Toward the Validation of Depth-Averaged, Steady-State Simulations of Fluvial Flows Using Three-Dimensional, Steady-State, RANS Turbulence Models

Pedro Abdiel Mateo Villanueva
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/etd

Part of the Mechanical Engineering Commons

Recommended Citation
https://digitalcommons.usu.edu/etd/826
TOWARD THE VALIDATION OF DEPTH-AVERAGED, STEADY-STATE SIMULATIONS OF FLUVIAL FLOWS USING THREE-DIMENSIONAL, STEADY-STATE, RANS TURBULENCE MODELS

by

Pedro Abdiel Mateo Villanueva

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Mechanical Engineering

Approved:

Dr. Robert E. Spall
Major Professor

Dr. Thomas H. Fronk
Committee Member

Dr. Thomas Hauser
Committee Member

Dr. Byron R. Burnham
Dean of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah
2010
Abstract

Toward the Validation of Depth-Averaged, Steady-State Simulations of Fluvial Flows Using Three-Dimensional, Steady-State, RANS Turbulence Models

by

Pedro Abdiel Mateo Villanueva, Master of Science
Utah State University, 2010

Major Professor: Dr. Robert E. Spall
Department: Mechanical and Aerospace Engineering

Calculations of fluvial flows are strongly influenced by geometry complexity and large overall uncertainty on every single measurable property, such as velocity and shear. Moreover, a considerable portion of the data obtained from computational simulations arose from two-dimensional, steady-state models. The present work states a different approach to perform computer-based simulations and analyze fluvial flows. For the first part, the suitability of OpenFOAM to be used as the main CFD solver to analyze fluvial flows is studied. Initially, two well documented channel configurations are computationally studied using OpenFOAM. Finally, these results are compared to the output obtained from one of the widely used quasi-3D CFD solvers used to perform studies about environmental hydraulics.

(141 pages)
To my mother CARMEN, my brother JAIME, and my sister MARIELLA, who stayed with me on this journey.

To KA YEE CHAN, for her support at all times.
Acknowledgments

I would like to thank Dr. Robert Spall, who provided me with timely advice during the development of this thesis.

I also acknowledge Dr. Thomas Hauser and Michal Hradisky, who first introduced me to scientific computing, for presenting me with key and insightful ideas relevant to the approach presented on this thesis.

I am also grateful to Dr. Thomas Fronk, who provided me with advice; and Dr. Christine Hailey, who provided me with invaluable support and advice during my education.

I would like to give special thanks to the government of the Dominican Republic and Doña Ligia Amada Melo, who provided me the wonderful opportunity to complete my Master’s degree at Utah State University.

Finally, I would like to express my gratitude to my parents, siblings, and friends who always had the right words that helped me realize I was capable to successfully complete this work and much more.

Pedro Abdiel Mateo Villanueva
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>v</td>
</tr>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>List of Figures</td>
<td>x</td>
</tr>
<tr>
<td>Notation</td>
<td>xv</td>
</tr>
<tr>
<td>Acronyms</td>
<td>xviii</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2 Fundamental Concepts</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Computational Fluid Dynamics</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Turbulence Modeling</td>
<td>6</td>
</tr>
<tr>
<td>2.3 Law of the Wall</td>
<td>10</td>
</tr>
<tr>
<td>2.4 Computational Mesh</td>
<td>17</td>
</tr>
<tr>
<td>2.4.1 Structured Mesh</td>
<td>18</td>
</tr>
<tr>
<td>2.4.2 Unstructured Mesh</td>
<td>18</td>
</tr>
<tr>
<td>2.4.3 Hybrid Mesh</td>
<td>18</td>
</tr>
<tr>
<td>2.5 Finite Volume Method</td>
<td>19</td>
</tr>
<tr>
<td>2.6 SIMPLE Algorithm</td>
<td>20</td>
</tr>
<tr>
<td>2.7 Parallel Computing</td>
<td>20</td>
</tr>
<tr>
<td>3 Problem Description</td>
<td>22</td>
</tr>
<tr>
<td>3.1 Turbulent Flows Through Large Aspect-Ratio Channels</td>
<td>22</td>
</tr>
<tr>
<td>3.1.1 Test Case 1</td>
<td>23</td>
</tr>
<tr>
<td>3.1.2 Test Case 2</td>
<td>24</td>
</tr>
<tr>
<td>3.2 Laminar Flows Through Large Aspect-Ratio Channels</td>
<td>25</td>
</tr>
<tr>
<td>3.2.1 Test Case 1</td>
<td>26</td>
</tr>
<tr>
<td>3.2.2 Test Case 2</td>
<td>26</td>
</tr>
<tr>
<td>3.3 Turbulent Fluvial Flows</td>
<td>26</td>
</tr>
<tr>
<td>4 Thesis Project Motivation and Objectives</td>
<td>31</td>
</tr>
<tr>
<td>5 Formulations and Computational Models</td>
<td>33</td>
</tr>
<tr>
<td>5.1 Setup of Test Case 1 Using Turbulence Models in OpenFOAM</td>
<td>33</td>
</tr>
<tr>
<td>5.1.1 Computational Mesh and Geometry Considerations</td>
<td>33</td>
</tr>
<tr>
<td>5.1.2 Boundary Conditions</td>
<td>35</td>
</tr>
<tr>
<td>5.2 Setup of Test Case 1 Using Turbulence Models in MD_SWMS</td>
<td>38</td>
</tr>
</tbody>
</table>
5.2.1 Computational Mesh and Geometry Considerations
5.2.2 Boundary Conditions
5.3 Setup of Test Case 1 Using Laminar Models in OpenFOAM
5.3.1 Computational Mesh and Geometry Considerations
5.3.2 Boundary Conditions
5.4 Setup of Test Case 1 Using Laminar Models in MD_SWMS
5.4.1 Computational Mesh and Geometry Considerations
5.4.2 Boundary Conditions
5.5 Setup of Test Case 2 Using Turbulence Models in OpenFOAM
5.5.1 Computational Mesh and Geometry Considerations
5.5.2 Boundary Conditions
5.6 Setup of Test Case 2 Using Turbulence Models in MD_SWMS
5.6.1 Computational Mesh and Geometry Considerations
5.6.2 Boundary Conditions
5.7 Setup of Test Case 2 Using Laminar Models in OpenFOAM
5.7.1 Computational Mesh and Geometry Considerations
5.7.2 Boundary Conditions
5.8 Setup of Test Case 2 Using Laminar Models in MD_SWMS
5.8.1 Computational Mesh and Geometry Considerations
5.8.2 Boundary Conditions
5.9 Validation of the Numerical Model

