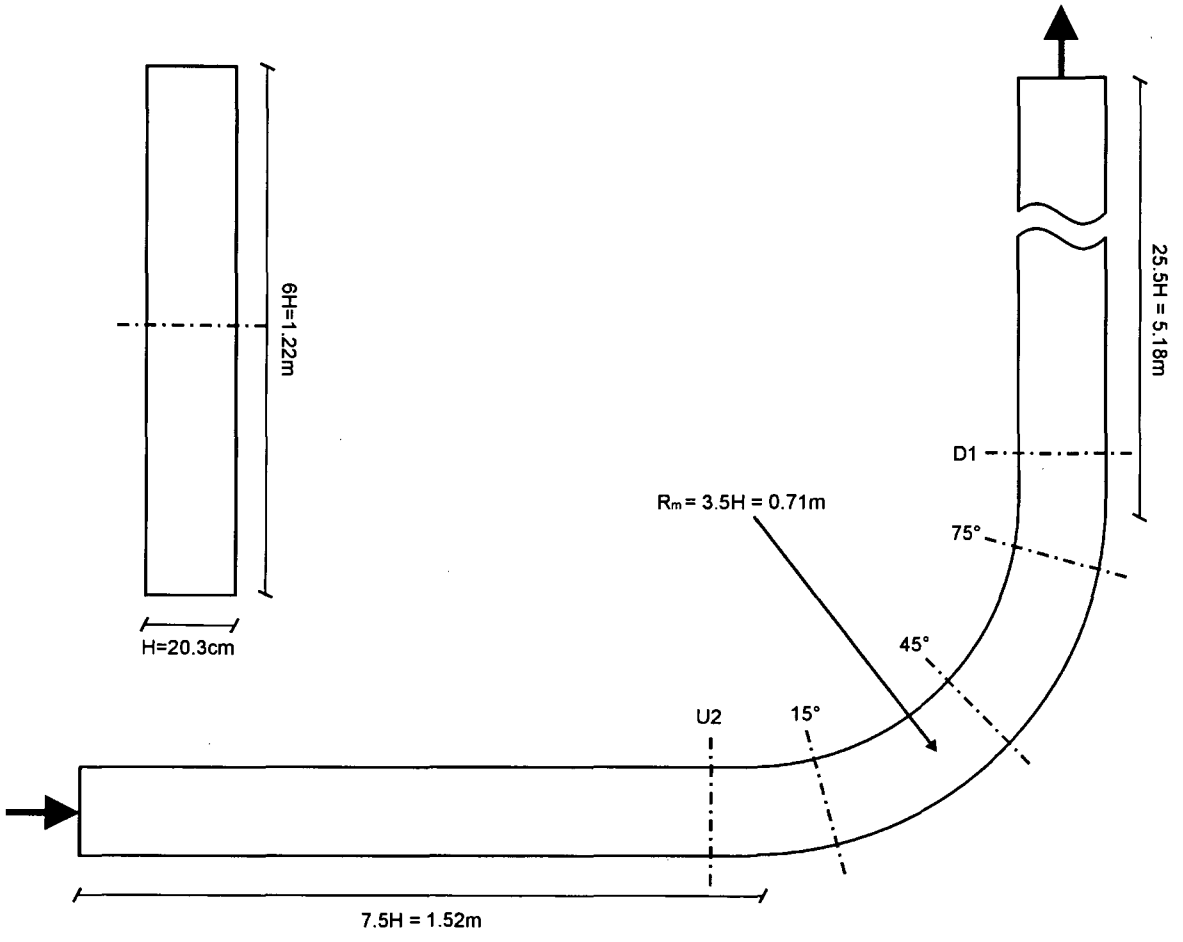

Figure 5.3: Layout of Kim & Patel's experiment

The experimental setup is depicted in figure 5.3. The physical model consists of a rectangular duct H=20.3cm wide and $6H$=1.22m high. It features an upstream section of $7.5H$=1.52m in length, followed by a 90° bend with a mean radius of curvature being $3.5H$=71.1cm, and a long downstream section of $25.5H$=5.18m. The duct is run as a wind-tunnel, using air with a

kinematic viscosity of $\nu = 1.45 \cdot 10^{-5} m^2/s$ and a density of approximately $\rho = 1.25 kg/m^3$. The freestream velocity near the middle of the upstream section was found to be $U_0 = 16 m/s$, resulting in a duct-based Reynolds number of $U_0 H/\nu = 224,000$.

As a measurement of wall roughness, a friction coefficient $C_f = 0.0038$ is provided. In a first step, this friction coefficient must be converted to the concept of equivalent sand roughness to be of use in the present work. Starting at the definition of the friction coefficient,

$$C_f = \frac{\tau_w}{0.5 \rho U^2} \tag{5.3}$$

where τ_w denotes the wall shear stress and U is the velocity outside of the boundary layer, we can evaluate $\tau_w = 0.61 N/m^2$. Introducing the shear velocity u^* as defined by equation 4.89, we obtain $u^* = 0.7 m/s$. Now evaluating equation 4.91 for the wall-distance where the freestream velocity is to be expected (i.e. the duct half width $H/2$), a fictitious sand roughness of $k_s = 2 \cdot 10^{-5}$ is obtained that can be used in the present study. Of course, one must be fully aware that the concept of an equivalent sand roughness is actually not applicable to air flows, which may lead to errors in the results due to a wrong assessment of the wall's influence on the mean flow.

Measurements are provided at cross-sections U2, 15°, 45°, 75° and D1. In this chapter an analysis of cross-section 45° is presented; results for all other cross-sections can be found in appendix B. Except for the contour plots presented, all evaluated quantities are non-dimensionalised by the freestream velocity U_0 and the duct width H.

In *Kim & Patel* (1994) [37] it is recommended to use velocity and turbulence measurements at the upstream cross-section as inlet conditions for numerical studies and do the computations for a reduced model only. However, since the numerical model should actually be able to return these values if the full experimental domain is represented in the computational model and all boundary conditions are applied correctly, it was decided to do the computations on the full domain. With this assumption, a freestream velocity of 16.4m/s was obtained at the reference location – which is not far from the freestream velocity given in *Kim & Patel* (1994) [37] – and so the non-dimensionalisation of the computational results was performed using this value.

The measurements and also previous studies indicate that strong gradients in many flow properties are to be expected along the bend. Since it is of utmost importance to capture these gradients correctly, the computational grid must be refined in the vicinity of the side walls. On the other hand, a fine discretisation in the flow direction is not needed. This leads to the spatial discretisa-

tion using a grid based on quadrilateral regions since cells having a larger number of sides would become seriously distorted under these circumstances. A detail of the computational grid along the bend is depicted in figure 5.4. It can be seen that an area occupying 20% of the flow domain both near the inner and outer walls is discretised by five regions while the centre area is allocated the same amount of regions. A small number of cells near the entrance and the exit of the bend are slightly distorted due to the grid generation algorithm, but the number is too small to expect a negative influence on the results. In the vertical direction, the channel is symmetric and so a symmetry boundary was introduced at $3H$, allowing us to represent only half of the experimental domain in the numerical model. This domain is divided into eleven cells (see figure 5.4, inset) with the top and bottom cells each occupying 5% of the height and all other cells using 10%. This results in a total of 37,906 cells. The solution converged after 8,100 iterations in 10 hours on the reference system (see section 5.1).

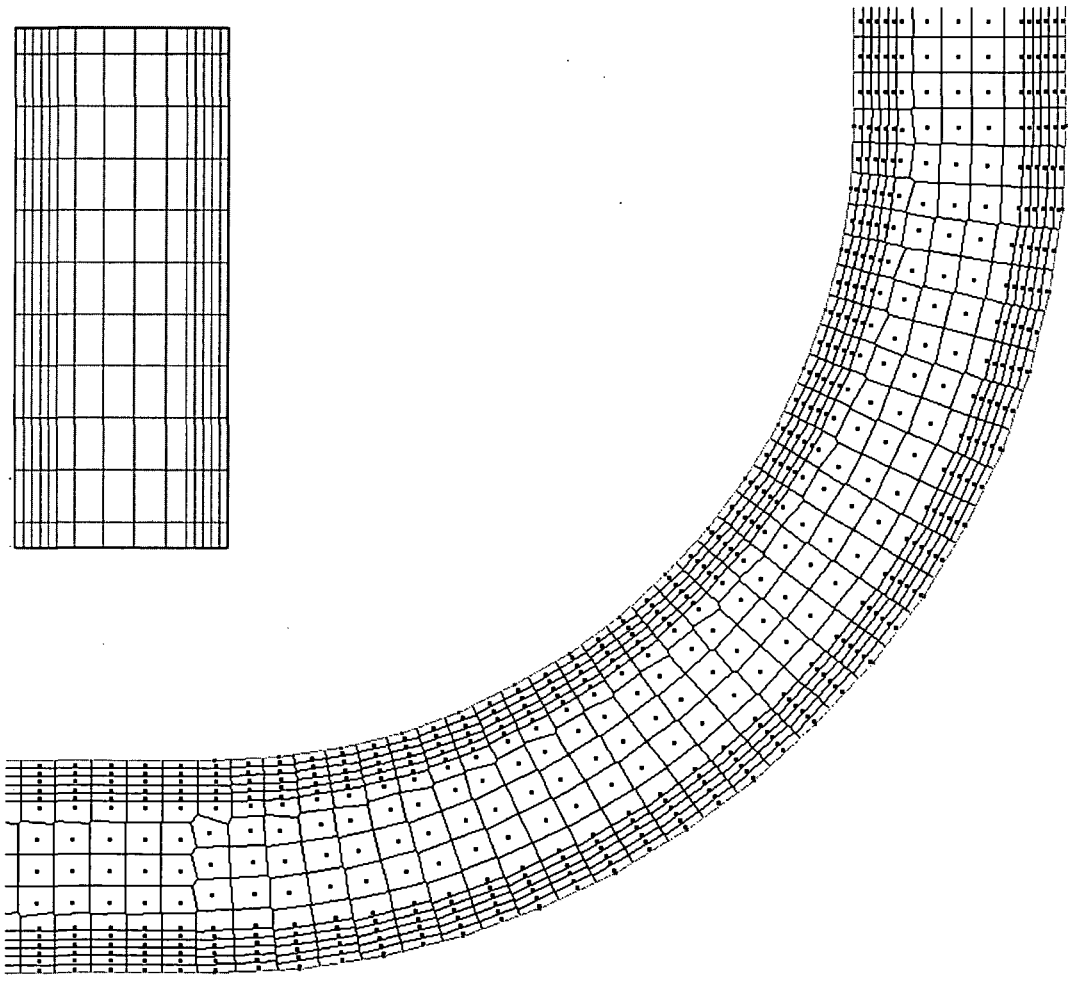

Figure 5.4: Detail of computation grid for Kim & Patel's experiment

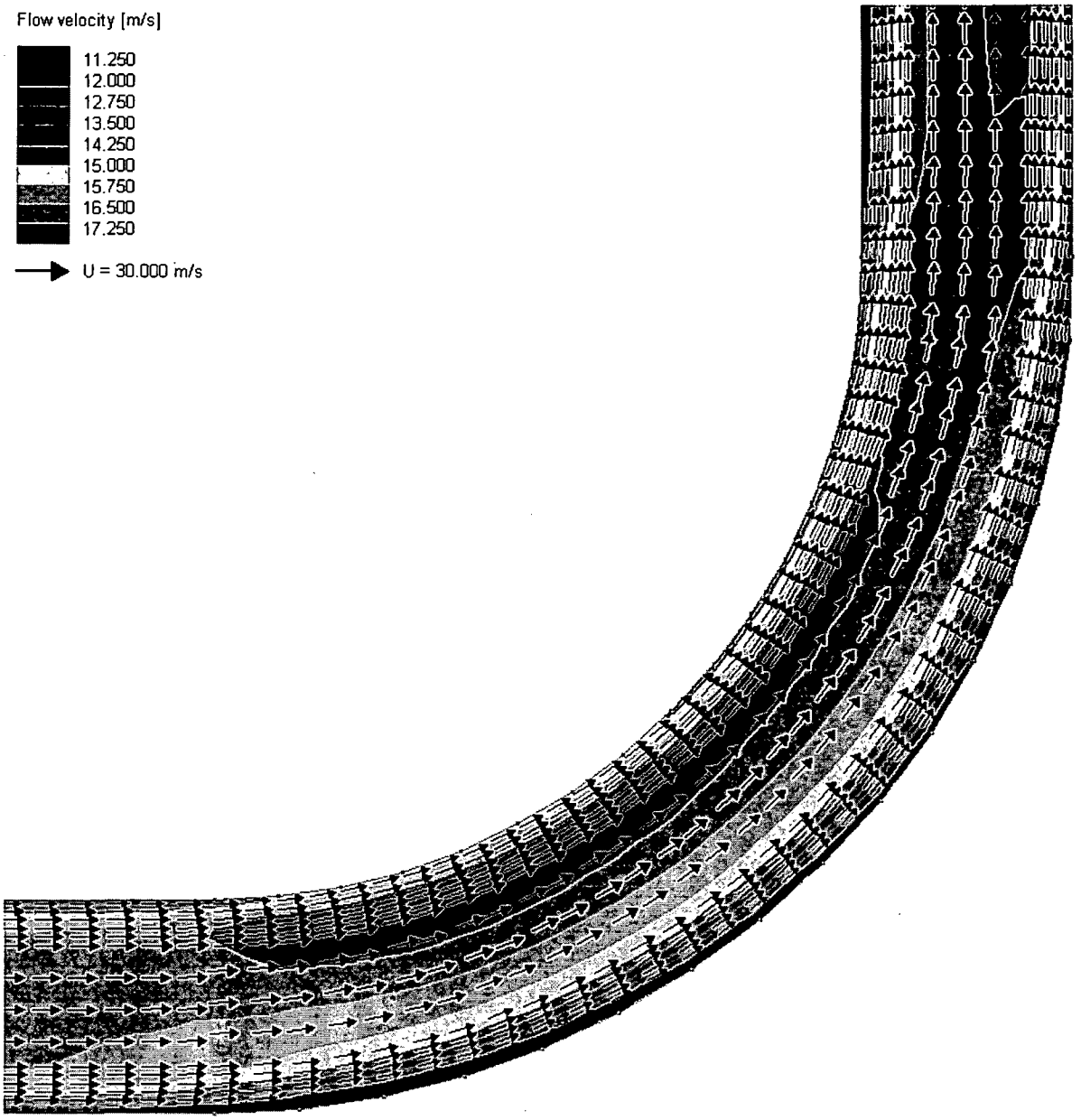

Figure 5.5: Computed depth-averaged flow velocity for Kim & Patel's experiment

Figure 5.5 depicts the computed depth-averaged flow velocity along the bend. This and subsequent images were produced using the algorithm described in *Bourke* (1987) [11]. It can be seen that minima occur close to the outer wall along the bend and at the inner wall in the downstream section of the duct. These minima take values of approximately 60% of the maxima observed in the corresponding cross-section. An assessment of these results can be done when the computed results are compared with the measurements by Kim and Patel.

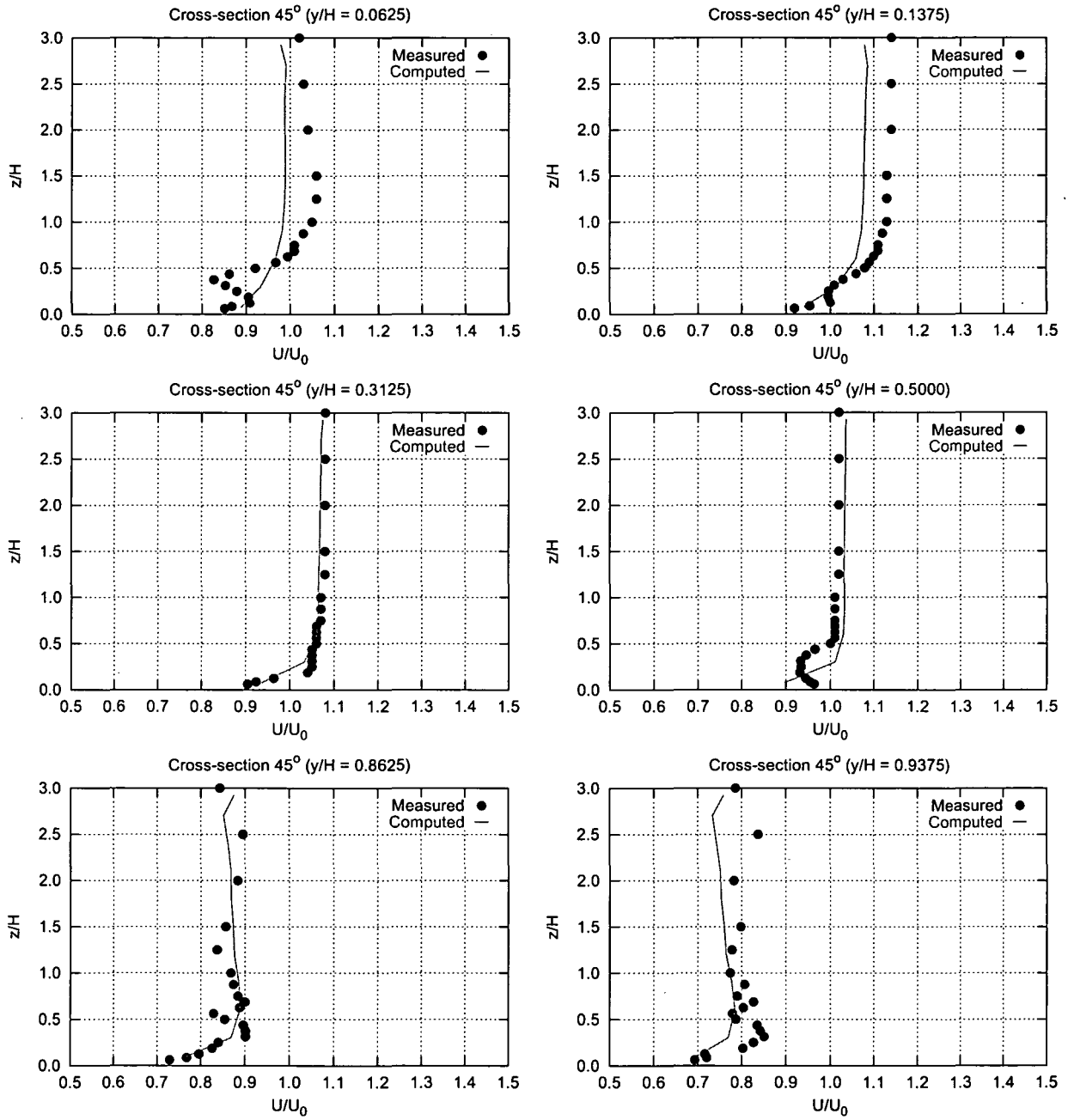

Figure 5.6: Longitudinal velocity profiles for cross section 45° of Kim & Patel's experiment

This is shown in figure 5.6 for the longitudinal velocity profiles. One can see a good agreement between computation and measurements in the region close to the center line and still a reasonable agreement in the outer regions. The vortex in the vicinity of the bottom wall, leading to distorted shapes of the velocity distributions in that area, is not captured by the model.

Sotiropoulos & Patel (1995) [75] credit this effect to the weak secondary motion predicted by the $k - \varepsilon$ turbulence closure, so that no longitudinal vortex forms.

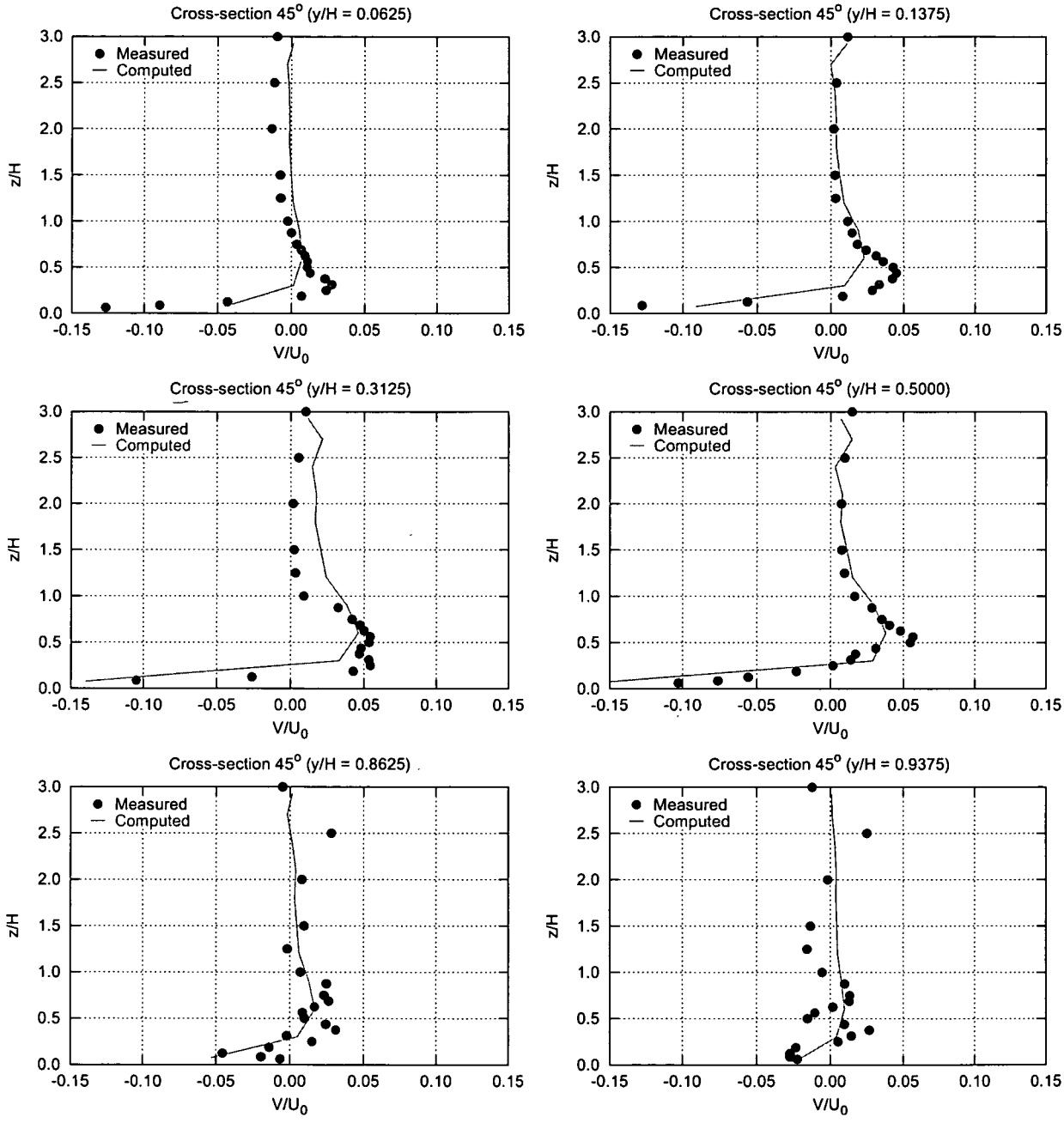

Figure 5.7: Transversal velocity profiles for cross section 45° of Kim & Patel's experiment

Another important flow property in curved channels or ducts is the secondary movement that can be seen in the transversal velocities as depicted in figure 5.7. The measurements exhibit strong minima close to the bottom wall that take values of up to $-0.15U_0$ while maxima of about $0.05U_0$ occur at $z/H = 0.5$. At the outer wall the flow pattern becomes quite complex as several vortices are evolving. It can be seen that the model captures the overall distribution of secondary velocity correctly, but fails to predict the exact vortex pattern close to the outer wall. This effect can be credited to deficiencies in the $k - \varepsilon$ turbulence closure (*Sotiropoulos & Patel* (1995) [75]), the second order upwind scheme employed in the present work, and the vertical resolution of the model.

While the minima of the velocity profile are correctly represented – at least in the inner regions of the duct – the maxima are underestimated to some extent. This has also been found and documented by *Sotiropoulos & Patel* (1992) [74] who conclude that the $k-\varepsilon$ model underpredicts the strength of the secondary motion.

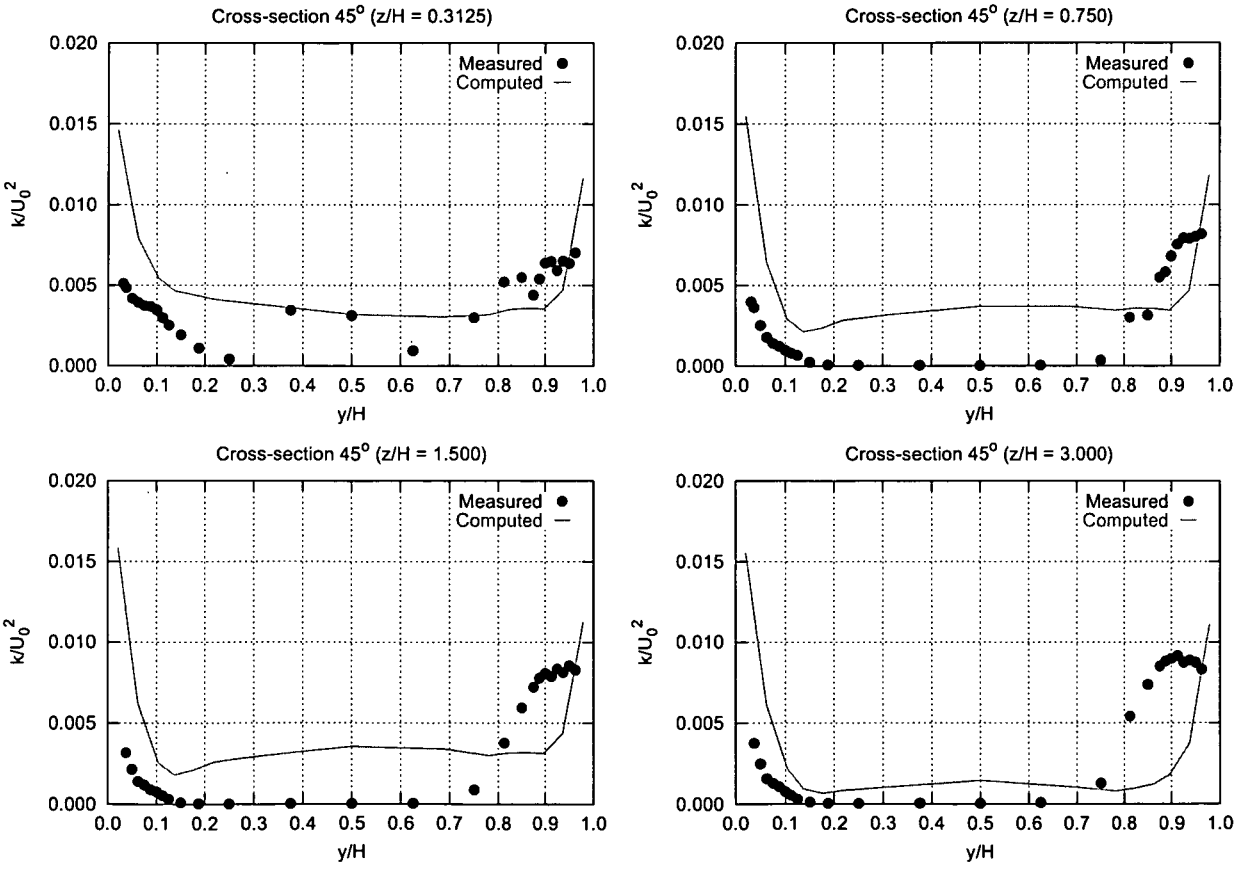

Figure 5.8: Distribution of non-dimensionalised turbulent kinetic energy for cross section 45° of Kim & Patel's experiment

Figure 5.8 finally depicts the distribution of the non-dimensionalised turbulent kinetic energy in the 45° cross-section in four distinct vertical layers. The measurements exhibit maxima at the side walls and minimal values in the inner flow region. The computational results also follow this pattern but vastly overpredict the turbulent kinetic energy almost everywhere in the flow domain – a feature of the $k-\varepsilon$ model that has also been found by other authors in the past. Furthermore, the distribution along the outer wall is not very well represented in the model. It should be mentioned though that the model correctly captures three distinct local maxima at the layer $z/H = 0.3125$ so it can be concluded that the model is at least able to represent certain characteristics of the turbulent flow pattern. However, the overall agreement between model results and measurements is not too good; a fact which does not matter for the engineer as long as he is mostly interested in flow patterns and water surface elevations but that should be considered when the distribution of turbulent kinetic energy is of interest.

5.3 Flow in a Sharply Curved Channel

After the model itself has been validated to comply with different measurements in section 5.2, in this section the influence of different grid types on the results is investigated. For this purpose a validation experiment was chosen that has been used extensively in the past in the course of the development of numerical models (e.g. *Leschziner & Rodi* (1979) [42], *Ammer* (1993) [4], *Lien et al.* (1999) [43], *Nguyen* (2000) [51], *Wu et al.* (2000) [92], *Ghamry & Steffler* (2002) [29]): *Rozovskii* (1961) [69] investigated the flow characteristics of a sharp 180° bend with a ratio of width to mean radius of curvature of 1.0. A curve with a width-to-mean radius ratio of 0.4 and more is considered to be sharp, as pointed out by *Lien et al.* (1999) [43], hence resulting in a highly three-dimensional flow situation.

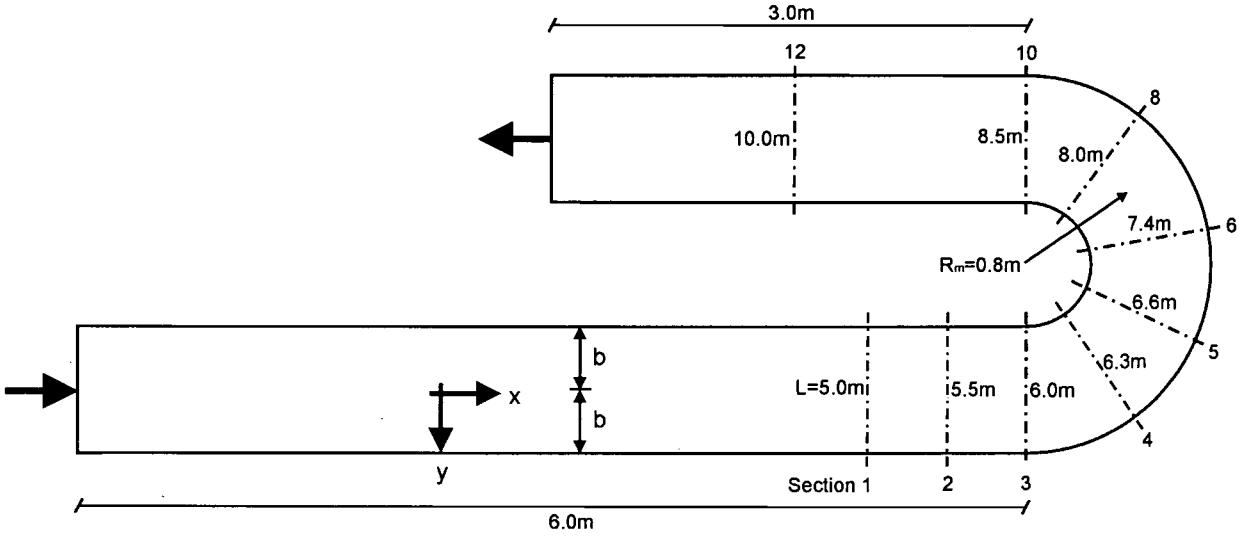

Figure 5.9: Layout of Rozovskii's experiment

The experimental setup is depicted in figure 5.9. An approach channel of 6m in length is followed by a 180° bend with a mean radius of 0.8m and an exit channel of 3m. The channel is horizontal and has a rectangular cross section with a width of 0.8m. The discharge in the channel is constant at 0.0123 m^3/s and the water depth at the *inlet* was documented to be 0.063m by *Rozovskii* (1961) [69], resulting in an average velocity of $UM = 0.265m/s$. As a measure of wall roughness the experimenter gives a Chezy coefficient of $C = 60m^{\frac{1}{2}}/s$. The Chezy and Strickler coefficients are related by the equation

$$k_{St} = \frac{C}{R_h^{\frac{1}{6}}} \tag{5.4}$$

where R_h denotes the hydraulic radius. Using this formula, the Strickler coefficient is found to

take a value of $k_{St} = 98m^{\frac{1}{3}}/s$ which in turn corresponds to a roughness height of $k_s = 0.0004m$. The *downstream* water depth required for the numerical model was introduced as a calibration parameter, with the final result after a number of runs yielding 0.053m. This value perfectly agrees with *Ghamry*'s (2002) [29] result.

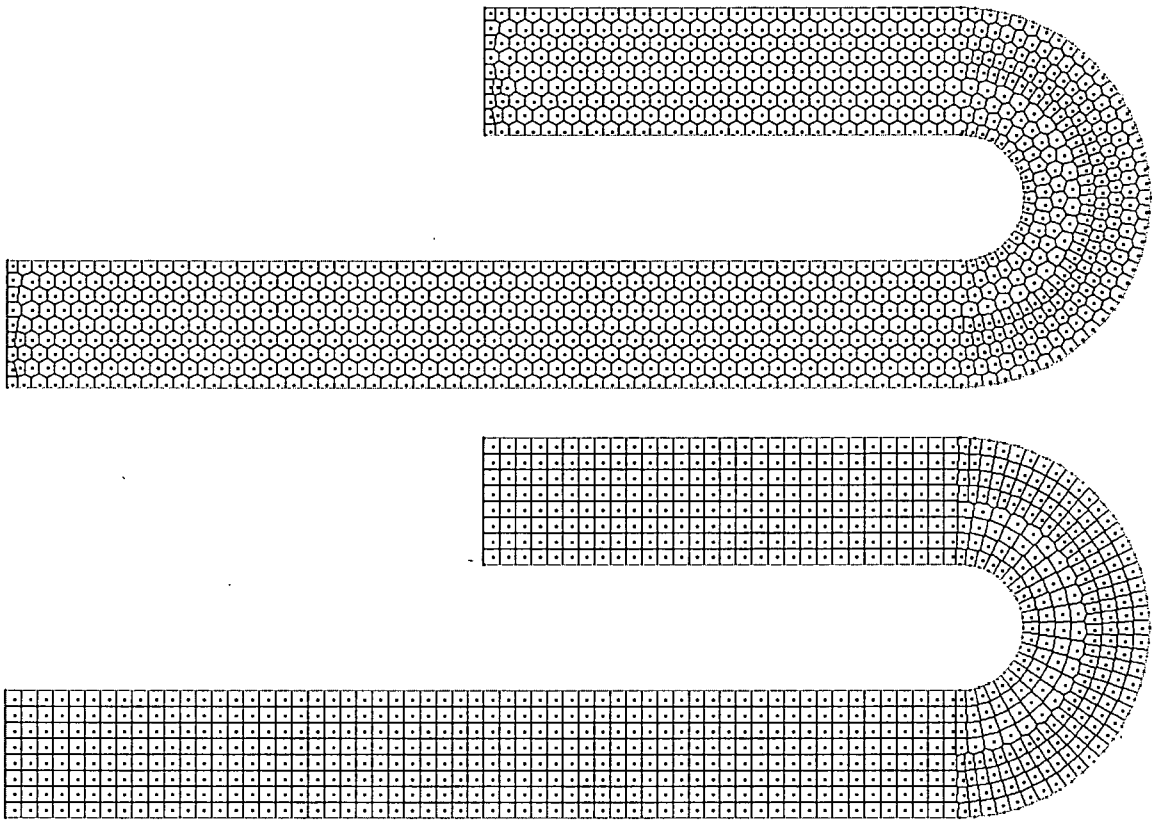

Figure 5.10: Computation grids for Rozovskii's experiment

In terms of spatial discretisation two different grids were employed to investigate the influence of the cell shape on the results:

- A first grid (see fig. 5.10, top) based on regions of hexagonal shape with a longitudinal grid point distance of 0.1m and a transversal grid point distance of 0.0866m. The grid is vertically structured, with the top and bottom cells each occupying 5% of the flow depth while all other cells occupy 10%. This setup results in 12,089 computation cells.
- A second grid (see fig. 5.10, bottom) based on regions of quadrilateral shape with longitudinal and transversal grid distances being equal to 0.1m. The vertical structure is the same as in the first setup, resulting in 10,648 computation cells.

While the numerical experiment based on quadrilateral regions converged after approximately 41,000 iterations in 15 hours on the reference system (see section 5.1), the computations using hexagonal grid regions took significantly longer and converged after approximately 68,000 iterations in 37 hours. The results are depicted in figures 5.11 through 5.21.

Figure 5.11: Computed water surface elevations for Rozovskii's experiment (top: hexagonal grid regions; bottom: quadrilateral grid regions)

Figure 5.11 shows the computed water surface elevations for both grid types (top: hexagonal regions, bottom: quadrilateral regions). When the results are compared with figure 5.12, one can see a reasonably well qualitative agreement between computations and measurements, even though a quantitative comparison is not so straightforward since the experimental results are presented in a reference system which apparently is not based on the channel bed. The results between the two different grids are approximately equal, with the hexagonal grid having a tendency towards smoothing out extremal values along the outer bank and exhibiting higher extrema along the inner bank.

A more precise assessment of the results can be done if the water surface elevation along the

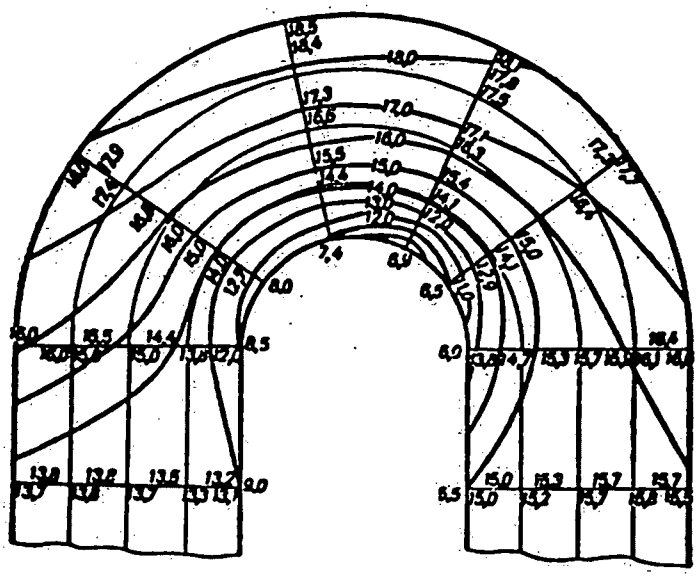

Figure 5.12: Measured water surface elevations by *Rozovskii* (1961) [69]

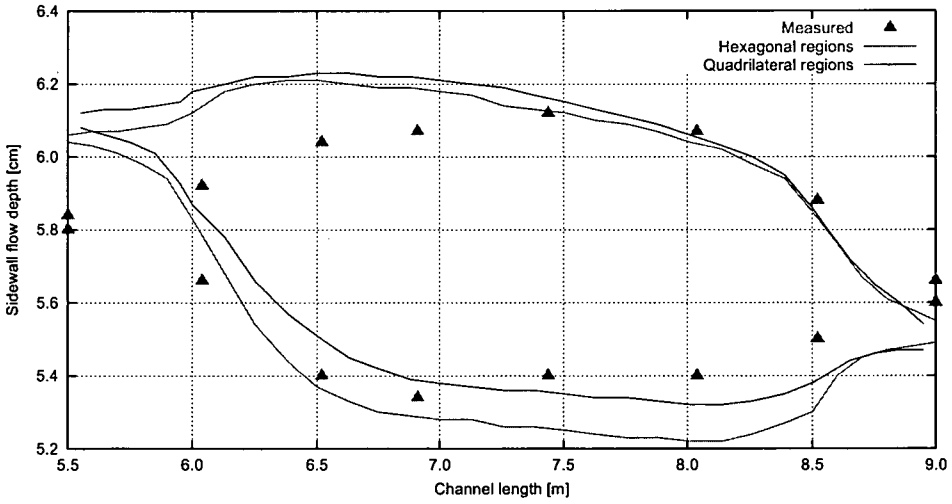

Figure 5.13: Sidewall flow depth for Rozovskii's experiment

channel walls is plotted for the bend as depicted in figure 5.13. The impression gained from the interpretation of the contour plots in fig. 5.11 is confirmed: the model using hexagonal regions shows a better agreement along the outer bank while it performs not so good along the inner bank. However, both grid types yield a relatively good agreement with the measurements. The differences between computations and measurements near start and end of the bend have been noticed by other authors, as well (e.g. *Lien et al.* (1999) [43]); they can be explained as results of the model calibration by making use of the inlet flow depth. Referring to these discrepancies,

Ammer (1993) [4] even suggests that the value given for the upstream water depth by *Rozovskii* (1961) [69] should be questioned in general.

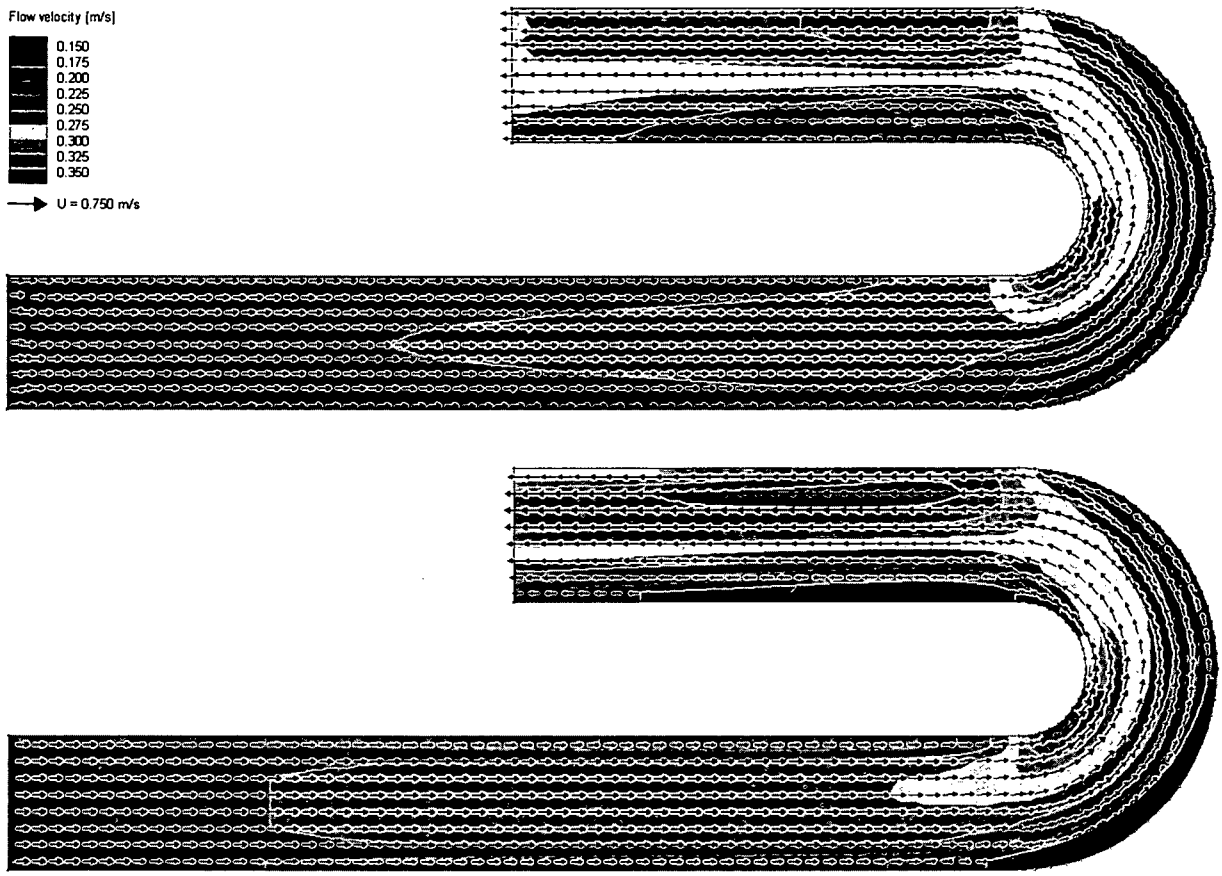

Figure 5.14: Computed depth-averaged flow velocity for Rozovskii's experiment (top: hexagonal grid regions; bottom: quadrilateral grid regions)

Figure 5.14 shows the depth-averaged flow velocities for the two grid setups. The grid based on quadrilateral regions exhibits stronger maxima and minima both along the bend and in the exit channel while these are mostly smoothed out in the hexagonal setup. Furthermore the decrease of flow velocity towards the side walls is stronger when quadrilateral regions are employed. In order to find out whether this smoothing of extremal values in the hexagonal grid is an effect known as *false diffusion* – predominantly to be seen when grid lines are not aligned with the main flow direction, resulting from one-dimensional interpolating in multi-dimensional domains – or whether it is actually a desired effect that increases the accuracy of the solution, the longitudinal velocity profiles in different cross sections are studied (figs. 5.15 and 5.16).

These figures depict the longitudinal velocity profiles for different points in a number of cross sections along the bend. The channel half-width (i.e. 0.4m) is denoted b, and so every point in

the cross-section can be assigned a value of y/b, with negative values ranging from the left bank to the centre line.

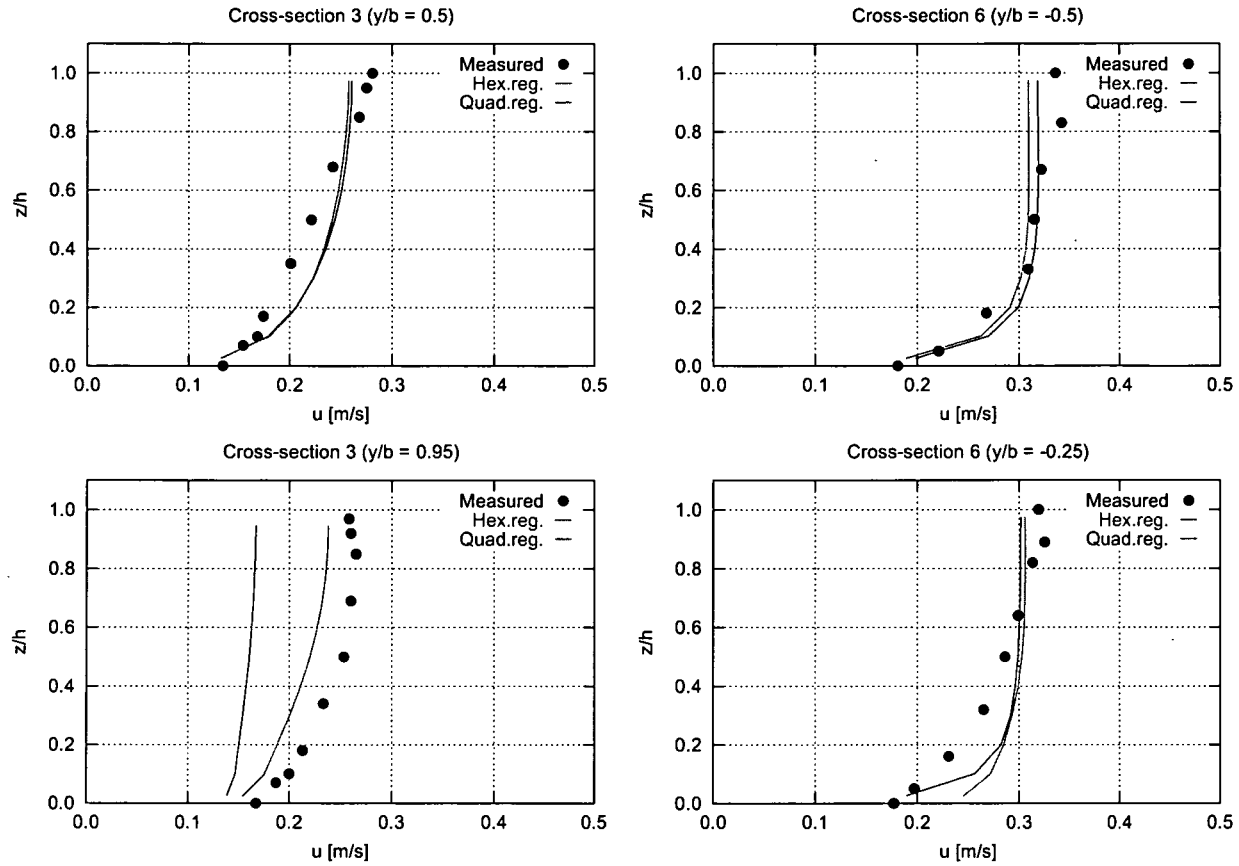

Figure 5.15: Selected longitudinal velocity profiles for cross sections 3 and 6 of Rozovskii's experiment

It can be seen that the agreement between measurements and computations is quite reasonable for both grids in the vicinity of the centre line, while the results notably deviate from the measurements in the region close to the banks. This effect can be credited to the lack of grid refinement in the bank regions which does not allow for the pressure gradients to be precisely captured, thus influencing the distribution of mass fluxes and velocities. It should also be noted that the curvature of the calculated velocity profile exhibits a different general tendency than the measurements. Considering the analogy to observations made in section 5.2 with regard to the longitudinal velocity profiles, it is very likely that this can be credited to the $k-\varepsilon$ model underestimating the strength of the secondary motion which in turn leads to deficiencies in the prediction of longitudinal velocities.

In general it was found that, except for the outer bank at cross-section 3 where the hexagonal grid

performs better, the quadrilateral grid returns results closer to the measurements. This allows the conclusion that actually a certain level of false or numerical diffusion is present when the hexagonal grid is being used, even though it is apparently not severe since the differences between the results on the two grids are generally not large.

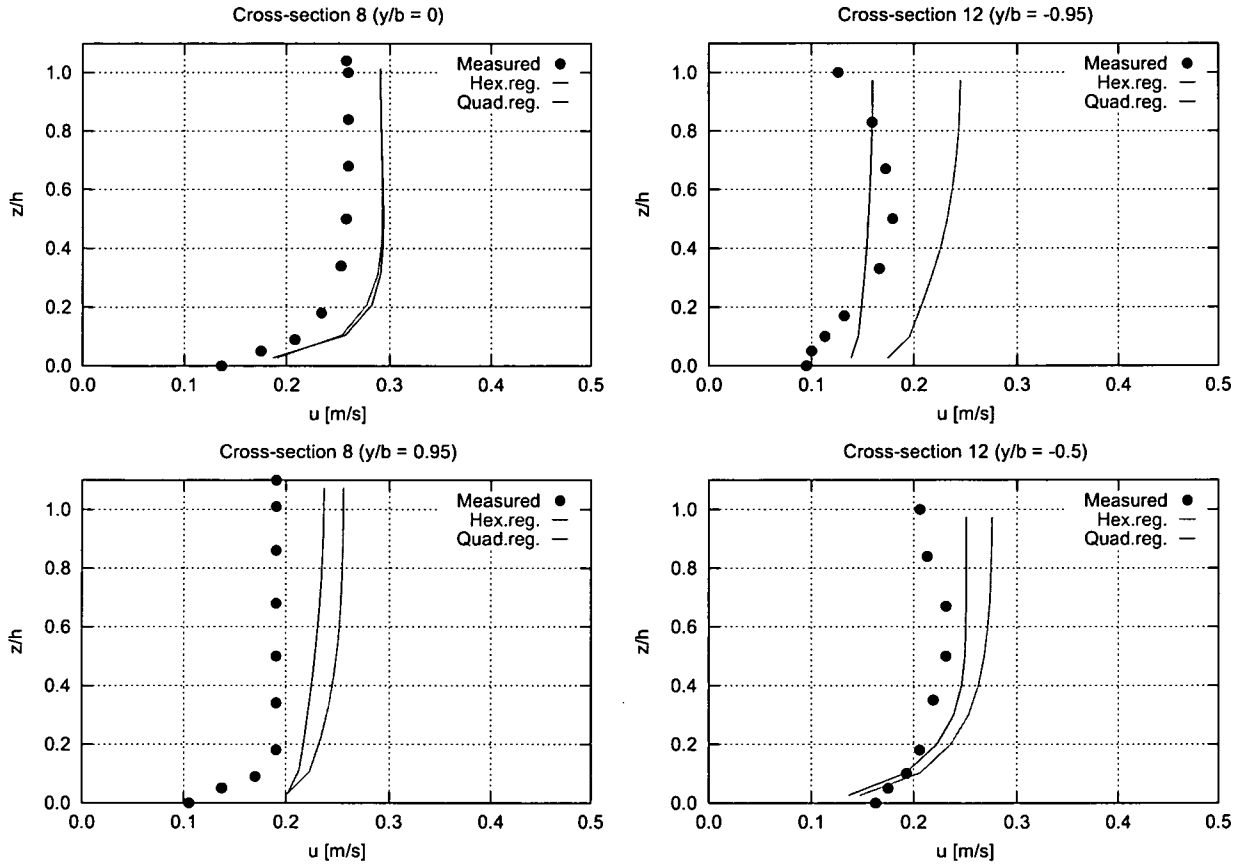

Figure 5.16: Selected longitudinal velocity profiles for cross sections 8 and 12 of Rozovskii's experiment

This impression is confirmed when the averaged velocity ratio U/UM is plotted for the cross-sections along the bend (fig. 5.17). While the results obtained on both grid setups exhibit an excellent agreement with the measured values, the quadrilateral grid still performs slightly better. In the vicinity of the banks the deviation from the measurements is larger than close to the centre of the channel.

Figures 5.18 through 5.20 depict the model results for the transversal velocities in all major cross-sections, using the two different grid shape approaches. Figure 5.21 finally compares the computed velocity profiles with the measurements published by Rozovskii.

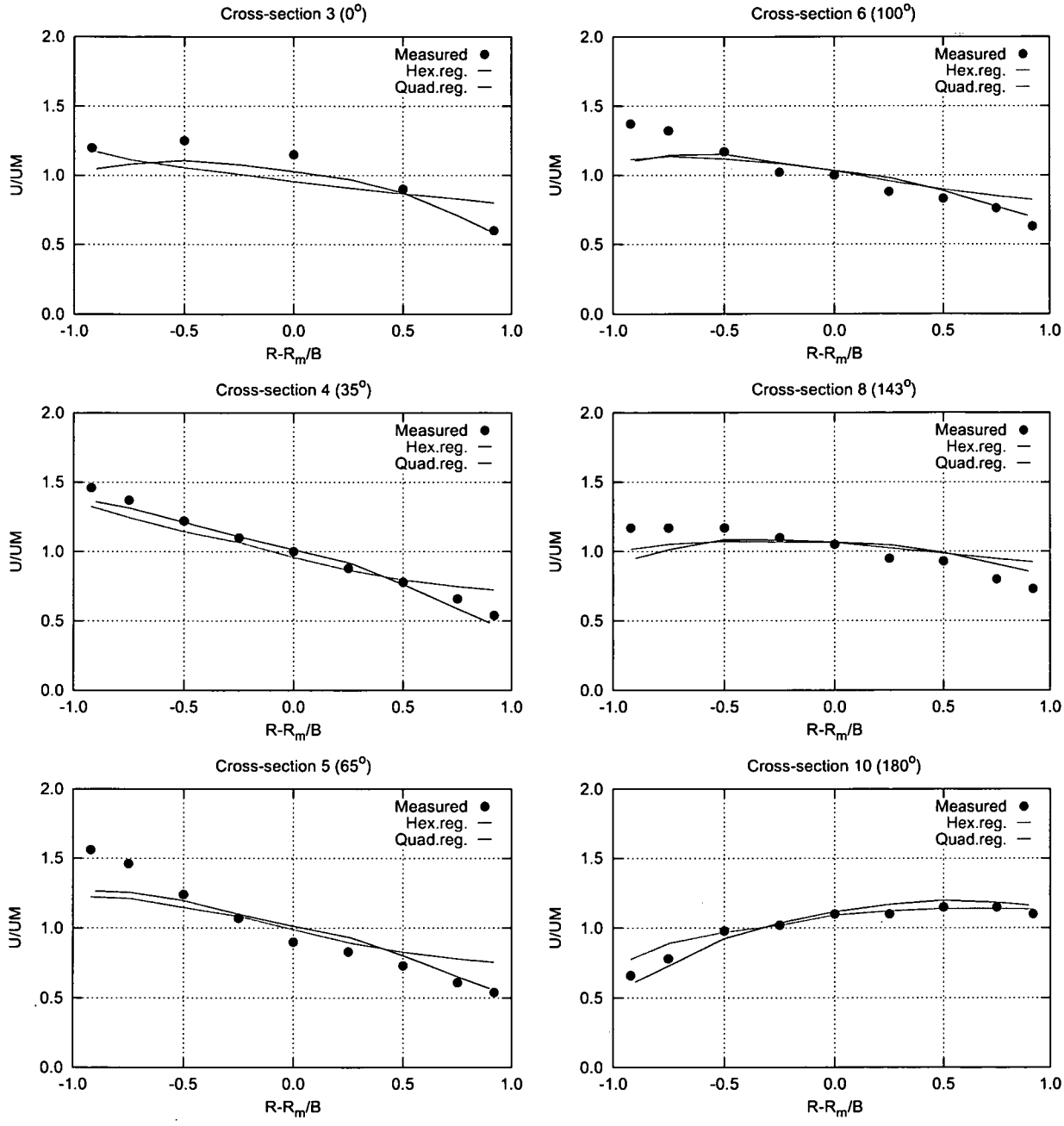

Figure 5.17: Averaged velocity ratio U/UM for Rozovskii's experiment

It is clearly visible that the flow exhibits a strong secondary motion throughout the bend. While the model predicts maxima of about 0.05m/s, measurements give values of up to 0.15m/s. Indeed the model correctly captures the direction of the motion, but underpredicts its strength – a fact that can be seen in other numerical codes as well (e.g. *Ghamry & Steffler* (2002) [29]). A closer

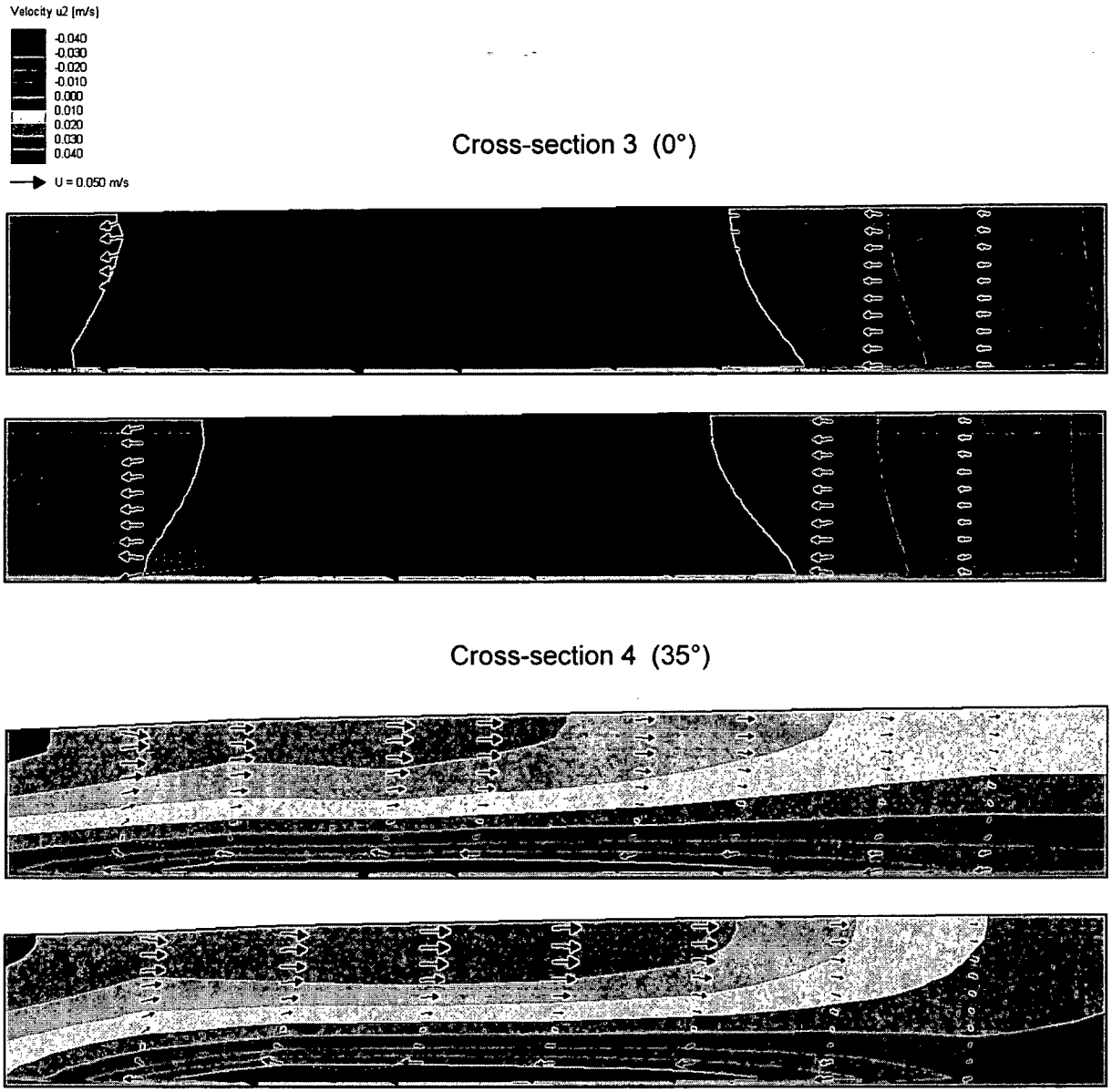

Figure 5.18: Contour plot of computed transversal velocity u_2 for cross-sections 3 and 4 of Rozovskii's experiment (top: hexagonal regions; bottom: quadrilateral regions) – figure scaled by the factor 2 in the vertical direction

look on the location where these extremal values occur reveals that they are found at the bed and close to the water surface. However, due to the model assumptions (i.e. zero velocity at the bottom, irrespective of flow direction) and also due to the vertical resolution of the model, it is virtually impossible to capture these maxima. Furthermore, it should be questioned whether extremal values right at the bed – as documented by *Rozovskii* (1961) [69] – are even physically

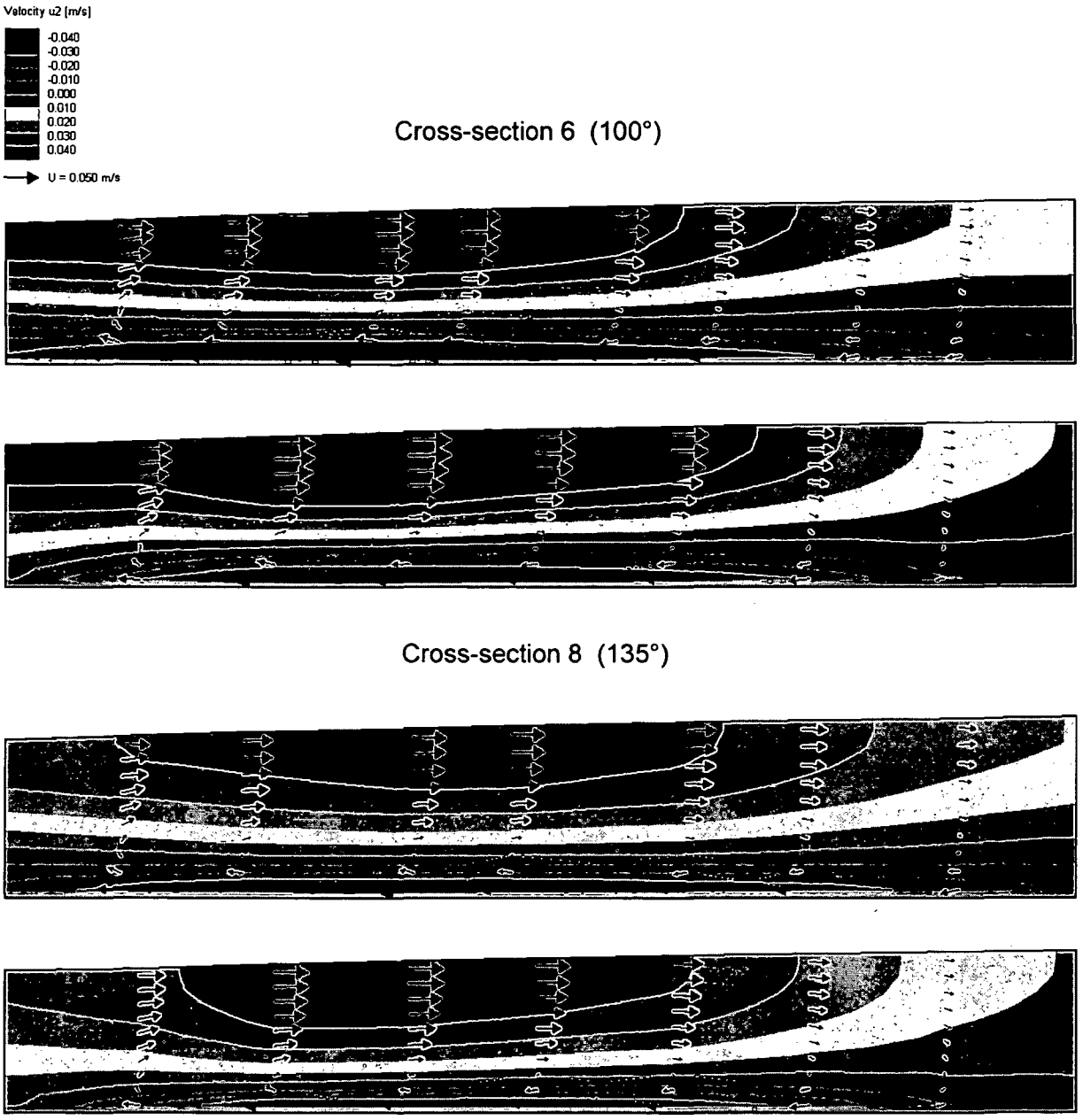

Figure 5.19: Contour plot of computed transversal velocity u_2 for cross-sections 6 and 8 of Rozovskii's experiment (top: hexagonal regions; bottom: quadrilateral regions) – figure scaled by the factor 2 in the vertical direction

possible. In contrast, section 5.4 discusses a flow situation where the maxima of the secondary motion were not measured at the bed but rather a significant distance above it – a behaviour which is correctly predicted by the model if a reasonable vertical resolution is employed.

The comparison between the results obtained on the two different grid shapes, as it is done in

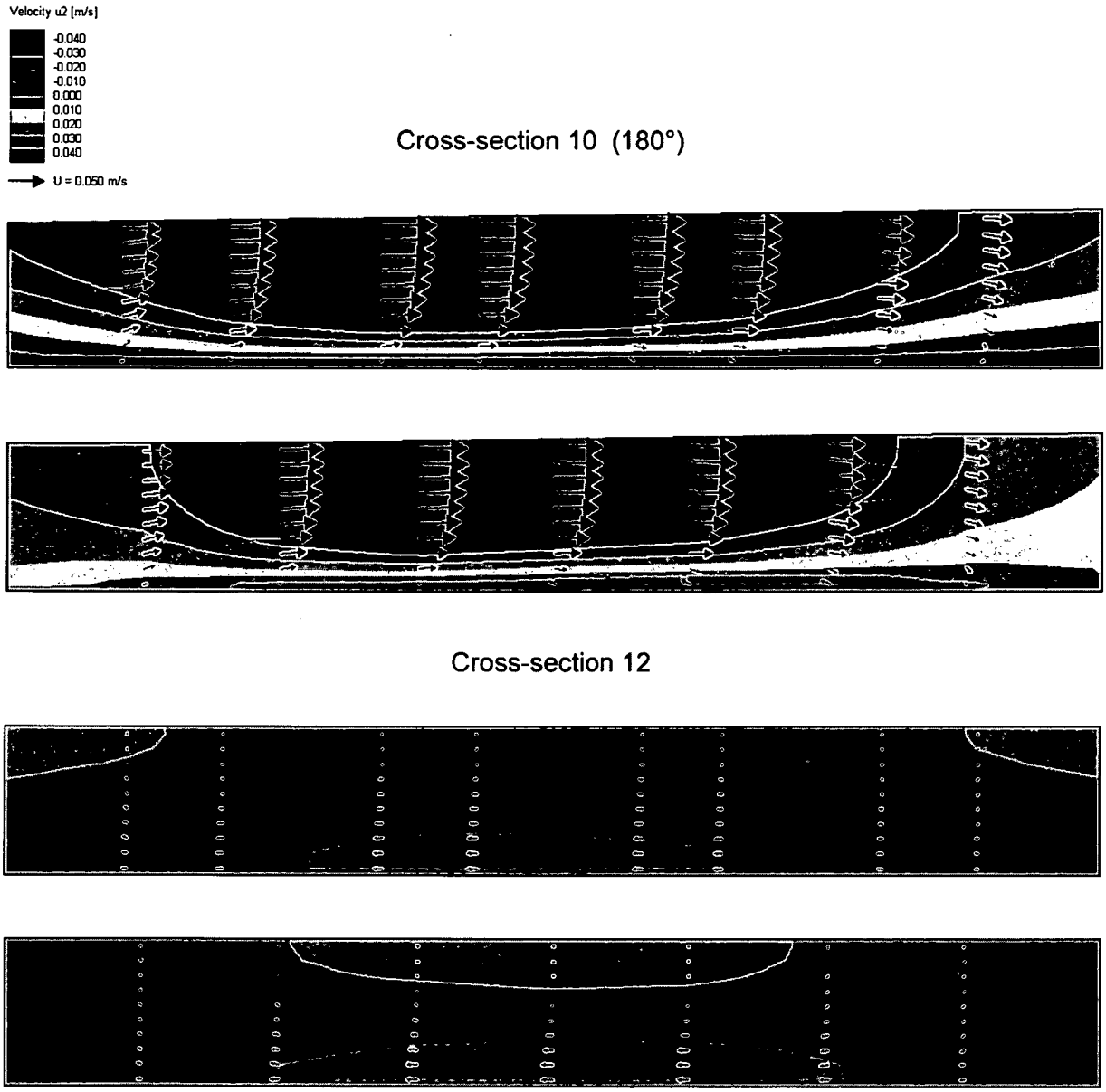

Figure 5.20: Contour plot of computed transversal velocity u_2 for cross-sections 10 and 12 of Rozovskii's experiment (top: hexagonal regions; bottom: quadrilateral regions) – figure scaled by the factor 2 in the vertical direction

figure 5.21, gives a very interesting result: throughout all locations evaluated, the transversal velocities obtained on the grid based on hexagonal regions are closer to the measurements than the solution on the quadrilateral grid regions. From this perception we can conclude that the hexagonal grid type actually yields a closer representation of mass fluxes in the transversal direction, which in turn leads to a better prediction of the secondary flow phenomena observed in

Rozovskii's experiment.

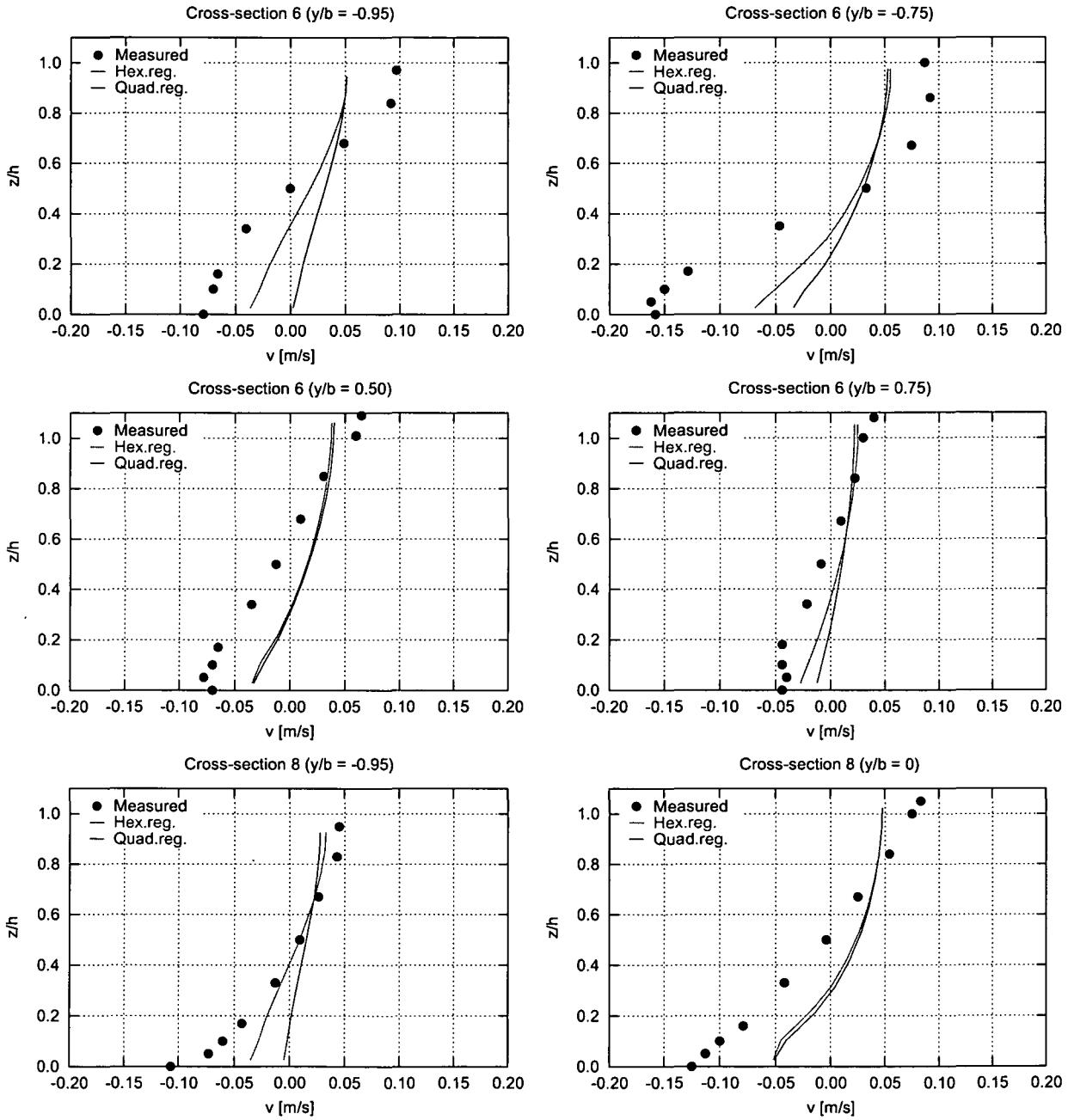

Figure 5.21: Selected transversal velocity profiles for Rozovskii's experiment

5.4 Flow in a 270° Bend with Moderate Curvature

In this section, a second numerical experiment will be conducted using two different grid shapes in order to assess the influence of the grid on the results obtained. The underlying physical experiment was done by Peter Steffler in 1984 and parts of the results were published in *Ghamry & Steffler* (2002) [29]. As can be seen in figure 5.22, the experimental channel has the shape of a 270° bend. Compared to Rozovskii's experiment, the curvature cannot be considered sharp since the width-to-mean radius ratio is around 0.3, hence it can be classified as moderate. Nonetheless the flow situation exhibits strong three-dimensional characteristics as will be shown later.