6 Results
6.1 Test Case 1: OpenFOAM using Turbulent Models
6.1.1 Mesh Quality
6.1.2 Initial Results
6.1.3 Mesh Refinement
6.1.4 Computational Uncertainties
6.2 Test Case 1: MD_SWMS using Turbulent Models
6.2.1 Initial Results
6.2.2 Mesh Refinement
6.3 Test Case 1: OpenFOAM using Laminar Models
6.4 Test Case 1: MD_SWMS using Laminar Models
6.5 Test Case 2: OpenFOAM using Turbulent Models
6.5.1 Mesh Quality
6.5.2 Initial Results
6.5.3 Mesh Refinement
6.5.4 Computational Uncertainties
6.6 Test Case 2: MD_SWMS using Turbulent Models
6.6.1 Initial Results
6.6.2 Mesh Refinement
6.7 Test Case 2: OpenFOAM using Laminar Models
6.8 Test Case 2: MD_SWMS using Laminar Models
6.9 Comparison of Turbulence Simulations: OpenFOAM and MD_SWMS
6.9.1 Test Case 1
6.9.2 Test Case 2
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>Comparison of OpenFOAM and MD_SWMS for Fluvial Flow Modeling</td>
<td>92</td>
</tr>
<tr>
<td>7.2</td>
<td>Usage of the Different OpenFOAM Versions on the Current Study</td>
<td>93</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Typical three-dimensional control volume used to derive the Navier-Stokes</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>equations in Cartesian coordinates.</td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>Law of the wall for a smooth plate.</td>
<td>16</td>
</tr>
<tr>
<td>2.3</td>
<td>Side view of the computational mesh of a cone.</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>Typical structured mesh.</td>
<td>18</td>
</tr>
<tr>
<td>2.5</td>
<td>Typical unstructured mesh.</td>
<td>19</td>
</tr>
<tr>
<td>2.6</td>
<td>Typical hybrid mesh.</td>
<td>19</td>
</tr>
<tr>
<td>3.1</td>
<td>Channel geometry for Test Case 1.</td>
<td>24</td>
</tr>
<tr>
<td>3.2</td>
<td>Channel geometry for Test Case 2.</td>
<td>25</td>
</tr>
<tr>
<td>3.3</td>
<td>Channel geometry for a section of the Strawberry River, Utah.</td>
<td>29</td>
</tr>
<tr>
<td>5.1</td>
<td>Computational mesh used to perform CFD analysis for Test Case 1.</td>
<td>34</td>
</tr>
<tr>
<td>5.2</td>
<td>Geometry considerations and description for Test Case 1.</td>
<td>34</td>
</tr>
<tr>
<td>5.3</td>
<td>Computational mesh used to perform CFD analysis for Test Case 2.</td>
<td>42</td>
</tr>
<tr>
<td>5.4</td>
<td>Geometry considerations and description for Test Case 2.</td>
<td>42</td>
</tr>
<tr>
<td>6.1</td>
<td>Pseudocolor plot: velocity magnitude scaled by the mean velocity of the</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>converged CFD simulation for Test Case 1 using OpenFOAM with turbulence</td>
<td></td>
</tr>
<tr>
<td></td>
<td>models.</td>
<td></td>
</tr>
<tr>
<td>6.2</td>
<td>Vector plot showing the recirculation regions upstream and downstream the</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>rectangular obstacle from the converged velocity field for Test Case 1, using</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OpenFOAM with turbulence models (obtained using VisIt [1]).</td>
<td></td>
</tr>
<tr>
<td>6.3</td>
<td>Pseudocolor plot: values of $y^+$ at the bottom wall of the converged CFD</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>simulation for Test Case 1, using OpenFOAM with turbulence models.</td>
<td></td>
</tr>
</tbody>
</table>
6.4 Two-dimensional velocity profiles of the depth-averaged, converged CFD simulation for Test Case 1 as a function of distance $y$. As presented; a) corresponds to a simulation setup with $4.1 \times 10^6$ cells, b) corresponds to a simulation setup with $3.2 \times 10^6$ cells, and plot c) corresponds to a simulation setup with $2.22 \times 10^6$ cells. Sub-figure A) is the velocity profile at $x = 10L_1$, B) is the velocity profile at $x = 15L_1$, and c) is the velocity profile at $x = 18L_1$. For all cases, the simpleFoam solver with turbulence models was used.

6.5 Pseudocolor plot: values of $y^+$ at the bottom wall of the converged results from the mesh-refined CFD simulation for Test Case 1, using OpenFOAM with turbulence models. As presented, a) correspond to a computational mesh with $2.22 \times 10^6$ cells and b) corresponds to a computational mesh with $4.1 \times 10^6$ cells.

6.6 Pseudocolor plot: velocity magnitudes scaled by the channel mean velocity for Test Case 1, obtained using OpenFOAM with turbulence models. As presented, a) corresponds to a meshed geometry with $2.22 \times 10^6$ cells, b) corresponds to a meshed geometry with $3.2 \times 10^6$, and c) corresponds to a meshed geometry with $4.1 \times 10^6$ cells.

6.7 Velocity distribution, scaled by the mean velocity, from the turbulent MD_SWMS solution for Test Case 1.

6.8 Velocity vector plots showing the main recirculation regions upstream and downstream the obstacle, obtained using the turbulent MD_SWMS solution.

6.9 Velocity distributions for different mesh sizes at different locations, obtained using the turbulent MD_SWMS solution. As shown, a) corresponds to a computational mesh with $5.0 \times 10^4$ cells, b) corresponds to a computational mesh with $7.0 \times 10^4$ cells, and c) corresponds to a computational mesh with $9.0 \times 10^4$ cells.

6.10 Velocity profile lines of the converged solutions, obtained using the turbulent MD_SWMS solution. As shown, A) corresponds to a location $x = 10L_1$, B) corresponds to a location $x = 15L_1$ and C) corresponds to a location $x = 18L_1$. The legend a) represents a computational mesh with $9.0 \times 10^4$ cells, b) represents a computational cell with $7.0 \times 10^4$ cells, and c) represents a computational mesh with $5.0 \times 10^4$ cells.

6.11 Scaled velocity distribution obtained using the laminar simpleFoam solution for Test Case 1.

6.12 Scaled velocity distribution obtained using the laminar simpleFoam solution for different mesh sizes. As shown, a) correspond to a mesh size of $2.22 \times 10^6$ cells, and b) corresponds to a mesh size of $4.1 \times 10^6$ cells.
6.13 Scaled velocity profile lines obtained using the laminar simpleFoam solution for different mesh sizes. As shown, a) correspond to a mesh size of 2.22x10^6 cells, and b) corresponds to a mesh size of 4.1x10^6 cells. As shown, A) corresponds to a location \( x = 10L_1 \), B) corresponds to a location \( x = 22.5L_1 \), and C) corresponds to a location \( x = 25L_1 \). .............................. 63

6.14 Scaled velocity distribution obtained using the laminar FaSTMECH solution for Test Case 1. ................................................................. 64

6.15 Scaled velocity distribution obtained using the laminar FaSTMECH solution for different mesh sizes used to study Test Case 1. As shown, a) correspond to a mesh size of 5.0x10^4 cells, and b) corresponds to a mesh size of 9.0x10^4 cells. .................................................. 65

6.16 Scaled velocity profile lines obtained using the laminar FaSTMECH solution for different mesh sizes (Test Case 1). As shown, a) correspond to a mesh size of 5x10^4 cells, and b) corresponds to a mesh size of 9x10^4 cells. As shown, A) corresponds to a location \( x = 22.5L_1 \), and B) corresponds to a location \( x = 25L_1 \). .......................................................... 66

6.17 Vector plot of the converged velocity field showing the different recirculation regions of Test Case 2, using solution obtained with simpleFoam with turbulence models. ............................................ 66

6.18 Pseudocolor plot: velocity magnitude scaled by the channel mean velocity of the converged solution for Test Case 2, using simpleFoam with turbulence models. ........................................... 67

6.19 Pseudocolor plot: values of \( y^+ \) at the bottom wall of the converged CFD simulation for Test Case 2, using simpleFoam with turbulence models. .... 67

6.20 Two-dimensional velocity profiles of the averaged, converged CFD simulation for Test Case 2 as a function of distance \( y \), obtained using simpleFoam with turbulence models. As shown, the label a) corresponds to a computational mesh containing 2.8x10^6 cells, b) corresponds to a computational mesh containing 2.1x10^6 cells, and c) corresponds to a computational mesh with 1.87x10^6 cells. Sub-figure A) is the velocity profile at \( x = 13.32L_2 \), B) is the velocity profile at \( x = 17.8L_2 \), and C) is the velocity profile at \( x = 30L_2 \). .............................. 68

6.21 Pseudocolor plot: scaled velocity magnitudes of the depth-averaged converged solution for Test Case 2, using simpleFoam with turbulence models. As shown, a) performed on a computational mesh with 1.87x10^6 cells, b) performed on a computational mesh with 2.1x10^6, and c) performed on a computational mesh with 2.8x10^6 cells. ........................................... 69
6.22 Pseudocolor plot: values of $y^+$ at the bottom wall of the converged solution from simpleFoam. As shown, a) performed on a computational mesh with $1.87 \times 10^6$ cells, b) performed on a computational mesh with $2.1 \times 10^6$, and c) performed on a computational mesh with $2.8 \times 10^6$ cells.