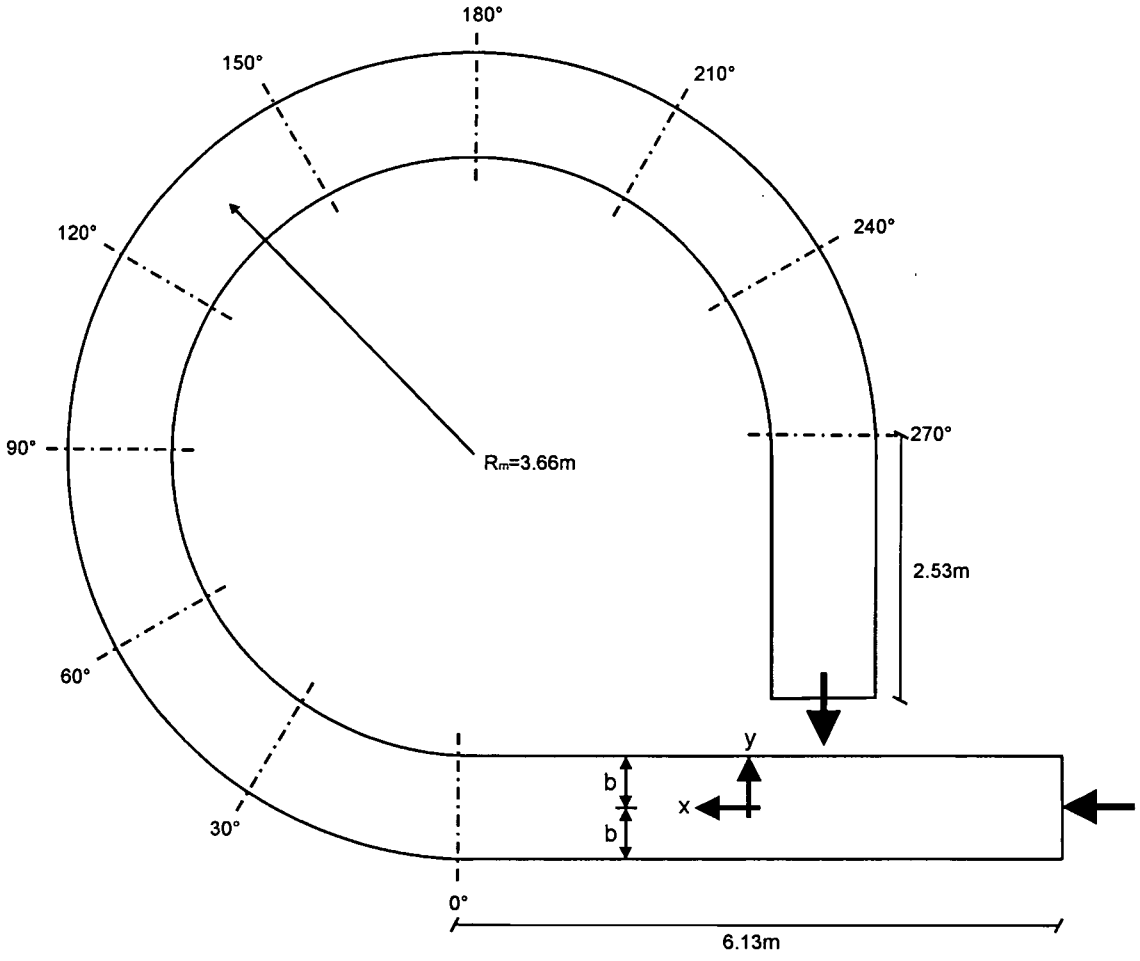

Figure 5.22: Layout of Steffler's experiment

The laboratory flume consists of an approach channel of 6.13m in length, followed by the 270° bend with a mean radius of 3.66m and an exit channel of 2.53m. The channel is 1.07m wide and features a bed slope of 0.00083. The total discharge is 0.0235 m^3/s; together with a specified wa-

ter depth of 0.061m at the outlet, an average flow velocity of $UM = 0.36m/s$ can be determined. *Ghamry & Steffler* (2002) [29] document an equivalent sand roughness value of $k_s = 0.0013m$ which is equivalent to a Strickler coefficient of $k_{St} = 80m^{\frac{1}{3}}/s$.

Once again, two different computation grids are being used to compare the results obtained on them:

- The first grid (fig. 5.23, top) is based on regions of hexagonal shape with a longitudinal grid point distance of 0.1m and a transversal distance of 0.089m. In accordance with previous computations, the grid is vertically structured: the top and bottom cells each occupy 5% of the flow depth while all other cells occupy 10%. This setup results in 34,639 computation cells.

- The second grid (fig. 5.23, bottom) uses quadrilateral grid regions with a longitudinal point distance of 0.1m and a transversal distance of 0.089m. The vertical structure is the same as in the first setup, resulting in a total of 34,584 computation cells.

The computation times until convergence of the results were significantly higher for Steffler's experiment compared with any other experiment investigated. On the reference system (see section 5.1), a solution for the grid based on hexagonal regions was obtained after 114,000 iterations in over 160 hours. The flow problem using quadrilateral grid regions reached equilibrium after 147,000 iterations in almost 200 hours – more than eight days. These extraordinary computation times, compared with Rozovskii's run, can be explained by the large number of computation cells which causes the solver to return results very slowly, which was discussed at the end of chapter 4. Furthermore both approach and exit channels point into negative directions of the global coordinate system. While this fact does not influence the result obtained, it slows down convergence because the numbering of cells is done according to the global coordinate system, and the solver module evaluates the equations according to the cell numbers. Hence, the reduction of errors in the solution is performed even more slowly if the numbering is exactly opposite to the flow direction.

The assessment of the solutions obtained on the different grids starts with the evaluation of the water depth as depicted in figure 5.24. No measurements of water levels are available, but the computations can be compared with each other. It can be seen that the overall shape of the water surface elevations is very similar, but the hexagonal grid setup results in a difference of about 1mm at the inlet. However, this difference, compared to the flow depth, is only around 1.5%,

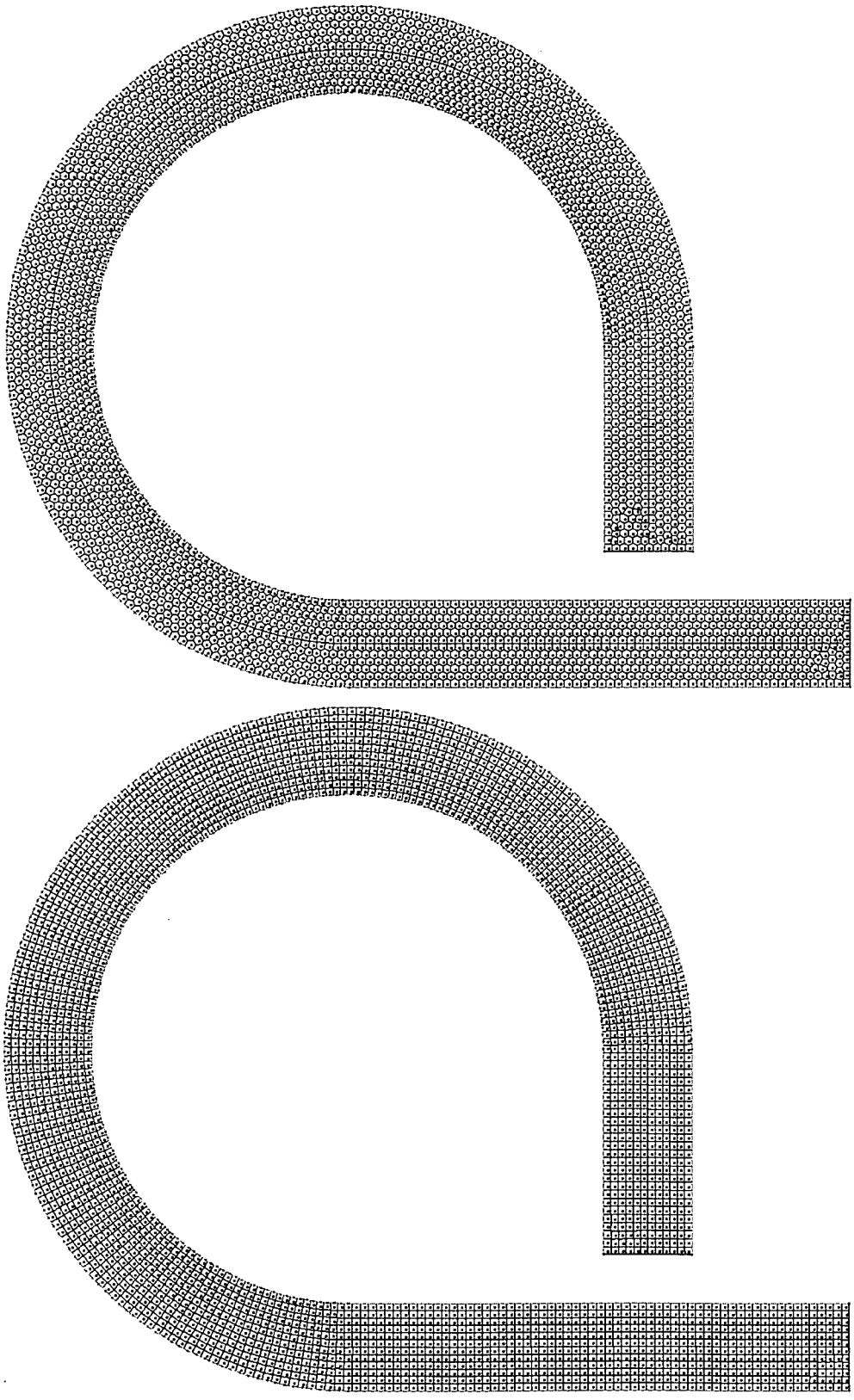

Figure 5.23: Computation grids for Steffler's experiment

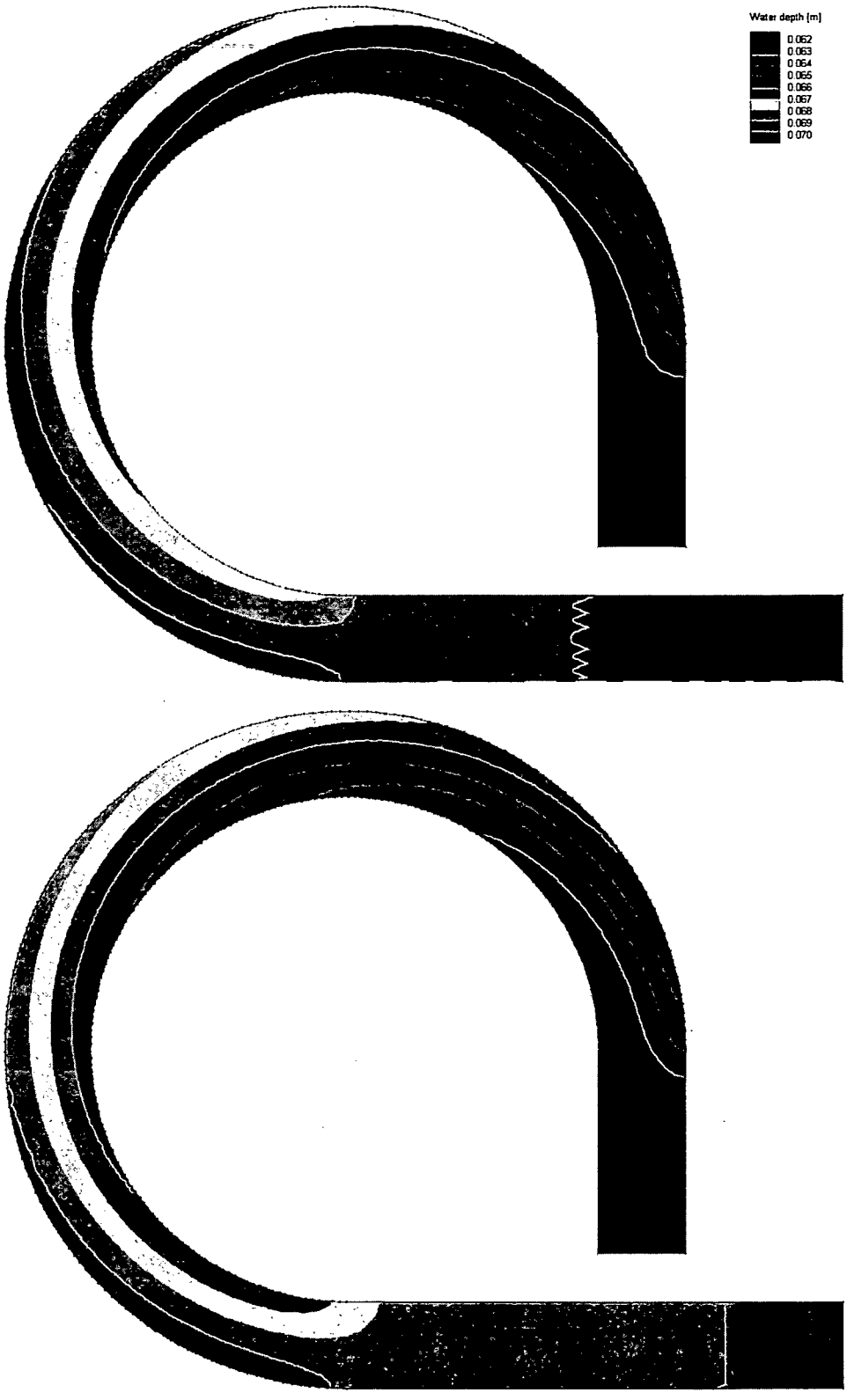

Figure 5.24: Computed water depths for Steffler's experiment (top: hexagonal grid regions; bottom: quadrilateral grid regions)

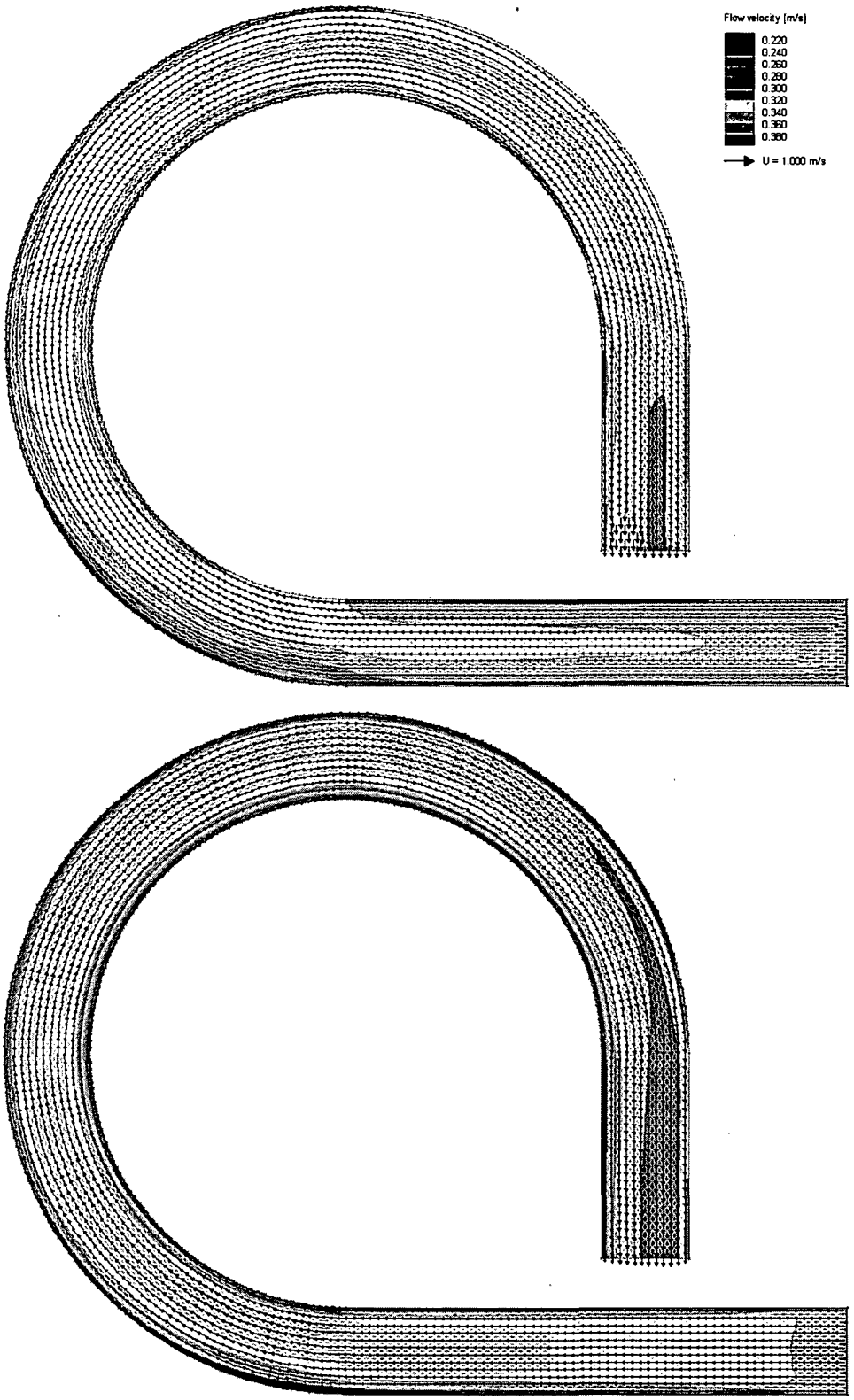

Figure 5.25: Computed depth-averaged flow velocity for Steffler's experiment (top: hexagonal grid regions; bottom: quadrilateral grid regions)

therefore the results can be judged as equal. The zigzag shape of some contour lines results from the graphical postprocessor's algorithms and is not a feature of the flow solver.

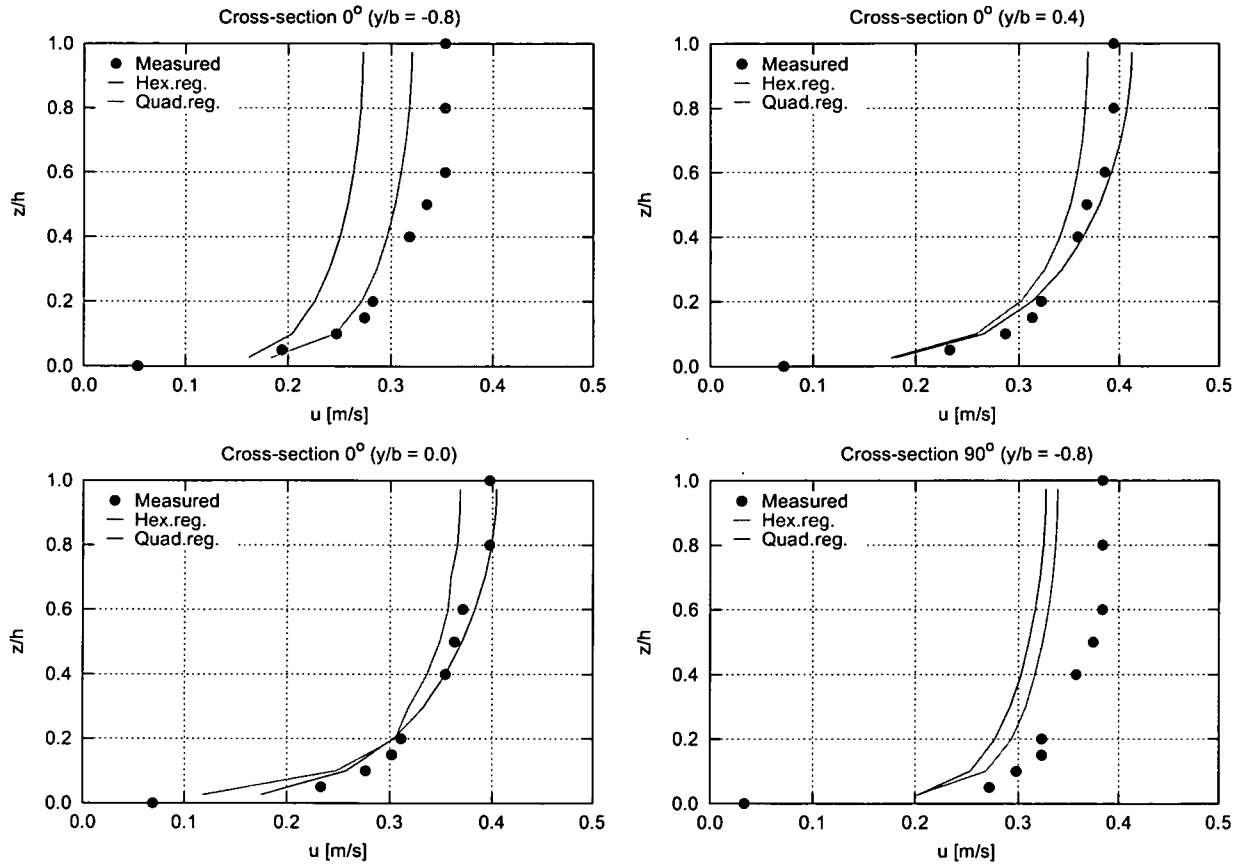

Figure 5.26: Selected longitudinal velocity profiles for cross-sections 0° and 90° of Steffler's experiment

Not quite as equal as the computed water surface elevations are the depth-averaged velocity distributions, as depicted in figure 5.25. Somehow the same impression arises as documented in previous section: extremal values are smoothed out in the hexagonal setup, as opposed to the grid based on quadrilateral regions. The latter grid exhibits very low flow velocities at the outer bank of the beginning of the bend. Later throughout the bend and in the exit channel, these low velocities are found along the inner bank. Compared to the maxima of flow velocity, they are roughly half their magnitude. An interesting result of the computation is the fact that both solutions are in agreement concerning the location of the maximum velocities; these are to be found along the exit channel. However, it must be noted that the length of the exit channel is actually too small to claim that no boundary condition errors would be able to propagate upstream and influence the results. But considering the way grid generation is performed (chapter 3), it is

not possible to assign a longer exit channel to the present model.

A more detailed assessment of the accuracy of the flow field can be done when interpreting figures 5.26 and 5.27. We can see that the minima along the outer bank at the beginning of the channel, as given by the grid using quadrilateral regions, are actually too low. Even though the hexagonal grid setup underestimates these velocities as well, this solution is much closer to the actual measurements. When we move towards the centre line of the channel, the results obtained on the quadrilateral grid become better, with an almost accurate prediction right at the center line of the 0° cross-section.

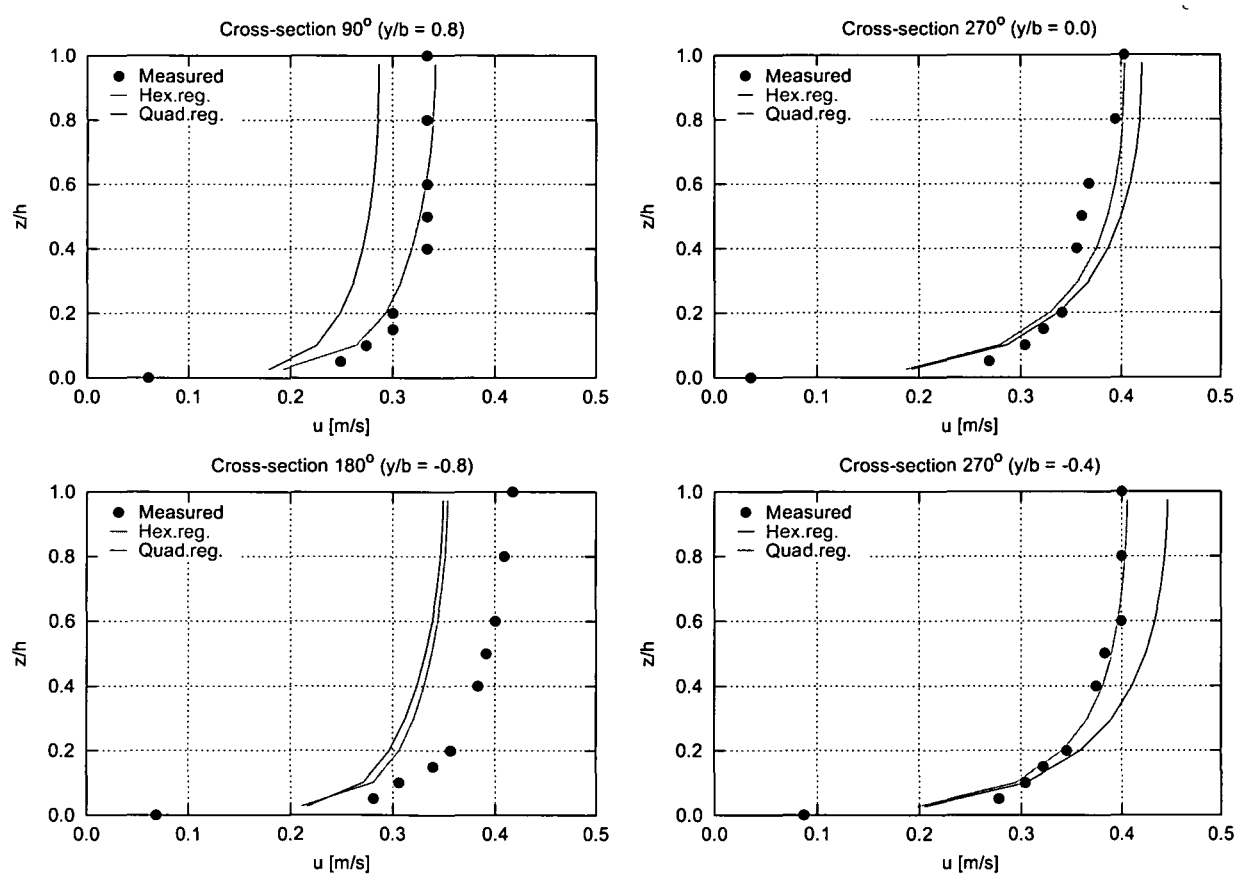

Figure 5.27: Selected longitudinal velocity profiles for cross-sections 90° , 180° and 270° of Steffler's experiment

At the cross-sections of 90° and 180°, the outer bank velocities are underestimated as well by both setups, with the hexagonal grid again returning better results. The main flow velocities at the inner bank are accurately predicted by the hexagonal grid; and this grid type also yields the best results in the 270° cross-section (fig. 5.27), even at the centre line. From this assessment we can conclude that – as far as the longitudinal velocity profiles are concerned – the hexagonal

grid type performs better than the quadrilateral one. This conclusion is fairly remarkable since it means that the smoothing of extremal values cannot be considered to be an effect purely raised by numerical diffusion, but that it is rather of a physical nature which is better captured by the hexagonally shaped grid.

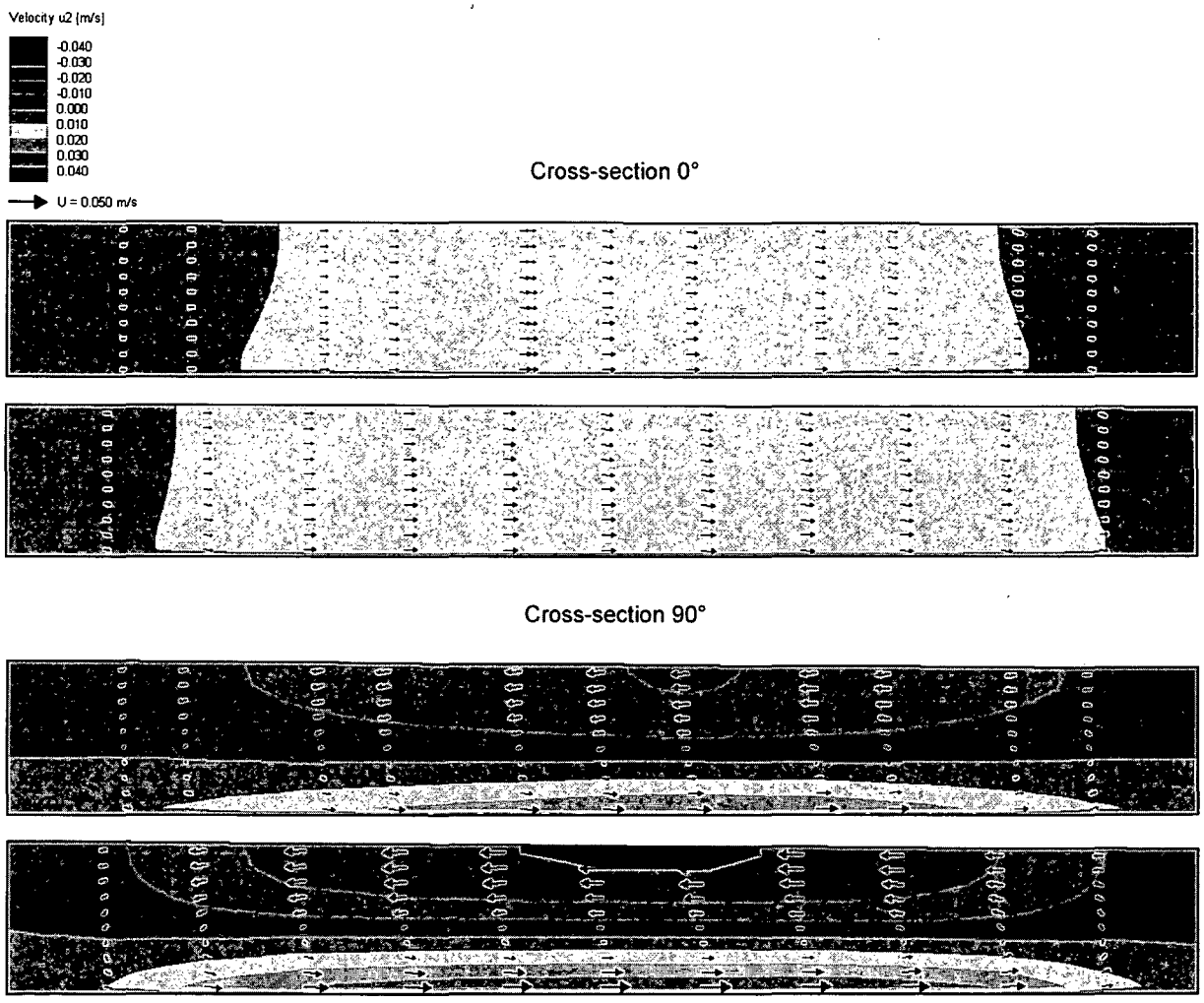

Figure 5.28: Contour plot of computed transversal velocity u_2 for cross-sections 0° and 90° of Steffler's experiment (top: hexagonal regions; bottom: quadrilateral regions) – figure scaled by the factor 2 in the vertical direction

Figures 5.28 and 5.29 depict the computed transversal velocities in a number of cross-sections. Again we can see that the solution obtained on the grid using quadrilateral regions exhibits stronger extremal values. A clear secondary motion is visible throughout the bend, with maxima of around 0.07m/s in measurements and 0.05m/s in the computation results.

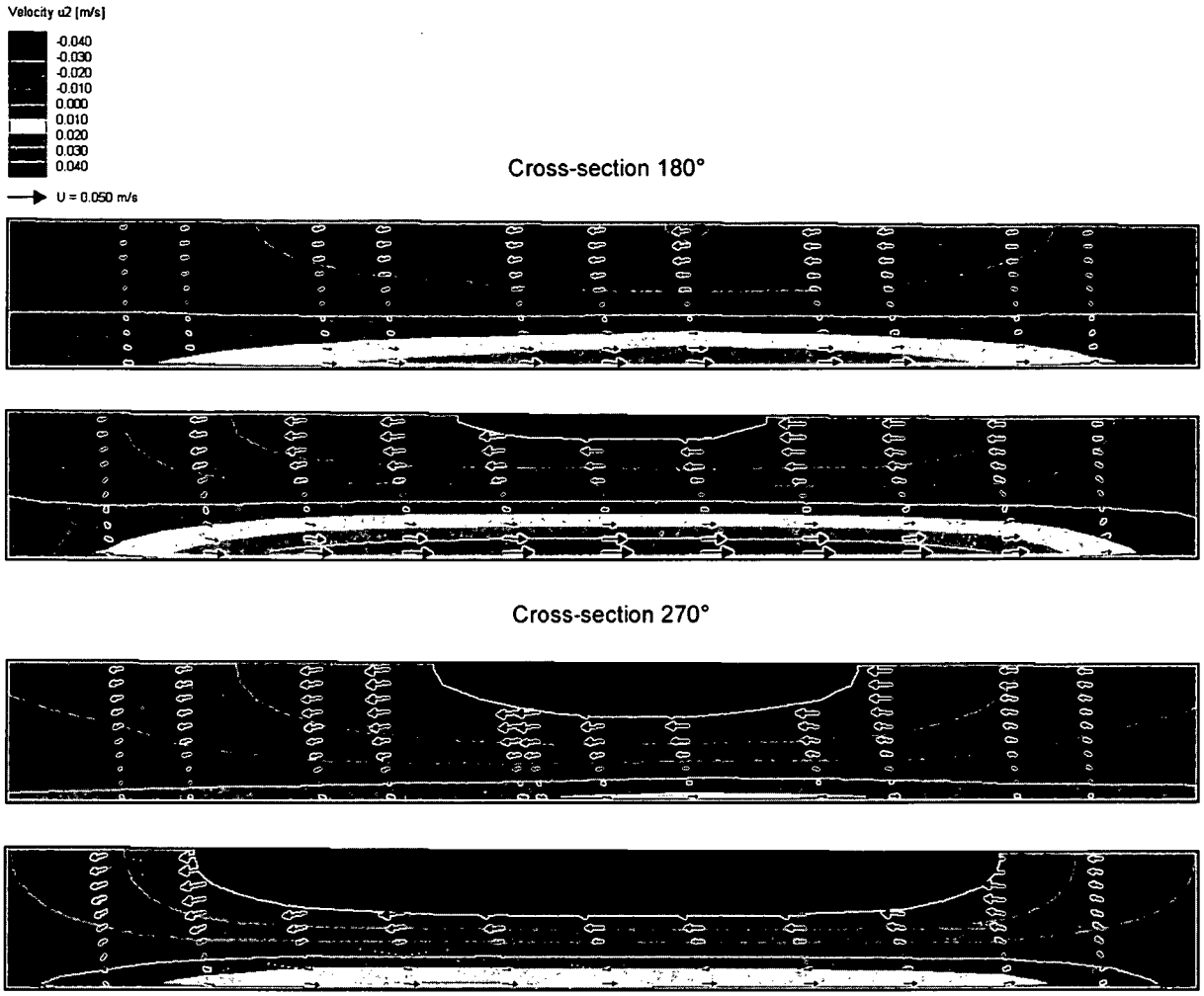

Figure 5.29: Contour plot of computed transversal velocity u_2 for cross-sections 180° and 270° of Steffler's experiment (top: hexagonal regions; bottom: quadrilateral regions) – figure scaled by the factor 2 in the vertical direction

Figure 5.30 allows for a comparison with Steffler's measurements. On a first glimpse, the impression is the same as in Rozovskii's experiment, with the computed transversal velocities not yielding a very good agreement with the measurements and the strength of the secondary motion being vastly underestimated. However, a closer look at the measured velocities reveals that the maxima are not to be found right at the bed but rather a certain distance above it – a fact that also represents the real situation better than the measurements done by Rozovskii (section 5.3). And exactly these velocities at the bed seem to be represented well in the present model. Since the wall boundary condition – which is applied at the bed and the banks – is a very strong boundary condition that highly influences the computational results, and since the vertical resolution

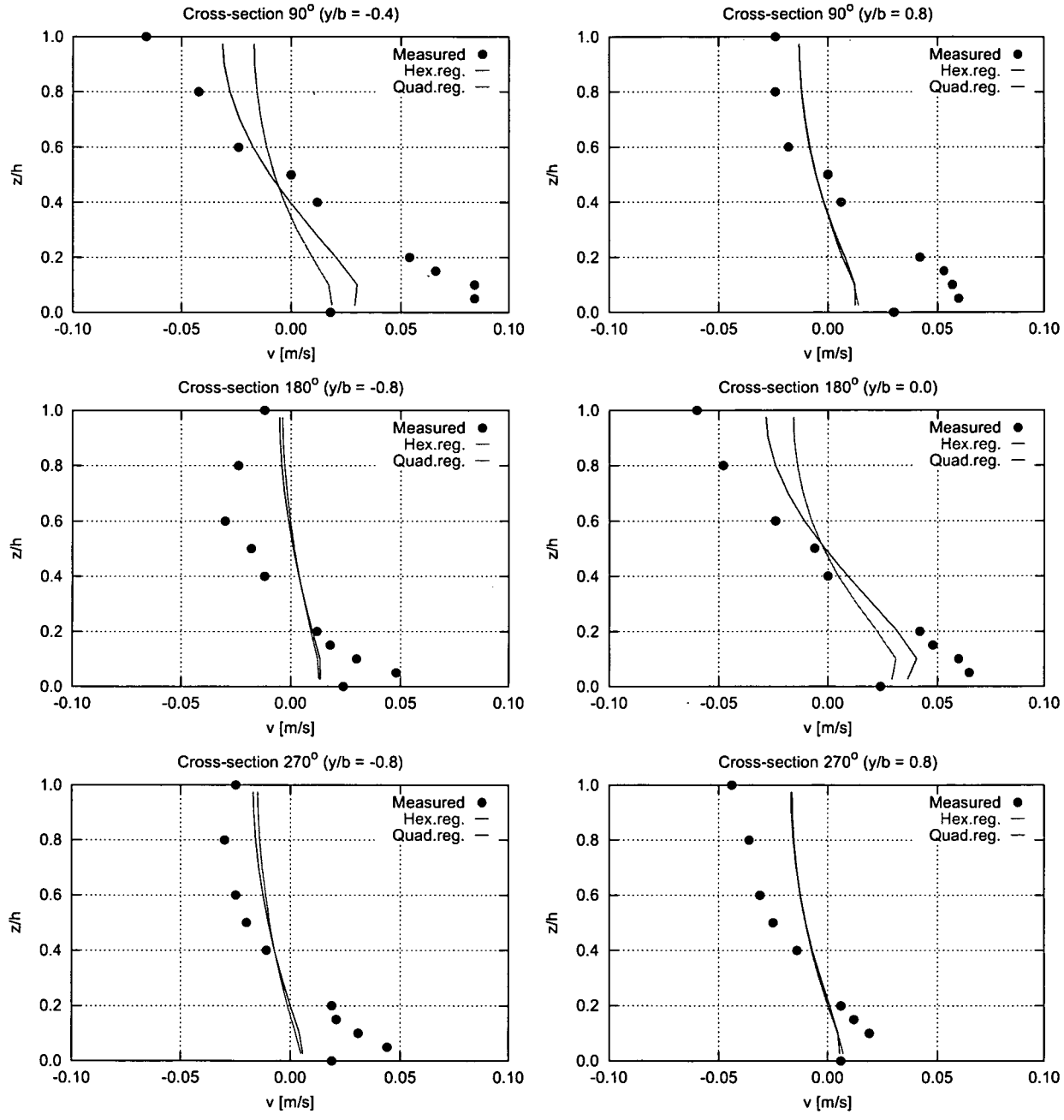

Figure 5.30: Selected transversal velocity profiles for Steffler's experiment

of the model is not so dense as to capture the extremal values visible in the diagrams, one can conclude that actually a better representation of the secondary velocities is not possible in the present setup. If a better vertical resolution would be employed, it can be expected that the secondary flow motion would be represented more exactly, but on the other hand the computation

times would become prohibitively high in order to achieve this goal.