6.23 Velocity distribution of the turbulent solution obtained from FaSTMECH, converged on a mesh size of $5.0 \times 10^4$ cells.

6.24 Velocity distributions of the turbulent solution obtained from FaSTMECH. As shown, a) corresponds to a solution converged on a computational mesh with $5 \times 10^4$ cells, and b) corresponds to a solution converged on a computational mesh with $8.2 \times 10^4$ cells.

6.25 Velocity profile lines of the turbulent solution obtained from FaSTMECH. The legend a) corresponds to a solution converged on a computational mesh with $8.2 \times 10^4$ cells, and b) corresponds to a solution converged on a computational mesh with $5 \times 10^4$ cells. As shown, sub-figure A) is the velocity profile at $x = 13.32L_2$, B) is the velocity profile at $x = 17.8L_2$, and C) is the velocity profile at $x = 30L_2$.

6.26 Velocity vector plot showing the recirculation regions from the FaSTMECH solution.

6.27 Scaled velocity distribution obtained using the laminar simpleFoam solution for Test Case 2.

6.28 Velocity vector plot showing the recirculation regions from the laminar simpleFoam solution for Test Case 2.

6.29 Scaled velocity distribution obtained using the laminar simpleFoam solution for different mesh sizes. As shown, a) correspond to a mesh size of $1.87 \times 10^6$ cells, and b) corresponds to a mesh size of $2.8 \times 10^6$ cells.

6.30 Velocity profile lines of the laminar solution obtained from simpleFoam. The legend a) corresponds to a solution converged on a computational mesh with $2.8 \times 10^6$ cells, and b) corresponds to a solution converged on a computational mesh with $1.87 \times 10^6$ cells. As shown, sub-figure A) is the velocity profile at $x = 13.32L_2$, and B) is the velocity profile at $x = 48L_2$.

6.31 Scaled velocity distribution obtained using the laminar FaSTMECH solution for different mesh sizes. As shown, a) correspond to a mesh size of $5 \times 10^4$ cells, and b) corresponds to a mesh size of $9 \times 10^4$ cells.

6.32 Velocity vector plot showing the recirculation regions from the laminar FaSTMECH solution for Test Case 2.
6.33 Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 1, using turbulence models, plotted against the scaled dimensionless distance $y/y_{max}$, corresponding to a location $x = 10L_1$. . . . 79

6.34 Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 1, using turbulence models, plotted against the scaled dimensionless distance $y/y_{max}$, corresponding to a location $x = 15L_1$. . . . 80

6.35 Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 1, using turbulence models, plotted against the scaled dimensionless distance $y/y_{max}$, corresponding to a location $x = 18L_1$. . . . 81

6.36 Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 2, using turbulence models, plotted against the scaled dimensionless distance $y/y_{max}$, corresponding to a location $x = 13.32L_2$. . . 83

6.37 Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 2, using turbulence models, plotted against the scaled dimensionless distance $y/y_{max}$, corresponding to a location $x = 17.8L_2$. . . 84

6.38 Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 2, using turbulence models, plotted against the scaled dimensionless distance $y/y_{max}$, corresponding to a location $x = 30L_2$. . . . 85

6.39 Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 1, using laminar models, plotted against the scaled dimensionless distance $y/y_{max}$, corresponding to a location $x = 10L_1$. . . . 86

6.40 Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 1, using laminar models, plotted against the scaled dimensionless distance $y/y_{max}$, corresponding to a location $x = 22.5L_1$. . . 87

6.41 Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 2, using laminar models, plotted against the scaled dimensionless distance $y/y_{max}$, corresponding to a location $x = 13.32L_2$. . . 88

6.42 Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 2, using laminar models, plotted against the scaled dimensionless distance $y/y_{max}$, corresponding to a location $x = 48L_2$. . . . 89
### Notation

#### English letter symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Channel cross sectional area</td>
</tr>
<tr>
<td>C</td>
<td>Constant of integration for the Law of the Wall</td>
</tr>
<tr>
<td>(D_H)</td>
<td>Hydraulic diameter</td>
</tr>
<tr>
<td>(f_s)</td>
<td>Flow stage</td>
</tr>
<tr>
<td>(F_b)</td>
<td>Body forces on a fluid per unit volume</td>
</tr>
<tr>
<td>(g_0)</td>
<td>Acceleration of gravity at sea level</td>
</tr>
<tr>
<td>(H)</td>
<td>Geometric altitude</td>
</tr>
<tr>
<td>(i_x)</td>
<td>Unit vector oriented along the (x)-axis</td>
</tr>
<tr>
<td>(i_y)</td>
<td>Unit vector oriented along the (y)-axis</td>
</tr>
<tr>
<td>(i_z)</td>
<td>Unit vector oriented along the (z)-axis</td>
</tr>
<tr>
<td>(k)</td>
<td>Turbulent kinetic energy</td>
</tr>
<tr>
<td>(l)</td>
<td>Prandtl’s mixing length</td>
</tr>
<tr>
<td>(L_1)</td>
<td>Characteristic length for Test Case 1</td>
</tr>
<tr>
<td>(L_2)</td>
<td>Characteristic length for Test Case 2</td>
</tr>
<tr>
<td>(p)</td>
<td>Thermodynamic pressure</td>
</tr>
<tr>
<td>(\bar{p})</td>
<td>Mean thermodynamic pressure</td>
</tr>
<tr>
<td>(\tilde{p})</td>
<td>Fluctuating thermodynamic pressure</td>
</tr>
<tr>
<td>(\bar{p})</td>
<td>Pseudo-mean pressure</td>
</tr>
<tr>
<td>(p^+)</td>
<td>Dimensionless wall-scaled axial pressure</td>
</tr>
<tr>
<td>(P)</td>
<td>Wetted perimeter</td>
</tr>
<tr>
<td>(Q)</td>
<td>Flow discharge</td>
</tr>
<tr>
<td>(R_E)</td>
<td>Radius of Earth</td>
</tr>
<tr>
<td>(Re)</td>
<td>Reynolds number</td>
</tr>
<tr>
<td>(\tilde{S}(\nabla))</td>
<td>Strain-rate tensor</td>
</tr>
<tr>
<td>(t)</td>
<td>Time</td>
</tr>
<tr>
<td>(u_r)</td>
<td>Friction velocity</td>
</tr>
<tr>
<td>(u^+)</td>
<td>Dimensionless wall-scaled velocity</td>
</tr>
<tr>
<td>(\mathbf{V})</td>
<td>Velocity vector</td>
</tr>
<tr>
<td>(\mathbf{\nabla})</td>
<td>Mean velocity vector</td>
</tr>
<tr>
<td>(\tilde{\mathbf{V}})</td>
<td>Fluctuating velocity vector</td>
</tr>
<tr>
<td>(V_x)</td>
<td>Velocity component in the (x)-direction</td>
</tr>
<tr>
<td>(\mathbf{\nabla}_x)</td>
<td>Mean velocity component in the (x)-direction</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>$\tilde{V}_x$</td>
<td>Fluctuating velocity component in the $x$-direction</td>
</tr>
<tr>
<td>$V_y$</td>
<td>Velocity component in the $y$-direction</td>
</tr>
<tr>
<td>$\overline{V}_y$</td>
<td>Mean velocity component in the $y$-direction</td>
</tr>
<tr>
<td>$\tilde{V}_y$</td>
<td>Fluctuating velocity component in the $y$-direction</td>
</tr>
<tr>
<td>$V_z$</td>
<td>Velocity component in the $z$-direction</td>
</tr>
<tr>
<td>$\overline{V}_z$</td>
<td>Mean velocity component in the $z$-direction</td>
</tr>
<tr>
<td>$\tilde{V}_z$</td>
<td>Fluctuating velocity component in the $z$-direction</td>
</tr>
<tr>
<td>$x$</td>
<td>Directional coordinate along the $x$-axis (Cartesian coordinates)</td>
</tr>
<tr>
<td>$y$</td>
<td>Directional coordinate along the $y$-axis (Cartesian coordinates)</td>
</tr>
<tr>
<td>$y^+$</td>
<td>Dimensionless wall distance</td>
</tr>
<tr>
<td>$z$</td>
<td>Directional coordinate along the $z$-axis (Cartesian coordinates)</td>
</tr>
<tr>
<td>$Z$</td>
<td>Geopotential altitude</td>
</tr>
</tbody>
</table>
Greek letter symbols

δ  Velocity boundary layer thickness
\delta  Kronecker delta
Δx  Length of computational cell along the x-direction
Δy  Length of computational cell along the y-direction
Δz  Length of computational cell along the z-direction
ε  Turbulent kinetic energy dissipation rate
κ  von Kármán constant
µ  Dynamic viscosity
µt  Dynamic eddy viscosity
ν  Kinematic viscosity
νt  Kinematic eddy viscosity
ν+  Ratio of eddy viscosity to molecular viscosity
ρ  Density
τ  Local shear stress
τ_{total}  Total shear stress
τ_w  Wall total shear stress
\overline{\tau}  Reynolds stress tensor
\overline{\sigma}  Pseudo-molecular fluid stress tensor
Acronyms

1D One-Dimensional
2D Two-Dimensional
3D Three-Dimensional
CFD Computational Fluid Dynamics
DES Detached Eddy Simulation
DNS Direct Numerical Simulation
DM Discretization Method
EAARL Experimental Advanced Airborne Research LiDAR
FaSTMECH Flow and Sediment Transport and Morphological Evolution of Channels
FDM Finite Difference Method
FEM Finite Element Method
FVM Finite Volume Method
LES Large Eddy Simulation
LiDAR Light Detection And Ranging
MD_SWMS Multi-Dimensional Surface-Water Modeling System
NS Navier-Stokes
PDE Partial Differential Equation
PIV Particle Image Velocimetry
RANS Raymonds-Averaged Navier-Stokes
RAS Reynolds-Averaged Stress
SIMPLE Semi-Implicit Method for Pressure Linked Equations
SToRM System for Transport and River Modeling
Chapter 1

Introduction

Modeling, monitoring and predicting the behavior of fluvial flows using Computational Fluid Dynamics (CFD) is becoming the target of many computational scientists. One of the main contributors for this tendency is the fact that because of the nature of the flow, it can develop different turbulent properties depending on the local Reynolds number of the fluid. These turbulent properties are known to have a large impact in modern life, such as landscape modification, bridge analysis, ship design, sediment prescriptions, vegetation patterns, and migratory activities of fish and other species.

Thus, our motivation is to study the rich dynamics encountered when fluvial flows are studied more in depth using computational resources. Nevertheless, this task is more complex than the common industrial engineering applications of CFD. Some of the factors influencing on the complexity of modeling fluvial flows are: very large levels of energy dissipation due to turbulence; the changing geomorphology of the channel bed and the walls; the flow is largely unidirectional; the exact geometries cannot be simplified to easily meshable surfaces; boundary conditions are usually unknowns; due to the rich interactions between the flow and the material properties of the surface and channel bed, drag coefficients and the roughness of the channel bed vary significantly spatially and temporally; the driving forces of the flow are highly variable; length and time scales can vary from being very large to being very small; and the channel length largely exceeds the channel width and the flow depth.

All the factors contributing to the relative increase in computational resources and the complexity of modeling fluvial flows in order to determine their behavior with some degree of accuracy are also the factors contributing to the main interest of their study. The large levels of energy dissipation due to turbulence and the changes of the bed of a river are two of...
the main reasons why the foundation of bridges’ supports are catastrophically failing before
the estimated time, and the transport of particles due to the high kinetic energy present is
widely known as being responsible of changes on local geomorphology.

Furthermore, computer-based simulations of fluvial flows are also characterized by con-
siderable uncertainty over almost every single aspect of the modeling process. Moreover,
the validation data to test the relevant aspects of the flow in order to determine the perfor-
mance level of the computer-based models may not be available, or may be very intricate
and expensive to obtain. Due to this uncertainty, model validation for fluvial flow situations
is a compromising task as well, and this is the reason why they are becoming one of the
new challenges for computational researchers.

Based on the fact that the length of a common river can be several orders of magnitude
larger than its depth and its width, a simple large aspect ratio channel may be considered
as a good approximation of the physical conditions appreciated on rivers. Additionally, in
order to understand the dynamics encountered on fluvial flows, the irregularities observed
on rivers can be initially simulated by simply rectangular blocks mounted the side walls.

Therefore, a computer-based model capable of describing and predicting the behavior
of fluvial flows will have a good start on validating the characteristics of the turbulent flow
past a long rectangular, surface-piercing block mounted at the side wall of a large aspect
ratio channel. The general behavior of such a flow was successfully observed in the past by
Chrisohoides and Sotiropoulos [2].

The physical configuration of a large aspect ratio channel with a long rectangular,
surface-piercing block mounted at a side wall will generate coherent structures around the
rectangular block. These three-dimensional coherent structures will, in fact, be the ac-
countable for the transportation of particles within the flow and eventually have a great
contribution on the scouring of bridge’s foundations, the establishment of new geomorpho-
logical entities and ecosystems, and the migration of living species. Thus, if a computational
model can predict the coherent vortical structures present in the physical configuration pre-
viously described, it will be suitable to be implemented on a geometry that describes an
actual fluvial flow.

Similarly, the coherent structures of the flow over a set of three surface-mounted cubic obstacles located on the upper and lower walls in a turbulent large-aspect ratio channel would also be a beneficial case study when trying to validate a computer-based model which is intended to be used to analyze fluvial flows. This case offers a fast-changing interaction between the fluid and the boundary surfaces, which can often be found on physical structures built on rivers (being bridges the most obvious scenario).

With the increase on computational resources and the accumulated experience, different applications have been developed in order to obtain some useful information about the properties of fluvial flows. However, it is common to simplify the actual computational analysis to a depth-averaged, steady flow case study. A comparison between how closely the results obtained using these assumptions and the results obtained from full three-dimensional, steady-state simulations, using both laminar and turbulent models, have not been rigorously done. Thus, two test cases were prepared to validate the depth-averaged approach commonly used to analyze fluvial flows. The initial configuration includes a large-aspect ratio channel with a long, surface-piercing rectangular obstacle mounted at a side wall. The final configuration includes a large-aspect ratio channel with upper and lower walls mounted cubic obstacles.

Many studies have been completed using similar channel geometries. These studies include heat transfer [3–6], development and comparison of computational models [7,8], and the analysis of the flow and the vortical structures developed on the channels [3, 4, 9–25], specially for cube arrays and cylinder arrays. These studies will be used as part of the validation of the computational models used on this study.
Chapter 2
Fundamental Concepts

2.1 Computational Fluid Dynamics

Computational Fluid Dynamics (CFD), as stated by Versteeg and Malalasekera [26], is the analysis of physical and theoretical systems involving fluid flow, heat transfer, and associated phenomena by the means of computer-based simulations. The implementation of CFD requires three stages: pre-processing, solver, and post-processing.