As far as the comparison of secondary velocities between hexagonal and quadrilateral grid regions is concerned, there is not much difference in the results except for two locations close to the centre line of the channel where the quadrilateral setup yields slightly better results. On the other hand, the computation time was lower for the hexagonal grid setup and this configuration also yielded longitudinal velocities in better agreement with the measurements. Therefore, in terms of an overall assessment, we can conclude that the hexagonal grid setup would actually be the better choice for an engineering application in a problem similar to Steffler's experiment.

5.5 Flow in an S-Shaped Trapezoidal Channel

So far only channels with rectangular cross-section have been used for validation of the numerical code. The final validation experiment now leads us to a channel with trapezoidal profile, thus making a step towards natural flow situations. The physical experiment was set up at the Water Resources Institute of the University of Innsbruck, consisting of an approach channel 3.0m in length followed by two consecutive bends with a mean radius of 4.0m and apex angle of 60 degrees each, and finally an exit channel being 2.8m long. The banks are sloped by a ratio of 2:3. This configuration corresponds to characteristic values for alpine sinuously trained rivers (*Vigl* (1990) [86]). Initially a bed slope of S=0.01 was chosen to simulate alpine conditions, exhibiting Froude numbers larger than unity, but later the physical model was adapted to a slope of S=0.005 (*Feurich & Schöberl* (2003) [22]). The model's cross-sectional geometry consists of a fixed bed 0.40m in width and two banks, each 0.80m wide; into this geometry the actual bed layer is introduced by filling it with sediment to a certain height, either fixing it afterwards or allowing for sediment transport (erosion and deposition). *Feurich* (2002) [21] discusses a series of experiments with varying discharge, bed layer depth and fixed or mobile sediment at the bed.

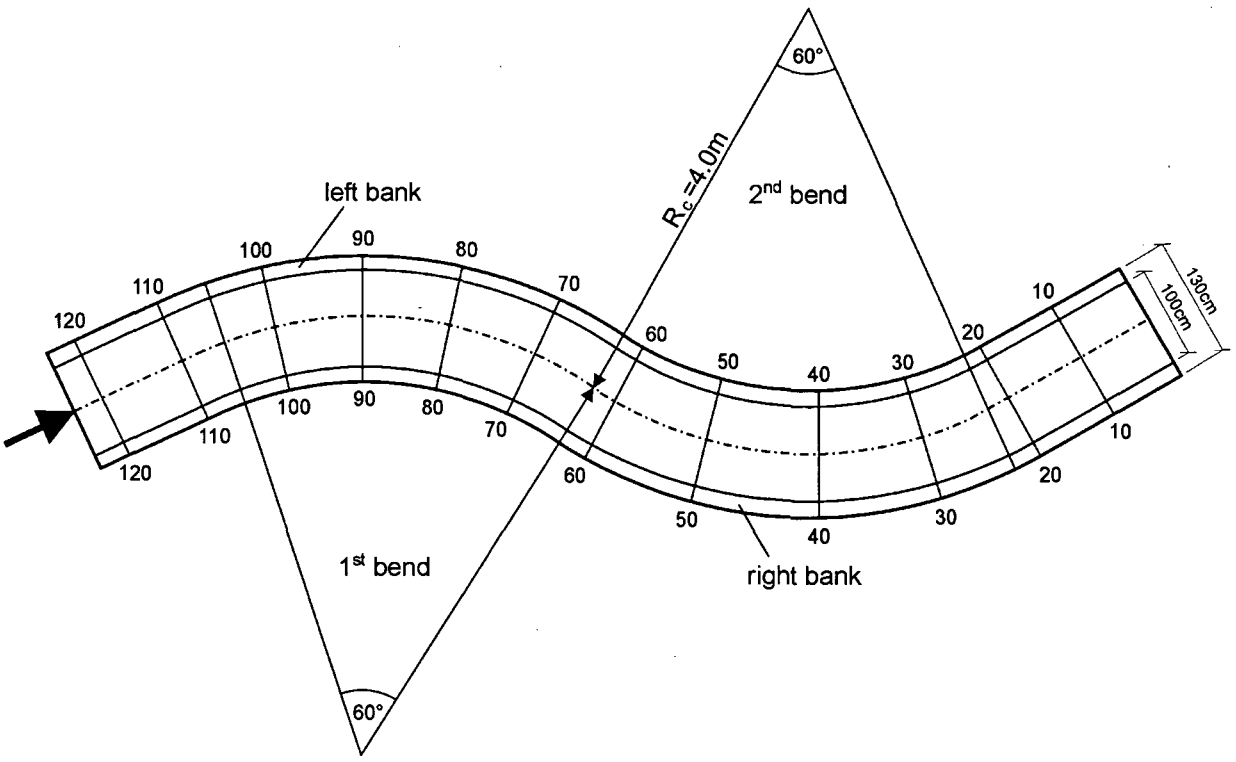

Figure 5.31: Layout of the computational equivalent of the S-shaped trapezoidal channel

The present work focuses on only one of these variations: the bed layer of this experiment is 20cm thick, resulting in the geometry illustrated in figure 5.31 with a 100cm wide fixed bed. Discharge is 0.09m^3/s, and roughness conditions are documented to be k_s=0.0047m at the bed and k_s=0.0074m along the banks. The physical experiment is subdivided into 120 cross-sections along which measurements were performed.

In terms of spatial discretisation, a grid composed of hexahedral cells is employed in the present work. This approach was chosen for two major reasons: first, the experiment allows for identification of a clear main direction of the main flow; second, a high-resolution discretisation in transversal direction is very important to capture the geometry at the side walls correctly. If a polyhedral grid based on hexagonal regions had been employed, either a prohibitively large number of cells would have resulted, or it would have been necessary to reduce the transversal resolution. Since neither option was acceptable, it was found that a brick-type grid composed of hexahedral cells was best suited for studying this physical experiment in a numerical model.

The computation grid starts at cross-section 120 and ends at cross-section 10, with the longitudinal cell dimension being equal to the cross-section distance, resulting in a length of approximately 110mm. *Feurich* (2002) [21] conducted a sensitivity study using the SSIIM model (*Olsen* (2000) [57]) and found that computed water surface and flow velocities obtained on a grid with extended entrance and exit channels did not differ from the corresponding results using a grid representing the area between cross-sections 10 and 120, respectively. In other words, no significant influence of the upstream and downstream boundary conditions was found that would require extensions of approach and exit channels in the present configuration.

For the transversal cell dimension, a distance of 35mm was chosen. Hence, the ratio of longitudinal and transversal grid dimension is about 3:1, which is in the recommended range found in literature. The average flow depth in the flume was found to be approximately 100mm which,

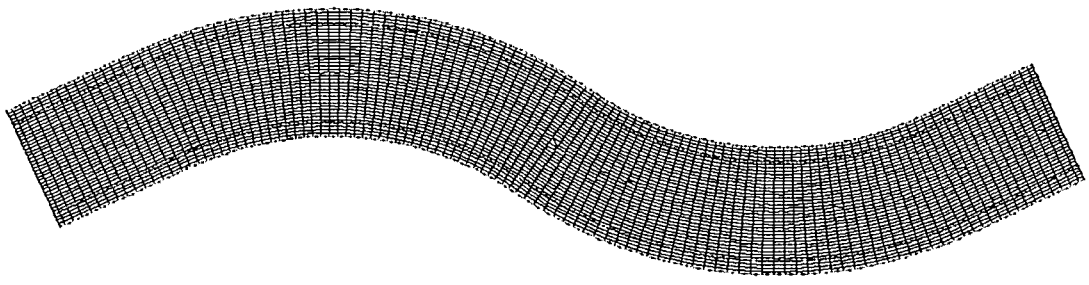

Figure 5.32: Computation grid for the S-shaped trapezoidal channel

due to the bank slope of 2:3, results in a wetted bank width of 150mm each.

Chapter 3.7 discussed the conditions under which a cell pile can become wet: basically, its base point must exhibit a higher elevation than the terrain surface. Given the transversal cell dimension of 35mm and the bank slope of 2:3, the centre point of an outside region would only turn wet if the water level rises for more than 12mm. For all other cells, a rise in water levels results in a change of vertical elevation only. It was found that the condition of a rise in water levels of more than 12mm was met only in a very small number of grid regions. However, these regions would subsequently exhibit water depths of only a few millimeters, and after vertical subdivision the bottom grid point would be located less than a millimeter above the terrain surface. In combination with a roughness height of k_s= 7.4mm, this leads to severe instabilities in those cells since the equations that make up the boundary conditions for the flow module (chapter 4) contain the ratio between roughness height and surface distance. This is the reason why most commercial models come with a restriction as far as the minimum cell height is concerned. Since such a restriction was not implemented in the present model, it must be enforced by proper geometrical choices, which finally justifies the cutting of geometry at the wetting line of the mean flow, leading to the actual geometry as illustrated in figure 5.31 with the corresponding computation grid depicted in figure 5.32. The consequence of such a reduced geometry is a slight increase in water depth, since continuity must be enforced by the model; however, the impact was found to be actually very little, which is also in line with *Feurich*'s (2002) [21] findings who employed a similar procedure for representing the geometry. *Nguyen* (2000) [51] also reports the need for a small "vertical wall" at the banks of a numerical model to avoid instabilities in a trapezoidal channel.

During test computations required to find the optimum numerical parameters (e.g. relaxation factors) it was found that the trapezoidal channel experiment actually exhibits numerical instabilities that can be credited to the cells with a low distance from the channel bed to the cell centroid. These instabilities could be reduced by making use of comparatively low relaxation factors (velocities were relaxed by 0.5, pressure/continuity by 0.03 and turbulence by 0.25). However, still the numerical algorithm exhibited severe instabilities upon the first update of the water surface, leading to divergence irrespective of the surface relaxation factor used. Even though the exact reason for this behaviour could not be found, it could be traced to the influence of roughness along the banks on the $k - \varepsilon$ model equations, resulting in large absolute values for turbulent kinetic energy and dissipation in regions of low water depth. To avoid this phenomenon causing instabilities in the solution procedure, other flow simulation models (e.g. Fluent) limit the ratio of eddy viscosity and kinematic viscosity. Another approach is to lower the roughness height in

regions of low water depth by introducing a mean roughness value for the entire flow domain instead of the two different roughness heights in bed and banks. *Feurich* (2002) [21] discusses the theoretical background for obtaining such a mean roughness for trapezoidal channels and comes up with k_s=5,4mm as a result for the present case, noting that he found hardly any difference in the results obtained using this approach. By applying this value to the entire flow domain, it was possible to solve the flow equations until convergence.

The 4,218 grid regions depicted in figure 5.32 were vertically structured into six cells, each occupying exactly one sixth of the flow depth, resulting in 25,308 cells. This setup was preferred over the one used so far in order to keep the total number of cells relatively low and avoid cell centroids being located too close to the solid walls bounding the flow domain. Nonetheless about 58,000 iterations and approximately 65 hours of computation time were needed to obtain equilibrium, this being mostly the result of the low relaxation factors employed.

Figure 5.33 depicts the computed water surface along with the measured values and the results of the SSIIM model as found by *Feurich* (2002) [21]. The most notable finding is the perfect agreement between the simulation results of the two numerical models; they are virtually indistinguishable from each other. However, simulation and measurements are not in perfect agreement. In the second (downstream) bend, the results fit the measurements very well, with the rise of the water level along the right bank and the fall of the water surface along the left bank being correctly predicted. In the first bend both numerical models slightly underpredict the maximum of the water level along the left bank. But a closer look on the measurements reveals that right downstream of the inlet a depression in the water surface can be found which of course gives rise to a more distinctly visible peak along the left bank thereafter. The reason for this depression in the measured surface is unclear; it could be the result of a slight bump in the laboratory flume, an imperfection in bed roughness, or the overall situation at the inlet. This local minimum in the water surface is also the reason for the lack of agreement between measurements and simulation in the approach channel itself; however, since this region is heavily influenced by the boundary conditions – both in the laboratory and the numerical experiments – it will not be used for a comparison after all. Hence, in summary it can be said that the computed water surface is actually in good agreement with the measurements.

Unfortunately, measurements of flow velocities for this case were performed with an instrumentation that was later found to produce unreliable results by the experimenter. However, reliable measurements are available for the same flow situation on a mobile bed; these were found to be in good agreement with the corresponding SSIIM model results (*Feurich* (2002) [21]). Therefore we can expect the SSIIM model results for the fixed bed situation to be reliable as well and use

that data as basis for validation.

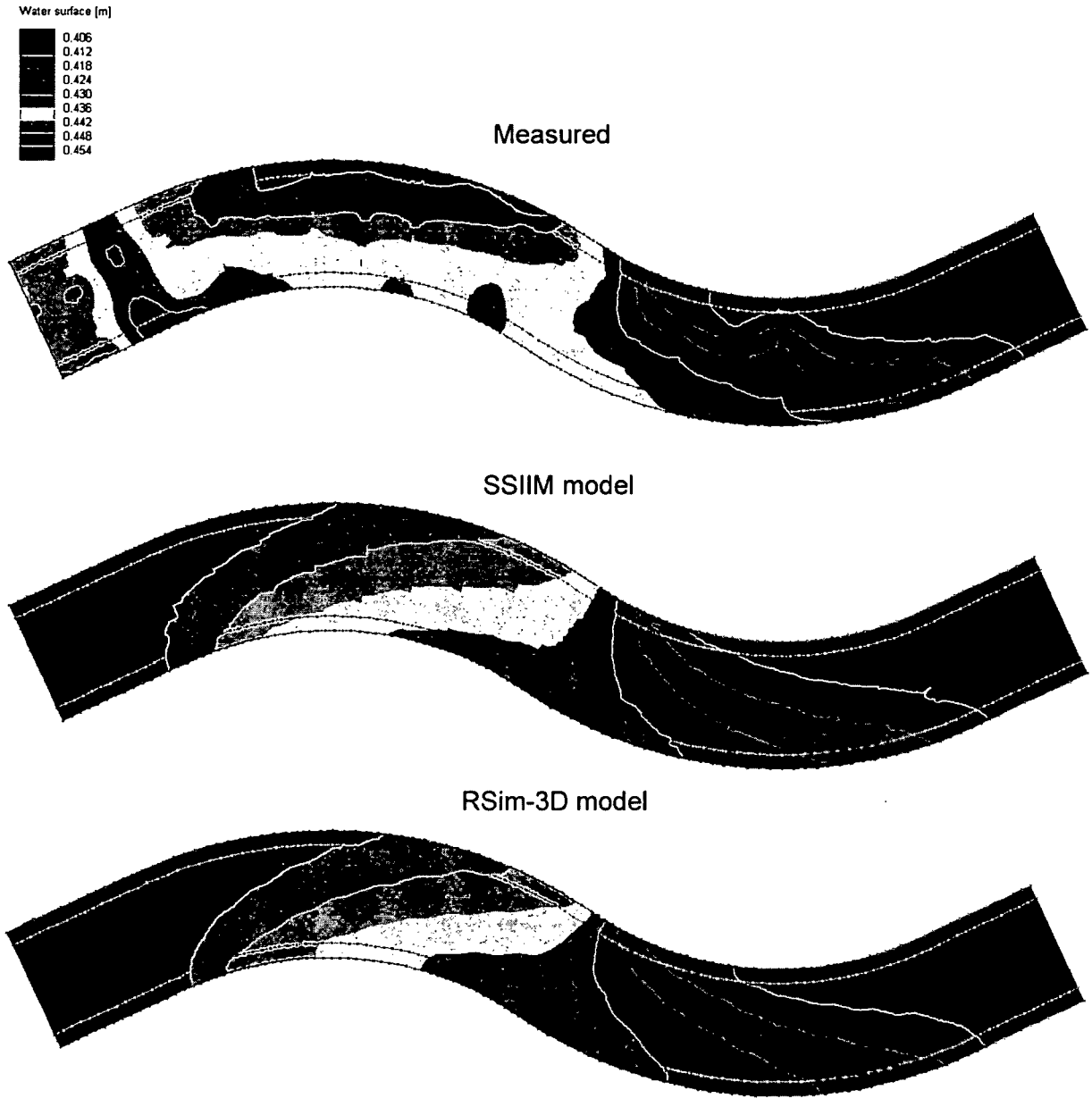

Figure 5.33: Water surface elevations for the S-shaped trapezoidal channel

Figure 5.34 allows for a comparison between the depth-averaged flow velocity obtained through the SSIIM and RSim-3D models, respectively. It should be noted that the available SSIIM model data only covered the flow area between the base points of the embankment, illustrated by the inner two green lines, and therefore the plotted data along the banks had to be obtained by means

of extrapolation. This is the reason why the effect of surface roughness in those regions of the embankment that exhibit low water depths cannot be seen in correspondingly low flow velocities in the illustration. Hence, a comparison is only useful in the region between the two base lines of the embankment. It is visible that the maximum of the flow velocity in the second bend is predicted correctly, both in location and magnitude (see also fig. 5.35). Also the velocity distribution downstream from this location is in very good agreement. Besides small differences in the approach channel, once again resulting from the influence of the boundary condition at the inlet, a good agreement can also be found in the first bend, with maximum and minimum flow velocities being correctly predicted.

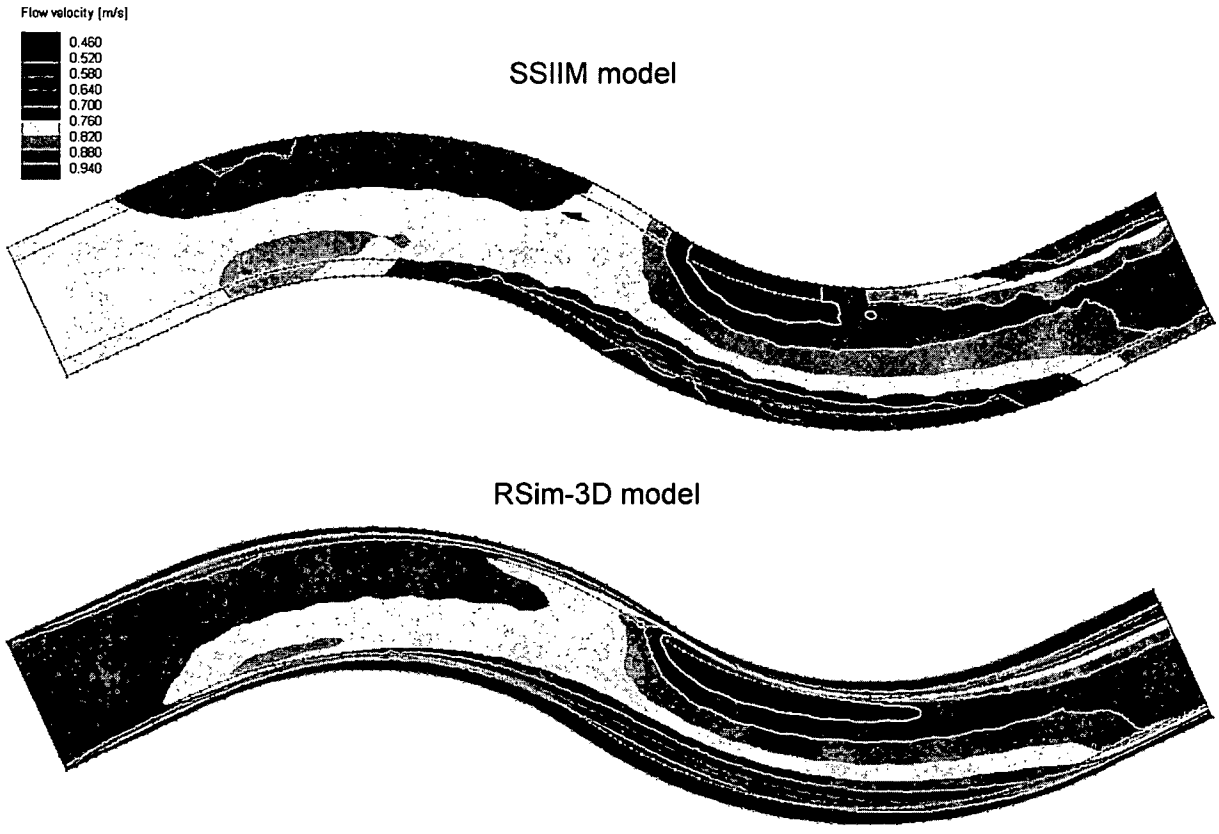

Figure 5.34: Depth-averaged flow velocity for the S-shaped trapezoidal channel

Finally, figure 5.36 depicts a series of cross-sections throughout the channel:

- start of the first bend (cross-section #105),
- middle of the first bend (cross-section #85),
- point of inflection (cross-section #64),

- middle of the second bend (cross-section #44), and
- end of the second bend (cross-section #23).

It can be seen that a strong secondary motion develops in both bends with maxima of around 0.15 m/s pointing towards the outer bank near the surface and into the opposite direction close to the bed. There are also two geometric features that can be noted: First, the intersection between bed and embankment is not a sharp line but appears to be smoothed out to some extent. The reason for this is to be found in the way the polyhedral grid was generated using the Kriging methodology from terrain elevation data provided. However, the resulting error is obviously not large. Second, the geometry near the surface is well preserved along the inner banks while the outer banks appear to be lacking a small part of the flow domain. This phenomenon was already discussed in the introductory paragraphs of this chapter; after an entire computation region has turned wet, it still takes a reasonable amount of rise in water levels for the algorithm to declare the neighbouring computation region wetted. This rise can only take place in the vertical direction during that phase, and that's what can be seen in figure 5.36. Of course, due to the continuity equation, the water level will turn out to be slightly higher if the geometry is not exactly preserved, but on the other hand this error is small, hence almost negligible.

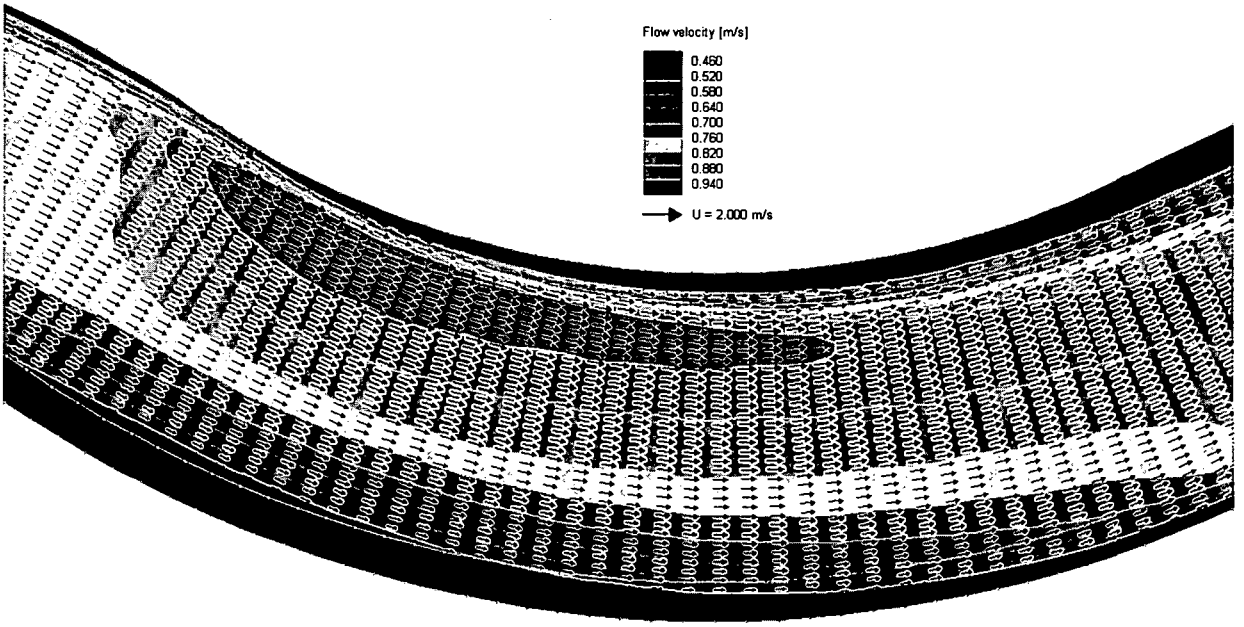

Figure 5.35: Flow detail in the second bend

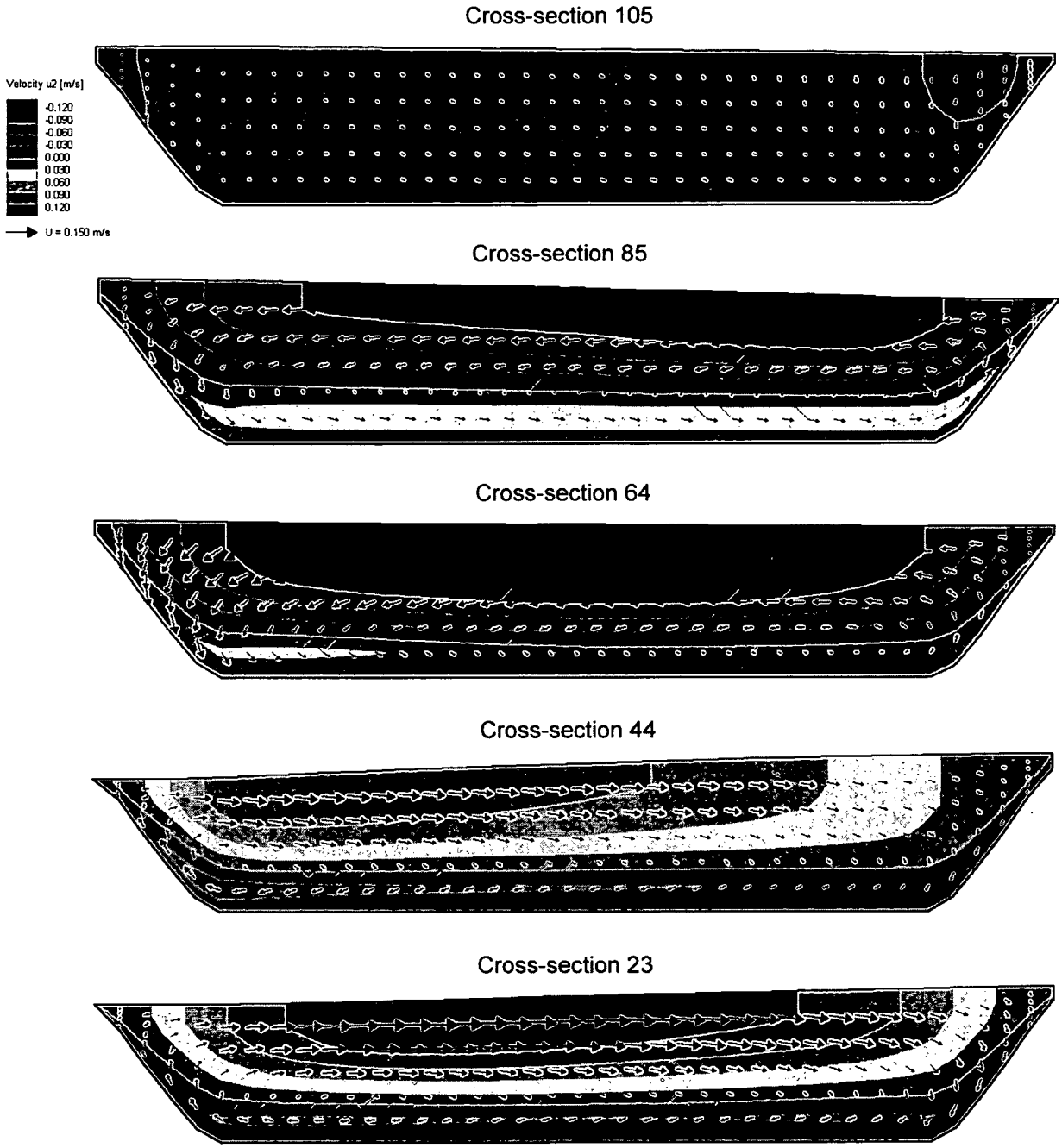

Figure 5.36: Selected cross-sections of the S-shaped trapezoidal channel – figure scaled by the factor 2 in the vertical direction

6 River Application

6.1 Introduction

In August 2002, a catastrophic flood event following several days of heavy rainfalls struck large parts of Central and Eastern Europe. Many rivers in this region reached flood peak levels that exceeded the highest maxima historically known, causing severe losses which sum up to approximately 3.1 billion Euro in Austria (*ZENAR* (2003) [94]). A significant portion of the damage can be attributed to the river Danube, where a 100 year's flood was encountered, and its tributaries, in some of which the return period of the flood event reached several thousand years (e.g. river Kamp, *Gutknecht et al.* (2002) [33]).

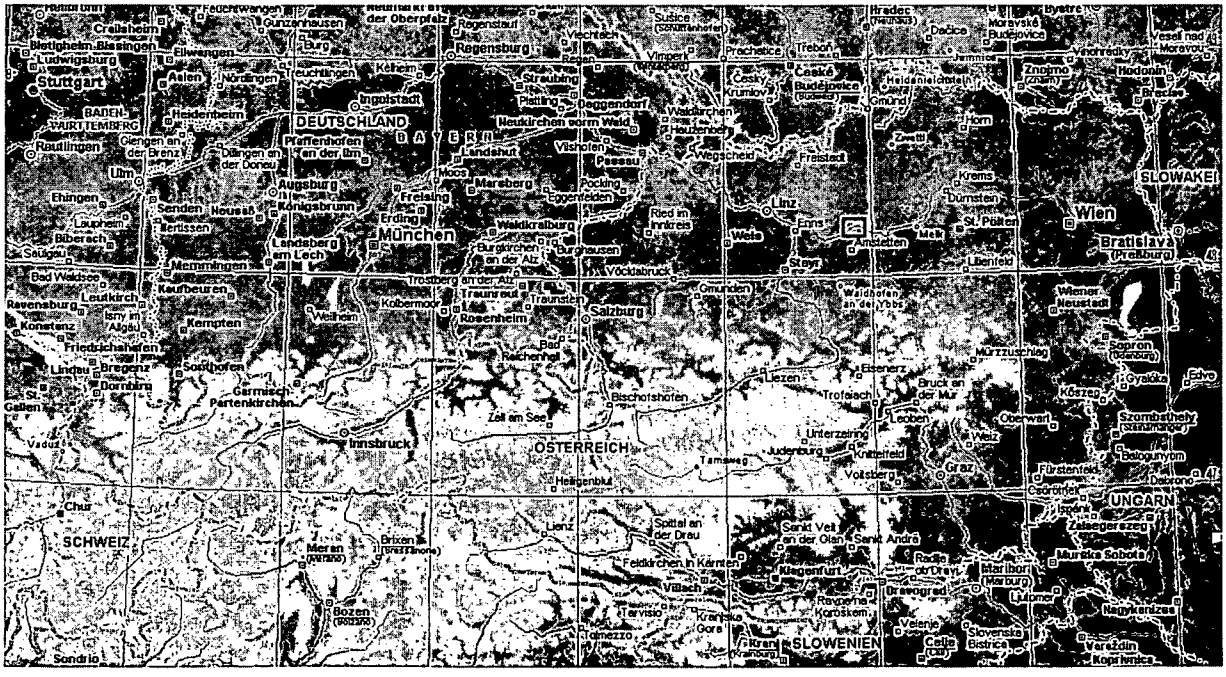

Figure 6.1: Location of the Danube river bend at Grein in Austria

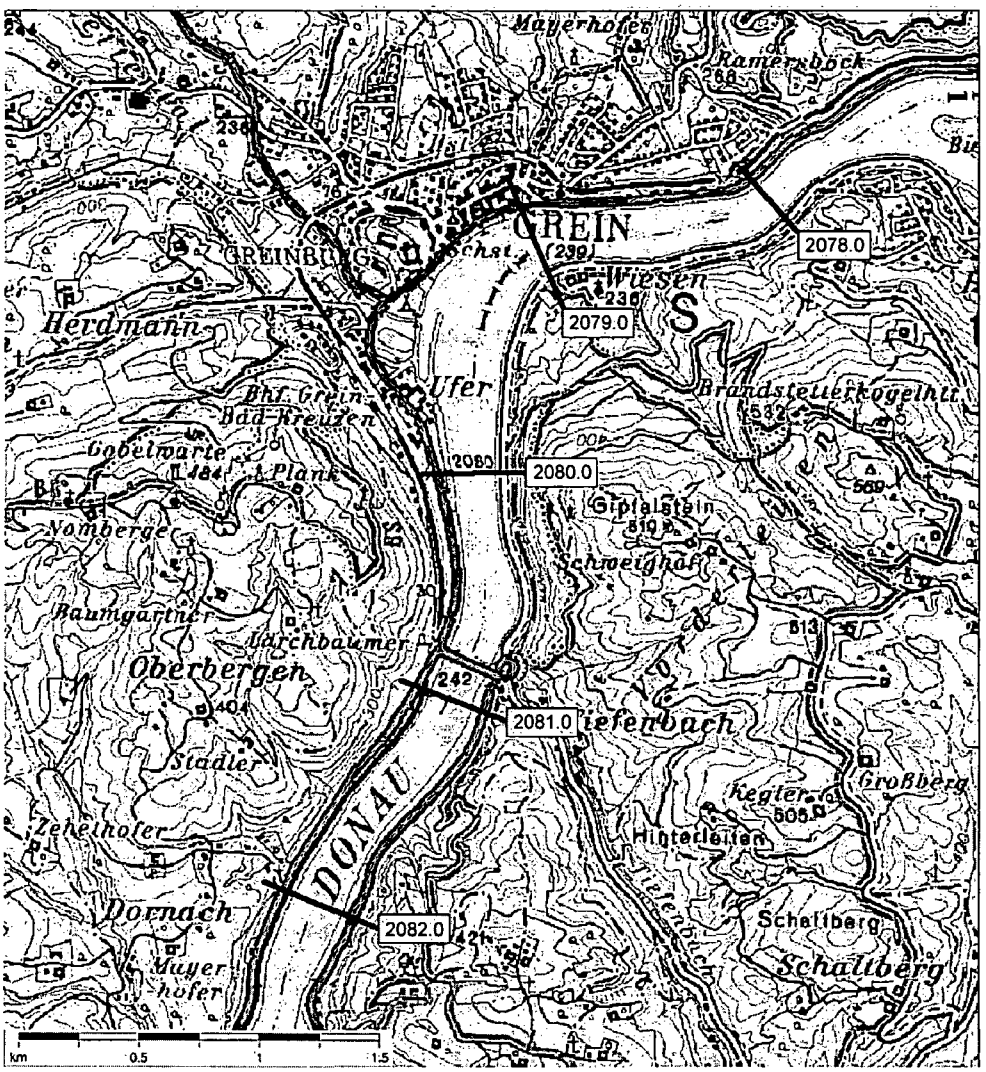

Figure 6.2: Detailed map of the river bend at Grein (based on Austrian Map by BEV)

Among the inundated villages along the river Danube was the municipality of Grein, located in the administrative division of Upper Austria (fig. 6.1). Figure 6.2 depicts a detailed map of the region in question. It can be seen that the municipality is located along the outer bank of a 90 degree river bend near river station 2079.0; the flow direction is from South-West to North-East in this region. However, even though the flood protection of the village was dimensioned to withstand a 100 year's flood, it was overtopped and, as a consequence, parts of the municipality were flooded. During the flood event a difference in water surface elevations between left and right bank was measured which amounted to 80cm. That is the reason why the river bend at Grein was included in a list of problematic places along the river Danube in the FloodRisk project of

2003/2004 (*Habersack (ed.)* (2004) [34]). In its assessment of the situation, the project team recommended the use of a 3D flow simulation model to reproduce the water surface slope in this region since the usually applied 1D models are not adequate for this purpose – that's also the reason why the maximum water level at the outer bank corresponding to a flood return period of 100 years was inaccurately predicted in the past.

In this chapter of the present work, the flow situation in the river bend at Grein is analysed with three different simulation models (RSim-3D, SSIIM and Fluent). The project domain was established between river stations 2082.0 (upstream) and 2078.0 (downstream), resulting in a river reach four kilometers in length. The area of specific interest is – as previously mentioned – located near station 2079.0, hence the approach section of the river is approximately three kilometers long. This selection was made to ensure that proper flow conditions, independent of flow boundary conditions, are to be found near the municipality of Grein. In this context it should be mentioned that it would have been desirable to move the outflow boundary further downstream as well, but unfortunately the river bed becomes bifurcated several hundred meters after the selected outflow (see fig. 6.2), causing this region to be unsuitable for a flow boundary.

Figure 6.3 depicts the terrain elevation data which was available for setting up the numerical model: terrain elevations are known at all data points coloured in blue. Within the river bed, the elevations are frequently measured in cross-sections, one per 100m river length. For the current study, data gathered in the year 1999 was used; the measurement points in each cross-section were available in a density of one point per meter cross-section length. In addition to the bed elevations, terrain elevation data was provided by the Austrian Federal Waterways Authority. This data had been derived from aerial views of the region to extend the river cross-sections across embankments and floodplains. The resulting digital terrain model is visualised in figure 6.3 by yellow contour lines every 5 meters.

Meshing the whole area of available terrain elevation data would have resulted in a large number of cells and nodes, requiring unnecessarily high computational effort irrespective of the numerical model used. Therefore it was decided to perform a 1D backwater computation using the RSim-3D model to approximate the bank line for the 100 year's flood and subsequently use this line as domain boundary (red line in fig. 6.3). Figure 6.4 allows for a comparison with the actual bank line (coloured in yellow) encountered during the passage of the flood in August 2002: the aerial view shows that the overall shape of the flooded terrain is well predicted; two small basins at km 2078.9 and km 2078.5 are correctly represented, and so is also the bank line both in the village of Grein and at the opposite shore. Only the large basin at the left bank near km 2079.4 is underestimated in its size. This error is to be attributed to the unavailable terrain data in that

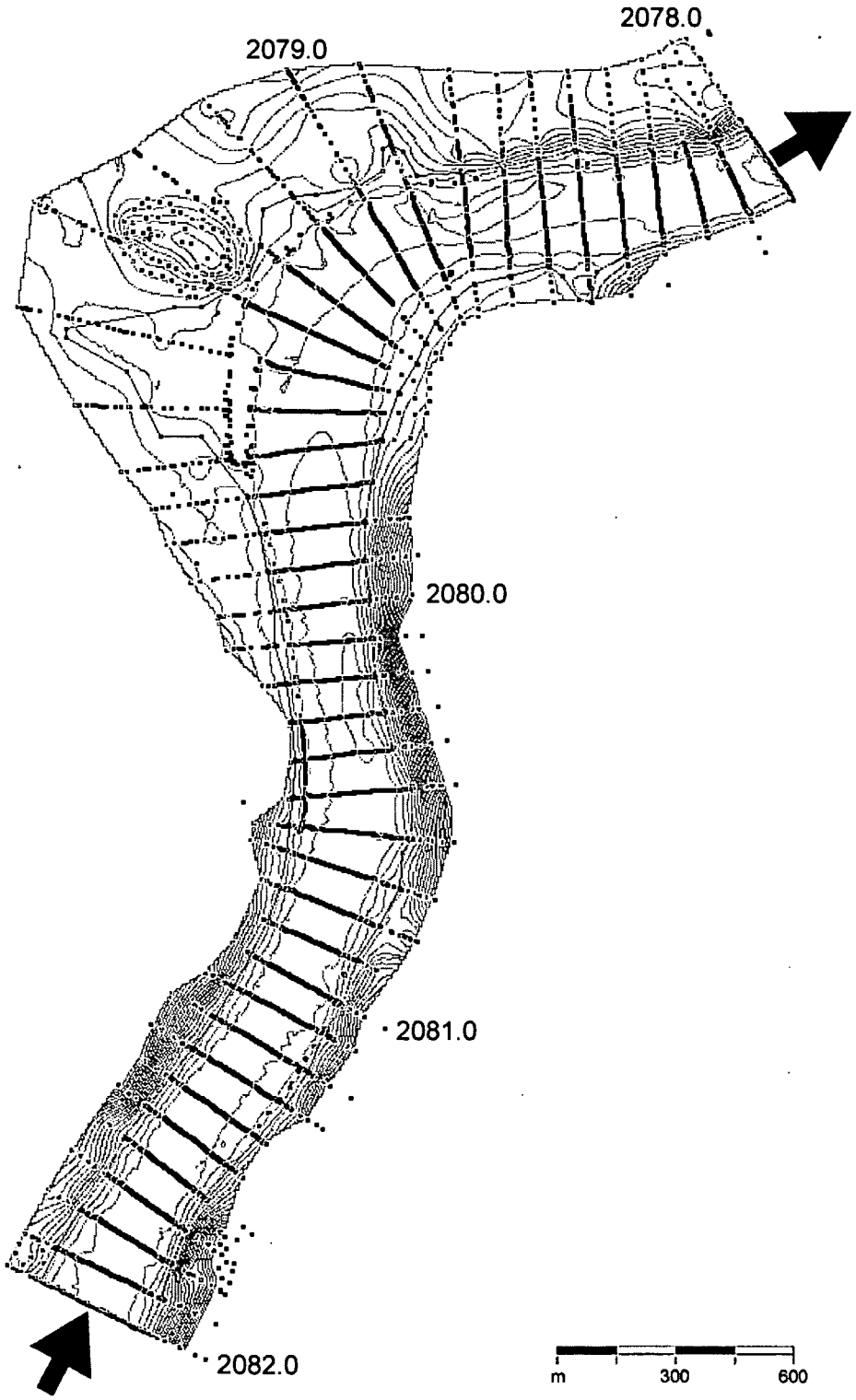

Figure 6.3: Terrain elevation data and boundary of the numerical model

region, since terrain elevations are known only in extension lines of river cross-sections and the region in question lies between two of them. Hence, the Kriging approach could not reproduce the exact terrain in this area. However, after evaluating the aerial view and the flow patterns discussed later in this chapter, it becomes clear that water depths and flow velocities in said region are very low so that the area does not contribute to the overall conveyance, therefore the results of the flow simulations are not affected.

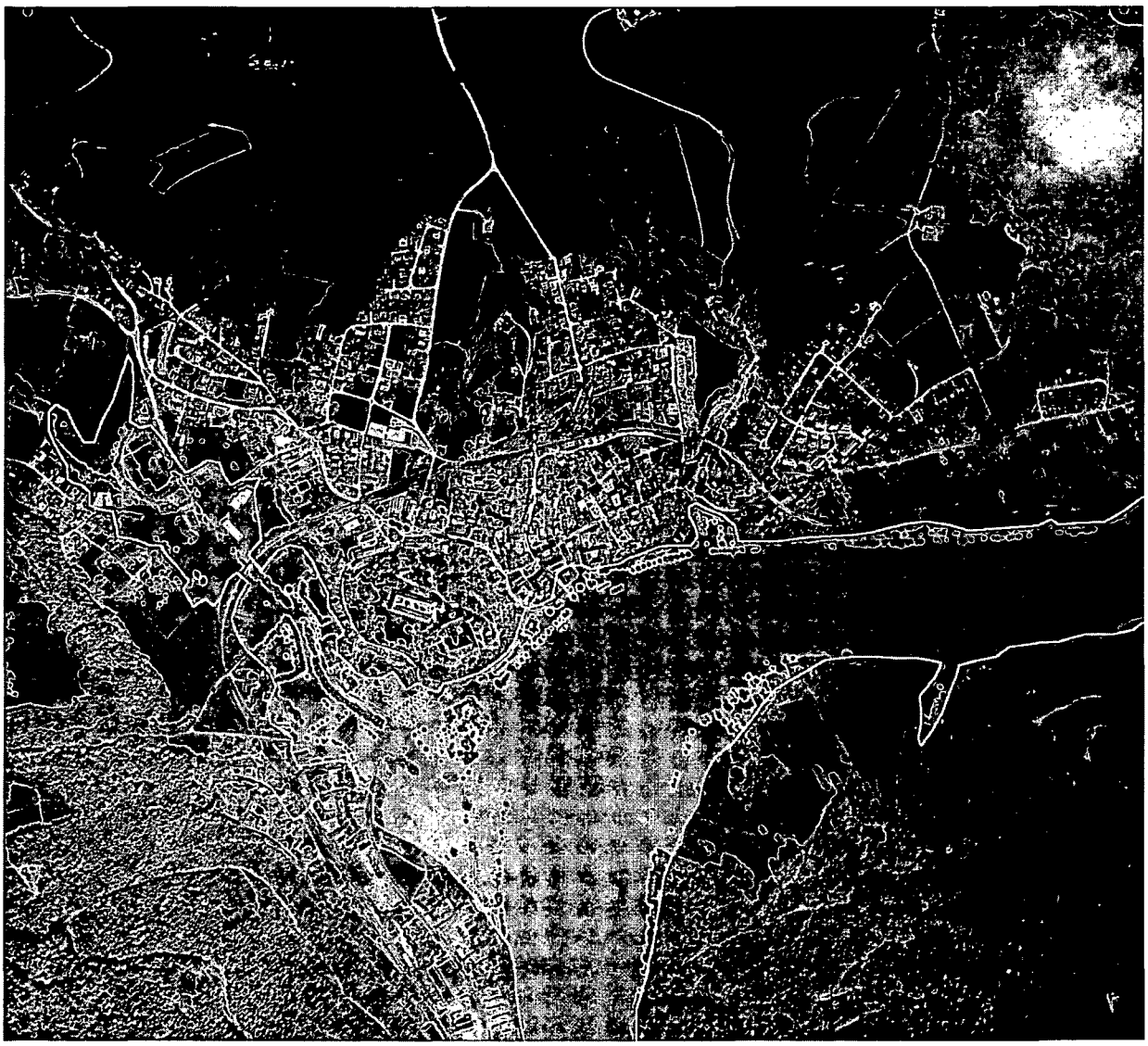

Figure 6.4: Aerial view of Grein in August 2002 (based on *BEV* (2002) [13])

The maximum water surface elevations corresponding to specific discharges are published by the Austrian Federal Waterways Authority for every stream-kilometer along the river Danube

in Austria. Known as KWD (*"K*ennzeichnende *W*asserstände der *D*onau*"*, characteristic water levels of the River Danube), they can be used as flow boundary conditions for numerical models. After the flood event of August 2002, the KWD values for the 100 year's flood were revised to reflect the maximum water levels observed. Therefore these values are not only useful as boundary conditions, but they are also good indicators for the assessment of the output of numerical models. Table 6.1 gives the KWD above sea level for mean discharge (MQ), highest navigable discharge (HSQ) and discharge with a return period of 100 years (HQ100) in the project region.

Stream-km	MQ [m^3/s]	MW [m]	HSQ [m^3/s]	HSW [m]	HQ100 [m^3/s]	HW100 [m]
2082.0	1 830	226.74	4 770	228.79	11 050	235.03
2081.0	1 830	226.67	4 770	228.56	11 050	234.77
2080.0	1 830	226.61	4 770	228.32	11 050	234.51
2079.0	1 830	226.58	4 770	228.23	11 050	234.38
2078.0	1 830	226.52	4 770	227.95	11 050	233.65

Table 6.1: Characteristic water levels of the River Danube (KWD) near Grein

In section 6.2, first the characteristics of establishing a numerical model within each simulation software are discussed. To allow for a comparison of both computation times and results, approximately the same number of cells was used in the same geometric framework for all models, leading to a comparative analysis of operational characteristics at the end of section 6.2. Section 6.3 provides the results of the numerical simulations for all models and discusses the differences. The results discussed include water surface elevations, mean flow velocities and secondary flow patterns in selected cross-sections. Finally, section 6.4 gives a short summary of this chapter.

6.2 Numerical simulation

6.2.1 RSim-3D

An unstructured grid based on hexagonal regions with a base distance of 20m was used to fill the area given by the bank line as obtained from a 1D backwater computation (see previous section for details). Along the boundary line, two rows with hexagonal regions characterised by a point distance of 10m were used to obtain a finer discretisation. Terrain elevations for the computation points were gathered by applying the Kriging approach discussed in chapter 3.4. In the vertical direction, the grid was subdivided into six equidistant layers, resulting in approximately 31,000 cells. To ensure numerical stability, an algorithm was added to the model that deactivates regions

with a water depth – measured above the computation points – of less than 20cm, which results in a maximum cell depth of approximately 3cm.

At the inflow cross-section at stream-km 2082.0, a constant discharge of 11,050 m^3/s was prescribed; the model automatically adjusts the corresponding inflow velocities according to the current flow area based on the water level, hence the inflow velocities did not have to be prescribed directly. The downstream boundary condition consists of a constant water depth of 233.65m above sea level according to table 6.1 for km 2078.0. As opposed to comparable models, RSim-3D enforces the outflow water depth at all regions next to the outflow boundary. This may result in differences in water surface elevations compared to other models, as will be discussed later.

While the water surface is treated as a symmetry boundary in RSim-3D and the corresponding elevations are found by evaluating the pressure equation, the wall boundary condition had to be calibrated in order to find the appropriate roughness coefficient. Disregarding the first kilometer within the flow domain – a region which was only modelled to obtain realistic flow conditions – the roughness coefficient was varied to obtain a good accordance with the KWD value at river station 2081.0. For the RSim-3D model, a Strickler coefficient of 35.0 most closely met this criterion. However, calibration of the roughness height for the flooded regions of the municipality of Grein was not possible because this parameter did not exhibit a notable influence on the water surface elevations in other parts of the flow domain. In the absence of measurements of water surface elevations and flow velocities in the inundated terrain to be used in a local calibration procedure, a value for the roughness height had to be selected from literature.

In *Vionnet et al.* (2004) [87] the selection of floodplain roughness coefficients for Besos River in Spain is performed by comparing data from physical model experiments and calibration results of a two-dimensional numerical model. As result a Manning coefficient of around 0.05 is obtained which corresponds to a Strickler coefficient of 20.0. *Nicholas & Mitchell* (2003) [53] apply a numerical model to a floodplain region of River Culm in the UK. Their calibration procedure results in a Manning coefficient of 0.06 (Strickler coefficient of 16.7) giving the best fit with measured data. In *Mason et al.* (2003) [45] a node-based friction parameterisation of floodplains is proposed which, based on data obtained by airborne scanning laser altimetry, classifies vegetation as short (<1.2m), intermediate or tall (>5m). Short vegetation comprehends most crops and grasses while hedges and shrubs represent intermediate vegetation; trees and buildings are classified as tall vegetation. According to *Mason et al.* (2003) [45], a floodplain containing a mix of grasses, crops, hedges and trees typical for the UK is characterised by a Manning coefficient of 0.06 (Strickler coefficient of 16.7). *Gutknecht* (2004) [32] gives a range of 10.0 to 25.0 for Strickler coefficients in typical floodplains, and *Arcement & Schneider* (2003) [7] show that

dense alluvial forests can take Strickler coefficients ranging as low as from 5.0 to 10.0.

From the aerial view in figure 6.4 it can be seen that large trees occupy parts of the inundated region within the municipality of Grein while houses had been built in other parts and even other parts appear completely without any vegetation or man-made structure. However, figure 6.2 shows that a woodless campsite and a road are located at the left bank of the river, smoothening the floodplains significantly. Therefore the actual Strickler coefficient can be expected to be slightly higher than the values given in the literature. Based on these considerations a roughness height of 1000mm was selected which corresponds to a Strickler coefficient of around 26 according to equation 4.94.

In terms of numerical characteristics, the second-order upstream method is employed for the discretisation of convective terms and the SIMPLE algorithm is used for pressure-velocity coupling. The standard k-ε model with default constants provides turbulence closure. These characteristics are the same for all three simulation models to allow for a comparison of the results obtained.

6.2.2 SSIIM

The second numerical simulation was performed with the SSIIM model (see chapter 2.2.11). This model comes in two versions which are capable of dealing with both structured and unstructured grids. Due to issues of numerical stability in the context of a flow problem with a free water surface, the structured grid version was preferred over the unstructured one for the present study. The structured grid was formed by subdividing the area between two river cross-sections – which are 100m apart – into five cell rows, and by dividing each cross-section into 25 cell columns. After subdividing each cell pile into six cells, a total of 30,000 cells results, which is a comparable number to the one which was used in RSim-3D. However, it should be mentioned that the usage of a structured grid results in an enormous speed-up of the numerical model, at the cost of the inability of using the model in complex geometries. Therefore, the bounding polygon of figure 6.3 had to be smoothed in the flood basins near Grein to allow for an application of the model to this problem.

Terrain elevations for all cell regions were obtained by applying a built-in longitudinal cross-section interpolation method to the data points depicted in figure 6.3. This method, which can be applied to structured grids only, interpolates along the vector from one cell row to the next, which usually results in a reasonably smooth terrain.

Again, a constant discharge of 11,050 m^3/s was prescribed at the inflow cross-section. At the

centre cell of the downstream boundary a constant water depth of 233.65m above sea level was enforced; the model allows the surface to move at all other cells near the outflow. The free water surface was modelled as a symmetry boundary, and the bed roughness was once again subject to calibration. After several calibration runs, a Strickler roughness parameter of 35.0 was found to give the optimum water surface position at river station 2081.0. For the floodplain roughness a value of 1000mm was used as discussed above.

6.2.3 FLUENT

The third software package used to simulate the flow conditions in the river Danube near the municipality of Grein was Fluent (see chapter 2.2.8). While RSim-3D and SSIIM were run on a regular PC with a clock rate of 2.8 GHz and 1GB of RAM, Fluent was installed on an Alpha workstation cluster at the computing centre of Vienna University of Technology. The comparison of computation times, as it is done in the next subsection of this work, can therefore only be seen as a guideline, as far as the Fluent model is concerned.

Grid generation for Fluent is done with a software called Gambit. However, it must be pointed out that this software is mainly of use for applications in mechanical engineering where the computational domain is bounded by pipes and other structures that follow clear geometrical rules. Creating the grid for a natural channel with a complicated bed geometry and a free water surface turned out to be an extremely time-consuming task. Fluent can operate both on structured and unstructured grids, but will generally produce results much faster when supplied a structured grid. Therefore it was decided to set up a structured grid similar to the one used by SSIIM. The bed geometry was imported from SSIIM, side walls were constructed manually in regions where the initial water level was located at a higher elevation than bed data points existed, and finally the resulting volume was decomposed into 30,000 cells following the pattern discussed in previous subsection.

While both SSIIM and RSim-3D use the pressure equation to relate pressure to changes in the water surface elevation for every grid region, Fluent does not provide such an algorithm. Instead, it uses the VOF (volume of fluid) method to predict the position of a surface that separates two phases (air and water in the case of a river). Unfortunately, this method requires the existence of separate inlets for both phases which must be filled entirely by either phase. For the position of the phase boundary is constantly on the move, the only way to simulate a water inlet that is entirely surrounded by water is by making water enter the computational domain through the river bed long before the actual area of interest (*Krouzecky* (2002) [39]). This procedure was

found to require long approach channels before natural flow conditions are obtained, furthermore it needs computation times of approximately three weeks for a typical river stretch, as the one in the present work, to result in a converged solution (*Krouzecky* (2002) [39]). Due to these severe limitations in the handling of the VOF method it was decided to employ a different technique: an initial grid was constructed using both bed geometry and surface shape obtained from a converged run of the SSIIM model. Then, Fluent was run using this grid, and after obtaining a converged solution, the pressure at the water surface was evaluated and translated into changes in the elevation level of the corresponding grid regions. This modified surface was again run through the Gambit grid generator before the whole process was repeated. It turned out that after three iterations of this kind the pressure difference was below 100 Pa in every single surface cell, corresponding to an accuracy of the resulting surface of 1cm.

The handling of the free surface was not the only problem encountered during the simulation runs using the Fluent model: regions with low water depth exhibited instabilities as far as turbulent kinetic energy and dissipation were concerned, leading to model divergence. Problems of this kind have also been reported by *Hodskinson* (1996) [36] and *Nicholas & Sambrook Smith* (1999) [54]. *Nicholas* (2001) [52] notes that these difficulties result from the existence of an upper limit on the roughness height k_s for a given near-bed cell thickness: for the Fluent model, k_s should not exceed the distance to the centroid of the near-bed grid cell. The implication of this is that the thickness of the near-bed cell limits the maximum shear velocity at the bed, so that near-bed velocities may be overpredicted in field situations involving high relative roughness (*ibid.*).

Similar problems were encountered in the RSim-3D model but could be resolved by deactivating regions with low water depths. Since such an option did not exist in the Fluent model, it was finally decided to reduce the model geometry by excluding the floodplains and using only the data points within the river bed. This procedure ensures that low water depths cannot occur, hence avoiding instabilities of the kind observed. However, additional errors are introduced into the simulation and so the Fluent results are not directly comparable to those obtained from other simulation models.

The boundary conditions employed for Fluent were as follows: an inlet boundary of type *mass flow inlet* was used, prescribing $1.105 \cdot 10^7$kg/s as mass flow. This type of inlet ensures that the inlet velocity is automatically adjusted for every new grid fed into the model. The usual zero-gradient outflow condition was used at the outlet, and symmetry conditions were prescribed at the water surface. The wall roughness was calibrated in the usual way, with a roughness height of 0.20m yielding the correct water surface position at river station 2078.0 – since Fluent does not enforce a given downstream water surface elevation, the model must be calibrated by prescribing

the water surface at the inlet and using the outlet as monitoring location. Using equation 4.94, the roughness height of 0.20m translates into a Strickler coefficient of 34.5, which is perfectly in line with the parameters obtained by the other two models.

6.2.4 Comparison

The first parameter of relevance in the context of an engineering application is the result of the model calibration: the bed roughness. The values obtained for this parameter are summarised in table 6.2.

	RSim-3D	SSIIM	FLUENT
Bed roughness (1D calibration) [K_{St}]	30.0	31.0	–
Bed roughness (3D calibration) [K_{St}]	35.0	35.0	34.5
Floodplain roughness [m]	1.0	1.0	–

Table 6.2: Roughness values as result of model calibration

From this table it can be seen that there are no significant differences in the bed roughness obtained by a 3D model calibration. This is due to the fact that the Strickler coefficient is translated into a roughness height by both RSim-3D and SSIIM – Fluent directly operates on a roughness height – and that the roughness enters the momentum equations by similar equations in all three models. Both RSim-3D and SSIIM also offer the possibility to obtain an initial guess for the water surface elevation by running a 1D backwater computation. Again, there is no big difference between the roughness parameters obtained on the two models, but it becomes clear that a significant difference exists between 1D and 3D model calibrations; the Strickler coefficient obtained by 1D calibration is lower than in 3D. The main reason for this difference lies in the methodology that is used to take bed roughness into account in different model dimensions: in one-dimensional computations the roughness coefficient also covers cross-sectional effects, i.e. the influence of secondary motion, while it is purely a measure of actual surface roughness in three-dimensional calculations.

The time until a solution to a specific flow problem can be obtained and the effort, both in human and computational resources, which must be spent on the problem, is another issue of high relevance to the engineer. Therefore the time spent on distinct tasks while working on the project was measured and is summarised in table 6.3:

- *Grid generation*: the time needed to produce a computational grid from geospatial data,

and the time required for subsequent operations on the same grid (i.e. manual changes to reflect surface changes),

- *Model handling*: the time required to write input files or set the appropriate numerical parameters in a graphical user interface, and the time for user interaction during model calibration (many of these tasks are of the nature of trial-and-error),
- *Computation time*: the actual time the software runs to return a converged solution; it should be pointed out that due to the fact that the Fluent model was run on a different computer hardware architecture, it is not possible to draw conclusions about the solver efficiency from this value for that software.

	RSim-3D	SSIIM	FLUENT
Grid generation [h]	3.0	4.0	70.0
Model handling [h]	5.0	2.0	25.0
Computation time [h]	61.0	1.0	0.75

Table 6.3: Time spent on distinct modelling tasks

Three major conclusions can be drawn from the figures in table 6.3:

1. While grid generation can be done rather quickly in both RSim-3D and SSIIM, this is a time-consuming task in Fluent. As already mentioned, this is mostly due to the fact that no options for converting measured data points (bed and surface) into a three-dimensional grid exist in Fluent, so that many operations must be done manually, requiring a fair amount of user interaction. Additionally, some small bugs in the 3D grid generation routines of Gambit (Fluent's graphical grid generator) led to inconsistencies in the solver module later. The respective errors in the grid had to be found and corrected manually before a converged flow solution could be obtained. Of course, the advantage of RSim-3D and SSIIM in terms of grid generation is not only that these programs were actually written to work on measured real-world terrain data, but also the fact that the author has much more experience with these models than with the Fluent software. Nonetheless it can be said that the creation of a suitable grid featuring a river stretch is a time-consuming and complex task in Fluent.
2. The handling of the model itself does not require much interaction in SSIIM, a little more in RSim-3D and is most time-consuming in Fluent. As mentioned, the author has worked

with SSIIM at numerous occasions in the past (e.g. *Tritthart* (2000) [79], *Scheuerlein et al.* (2004) [70]) and so the experience with the software facilitates a short interaction time during the runs of the model. As author of the RSim-3D model, the same thing holds true; however, it must be said that RSim-3D is still "work in progress", hence it requires more interaction time to get some parameters right to avoid instabilities. As far as Fluent is concerned, the comparably long interaction time is a result of the way in which the free water surface was treated, but also – to a lesser amount – the lack of detailed experience in using this model.

3. Once a suitable grid is supplied to the model, both Fluent and SSIIM deliver results rather quickly while RSim-3D needs long periods of time until a state of convergence is reached. The reasons for this are manifold:

   - Solvers operating on structured grids deliver results significantly faster than those using unstructured grids. Structured grids do not require a table storing the position of every cell in the continuum and the connecting faces between the cells; all of these values are known implicitly by supplying the index of each cell. Furthermore, cells in hexahedral shape usually do not exhibit geometric distortion, facilitating a fast solver progress. Experience with unstructured grids using the Fluent solver (e.g. *Krouzecky* (2002) [39]) shows that computation times become extremely long using unstructured grids. However, the advantage in representing complex geometries exactly by using unstructured grids, is a decisive reason to favor this grid type despite its higher computational cost.
   - RSim-3D's solver is not using the fastest algorithm available. For smaller flow problems, it is reasonably fast, but for larger numbers of cells it becomes inefficient (see chapter 4.6). This may be a starting point for a possible improvement of the model in the future.
   - Relaxation factors significantly influence the time until convergence is reached. Due to instabilities in regions with low flow depths, RSim-3D had to use lower relaxation factors than SSIIM or Fluent. While the latter model even required the removal of all regions with low flow depth (i.e. floodplains) before reasonable relaxation factors could be used, SSIIM did not exhibit stability problems at all and operated using the standard relaxation factors.
   - The models employ different convergence criteria. SSIIM uses unscaled residuals while RSim-3D and Fluent employ scaled residuals (chapter 4.5.3). Depending on the

flow situation encountered, the latter can be a much stronger criterion. Furthermore, by default, the SSIIM model declares a solution converged as soon as the maximum of all residuals is below 10^{-3}, while RSim-3D uses 10^{-4}. Recalling figure 4.8 in chapter 4.6, a drop in residuals is linear on a logarithmic scale, i.e. it takes approximately the same number of iteration cycles to reduce the residuals from 10^{0} to 10^{-1}, as from 10^{-3} to 10^{-4}.

6.3 Results

6.3.1 Water surface

The resulting water surface elevation for the discharge with a return period of 100 years is depicted in figures 6.5, 6.7 and 6.8 for the RSim-3D, SSIIM and Fluent models, respectively. Figure 6.6 visualises the water depth within the project domain. It should be noted that the first stream kilometer from km 2082.0 to 2081.0 was only modelled to obtain natural flow conditions in the reach thereafter; hence, depending on the model used, the water surface elevation in this first kilometer exhibits reasonable differences. However, after approximately 1.5 kilometers all models show pretty much the same water surface pattern with a significant rise at the outer bank near the municipality of Grein and a drop in water surface elevations at the opposite bank. The water surfaced obtained from the SSIIM model appears smoother than the one computed with RSim-3D, which can be attributed to the different terrain interpolation techniques employed in these models. This is also the obvious reason for the slightly higher extrema in figure 6.5.

Some spots near the left bank in figure 6.5 appear red (high altitude) or blue (low altitude). This is because RSim-3D displays inactive grid regions along with the active ones, and so the red spots are just dry areas with an altitude higher than the maximum water surface elevation, while the blue spots are areas that have been automatically deactivated due to very low water depth.

When comparing the water surface obtained from Fluent (fig. 6.8) with the other figures, it can be seen that the general water surface pattern shows no significant differences, even though the floodplain regions were excluded from the computational domain. Only the absolute elevation above sea level is slightly lower, but this stems from the model having been calibrated from an upstream location, as opposed to a downstream location in RSim-3D and SSIIM.

Figure 6.9 shows the resulting water surface elevations at the left and right banks for all three simulation models along with the characteristic water levels (KWD). Since the KWD values are

Figure 6.5: Water surface in the Danube bend near Grein (RSim-3D model)

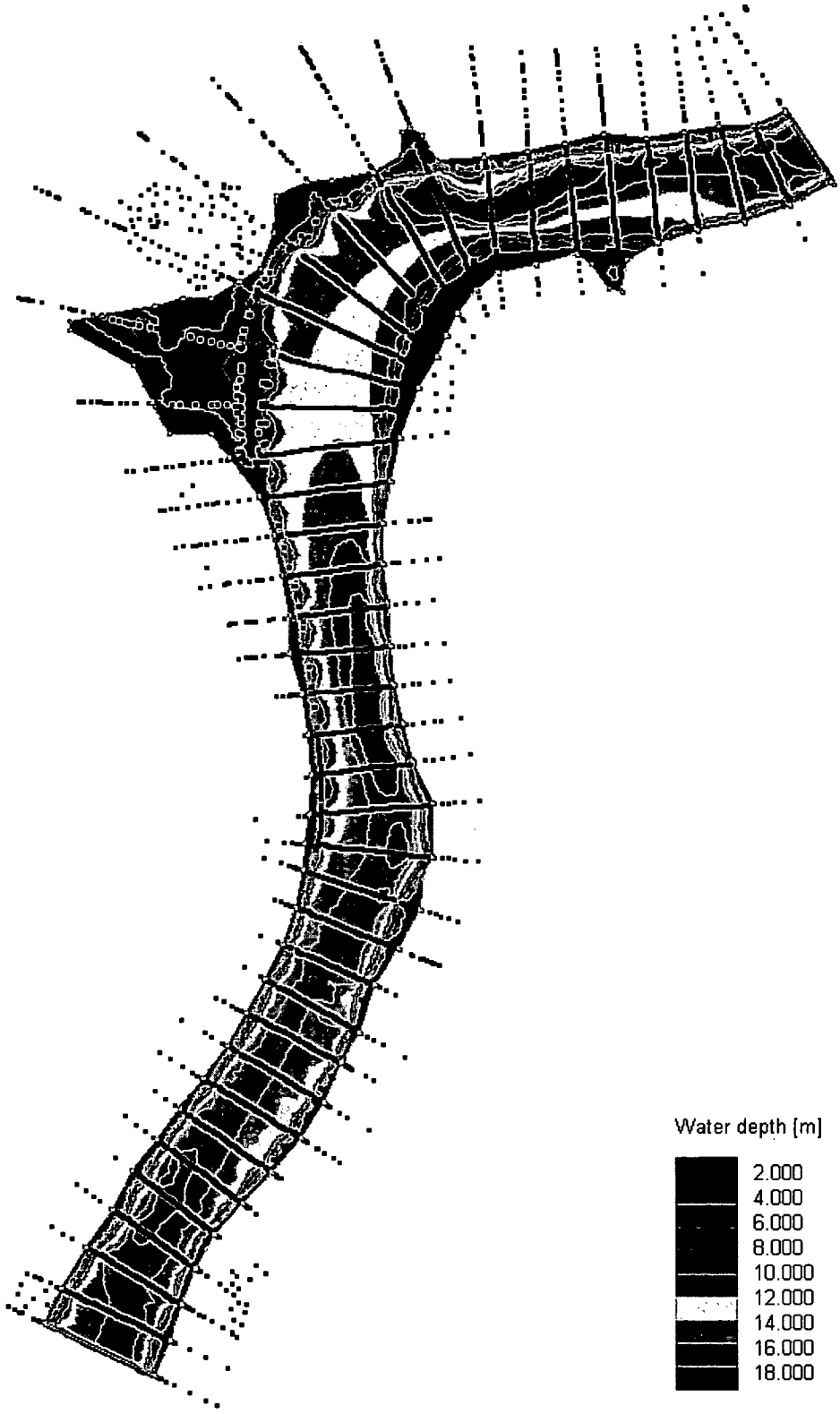

Figure 6.6: Water depth in the Danube bend near Grein (RSim-3D model)

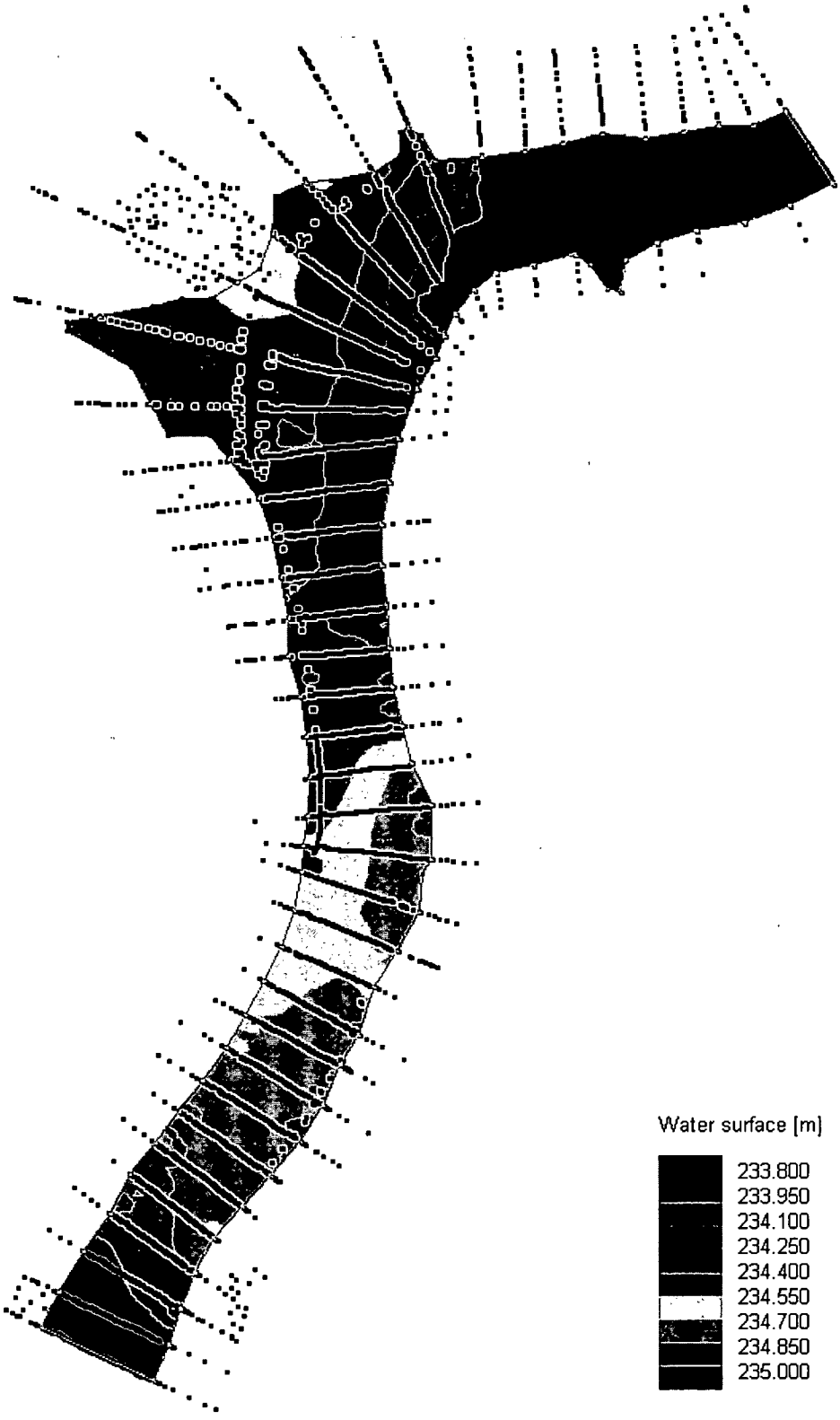

Figure 6.7: Water surface in the Danube bend near Grein (SSIIM model)

Figure 6.8: Water surface in the Danube bend near Grein (Fluent model)

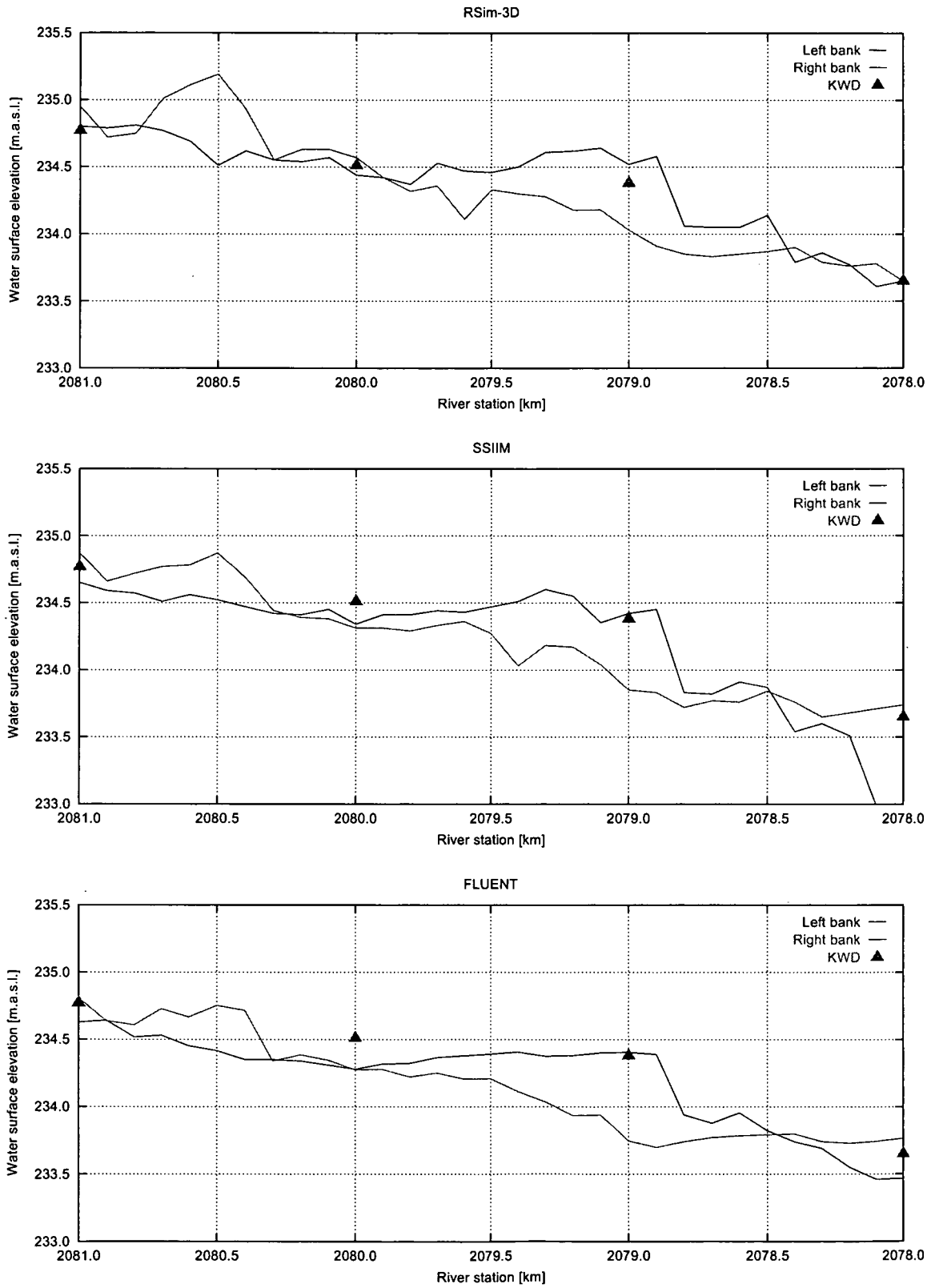

Figure 6.9: Water surface elevations along left and right banks

based on the *highest water levels* observed in August 2002, the computed water surfaces should always match these values at one bank while being lower or equal along the other bank. These characteristic values are now analysed one at a time:

- *km 2081.0*: RSim-3D gets the KWD right at the left bank, the water surface at the right bank is overpredicted by approximately 10cm. In the SSIIM model the KWD value lies right between the left and the right bank, with the latter being about 5cm too high. Fluent is the only model to exactly predict the KWD, but this is to no surprise since this location was used as monitoring location during model calibration.