During the Pre-processing stage, the input of the problem to a CFD solver takes place. Definition of the geometry of the region, generation of the computational mesh, determination of the fluid flow/heat transfer phenomena to be solved, definition of necessary physical properties of the fluid/material, and specification of the appropriate boundary conditions are some of the basic tasks performed during pre-processing.

The Solver stage consists of the integration of the governing equations over the computational grid of the domain. This task requires the discretization of the governing equations (converting the set of governing differential equations into a set of algebraic equations) and then iterate through the computational domain in order to get the final solution.

After a solution is obtained in the Solver stage, the results are visualized using a graphic unit interface during the Post-processing stage. Even though the simulations are completed at this point, the Post-processing stage demands significant computational power. Most of the time, the datasets output from CFD simulations can be large. Is not uncommon to generate over one TB (1024 gigabytes) of data from a transient CFD simulation. In order to handle all this information, a powerful workstation is needed. Furthermore, since most of the time the CFD simulations are completed with the aid of parallel computing, users need to connect remotely to a computer cluster and thus, connectivity speed also plays an important role in the Post-processing stage.
Due to the limited computational resources that characterized the past, CFD was limited to coarse solutions and small domains. Nevertheless, as the hardware keeps improving, the applications of CFD continue diversifying. From nano-tubes to space shuttles, many applications are using the advantages of CFD.

The governing of fluid motion are known as the Navier-Stokes (NS) equations. The NS equations express the principle of conservation of momentum in fluid flows, and can be easily derived using a control volume such as the one illustrated in Figure 2.1.

Along with the Navier-Stokes equations, the Continuity equation and the Energy equations are also used to solve for the velocity and temperature profiles of fluid systems. These three set of equations are capable of fully describing the classical behavior of fluid flow. Moreover, if the fluid is assumed to be isothermal then the energy equation is not necessary in order to describe the flow properties and characteristics. The Navier-Stokes equations can be written as

$$\rho \left[ \frac{\partial \mathbf{V}}{\partial t} + (\mathbf{V} \cdot \nabla) \mathbf{V} \right] = -\nabla (p + \frac{2}{3} \mu \nabla \cdot \mathbf{V}) + \nabla \cdot [2\mu \tilde{S}(\mathbf{V})] + \mathbf{F}_b$$ (2.1)

where \( t \) is time, \( \rho \) is density, \( \mathbf{V} \) is the velocity vector, \( \nabla \) is the gradient operator, \( \nabla \cdot \) is the divergence operator, \( p \) is the thermodynamic pressure, \( \mu \) is the dynamic viscosity, \( \tilde{S}(\mathbf{V}) \) is the strain-rate tensor, and \( \mathbf{F}_b \) are the body forces per unit volume. If the only body forces affecting the flow are the gravitational forces due to Earth’s attraction then

$$\mathbf{F}_b = -\rho g_0 \nabla Z$$ (2.2)

where \( g_0 \) is the acceleration of gravity at sea-level and \( Z \) is known as the geopotential altitude,

$$Z \equiv \frac{R_E H}{R_E + H}$$ (2.3)

where \( H \) is the geometric altitude. Similarly, the Continuity equation is
Fig. 2.1: Typical three-dimensional control volume used to derive the Navier-Stokes equations in Cartesian coordinates.

\[ \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{V}) = 0. \]  

\text{(2.4)}

Unfortunately, for most of the applications studied up to date, this set of equations does not have an analytical solution. Mathematicians have been trying to find an analytical of closed-form solution for the primitive variables (velocity and pressure) present on this set of equations with no success. Nevertheless, approximate solutions to this set of equations can be obtained by using CFD. To obtain the approximate solution the Continuity equation and the NS equations are discretized over a finite computational domain, and evaluated at specific points (nodes).

For each node, the governing partial differential equations (PDEs) are manipulated and turned into a set of algebraic equations. This algebraic equations include the nature of the original differential problem, as well as the boundary conditions.

The process in which the PDEs are manipulated and turned into algebraic equations is known as the discretization method (DM). Some of the widely used discretization methods include the Finite Difference Method (FDM), the Finite Element Method (FEM), and the Finite Volume Method (FVM).

2.2 Turbulence Modeling

The difference between laminar and turbulent flows is given by the Reynolds number, \( Re \). That is, when the Reynolds number becomes high enough, fluids will drastically change their behavior and become turbulent. When the fluid becomes turbulent, the techniques
used to perform numerical analysis for laminar flows are not valid anymore. One technique used to perform numerical studies of turbulent flows is known as turbulence modeling.

Turbulence modeling is one of the great unsolved problems in classical physics. When fluids are said to be in turbulent motion, the velocity and pressure fields are not unique functions of space and time. When turbulent motion takes place, the velocity and pressure fields have fluctuating components associated with the vortex eddies that develop at high Reynolds numbers. Thus, statistically distinct average values can be discerned for various quantities. Another important characteristic of turbulent flows is their ability to transport and mix fluid properties faster and more effectively than laminar flows.

In turbulent flows, the velocity vector and the thermodynamic pressure can be written as the sum of a mean value and a fluctuating component,

\[ \mathbf{V} = \overline{\mathbf{V}} + \tilde{\mathbf{V}} \]
\[ p = \overline{p} + \tilde{p}. \]  

(2.5)

It is important to mention that such fluctuations are shown to be considerably large (up to 25\% of the mean values). Moreover, the turbulent fluctuations also exhibit variations over a wide range of timescales. However, it is also noticeable that the fluctuating components of velocity and pressure are not completely random: the fluctuations never get to a point in which they become significantly larger than the mean; and the fluctuations always keep oscillating between around 25\% up and 25\% down the mean values, never spending a large amount of time on a particular side.

Using the definitions of Eq. (2.5), the Continuity equation for turbulent flows can be written as

\[ \frac{\partial \rho}{\partial t} + \nabla \cdot [\rho(\overline{\mathbf{V}} + \tilde{\mathbf{V}})] = 0. \]  

(2.6)

After applying the appropriate ensemble-averaging properties, the Continuity equation for turbulent flows and the Reynolds-Averaged Navier-Stokes (RANS) equations can be derived. The equation that states conservation of mass in turbulent flows is given by
\[
\frac{\partial \rho}{\partial t} + \nabla \cdot [\rho \vec{V}] = 0. \tag{2.7}
\]

Similarly, the conservation of momentum for turbulent flows can be expressed as
\[
\rho \left[ \frac{\partial \vec{V}}{\partial t} + (\nabla \cdot \vec{V}) \vec{V} \right] = -\nabla \left( p + \rho g_0 Z + \frac{2}{3} \mu \nabla \cdot \vec{V} \right) + \nabla \cdot [2\mu \tilde{S}(\vec{V})] + g_0 Z \nabla (\rho) - \rho \bar{\vec{V}} \cdot \nabla \bar{\vec{V}}. \tag{2.8}
\]