- *km 2080.0*: In both Fluent and SSIIM, the water surface elevation at this river station is underestimated by 15cm (SSIIM) to 20cm (Fluent). RSim-3D performs very well at this location, getting the KWD approximately right.

- *km 2079.0*: This river station is located right near the municipality of Grein at the apex of the bend. RSim-3D overpredicts the water surface elevation by several centimetres while SSIIM and Fluent give the correct value. However, as far as the Fluent model is concerned, this result must be assessed very critically: considering the simplifications required to obtain a converged solution, the numerically correct value could be mere coincidence. RSim-3D's overprediction of the actual situation can be attributed to the different terrain interpolation technique employed in that software package.

- *km 2078.0*: Used as monitoring location in RSim-3D and SSIIM, it is no surprise that these two models perform slightly better than Fluent. However, while the downstream water surface elevation is enforced over the whole cross-section width in RSim-3D, SSIIM uses only one reference cell for this purpose. Since this reference cell is usually placed right at the centre line of the river, the water surface is free to move at both banks. It can be seen that SSIIM predicts a significant drop in water levels along the left bank. Since the same bed elevations were used for SSIIM and Fluent – and the latter model does not exhibit this water surface minimum – a wrongly interpolated channel bed can be excluded from the list of possible reasons to cause this effect.

It remains to analyse the maximum difference in water levels between left and right bank near the village of Grein. The introductory section of this chapter already discussed the fact that during the flood event of August 2002 a difference of 80cm was observed. Table 6.4 summarises the simulation results for the cross-section at km 2078.9, where the largest differences were

computed by all models. It can be seen that none of the models yields a difference of 80cm, but both RSim-3D and Fluent exhibit differences close to 70cm – a reasonably good agreement with the observations.

	RSim-3D	SSIIM	Fluent
Difference left-right bank	68cm	62cm	69cm

Table 6.4: Differences in water surface elevations at river station 2078.9

6.3.2 Depth-averaged flow velocities

Figures 6.10 through 6.12 depict the depth-averaged flow velocities for the three simulation models. Again, the overall flow pattern is very similar for all models. The most significant difference between RSim-3D and SSIIM on one hand and Fluent on the other is that the velocity pattern in the flow domain appears much smoother in the output of Fluent. This is due to the excluded floodplains and the terrain data points that were not used in the bed interpolation algorithm. Comparing the output of RSim-3D and SSIIM it can be seen that the shape of the velocity distribution near the banks is slightly smoother in the latter model. This can be attributed to the different terrain interpolation methods.

Still, all three models agree on a maximum velocity magnitude between 3.6 and 4.0 m/s, with maxima to be found at the straight river reach near km 2080.0 and near the downstream model boundary at km 2078.0. The mean flow velocity slows down to some extent while passing by the municipality of Grein where the inundated terrain enlarges the river cross-section; this effect cannot be seen in the Fluent model output since the floodplains were not modelled.

An area of specific interest is the inundation area between the cross-sections of km 2079.6 and km 2079.3. It is depicted in detail in figure 6.13 (output of RSim-3D). It can be seen that an area of recirculating flow with a velocity magnitude of 0.5 to 1.0 m/s evolves which is rotating counter-clockwise being excited by the main flow. The same recirculating flow pattern is predicted by the SSIIM model, as well.

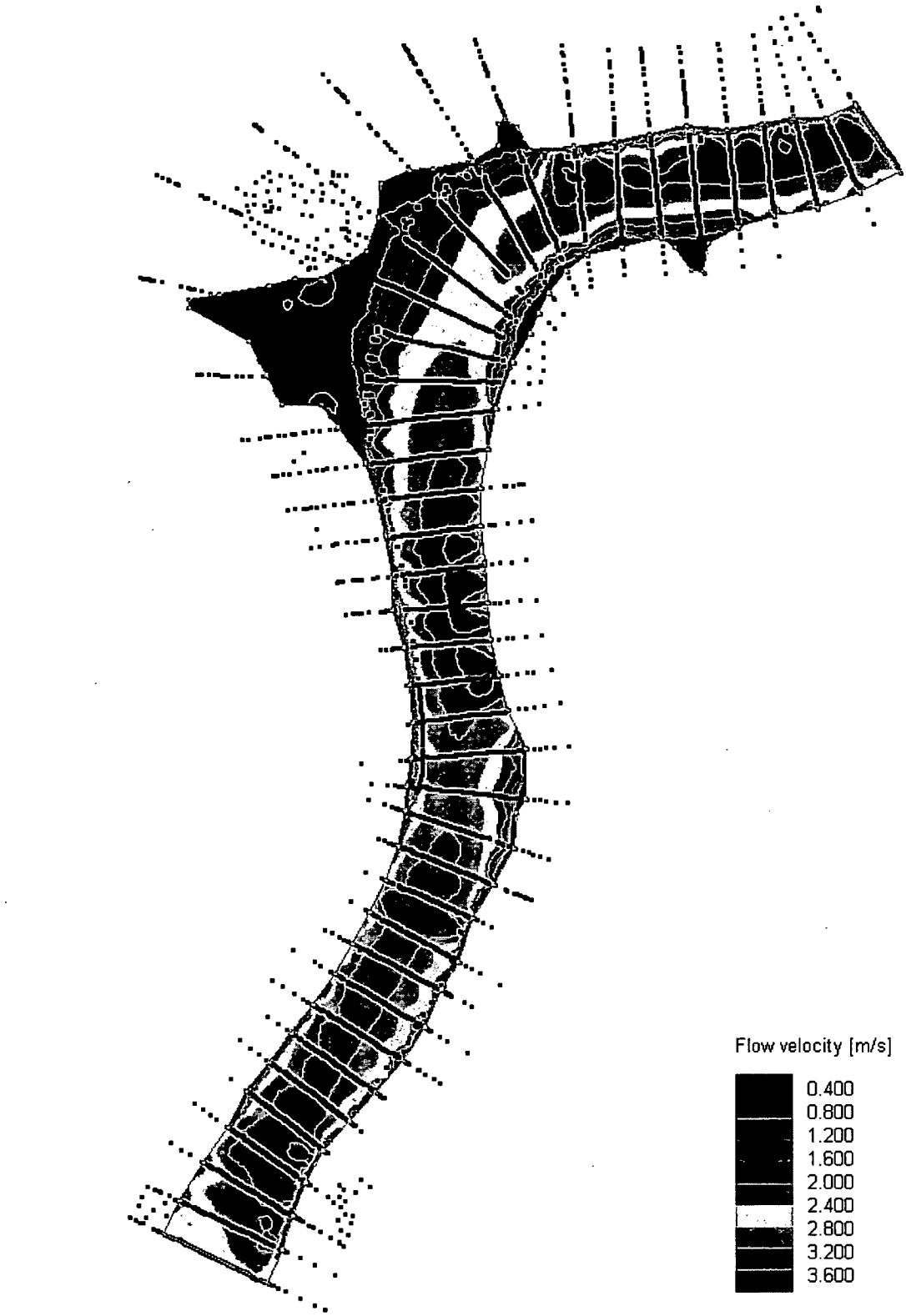

Figure 6.10: Depth-averaged flow velocity in the Danube bend near Grein (RSim-3D model)

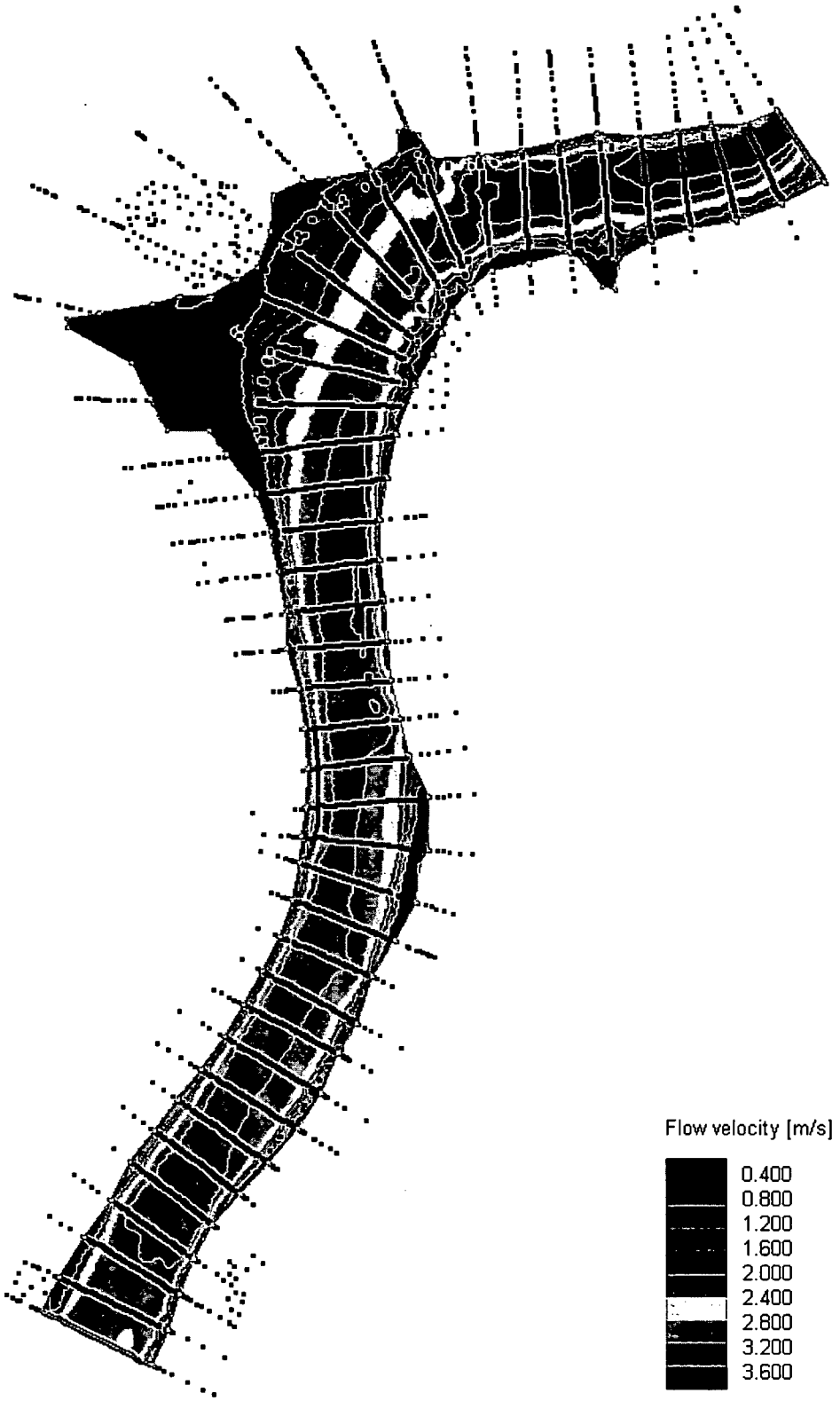

Figure 6.11: Depth-averaged flow velocity in the Danube bend near Grein (SSIIM model)

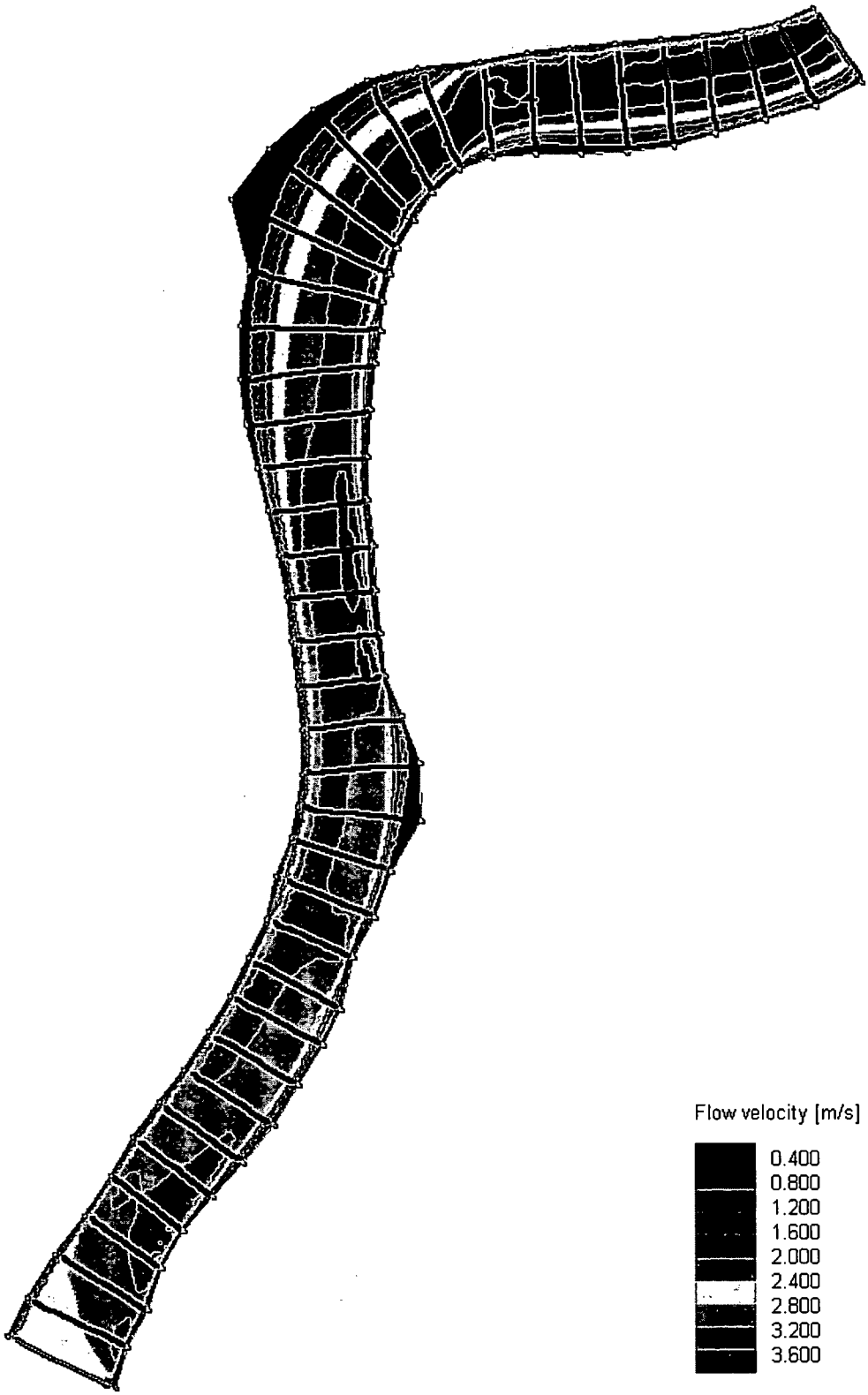

Figure 6.12: Depth-averaged flow velocity in the Danube bend near Grein (Fluent model)

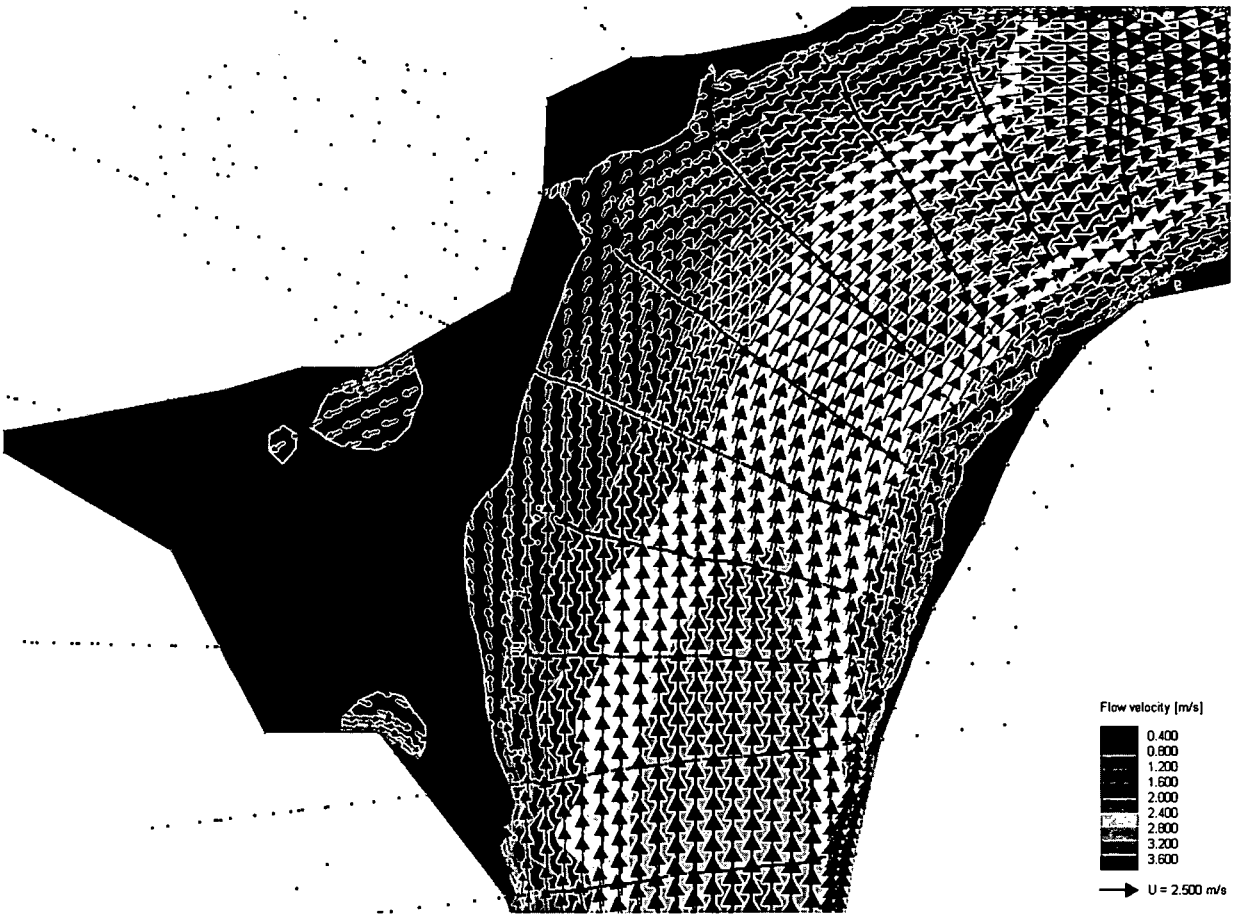

Figure 6.13: Flow detail in the inundation area near km 2079.5

6.3.3 Secondary flow

The results of the three models exhibit little to no differences between the secondary flow patterns. Therefore only the output of RSim-3D is depicted in this section. Figures 6.14 and 6.15 show cross-sections for every 200m between river stations 2079.8 and 2078.2, additionally the cross-section at km 2078.9 is depicted where the maximum difference in water surface elevations between left and right bank was encountered. All figures are scaled by the factor 2.0 in the vertical direction. In order to preserve the same velocity vector scale for all cross-sections it was necessary to apply different geometric scales.

At cross-sections 2079.8 and 2079.6 the secondary movement points towards the right bank with a velocity magnitude of up to 0.6 m/s, a consequence of the beginning bend. Then, at river stations 2079.4 and 2079.2 the secondary movement changes its direction and points towards the

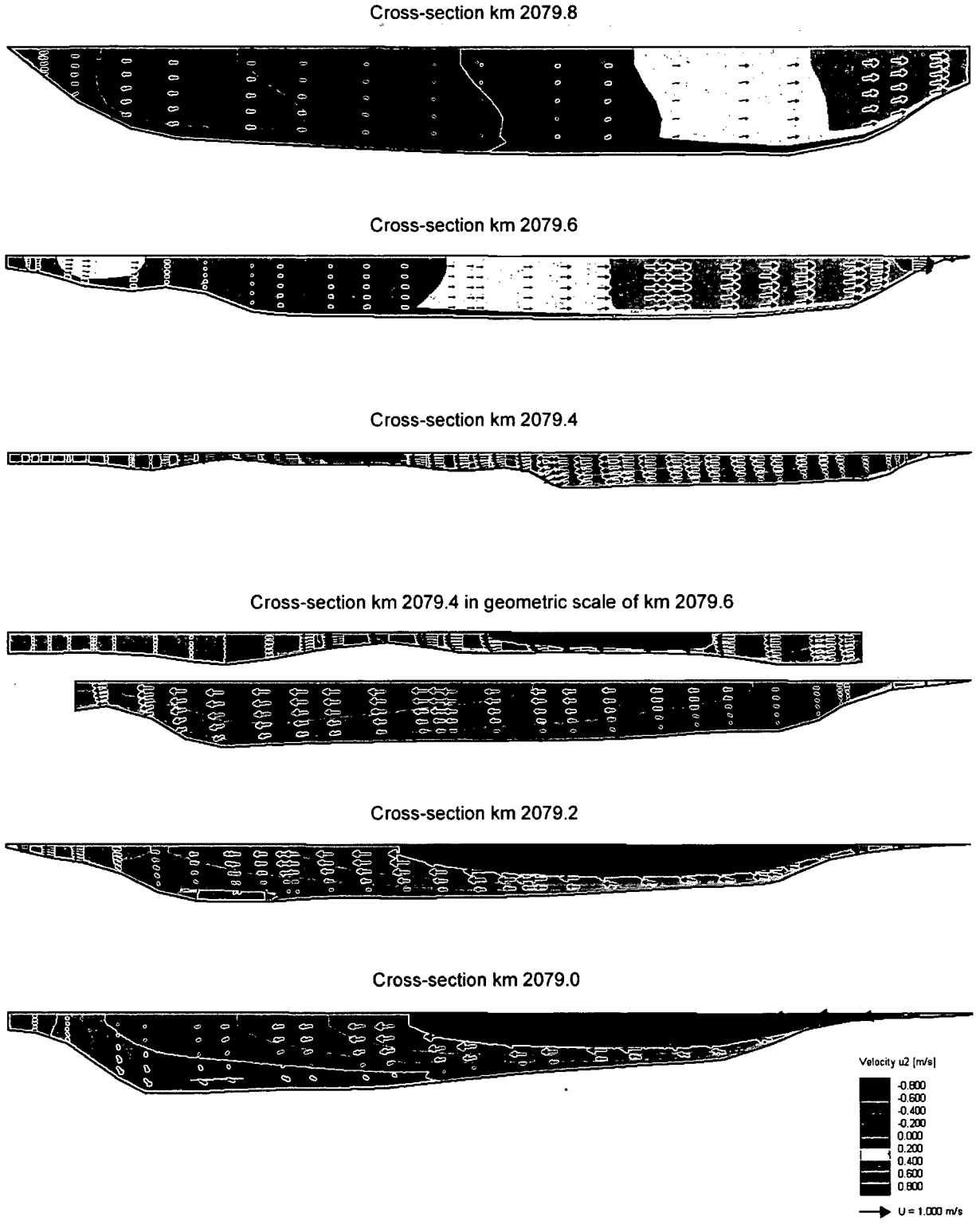

Figure 6.14: Secondary flow in cross-sections 2079.8 through 2079.0

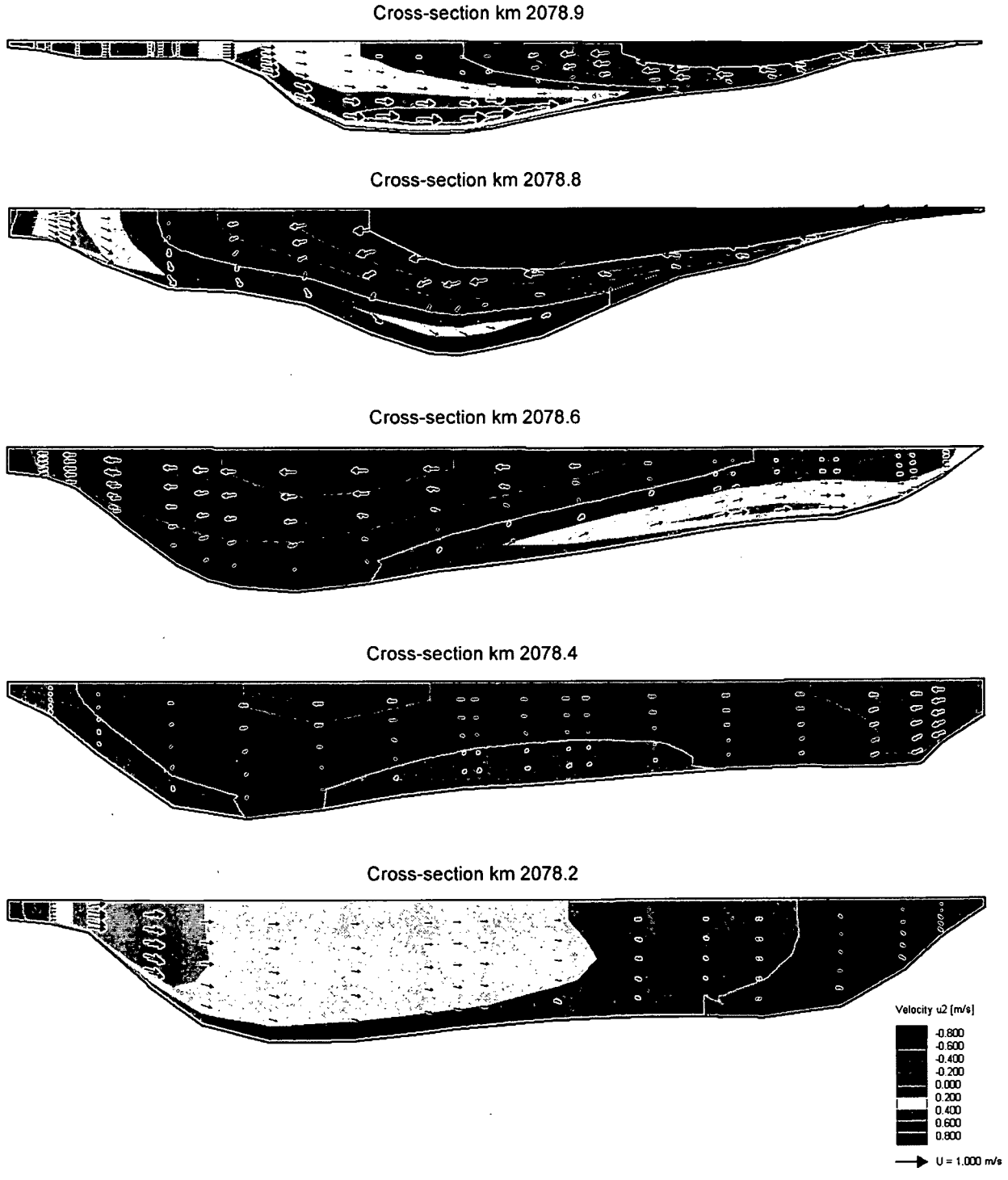

Figure 6.15: Secondary flow in cross-sections 2078.9 through 2078.2

left bank with velocity magnitudes of up to 0.8 m/s. This is caused by the flow from the river bed into the direction of the inundated terrain within the village of Grein. The first really distinct secondary flow pattern evolves at cross-section 2079.0, pointing towards the left (outer) bank near the water surface and into the opposite direction close to the bed. This pattern continues throughout cross-sections 2078.9 and 2078.8 with velocity magnitudes of up to 0.8 m/s. Cross-sections 2078.6 and 2078.4 still exhibit a distinct secondary flow pattern, and the velocity magnitude falls below 0.4 m/s. At km 2078.2 this pattern has disappeared, and the movement points towards the right bank in the entire cross-section.

6.4 Summary

A study of the flow conditions during the flood event of August 2002 in the Danube river bend near the municipality of Grein in Upper Austria was performed using three different simulation models: RSim-3D, SSIIM and Fluent. For the RSim-3D model an unstructured polyhedral computation grid based on hexagonal grid regions was used while the other two models employed structured grids consisting of hexahedra. It became apparent that the models using structured grids exhibit significant advantages in terms of computation time required to obtain a converged solution; however, this advantage is bought by geometry simplifications, imposing restrictions on the shape of the polygon bounding the project domain. On the other hand, it was found that a lot of manual work is required to build a grid for the Fluent model – a generic flow simulation code –, a task which is easier to perform in the typical river simulation models.

As far as results of the flow simulation are concerned, the shape of the computed water surface, the depth-averaged flow velocities and secondary flow patterns were evaluated. No significant differences were encountered in the computed water surfaces, even though the maxima computed by the RSim-3D model were slightly higher than those obtained through the other software packages, resulting in a closer match with the observations of August 2002 in certain places. However, this can either be attributed to the polygonal cell shapes, the different bed elevation interpolation technique employed in that model or a combination of both. All simulation results show the characteristic pattern of a rise in water surface elevations along the outer bank and a drop along the inner bank, even though the computed maximum difference of almost 70cm falls slightly short of the 80cm observed during the flood event.

In terms of the depth-averaged flow velocities, all three models agree on a maximum velocity magnitude between 3.6 and 4.0 m/s. The output of the Fluent model appears somewhat smoother

which is to be attributed to the reduced simulation domain employed for building the grid since areas of low water depth in the inundated regions led to instabilities during the simulation runs, requiring these regions to be removed from the computational domain. Both the RSim-3D and SSIIM models show a large recirculation area near the left bank at the beginning of the river bend with velocity magnitudes of up to 1.0 m/s. Finally taking a closer look at the secondary flow patterns, it was found that a distinct secondary movement can be found throughout the bend, pointing towards the outer bank near the water surface and into the direction of the inner bank close to the bed. The secondary movement reaches velocity magnitudes of up to 0.8 m/s.

In a final assessment of the performance of the RSim-3D model compared to the other flow simulation models used in this study it can be summarised that the computational cost is higher, but most of this can be attributed to the unstructured grid approach. The results are approximately equal to that of other models, delivering both a realistic water surface and flow pattern throughout the entire project domain.

7 Conclusions and Future Work

7.1 Conclusions

In this thesis, a 3D river flow simulation model based on the Finite Volume Method and using unstructured computation grids consisting of polyhedral cells was derived and implemented along with a software tool for pre- and post-processing tasks. The simulation model uses the Second Order Upwind scheme for the discretisation of convective terms in the Reynolds-Averaged Navier-Stokes equations, the SIMPLE method to couple the unknown pressure and velocity fields in the governing equations, and the standard $k - \varepsilon$ model for turbulence closure. The position of the free water surface is determined by evaluating the computed pressure at the water surface.

The simulation model was validated against laboratory data using four selected channel flow cases before it was compared to other numerical codes by applying it to a reach of the river Danube near the municipality of Grein in Austria, analysing the flow conditions during the flood event of August 2002 when a discharge with a return period of 100 years was encountered.

Purpose of the validation study was also to assess the difference between using polyhedral and the hexahedral cell shapes normally employed. It was found that in some cases the simulation results were closer to the observed values using polyhedral cells while other cases showed no significant differences to using hexahedral cells. When the polyhedral cells were arranged in the flow domain such that the resulting grid was rather coarse and the prevailing flow direction was not perpendicular to any of the cells' faces, some numerical diffusion was observed, even though not severe. This means that the problem of the flow solution depending on the exact arrangement of cells is not entirely solved, but polyhedral cells are capable of reducing its severity. A reasonable level of grid refinement can provide a remedy to the problem.

In the practical application of the model to a real flow situation it was found that the model based on polyhedral cells yields results of equal or higher quality – using the deviation from observed values as yardstick – than comparable models. However, since the usage of different terrain interpolation methods has a significant influence on the results obtained, the exact reason

for the model's advantage in some distinct places within the project domain cannot be clarified completely.

It was found that the implemented model, using a polyhedral grid approach, requires relatively higher computation times to obtain a converged solution than comparable models. A significant portion of this time is attributed to the unstructured grid which needs significantly more computational effort than its structured counterparts, but the general formulation of the discretised governing equations and the Gauss-Seidel solver algorithm also have their share in being computationally demanding.

On the other hand, many flow problems in complex bounded domains need to be simplified before they can be computed by making use of a solver based on a structured grid. The polyhedral grid approach, however, does not come with such restrictions. Furthermore the polyhedral cell shapes do not become distorted easily, not even when bounding polygons of very complicated shape are being used, which is another significant advantage of this modelling technique.

7.2 Future Work

The numerical model is perfectly operational using the formulae and algorithms presented in this thesis. However, of practical relevance is definitely the efficiency of the solver algorithm as it directly translates into the computational effort required to solve a particular problem. It is clear that this could be a starting point for potential future improvements of the model. The efficiency of different algorithms in the context of a typical river flow situation could easily be assessed to find and subsequently implement the one that has been found to be optimal.

Furthermore, the simulation model is not capable of dealing with unsteady flow conditions at this moment. Its use is therefore restricted to steady flows or weakly unsteady conditions which can be treated by a variation of boundary conditions alone. Transient flow problems require the discretisation and implementation of additional terms in the governing equations. It is not particularly difficult to discretise these terms and add them to the numerical model; implementation, however, requires extensive additional testing to prove that mass is actually conserved over several time steps.

The problem of sediment transport in rivers, lakes and reservoirs will probably be the most important challenge for the hydraulic and water resources engineer in the 21st century. Since physical experiments in this field are usually expensive in terms of time and money, and also complicated if the variation of sediment grain diameters is required in a study, it is predictable

that the focus of such experiments will shift towards numerical modelling soon. Due to the modular design of the RSim-3D model developed in this work, it is not difficult to add further transport equations and their respective boundary conditions to the source code. No special considerations will be needed for the implementation of a mobile bed geometry since the same approaches can be used that allow the water surface to move freely in the vertical direction.

Another challenge for the hydraulic and water resources engineer is the analysis of water quality and the transport of pollutants in inland waters. While the water quality of many rivers in Europe has improved notably during the past decades, there is still need for improvement in a significant number of waterbodies. Furthermore the real challenge may not even lie in the restoration of polluted rivers but in keeping the high water quality standard of all others, predicting the spread of pollutants once a disaster has taken place. This issue can be addressed in a numerical model by implementing the appropriate transport equations which are actually well-known.

Finally, it should be pointed out that RSim-3D's visualisation options are limited to plan views and cross-sections. Even though transects can be defined between two arbitrary points, allowing the user to analyse every place within the flow domain, this may turn out to be insufficient when the results of a flow simulation are going to be presented to the general public. Visualisation tools in three spatial dimensions, like particle tracking or streamline contours, are available which are capable of bridging the gap between the engineer and the general public. It would be very interesting, though also very challenging, to couple the simulation model with one of these visualisation techniques in the future.

Bibliography

[1] AEA TECHNOLOGY ENGINEERING SOFTWARE LTD: *CFX-5 User Documentation Version 5.5.1.* Waterloo, Ontario, Canada, 2002.

[2] AKIMA, H.: *Algorithm 526: Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Points.* ACM Transactions on Mathematical Software, 4(2):160–164, 1978.

[3] AKIMA, H.: *A Method of Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Points.* ACM Transactions on Mathematical Software, 4(2):148–159, 1978.

[4] AMMER, M.: *Finite-Element-Modellierung dreidimensionaler Strömungen mit freier Oberfläche.* PhD thesis, TU München, 1993.

[5] ANDERSON, M.G. (ED.): *Special Issue: The TELEMAC Modelling System.* Hydrological Processes, 14(13):2207–2364, 2000.

[6] APSLEY, D.: *CFD Lecture Notes.* Class Notes, Univ. Manchester, 2003.

[7] ARCEMENT, G.J. and V.R. SCHNEIDER: *Guide for Selecting Manning's Roughness Coefficients for Natural Channels and Flood Plains.* Technical Report, USGS, 2003.

[8] ARMFIELD, S., S. NORRIS, P. MORGAN and R. STREET: *A Parallel Non-Staggered Navier-Stokes Solver Implemented on a Workstation Cluster.* In ARMFIELD, S., P. MORGAN and K. SRINIVAS (editors): *Computational Fluid Dynamics 2002*, pages 30–45. Springer, 2003.

[9] BARRETT, R., M. BERRY, T.F. CHAN, J. DEMMEL, J. DONATO, J. DONGARRA, V. EIJKHOUT, R. POZO, C. ROMINE and H. VAN DER VORST: *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, PA, USA, 2nd edition, 1994.

[10] BARTH, T.J. and D.C. JESPERSEN: *The Design and Application of Upwind Schemes on Unstructured Meshes*. In *Proc. 27th Aerospace Sciences Meeting*, Reno, Nevada, USA, 1989.

[11] BOURKE, P.: *CONREC: A Contouring Subroutine*. Byte, pages 143–150, 1987.

[12] BRADBROOK, K.F., S.N. LANE and K.S. RICHARDS: *Numerical Simulation of Three-Dimensional, Time-Averaged Flow Structure at River Channel Confluences*. Water Resources Res., 36(9):2731–2746, 2000.

[13] BUNDESAMT F. EICH- U. VERMESSUNGSWESEN: *Aerial Views of the Flood of August 2002 in Austria*, 2002.

[14] CRESWELL, R. and P. CROAKER: *The Practical Application of Polyhedral Finite Volume Methodology to Problems with Large Scale Discrepancies*. In ARMFIELD, S., P. MORGAN and K. SRINIVAS (editors): *Computational Fluid Dynamics 2002*, pages 783–784. Springer, 2003.

[15] CUK, R.: *Construction of Voronoi Diagrams Using Fortune's Method: A Look on an Implementation*. In *Proc. Central European Seminar on Computer Graphics*, Budmerice, Slovakia, 1999.

[16] DAVIDSON, L.: *A Pressure Correction Method for Unstructured Meshes with Arbitrary Control Volumes*. Int. J. Numer. Methods Fluids, 22:265–281, 1996.

[17] DAVIDSON, L. and P.V. NIELSEN: *Calculation of the Two-Dimensional Airflow in Facial Regions and Nasal Cavity Using an Unstructured Finite Volume Solver*. Rept., Chalmers University of Technology, Gothenburg, Sweden, 1995.

[18] DAVIDSON, L. and L. STOLCIS: *An Efficient and Stable Solution Procedure of Compressible Turbulent Flow on General Unstructured Meshes Using Transport Turbulence Models*. In *Proc. 33rd Aerospace Sciences Meeting*, Reno, Nevada, USA, 1995.

[19] ERCOFTAC: *NEXUS Database, Classic Collection.* Internet: <http://cfd.me.umist.ac.uk/ercoftac>, European Research Community on Flow, Turbulence and Combustion, Univ. Manchester, UK, 1995.

[20] FERZIGER, J.H. and M. PERIC: *Computational Methods for Fluid Dynamics.* Springer, Berlin, 3rd edition, 2002.

[21] FEURICH, R.: *Untersuchung der Strömungsverhältnisse in einem doppelt gekrümmten Gerinne.* PhD thesis, Univ. Innsbruck, 2002.

[22] FEURICH, R. and F. SCHÖBERL: *Flow Field Estimation in a S-Shaped Channel by High Resolution Measurements Combined with 3D Numerical Simulation.* In *Proc. XXX IAHR Congress*, volume C-I, pages 733–740, Thessaloniki, Greece, 2003.

[23] FISCHER-ANTZE, T., T. STOESSER, P. BATES and N.R.B. OLSEN: *3D Numerical Modelling of Open-Channel Flow with Submerged Vegetation.* J. Hydr. Res., 39(3):303–310, 2001.

[24] FLUENT INC.: *FIDAP 8 Theory Manual.* Lebanon, New Hampshire, USA, 1998.

[25] FLUENT INC.: *FLUENT 6.0 User's Guide.* Lebanon, New Hampshire, USA, 2003.

[26] FORTUNE, S.: *A Sweepline Algorithm for Voronoi Diagrams.* In *Proc. Symposium on Computational Geometry*, pages 312–322, Yorktown Heights, NY, USA, 1986.

[27] FORTUNE, S.: *Voronoi Diagrams and Delaunay Triangulations.* In *Computing in Euclidean Geometry*, pages 193–223. World Scientific Publ., 1992.

[28] FRANK, A.: *GIS Theory.* Class Notes, Inst. f. Geoinformation und Landesvermessung, TU Vienna, 2002.

[29] GHAMRY, H.K. and P.M. STEFFLER: *Effect of Applying Different Distribution Shapes for Velocities and Pressure on Simulation of Curved Open Channels.* J. Hydr. Eng., 128(11):969–982, 2002.

[30] GOUDA, M., K. KARNER and R. TATSCHL: *Dam Flooding Simulation Using Advanced CFD Methods.* In *Proc. WCCM 5*, Vienna, Austria, 2002.

[31] GRIEBEL, M., T. DORNSEIFER and T. NEUNHOEFER: *Numerische Simulation in der Strömungsmechanik.* Vieweg, Wiesbaden, 1995.

[32] GUTKNECHT, D.: *Technische Hydraulik I.* Class Notes, TU Wien, Vienna, Austria, 2004.

[33] GUTKNECHT, D., CH. RESZLER and G. BLÖSCHL: *Jahrtausend-Hochwasser am Kamp?* Rept., Institute of Hydraulics, Hydrology and Water Resources Management, TU Wien, Vienna, Austria, 2002.

[34] HABERSACK, H.: *Analyse der Hochwasserereignisse vom August 2002 - FloodRisk, Synthesebericht.* Rept., Inst. f. Wasserwirtschaft, Hydrologie u. Konstruktiven Wasserbau, Univ. Bodenkultur, Vienna, Austria, 2004.

[35] HERVOUET, J.-M., J.-L. HUBERT, J.-M. JANIN, F. LEPEINTRE and E. PELTIER: *The Computation of Free Surface Flows with TELEMAC: An Example of Evolution Towards Hydroinformatics.* J. Hydr. Res., 32(1):45–64, 1994.

[36] HODSKINSON, A.: *Computational Fluid Dynamics as a Tool for Investigating Separated Flow in River Bends.* Earth Surf. Process. Landforms, 21(11):993–1000, 1996.

[37] KIM, W.J. and V.C. PATEL: *Origin and Decay of Longitudinal Vortices in Developing Flow in a Curved Rectangular Duct.* J. of Fluids Engineering, 116:45–52, 1994.

[38] KRIGE, D.G.: *A Statistical Approach to some Basic Mine Evaluation Problems on the Witwatersrand.* Journal of the Chem. and Metall. Soc. of South Africa, 52:119–139, 1951.

[39] KROUZECKY, N.: *Wasserspiegelhebung zufolge nicht überströmter Buhnen bei fester Sohle.* Habil., TU Wien, Vienna, Austria, 2002.

[40] LAUNDER, B.E. and D.B. SPALDING: *The Numerical Computation of Turbulent Flows.* Comp. Methods in App. Mech. and Eng., 3:269–289, 1974.

[41] LEONARD, B.P.: *A Stable and Accurate Convective Modelling Procedure Based on Quadratic Upstream Interpolation.* Comp. Methods in App. Mech. and Eng., 19:59–98, 1979.

[42] LESCHZINER, M.A. and W. RODI: *Calculation of Strongly Curved Open Channel Flow.* J. Hydraul. Div., Am. Soc. Civ. Eng., 105(10):1297–1314, 1979.

[43] LIEN, H.C., T.Y. HSIEH, J.C. YANG and K.C. YEH: *Bend-Flow Simulation using 2D Depth-Averaged Model.* J. Hydr. Res., 125(10):1097–1108, 1999.

[44] LUIJENDIK, A.P.: *Validation, Calibration and Evaluation of Delft3D-FLOW Model with Ferry Measurements.* Master's thesis, University of Utrecht, The Netherlands, 2001.

[45] MASON, D.C., D.M. COBBY, M.S. HORRITT and P.D. BATES: *Floodplain Friction Parameterization in Two-Dimensional River Flood Models Using Vegetation Heights Derived from Airborne Scanning Laser Altimetry.* Hydrol. Process., 17:1711–1732, 2003.

[46] MENTER, F.R.: *Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications.* AIAA Journal, 32(8):1598–1605, 1994.

[47] MILBRADT, P.: *Algorithmische Geometrie in der Bauinformatik.* Habil., Univ. Hannover, 2001.

[48] MILLER, R., A. ROULUND, B.M. SUMER, J. FREDSOE, C. TRUELSEN and J. MICHELSEN: *3-D Numerical Modeling of Flow Around a Groin.* In *Proc. XXX IAHR Congress*, volume C-II, pages 385–391, Thessaloniki, Greece, 2003.

[49] MÜNCH, O.: *Das Voronoi-Diagramm in der Airline-Metrik: Untersuchung der strukturellen Eigenschaften und Veranschaulichung durch ein Java-Programm.* Master's thesis, Fernuniversität Hagen, 1998.

[50] NAUDASCHER, E.: *Hydraulik der Gerinne und Gerinnebauwerke.* Springer, 1987.

[51] NGUYEN, V.TH.: *Zur dreidimensionalen Berechnung turbulenter Strömungen in Gerinnekrümmungen.* PhD thesis, Univ. Karlsruhe, 2000.

[52] NICHOLAS, A.P.: *Computational Fluid Dynamics Modelling of Boundary Roughness in Gravel-Bed Rivers: An Investigation of the Effects of Random Variability in Bed Elevation.* Earth Surf. Process. Landforms, 26(4):345–362, 2001.

[53] NICHOLAS, A.P. and C.A. MITCHELL: *Numerical Simulation of Overbank Processes in Topographically Complex Floodplain Environments.* Hydrological Processes, 17(4):727–746, 2003.

[54] NICHOLAS, A.P. and G.H. SAMBROOK SMITH: *Numerical Simulation of Three-Dimensional Flow Hydraulics in a Braided Channel.* Hydrological Processes, 13(6):913–929, 1999.

[55] OLSEN, N.R.B.: *Computational Fluid Dynamics in Hydraulic and Sedimentation Engineering.* Class Notes, Norwegian University of Science and Technology, Trondheim, Norway, 1999.

[56] OLSEN, N.R.B.: *CFD Modeling of Bed Changes During Flushing of a Reservoir.* In *Proc. Hydroinformatics 2000*, Iowa, USA, 2000.

[57] OLSEN, N.R.B.: *A Three-Dimensional Numerical Model for Simulation of Sediment Movements in Water Intakes with Multiblock Option.* User's Manual, Norwegian University of Science and Technology, Trondheim, Norway, 2000.

[58] OLSEN, N.R.B.: *Estimating Meandering Channel Evolution Using a 3D CFD Model.* In *Proc. Hydroinformatics 2002*, pages 52–57, Cardiff, UK, 2002.

[59] OLSEN, N.R.B. and H.M. KJELLESVIG: *Three-Dimensional Numerical Modeling of Bed Changes in a Sand Trap.* J. Hydr. Res., 37(2):189–198, 1999.

[60] OUILLON, S. and D. DARTUS: *Three-Dimensional Computation of Flow Around Groyne.* J. Hydr. Eng., 123(11):962–970, 1997.

[61] PATANKAR, S.V.: *Numerical Heat Transfer and Fluid Flow.* Taylor and Francis, Bristol, 1980.

[62] PATANKAR, S.V. and D.B. SPALDING: *A Calculation Procedure for Heat, Mass and Momentum Transfer in Three-Dimensional Parabolic Flows.* Int. J. Heat Mass Transfer, 15:1787–1806, 1972.

[63] PATEL, V.C., W. RODI and G. SCHEUERER: *Turbulence Models for Near-Wall and Low Reynolds Number Flows: A Review.* AIAA Journal, 23(9):1308–1319, 1985.

[64] PREMSTALLER, G.: *Application of Computational Fluid Dynamics to Scour Problems in Uniform Cohesionless Sediment.* Master's thesis, Univ. Innsbruck, 2002.

[65] PRESS, W.H.: *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, Cambridge, UK, 1987.

[66] RHIE, C.M. and W.L. CHOW: *Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation.* AIAA Journal, 21(11):1525–1532, 1983.

[67] RODI, W.: *The Prediction of Free Turbulent Boundary Layers by Use of a Two-Equation Model of Turbulence.* PhD thesis, University of London, 1972.

[68] RODI, W.: *Turbulence Models and their Application in Hydraulics.* State-of-the-art paper, International Association for Hydraulic Research, Karlsruhe, 1984.

[69] ROZOVSKII, I.L.: *Flow of Water in Bends of Open Channels.* Israel Program for Science Translation, Acad. Sci. Ukrainian S.S.R., 1961.

[70] SCHEUERLEIN, H., M. TRITTHART and F. NUNEZ GONZALEZ: *Numerical and Physical Modelling Concerning the Removal of Sediment Deposits from Reservoirs.* In *Proc. International Conference on Hydraulics of Dams and River Structures*, Tehran, Iran, 2004.

[71] SCHLICHTING, H. and K. GERSTEN: *Grenzschicht-Theorie.* Springer, Berlin, 9th edition, 1997.

[72] SCHMID, B.H.: *Computational Hydraulics and Hydrology.* Class Notes, Institute of Hydraulics, Hydrology and Water Resources Management, TU Vienna, 2001.

[73] SHEWCHUK, J.R.: *Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator.* In *First Workshop on Applied Computational Geometry, ACM*, pages 124–133, Philadelphia, USA, 1996.

[74] SOTIROPOULOS, F. and V.C. PATEL: *Flow in Curved Ducts of Varying Cross-Section.* IIHR Report No. 358, Institute of Hydraulic Research, University of Iowa, USA, 1992.

[75] SOTIROPOULOS, F. and V.C. PATEL: *Turbulence Anisotropy and Near-Wall Modeling in Predicting Three-Dimensional Shear-Flows.* AIAA Journal, 33(3):504–514, 1995.

[76] SPALDING, D.B.: *PHOENICS-Beginner's Guide and User Manual.* CHAM, London, UK, 1986.

[77] STANDINGFORD, D.W.F. and S.A. FORTH: *A Discrete Sensitivity Solver for an Industrial CFD Code Via Automatic Differentiation.* In ARMFIELD, S., P. MORGAN and K. SRINIVAS (editors): *Computational Fluid Dynamics 2002*, pages 82–87. Springer, 2003.

[78] STEINRÜCK, H.: *Grundlagen der Numerischen Strömungsmechanik.* Class Notes, Institute of Fluid Mechanics and Heat Transfer, TU Vienna, 2002.

[79] TRITTHART, M.: *Anwendung von dreidimensionalen numerischen Methoden beim Sedimentmanagement in Talsperrenreservoirs.* Master's thesis, University of Innsbruck, Austria, 2000.

[80] TRITTHART, M.: *Polyhedral Finite Volumes as Basis for Three-Dimensional Numerical Modelling of River Flow.* In *Proc. 6th International Conference on Hydroinformatics*, pages 324–331, Singapore, 2004.

[81] TRITTHART, M. and P. MILBRADT: *A First Analysis of the Flood Events of August 2002 in Lower Austria by Using a Hydrodynamic Model.* In *Proc. XXX IAHR Congress*, volume C-II, pages 111–118, Thessaloniki, Greece, 2003.

[82] TUREK, S.: *Efficient Solvers for Incompressible Flow Problems*, volume 6 of *Lecture Notes in Computational Science and Engineering.* Springer, Berlin, 1999.

[83] US ARMY CORPS OF ENGINEERS: *Engineering and Design – Hydraulic Design of Flood Control Channels.* Engineer Manual, USACE, Washington, D.C., USA, 1994.

[84] VERSTEEG, H.K. and W. MALALASEKERA: *An Introduction to Computational Fluid Dynamics: The Finite Volume Method.* Pearson Education, Harlow, 6th edition, 2001.

[85] VIERMETZ, M.: *Animation of Common Algorithms in Computational Geometry using Java.* Rept., Institut f. Informatik, University of München, Germany, 2001.

[86] VIGL, A.: *Typische Bogenfolgen alpiner Flüsse.* PhD thesis, University of Innsbruck, 1990.

[87] VIONNET, C.A., P.A. TASSI and J.P. MARTIN VIDE: *Estimates of Flow Resistance and Eddy Viscosity Coefficients for 2D Modelling on Vegetated Floodplains.* Hydrol. Process., 18:2907–2926, 2004.

[88] VORONOI, G.: *Nouvélles Applications Des Paramètres Continus À la Théorie Des Formes Quadratiques, Deuxième Mémoire, Recherches sur Les Parallelloedres Primitifs.* J. Reine Angew. Math., 134:198–287, 1908.

[89] WESSELING, P.: *Principles of Computational Fluid Dynamics.* Springer, Berlin, 2001.

[90] WILCOX, D.C.: *Turbulence Modeling for CFD.* Technical Report, DCW Industries, La Canada, California, USA, 1994.

[91] WILHELM, M.: *Generierung von Delaunay Triangulationen.* Master's thesis, Univ. Stuttgart, 2000.

[92] WU, W., W. RODI and T. WENKA: *3D Numerical Modeling of Flow and Sediment Transport in Open Channels.* J. Hydr. Eng., 126(1):4–15, 2000.

[93] YAKHOT, V. and S.A. ORSZAG: *Renormalization Group Analysis of Turbulence.* J. Sci. Comput., 1:3–51, 1986.

[94] ZENTRUM F. NATURGEFAHREN U. RISIKOMANAGEMENT: *Ereignisdokumentation Hochwasser August 2002.* Rept., ZENAR, Univ. f. Bodenkultur, Vienna, Austria, 2003.

[95] ZIENKIEWICZ, O.C.: *The Finite Element Method in Engineering Science.* McGraw-Hill, London, 2nd edition, 1971.

Appendix A Flow charts and examples

Illustration of region and cell numbering schemes

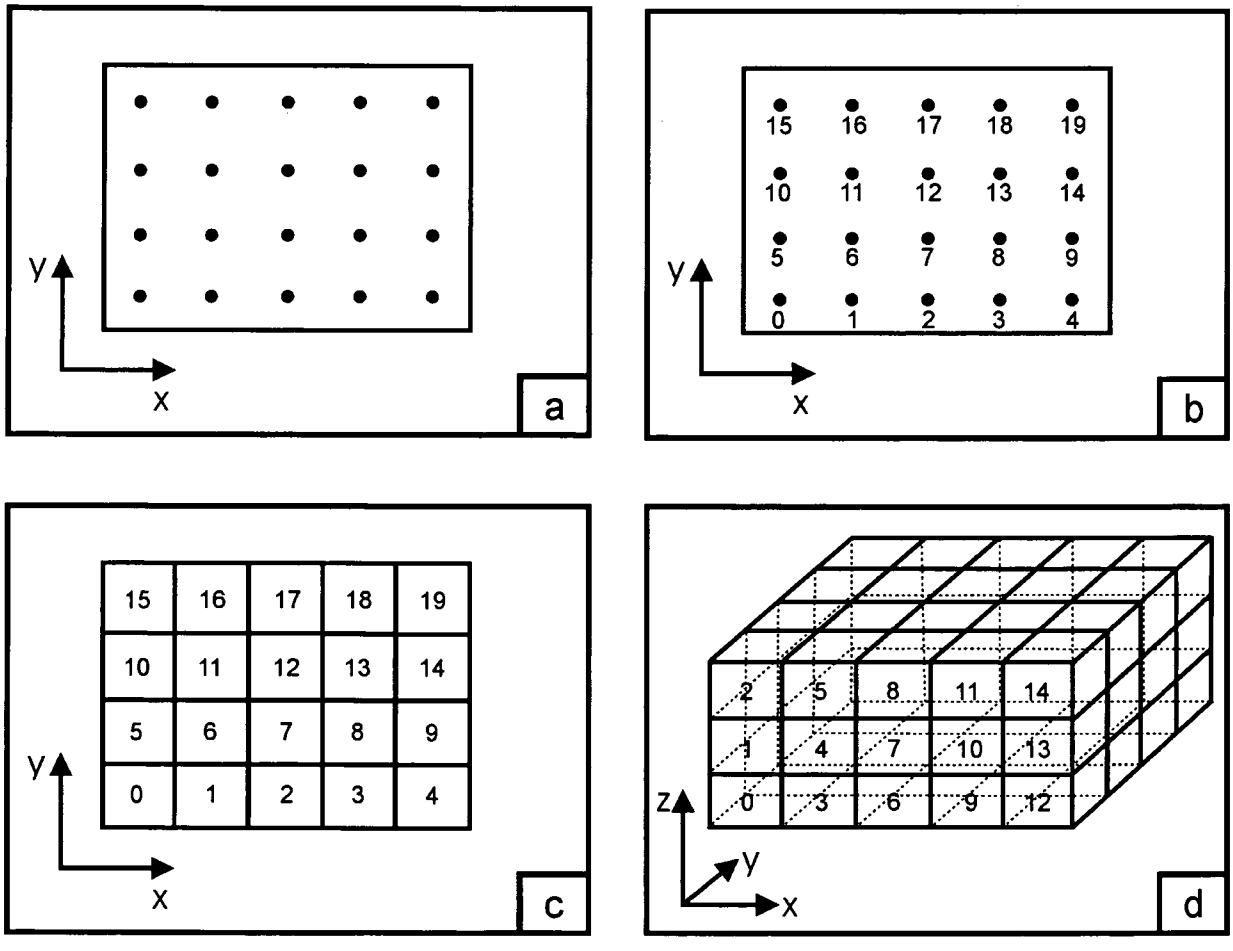

Figure A.1: Region and cell numbering schemes

Stages of the numbering process (fig. A.1)

a. *General situation:* Pattern of distributed base points within a bounding polygon.

b. *Sorting of points:* Fortune's algorithm requires the base points to be sorted/numbered in ascending order along the positive y-axis; points with equal distance along the y-axis are sorted along the positive x-axis.

c. *Numbering of 2D regions:* Two-dimensional grid regions receive the same identification number as the base point they belong to.

d. *Numbering of 3D cells:* Applying a structured vertical subdivision of each grid region into n cells, the cell number is derived from the region number by

$$c = n \cdot r + i$$

where c is the cell number, r the region number and i the vertical cell index in the range 0 to n-1. In order to store boundary conditions at the bed and the water surface, one extra cell is added on each side of the cell pile, therefore the internal value of n is equal to $(n_{\text{user}} + 2)$ where n_{user} is the user-supplied number of cells in each cell pile.

Illustrative example of the Kriging process

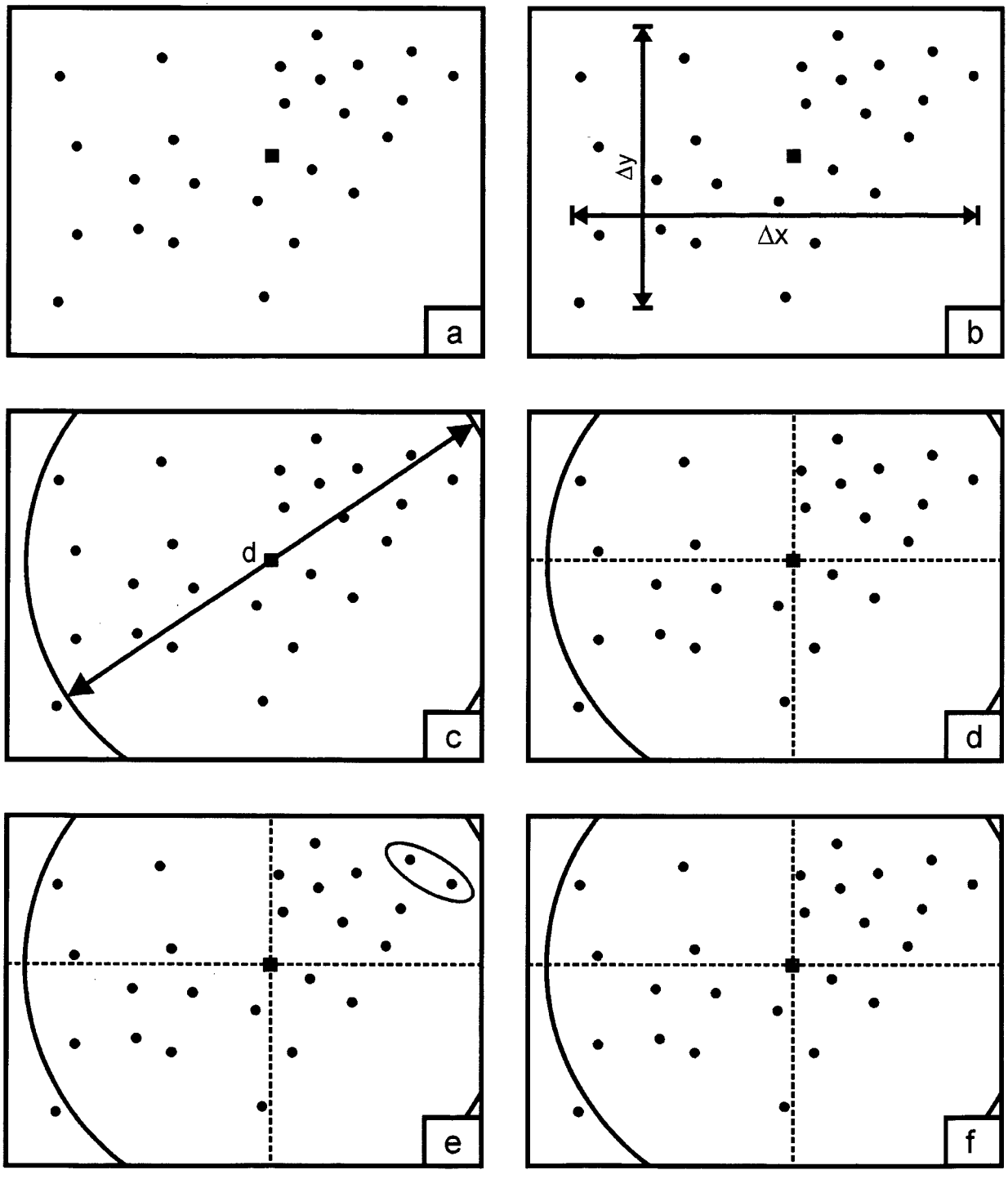

Figure A.2: Illustrative example of the Kriging process

Stages of the Kriging process (fig. A.2)

a. *General situation:* Distributed data points with known terrain elevation (black circles) surround a point with unknown terrain elevation (red box).

b. *Determine spatial dimensions:* The dimensions Δx and Δy of the terrain data set are computed.

c. *Apply search circle:* A circle with diameter $d = \sqrt{\Delta x^2 + \Delta y^2}$ is constructed on top of the point with unknown terrain elevation. Terrain data points lying outside of this circle are excluded from the data set.

d. *Sector subdivision:* The data set is subdivided into four sectors (quadrants).

e. *Sector search:* Between one and eight data points of every quadrant are used for the semi-variogram. If there are more than eight points available (as in the north-eastern quadrant), only the points with the shortest distance to the point with unknown terrain elevation stay in the data set; all other points (green ellipse) are excluded.

f. *Computation of semivariogram:* The semivariances – distances from every point to every other point – are computed for the remaining points in the data set (coloured in blue) before the linear equation set is solved as outlined in chapter 3.4.3.

In the computational implementation of the Kriging process these stages are executed in parallel as illustrated in figure A.3.

Flow chart of the Kriging algorithm

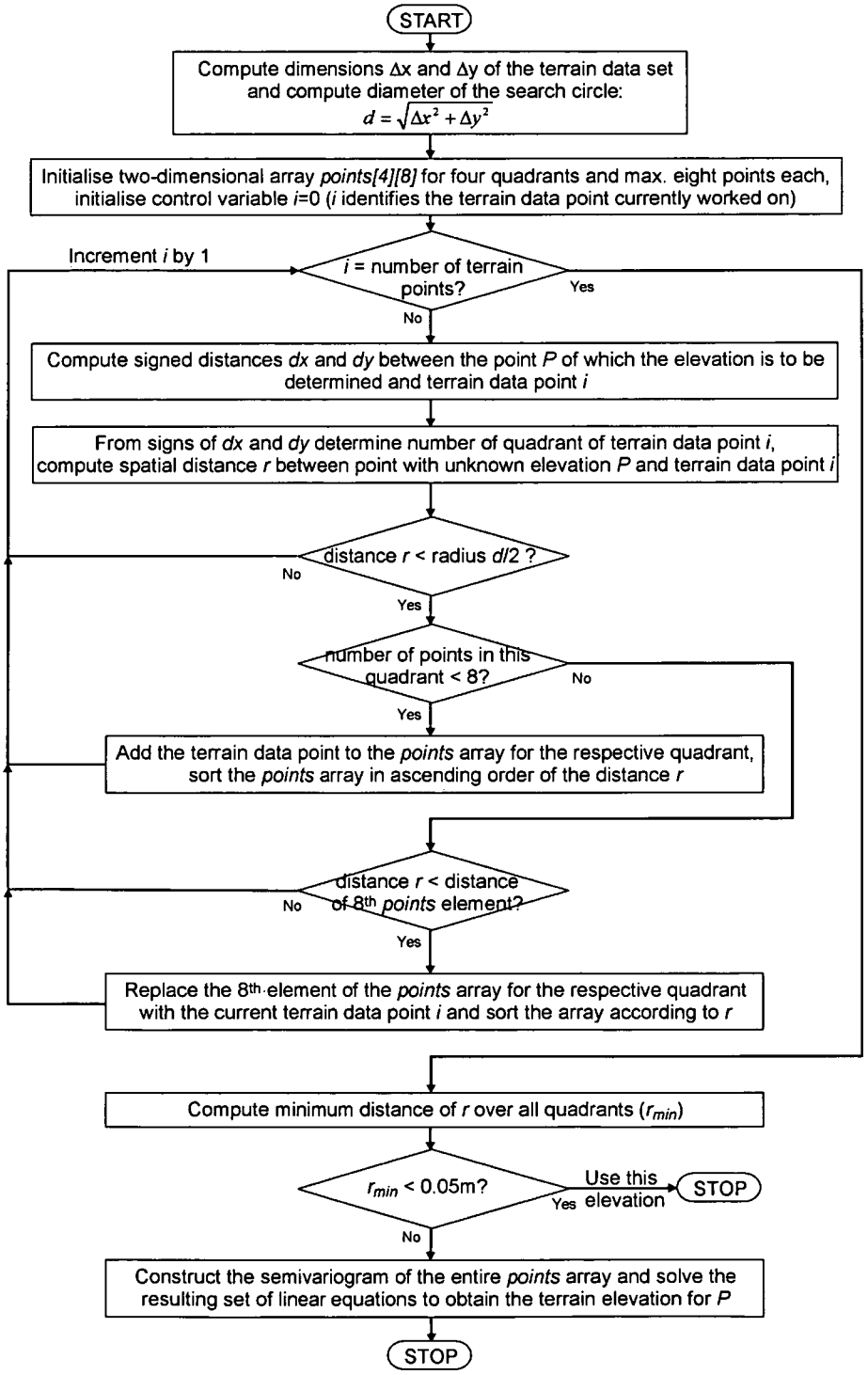

Figure A.3: Flow chart of the Kriging algorithm

Kriging vs. Bivariate Interpolation on cross-section data

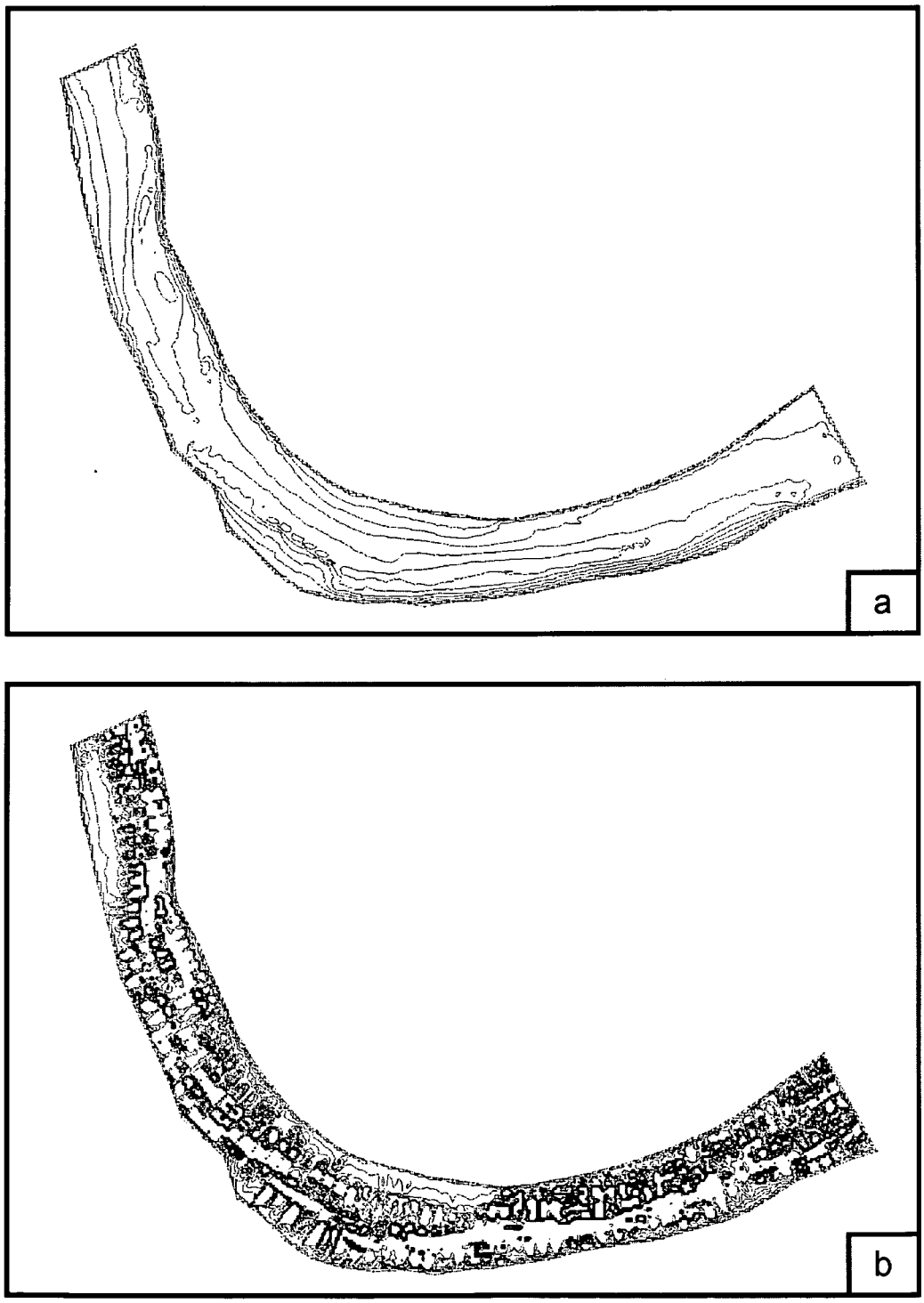

Figure A.4: Kriging (a.) and Bivariate Interpolation (b.) applied to a reach of the River Danube