The last term on the right-hand side of Eq. (2.8) accounts for the convective transport of momentum by the velocity fluctuations. By using vector identities and the Continuity equation, the convective transport of momentum due to turbulent velocity fluctuations can be rewritten as
\[
\rho (\bar{V} \cdot \nabla) \bar{V} = \nabla \cdot (\rho \bar{V} \bar{V}) = \nabla \cdot \left( \rho \begin{bmatrix} V_x V_x V_x V_x V_x V_x \\
V_y V_y V_y V_y V_y V_y \\
V_z V_z V_z V_z V_z V_z \end{bmatrix} \right) \tag{2.9}
\]

where \( V_x, V_y \) and \( V_z \) are the fluctuating components of the velocity vector in Cartesian coordinates. Equation (2.9) results in a Cartesian vector of the form
\[
\rho (\bar{V} \cdot \nabla) \bar{V} = \nabla \cdot (\rho \bar{V} \bar{V}) = \nabla \cdot \left( \rho \begin{bmatrix} \frac{\partial \rho V_x V_x}{\partial x} + \frac{\partial \rho V_y V_x}{\partial y} + \frac{\partial \rho V_z V_x}{\partial z} \\
\frac{\partial \rho V_x V_y}{\partial x} + \frac{\partial \rho V_y V_y}{\partial y} + \frac{\partial \rho V_z V_y}{\partial z} \\
\frac{\partial \rho V_x V_z}{\partial x} + \frac{\partial \rho V_y V_z}{\partial y} + \frac{\partial \rho V_z V_z}{\partial z} \end{bmatrix} \right) i_x
\]

\[
+ \left( \frac{\partial \rho V_x V_y}{\partial x} + \frac{\partial \rho V_y V_y}{\partial y} + \frac{\partial \rho V_z V_y}{\partial z} \right) i_y
\]

\[
+ \left( \frac{\partial \rho V_x V_z}{\partial x} + \frac{\partial \rho V_y V_z}{\partial y} + \frac{\partial \rho V_z V_z}{\partial z} \right) i_z \tag{2.10}
\]

where \( i_x, i_y, \) and \( i_z \) are the unit vectors defined in the Cartesian coordinate system.

It is important to notice that the turbulent velocity fluctuations and the molecular stresses affect the mean flow in a similar way. Thus, a pseudo-molecular stress tensor can be defined according to the relation
\[
\tilde{\sigma} = 2\mu \tilde{S}(\nabla) \left( \bar{p} + \frac{2}{3} \mu \nabla \cdot \vec{V} \right) \tilde{\delta} \tag{2.11}
\]
where \( \mathbf{\delta} \) is the Kronecker delta. The last expression can be further expanded to
\[
\mathbf{\sigma} = \begin{bmatrix}
2\mu \left( \frac{\partial V_x}{\partial x} - \frac{1}{3} \nabla \cdot \mathbf{V} \right) - p \\
\mu \left( \frac{\partial V_y}{\partial x} + \frac{\partial V_x}{\partial y} \right) \\
\frac{\partial V_z}{\partial x} + \frac{\partial V_x}{\partial z}
\end{bmatrix}
\begin{bmatrix}
2\mu \left( \frac{\partial V_y}{\partial y} + \frac{\partial V_x}{\partial z} \right) - p \\
\mu \left( \frac{\partial V_z}{\partial y} + \frac{\partial V_y}{\partial z} \right) \\
2\mu \left( \frac{\partial V_z}{\partial z} - \frac{1}{3} \nabla \cdot \mathbf{V} \right) - p
\end{bmatrix}
\]
and by manipulating Eq. (2.10), the turbulent stress tensor can be defined as
\[
\mathbf{\tau} \equiv -\rho \mathbf{\nabla} \mathbf{V} = -\rho \begin{bmatrix}
V_x V_x & V_x V_y & V_x V_z \\
V_y V_x & V_y V_y & V_y V_z \\
V_z V_x & V_z V_y & V_z V_z
\end{bmatrix}
\]
which means that the last term on the right-hand side of Eq. (2.8) can be expressed as
\[
\rho \left( \mathbf{\nabla} \cdot \mathbf{V} \right) \mathbf{\nabla} = -\nabla \cdot \mathbf{\tau}.
\]
Applying Eq. (2.14) to Eq. (2.8), the formulation of the RANS equations is complete. The final form of the RANS equations is given by
\[
\rho \left[ \frac{\partial \mathbf{V}}{\partial t} + (\mathbf{V} \cdot \nabla) \mathbf{V} \right] = -\nabla \left( p + \rho g_0 Z + \frac{2}{3} \mu \nabla \cdot \mathbf{V} \right) + \nabla \cdot [2\mu \mathbf{S}(\nabla) + \mathbf{\tau}] + g_0 Z \nabla(\rho).
\]
Equation (2.15) and Eq. (2.7) can produce a complete formulation to solve the mean velocity and pressure fields. However, the turbulent stress tensor (also known as the Reynolds stress tensor) contains six unknown fluctuating components. Thus, the objective of a turbulence model is to provide mathematical relations to express the Reynolds stress tensor in terms of the mean flow. The different turbulence models are just different ways to represent the turbulent stress tensor as a function of the mean flow properties.

Over the years many turbulence models have been developed. However, none of them
are found to be fully representative of general turbulent motion, and the fact that one specific model may be more suitable for a specific application is the strongest claim that can be made. Additional information about turbulence modeling can be found in Phillips [27] and Wilcox [28].

2.3 Law of the Wall

In 1930 Johann Nikuradse (one of the students of the famous scientist Prandtl, who first started to study the turbulent motion of fluids) demonstrated through experimentation that the streamwise velocity in the flows near the walls varies logarithmically with distance. Theodore von Kármán (another student under the supervision of Prandtl) suggested an algebraic equation to describe how the streamwise velocity varies for both internal and external flows behaves near the walls, and such expression is commonly known as the law of the wall.

The derivation of the law of the wall can be achieved by performing the parallel flow approximation. As stated before, the governing equations of turbulent fluid motion are given by Eq. (2.15) and Eq. (2.7). In order to model turbulence motion, Joseph Boussinesq introduced the concept of eddy viscosity, and developed a hypothetical behavior of the Reynolds stress tensor, which is given by

\[
\vec{\tau} = 2\mu_t \vec{S}(\nabla) - \frac{2}{3}(\rho k + \mu_t \nabla \cdot \nabla) \delta. \tag{2.16}
\]

Using Eq. (2.16) in Eq. (2.15) and rearranging the terms gives the Boussinesq-RANS equations

\[
\rho \left[ \frac{\partial \nabla}{\partial t} + (\nabla \cdot \nabla) \nabla \right] = -\nabla \left( \bar{p} + \rho g_0 Z + \frac{2}{3}(\mu + \mu_t) \nabla \cdot \nabla \right) + \nabla \cdot [2(\mu + \mu_t) \vec{S}(\nabla)] + g_0 Z \nabla (\rho). \tag{2.17}
\]

The Boussinesq-RANS equations can be further simplified (conveniently) using the definition of a pseudo-mean pressure
Using Eq. (2.18) in Eq. (2.17) the Boussinesq-RANS equations can be re-written as

\[ \rho \left( \frac{\partial \mathbf{V}}{\partial t} + (\mathbf{V} \cdot \nabla) \mathbf{V} \right) = -\nabla \hat{p} + \nabla \cdot [2(\mu + \mu_t)\tilde{\mathbf{S}}(\mathbf{V})] + g_0 Z \nabla (\rho). \] (2.19)

Under the assumption of steady-state, incompressible flow, the Boussinesq-RANS equations can be written as

\[ (\mathbf{V} \cdot \nabla) \mathbf{V} = -\frac{\nabla \hat{p}}{\rho} + \nabla \cdot [2(\nu + \nu_t)\tilde{\mathbf{S}}(\mathbf{V})] \] (2.20)

where \( \nu \) is the kinematic viscosity and \( \nu_t \) is the kinematic eddy viscosity. Similarly, the Continuity equation under the same conditions is given by

\[ \nabla \cdot \mathbf{V} = 0. \] (2.21)

For a steady-state, two-dimensional, incompressible flow in Cartesian coordinates, the mathematical formulation for the Boussinesq-RANS equations can be expanded to

\[ \frac{\partial \mathbf{V}_x}{\partial x} + \frac{\partial \mathbf{V}_y}{\partial y} = 0 \] (2.22)

\[ \mathbf{V}_x \frac{\partial \mathbf{V}_x}{\partial x} + \mathbf{V}_y \frac{\partial \mathbf{V}_x}{\partial y} = -\frac{1}{\rho} \frac{\partial \hat{p}}{\partial x} \plus \frac{\partial}{\partial x} \left[ 2(\nu + \nu_t) \frac{\partial \mathbf{V}_x}{\partial x} \right] + \frac{\partial}{\partial y} \left[ (\nu + \nu_t) \left( \frac{\partial \mathbf{V}_y}{\partial x} + \frac{\partial \mathbf{V}_x}{\partial y} \right) \right] \] (2.23)

\[ \mathbf{V}_x \frac{\partial \mathbf{V}_y}{\partial x} + \mathbf{V}_y \frac{\partial \mathbf{V}_y}{\partial y} = \frac{1}{\rho} \frac{\partial \hat{p}}{\partial y} + \frac{\partial}{\partial x} \left[ (\nu + \nu_t) \left( \frac{\partial \mathbf{V}_y}{\partial x} + \frac{\partial \mathbf{V}_x}{\partial y} \right) \right] + \frac{\partial}{\partial y} \left[ 2(\nu + \nu_t) \frac{\partial \mathbf{V}_y}{\partial y} \right]. \] (2.24)

Under the assumption that \( x \) is the streamwise direction and \( y \) is the normal coordinate (pointing outward, measured from a solid, smooth wall) the no-slip boundary conditions
are given by

\begin{align*}
\nabla_x(x,0) &= 0 \\
\nabla_y(x,0) &= 0.
\end{align*}

(2.25)

If the flow is fully developed, the changes in the mean velocity with respect to the \(x\)-coordinate are small compared to those in the \(y\)-coordinate. Thus, Eq. (2.22), Eq. (2.23), and Eq. (2.24) can be reduced, respectively, to

\begin{align*}
\frac{\partial V_y}{\partial y} &\approx 0 \tag{2.26} \\
V_y \frac{\partial V_x}{\partial y} &\approx -\frac{1}{\rho} \frac{\partial p}{\partial x} + \frac{\partial}{\partial y} \left[ (\nu + \nu_t) \frac{\partial V_x}{\partial y} \right] \tag{2.27} \\
\nabla_y \frac{\partial V_y}{\partial y} &\approx -\frac{1}{\rho} \frac{\partial p}{\partial y} + \frac{\partial}{\partial y} \left[ 2(\nu + \nu_t) \frac{\partial V_y}{\partial y} \right]. \tag{2.28}
\end{align*}

Integrating Eq. (2.26) and using the no-slip condition at the wall as the boundary condition yields

\begin{equation}
\nabla_y \approx 0 \tag{2.29}
\end{equation}

Applying Eq. (2.29) in Eq. (2.28) gives

\begin{equation}
\tilde{p} \approx \tilde{p}(x). \tag{2.30}
\end{equation}

Using Eq. (2.29) and Eq. (2.30) in Eq. (2.27) gives

\begin{equation}
\frac{\partial}{\partial y} \left[ (\nu + \nu_t) \frac{\partial V_x}{\partial y} \right] \approx \frac{1}{\rho} \frac{d\tilde{p}}{dx}. \tag{2.31}
\end{equation}

Integrating Eq. (2.31) with respect to \(y\), from the wall \((y = 0)\) to an arbitrary value of \(y\) gives
\[
\int_0^\nu \frac{\partial}{\partial y} \left[ (\nu + \nu_t) \frac{\partial V_x}{\partial y} \right] \, dy \cong \frac{1}{\rho} \frac{d\hat{p}}{dx} \int_0^\nu dy = \frac{1}{\rho} \frac{d\hat{p}}{dx} y. \quad (2.32)
\]

The expression in the left-hand side of Eq. (2.32) is related to the total shear stress (shear stress resulting from both the molecular effects and the turbulent fluctuations of the fluid motion),

\[
\tau_{\text{total}} = (\mu + \mu_t) \frac{\partial V_x}{\partial y}. \quad (2.33)
\]

The total shear stress evaluated at the wall can be denoted as \(\tau_w\). Thus, Eq. (2.32) and Eq. (2.33) can be combined to form an expression of the form

\[
(\nu + \nu_t) \frac{\partial V_x}{\partial y} - \frac{\tau_w}{\rho} \cong \frac{1}{\rho} \frac{d\hat{p}}{dx} y. \quad (2.34)
\]

For wall bounded flows, the friction velocity is an important velocity scale. It can be defined as

\[
u_t = \sqrt{\frac{\tau_w}{\rho}}. \quad (2.35)
\]

Thus, using Eq. (2.35) in Eq. (2.34) the parallel flow approximation can be formulated as

\[
\nabla_y \cong 0
\]

\[
(\nu + \nu_t) \frac{\partial V_x}{\partial y} \cong u_f^2(x) + \frac{1}{\rho} \frac{d\hat{p}}{dx} y. \quad (2.36)
\]

To fully formulate the near-wall approximations, the parallel flow approximation can be nondimensionalized using the wall-scaled dimensionless variables,
where $y^+$ is the dimensionless wall distance, $u^+$ is the dimensionless streamwise velocity, $p^+$ is the dimensionless axial pressure, and $\nu^+$ is the ratio of the eddy viscosity to the molecular viscosity. Using these variables, the near-wall approximation can be expressed as

$$\frac{dp^+}{dy^+} \approx 0$$

$$\frac{du^+}{dy^+} \approx \frac{1 + p^+ y^+}{1 + \nu^+}.$$  \hspace{1cm} (2.38)

The no-slip condition at the wall and a known pressure away from the wall (flat plate approximation) can be used as the necessary boundary conditions to complete the near-wall approximation,

$$u^+(0) = 0$$

$$p^+(\infty) = 0.$$  \hspace{1cm} (2.39)

Using the boundary conditions defined above, the value of $p^+$ can be evaluated,

$$p^+ = 0.$$  \hspace{1cm} (2.40)

Thus, the near-wall approximation for a flat plate can be written as

$$u^+(0) = 0$$

$$\frac{du^+}{dy^+} = \frac{1}{1 + \nu^+}.$$  \hspace{1cm} (2.41)

In order to solve Eq. (2.41), is necessary to develop special equations to find the quantities directly originated from turbulent motion. One of these equations is known as the
Prandtl’s mixing length theory. Prandtl stated that the kinematic viscosity should be equal to the product of a characteristic velocity and a turbulence mixing length,

\[ \nu_t = l^2 \left| \frac{\partial V_x}{\partial y} \right|. \quad (2.42) \]

When the ratio \( y/\delta \) is very small (where \( \delta \) is the velocity boundary layer thickness), the mixing length is negligible. Using this assumption Eq. (2.41) reduces to

\[
\begin{align*}
    u^+(0) &= 0 \\
    \frac{du^+}{dy^+} &= 1.
\end{align*}
\]

Equation (2.42) can be analytically integrated. This solution yields what is commonly known as the laminar sublayer

\[ u^+ \approx y^+, \quad y^+ < 5. \quad (2.44) \]

Experimental data taken by Prandtl’s students suggested that the mixing length is proportional to the distance from the wall. Thus, Eq. (2.42) can be written in dimensionless form as

\[ \nu^+ = (\kappa y^+)^2 \left| \frac{\partial u^+}{\partial y^+} \right|. \quad (2.45) \]

where \( \kappa \) is the von Kármán constant. In the region far from the wall the ratio of the eddy viscosity to the molecular viscosity is large, so \( 1 + \nu^+ \approx \nu^+ \). Using this approximation Eq. (2.41) can be algebraically manipulated to yield

\[ \frac{du^+}{dy^+} = \frac{1}{\kappa y^+}. \quad (2.46) \]

Equation (2.46) can also be analytically integrated, and the results of commonly known as the Law of the Wall.
\[ u^+ = \frac{1}{\kappa} \ln(y^+) + C \] (2.47)

where C is an unknown integration constant. The law of the wall is considered valid in the interval \( 30 < y^+ < 500 \), and this region is known as the log layer. Modern correlation techniques showed that the values that best described the turbulent flow over a flat plate in the log layer are \( \kappa \approx 0.41 \) and \( C \approx 5.0 \).