Flow chart of the solver algorithm

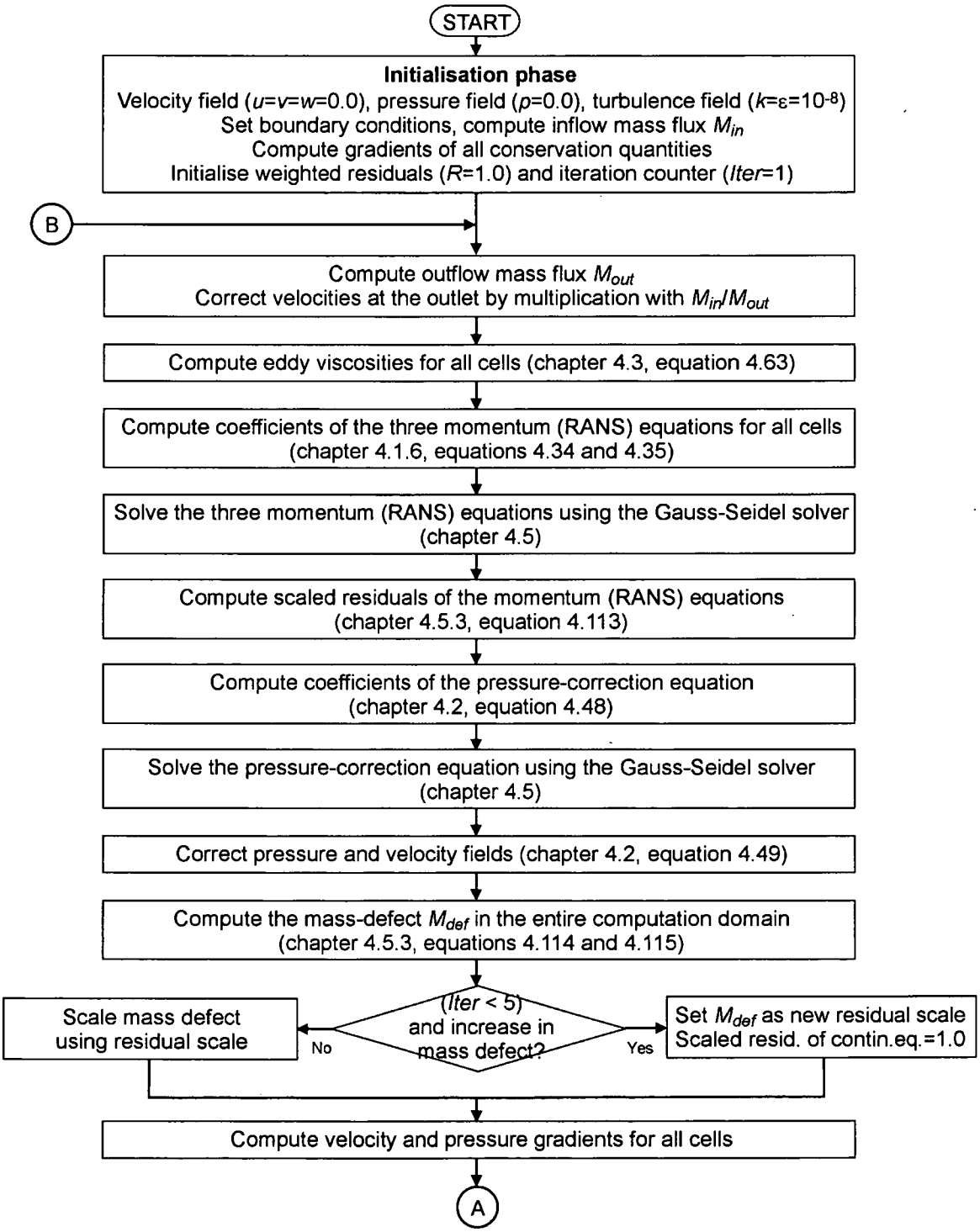

Figure A.5: Flow chart of the solver algorithm (part 1)

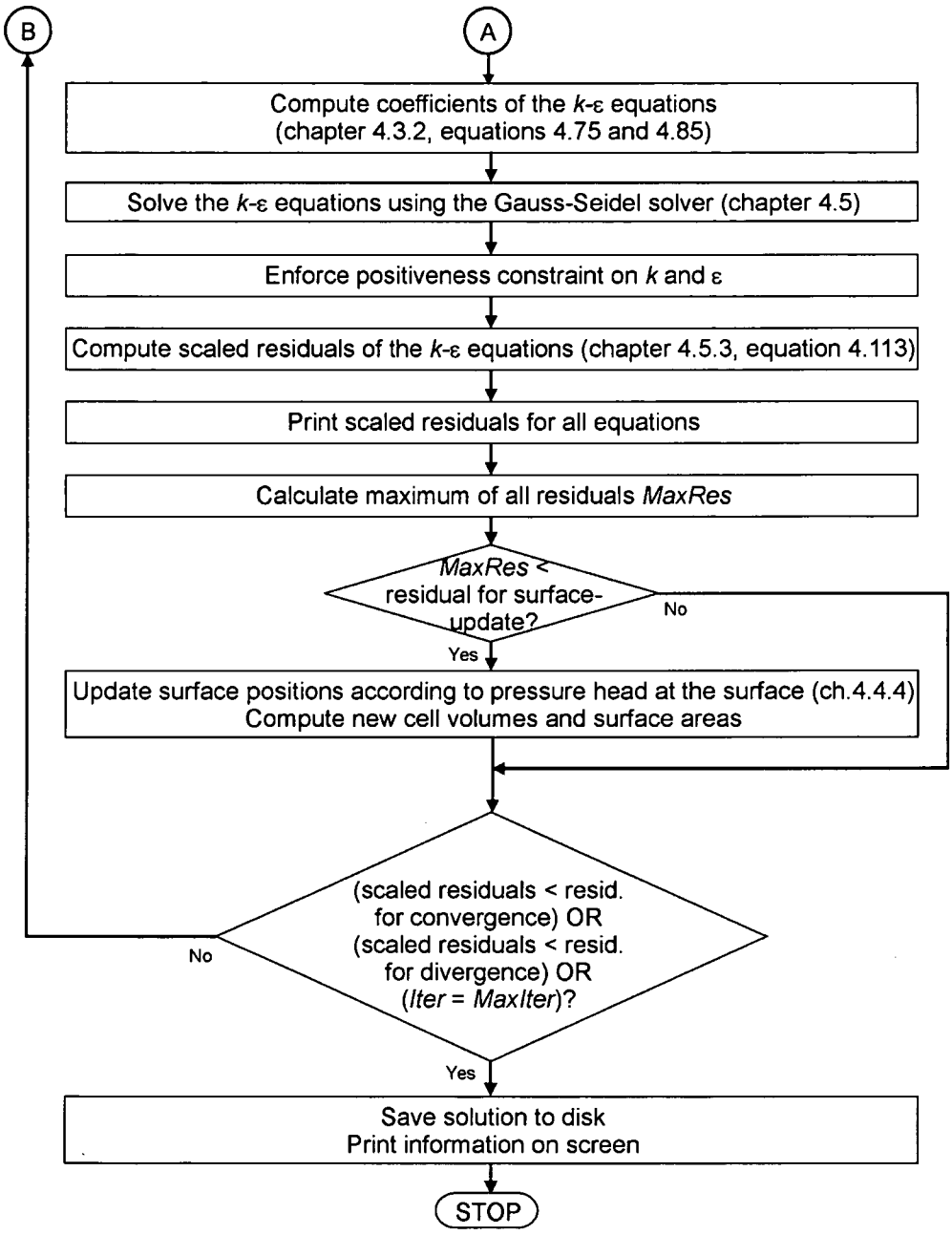

Figure A.6: Flow chart of the solver algorithm (part 2)

Appendix B Velocity profiles

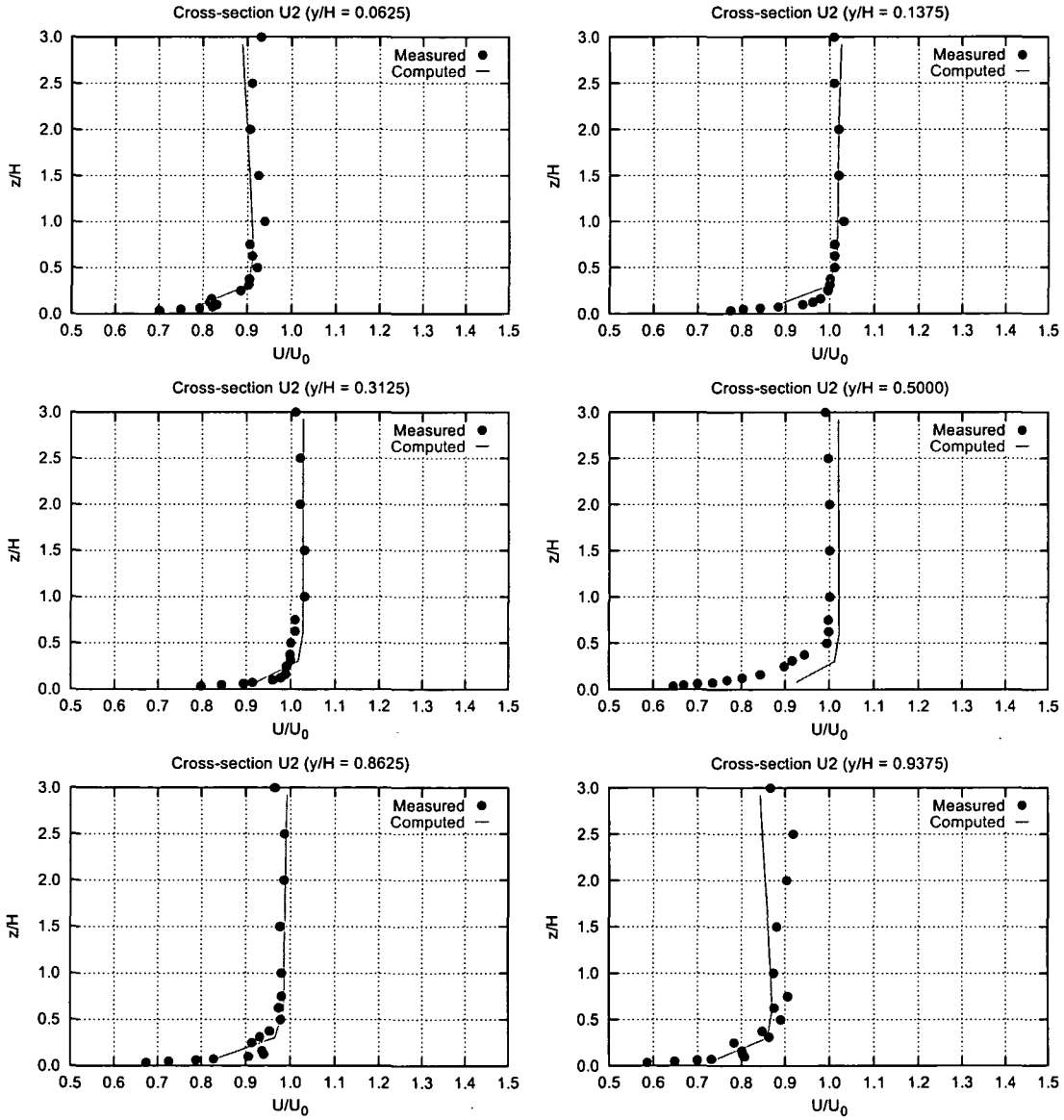

Figure B.1: Longitudinal velocity profiles for cross section U2 of Kim & Patel's experiment

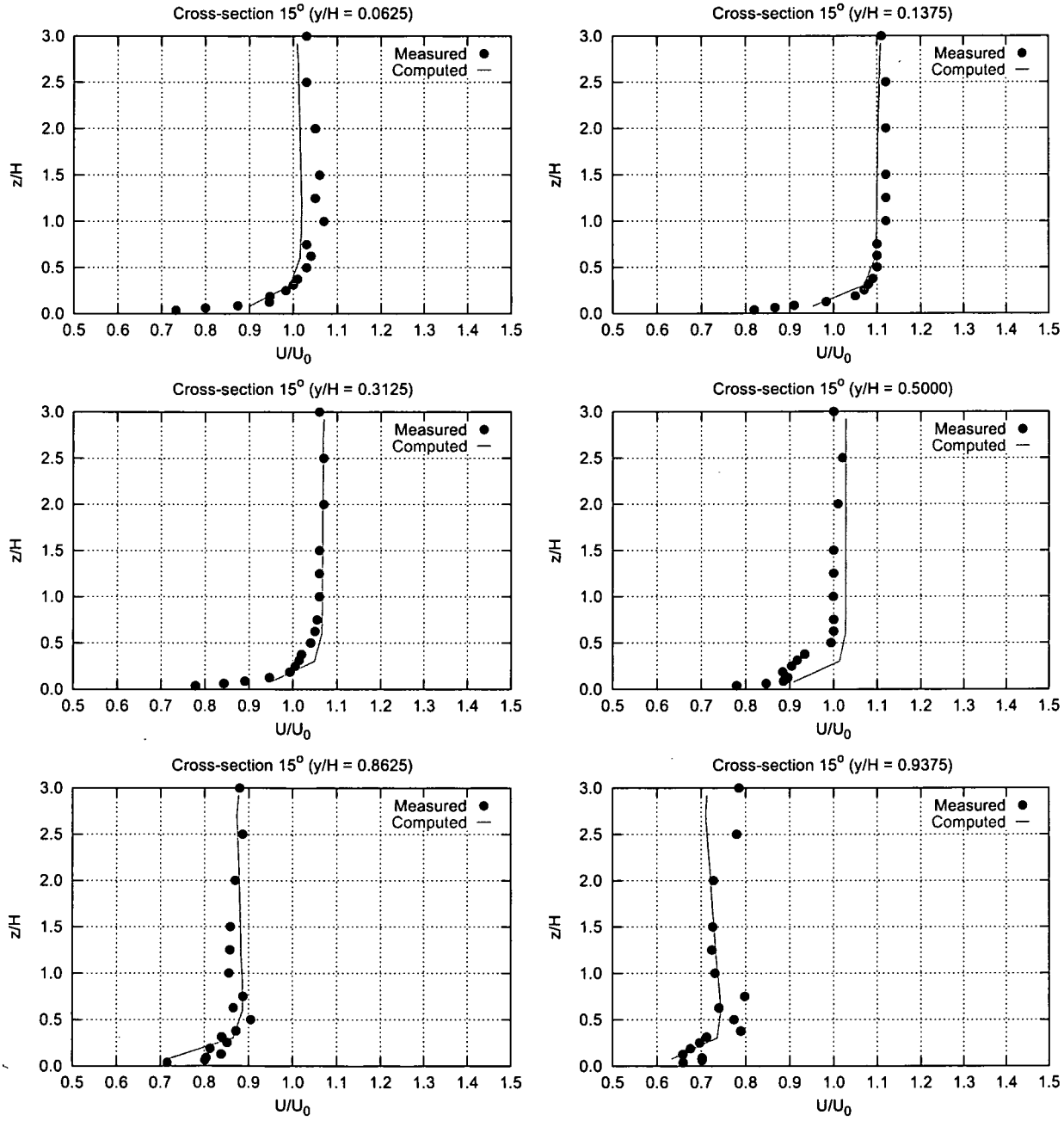

Figure B.2: Longitudinal velocity profiles for cross section 15° of Kim & Patel's experiment

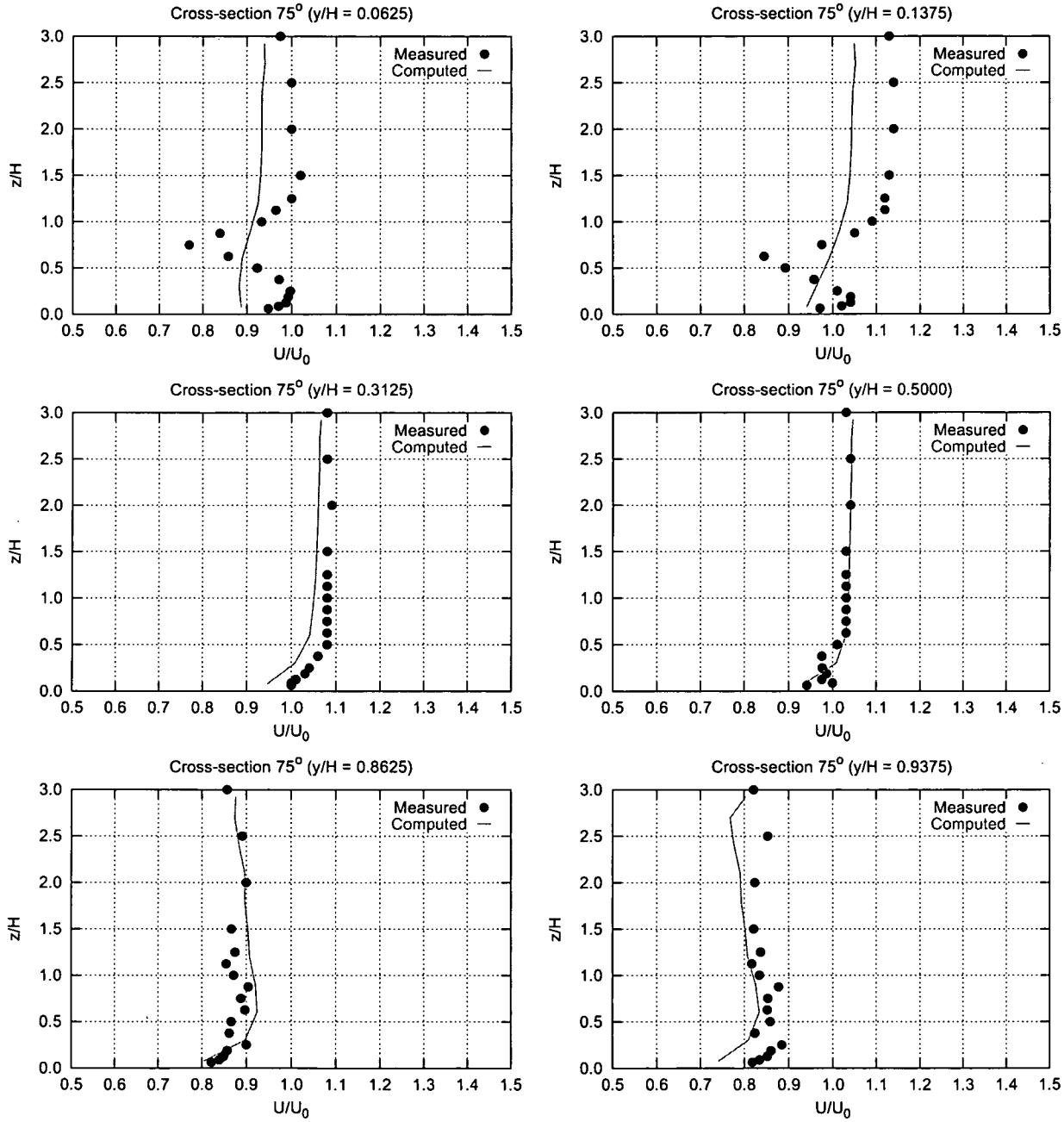

Figure B.3: Longitudinal velocity profiles for cross section 75° of Kim & Patel's experiment

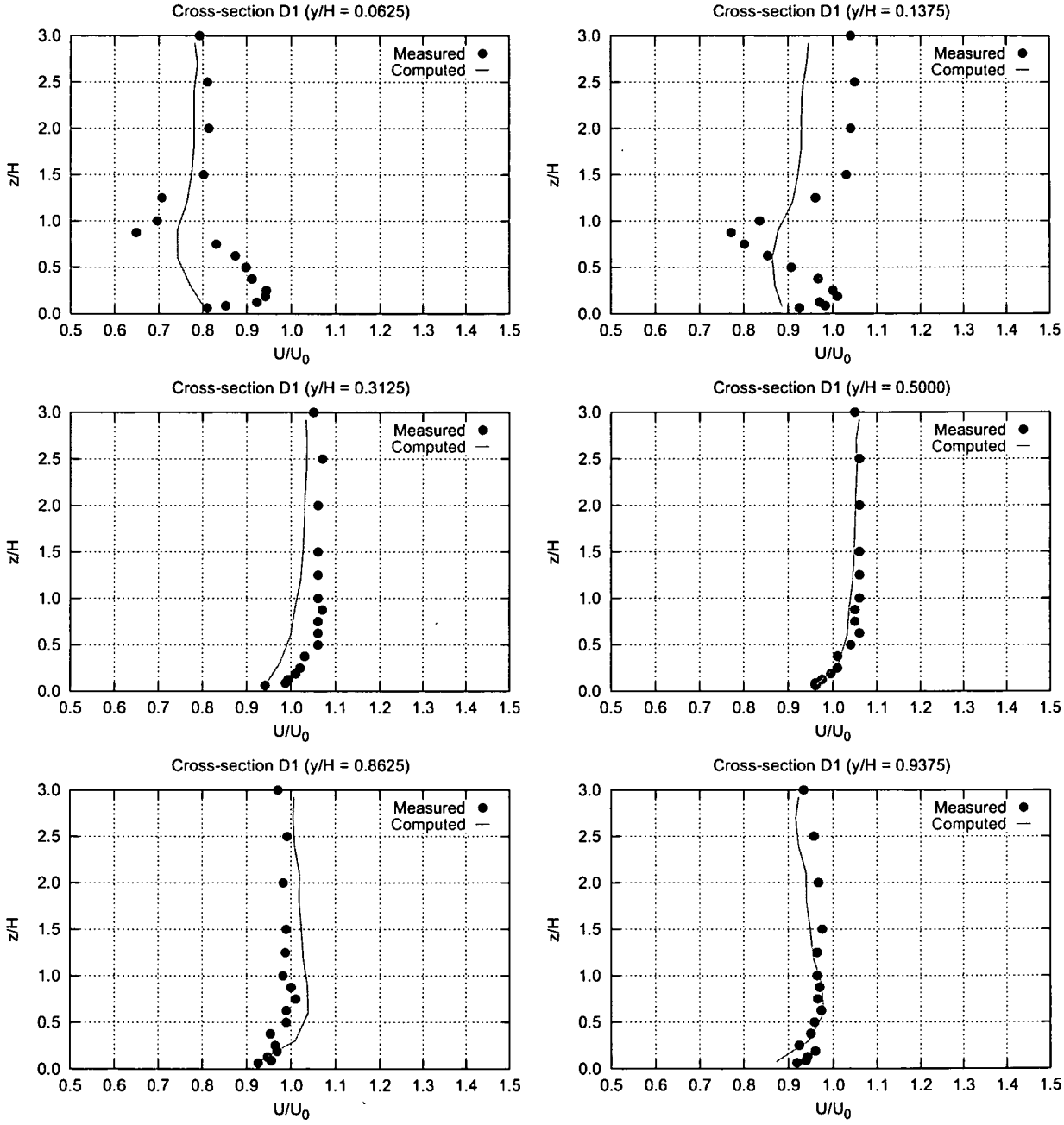

Figure B.4: Longitudinal velocity profiles for cross section D1 of Kim & Patel's experiment

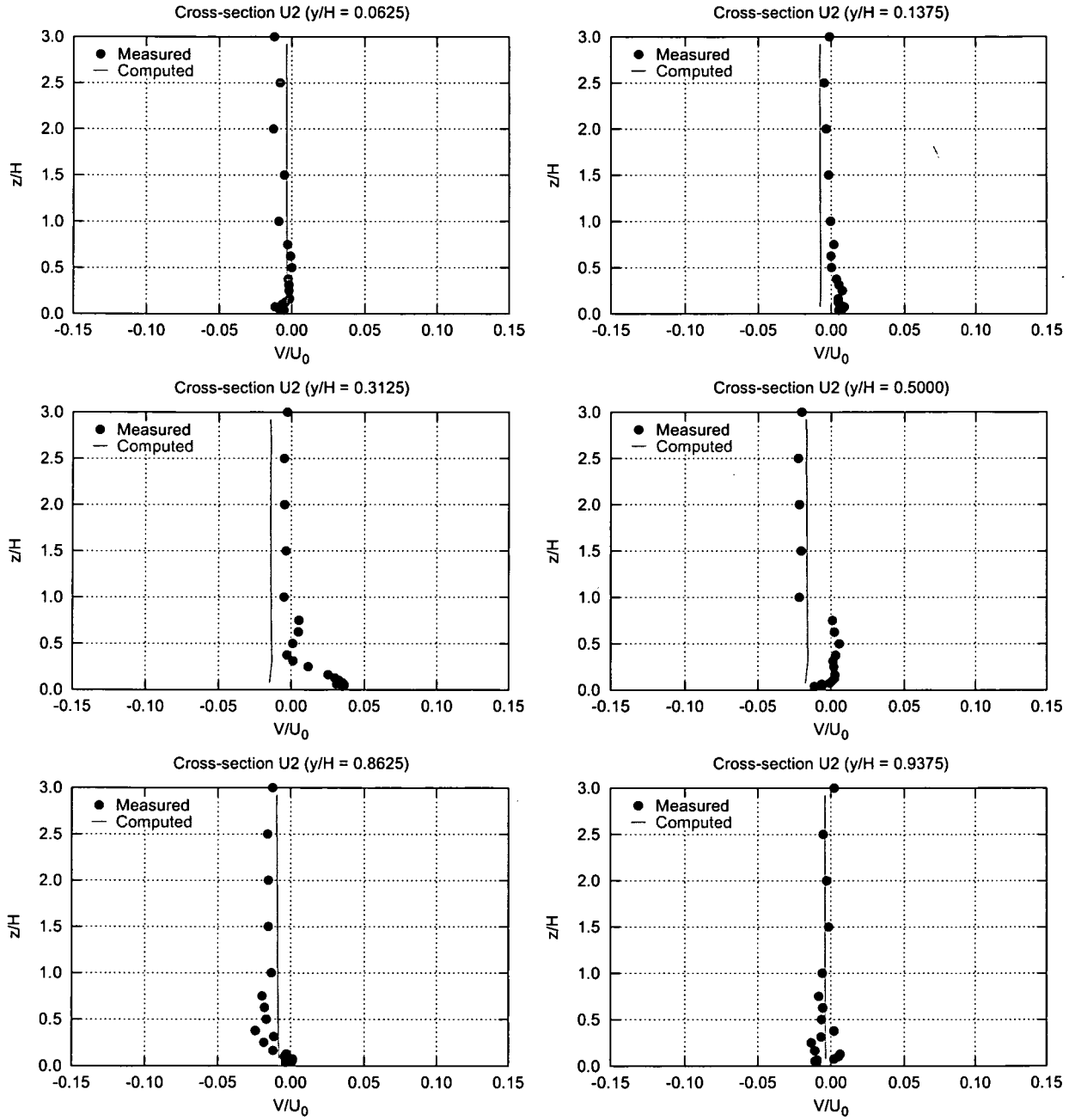

Figure B.5: Transversal velocity profiles for cross section U2 of Kim & Patel's experiment

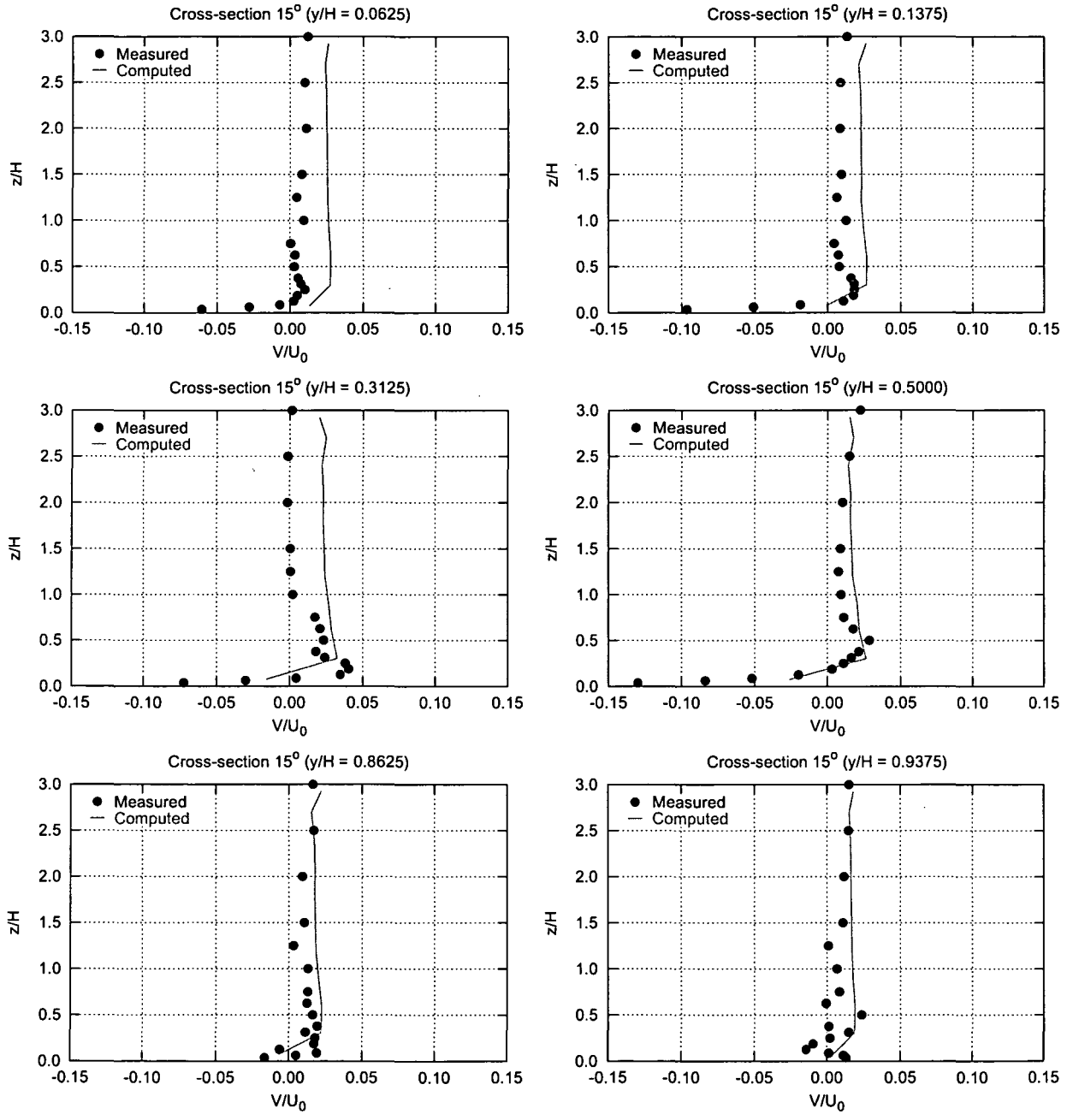

Figure B.6: Transversal velocity profiles for cross section 15° of Kim & Patel's experiment

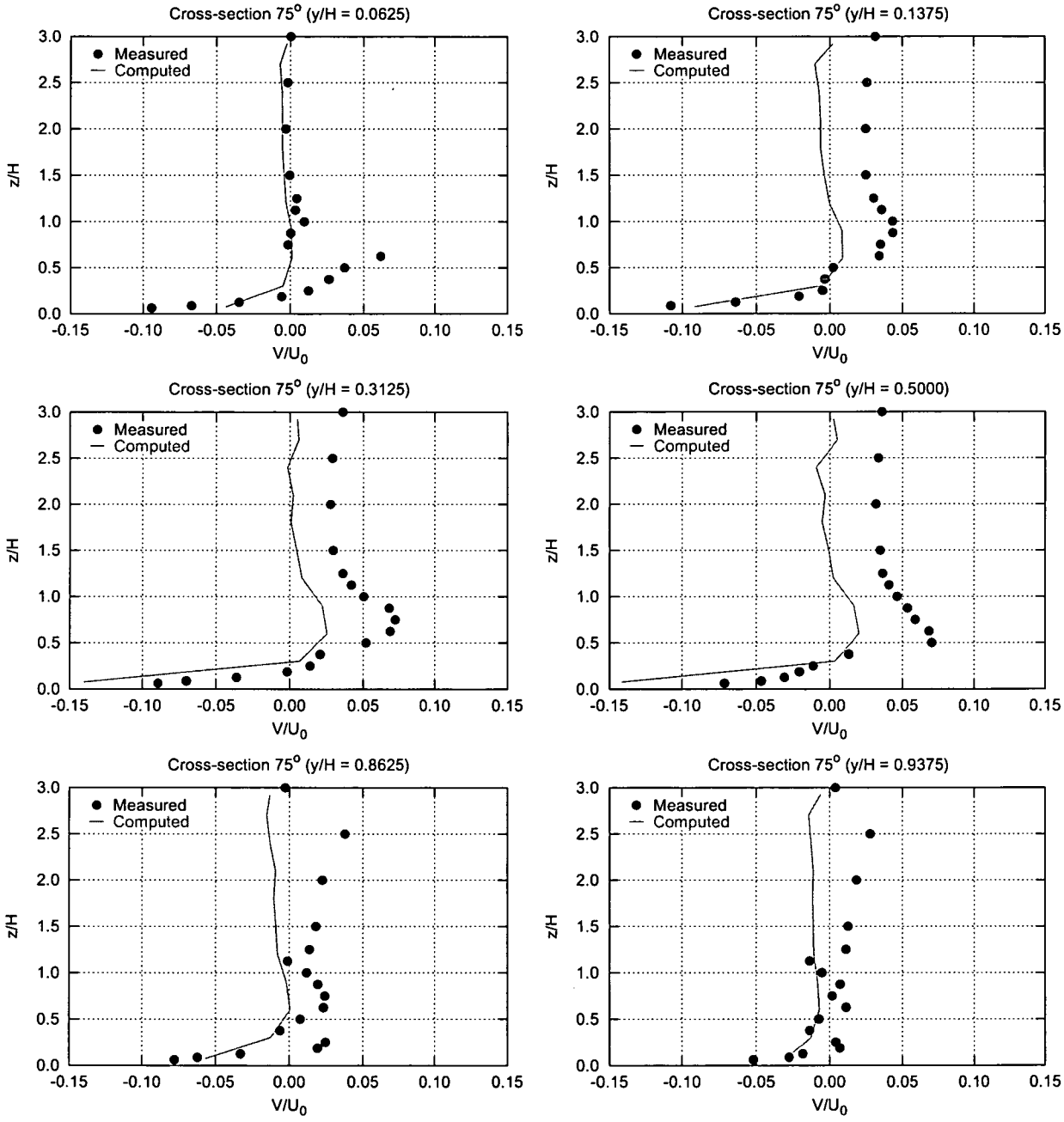

Figure B.7: Transversal velocity profiles for cross section 75° of Kim & Patel's experiment

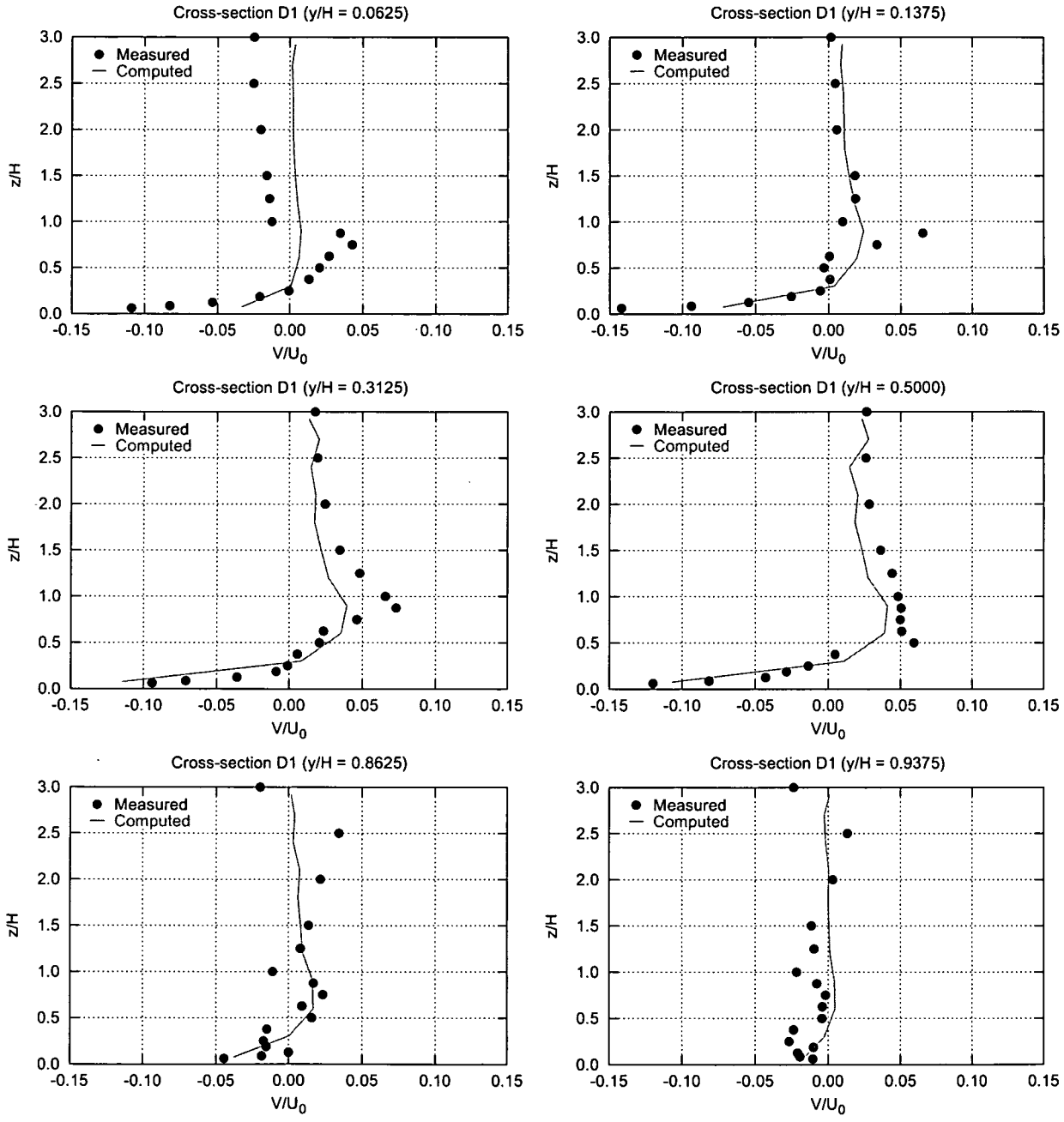

Figure B.8: Transversal velocity profiles for cross section D1 of Kim & Patel's experiment