As of today, \( u^+ = y^+ \) fits experimental data in the interval \( 0 < y^+ < 5 \) (laminar sublayer) and the law of the wall fits experimental data in the interval \( 30 < y^+ < 500 \) (log layer). However, neither of the two are able to predict the experimental data found over the interval \( 5 < y^+ < 30 \). The region between \( y^+ = 5 \) and \( y^+ = 30 \) is usually known as the transition layer. Figure 2.2 illustrates the logarithmic behavior of the law of the wall.

![Fig. 2.2: Law of the wall for a smooth plate.](image)

More information about the law of the wall and near wall approximations for fluid flows can be found in Phillips [27] and Wilcox [28]. Some of the remarks from the law of the wall are:

- In the range \( 0 < y^+ < 5 \), the Reynolds shear stress is negligible compared to the viscous stress.
• In the range $0 < y^+ < 50$, the viscous contribution to the shear stress is significant.

• In the range $50 < y^+ < 500$, the direct effects of viscosity on the velocity field are negligible.

2.4 Computational Mesh

The PDEs that govern fluid motion cannot be solved algebraically, except for very simple cases. Thus, the physical domain over which the fluid flow analysis is desired must be divided into smaller sub-domains in order to discretize the governing equations and solve them inside of each one of the sub-domains. The sub-domains are made up of primitive geometries (like hexahedra and tetrahedra in 3D, and quadrilaterals and triangles in 2D). The result of dividing the physical domain into smaller sub-domains is known as a computational mesh. A typical computational mesh is illustrated in Figure 2.3. According to how the elements are connected, a computational mesh can be classified into three categories: structured, unstructured, and hybrid meshes.

Fig. 2.3: Side view of the computational mesh of a cone.
2.4.1 Structured Mesh

A structured mesh is characterized by regular connectivity of the elements. The term applied for both 2D and 3D meshes. This kind of computational mesh is suitable for FDM and FVM solvers, but their applications are limited to simple geometries. A typical structured computational mesh is illustrated in Figure 2.4.

2.4.2 Unstructured Mesh

A unstructured mesh is characterized by irregular connectivity of the elements. The term applied for both 2D and 3D meshes. Virtually any geometry can be decomposed into sub-domains using unstructured meshes. They are suitable for FEM solvers and are considered to be very accurate. The downside of the unstructured meshes is that due to the connection irregularity of the elements and the necessity to explicitly save the neighborhood connectivity, they require additional storage (sometimes substantial) when compared to structured meshes. A typical unstructured computational mesh is illustrated in Figure 2.5.

2.4.3 Hybrid Mesh

A hybrid mesh is a computational mesh that contains both structured and unstructured portions. They are commonly used to mesh complicated portions of a physical geometry without burdening the rest. A typical hybrid computational mesh is illustrated in Figure 2.6.
2.5 Finite Volume Method

The FVM is a method in which PDEs are represented and evaluated by algebraic approximations used to enforce the integral conservation law. The values are evaluated at discrete points of a meshed geometry, known as nodes. The term “finite volume” refers to a very small volume surrounding each one of the nodes on a meshed geometry. The terms on the PDE’s are evaluated as fluxes going in and out of the control volumes. Also, since the flux going out of a finite volume is the same flux going in the adjacent finite volume, the FVM is said to be a conservative method. Mass, energy, momentum, and species are always conserved by this method, even in a relatively coarse mesh.

One of the advantages of the FVM over the FDM is that they can be used on both structured and unstructured meshes, while the FDM requires a structured mesh to be used.
The FVM has been proven to perform well when trying to solve PDEs with discontinuous coefficients. It can also be used for elliptic, parabolic and hyperbolic PDEs, making it a versatile discretization method suitable for many problems and engineering applications.

The major disadvantage of the FVM when compared to the FDM is the increased complexity of the models, the increased computational time, and decreased computational efficiency.

2.6 SIMPLE Algorithm

The SIMPLE algorithm, as stated by Versteeg and Malalasekera [26], is essentially a guess-and-correct procedure for the calculation of thermodynamic pressure over a staggered computational mesh arrangement. When the SIMPLE algorithm is used, the velocity field is approximated using the momentum equations and a guessed value of the pressure distribution, then the pressure distribution is obtained from the a formulation of the pressure equation, and finally the velocity field is corrected and a new set of fluxes is calculated.

2.7 Parallel Computing

Parallel computing is a computation technique in which calculations are carried out simultaneously by two of more computer processors. The basic operational principle is that a large problem can be divided into smaller problems, and these smaller problems can be solved simultaneously. Ideally, parallel computing allows computations to be performed faster because of the multiple computational processors being used. Nevertheless, in practice is not always possible to divide a task into several sub-tasks and have a processor complete each sub-task without interfering with each other.

Due to their complexity and the computational time required to simulate the interactions of fluids and gases, parallel computing is largely used as the ultimate resource to perform CFD simulations. However, sometimes billions of computations are required to catch the interactions of fluids and gases with their boundary surfaces, which leads to one of the major questions about parallel computing and CFD: scalability.
The terms scalability refers to the ability of a large problem to be decomposed into several sub-problems and still keep decreasing the computational time required to obtain the solutions. As the number of processors is increased, the internal communication to perform the calculations and obtain the solution of the problem is also increased. Thus, there is a point in which adding more processors does not decrease the computational time to solve a problem, and in many CFD applications going beyond this point can actually slow down the solutions.
Chapter 3

Problem Description

With the ultimate goal to analyze and comprehend fluvial flows, a two-step approach was employed through the development of this thesis. Step 1 involves the study and analysis of the flow through large aspect-ratio channels with obstacles normal to the streamwise direction using OpenFOAM [29] using laminar and turbulence models. Step 2 consists on the study and analysis of of the flow through large aspect-ratio channels with obstacles normal to the streamwise direction using MD_SWMS [30]. In order increase the scope of the study both laminar and turbulent models will be used, even though literature suggest that fluvial flows are strictly turbulent.

Moreover, the proposed study includes the numerical simulation of flows over open channels at high Reynolds number using the steady-state, isothermal assumption. To achieve this goal, two different open-channel geometries were prepared. The first geometry consists on a large aspect-ratio channel with a long, rectangular, surface-piercing obstacle mounted at the lower wall (referred from now on as Test Case 1). The second geometry (referred from now on as Test Case 2) consists on a large aspect-ratio channel with multiple cubic obstacles mounted at the upper wall (2 blocks) and lower wall (1 block).

3.1 Turbulent Flows Through Large Aspect-Ratio Channels

The turbulent flow through large aspect-ratio channels has been the subject of many researchers, and thus the available documentation about such flow is vast and objective. The flow through pen channels not only has been studied by using CFD but also by using experimental techniques such as PIV (Particle Image Velocimetry) [31]. Different experimental setups were also prepared in order to obtain more data about the behavior of fluid flows through large aspect-ratio channels. Most of the studies related to open-channel flows
have been performed using computationally demanding turbulence models such as DNS, LES, and DES, as discussed in [14,16,22,23,32–34]. These studies also revealed the details of channel flows, such as vortical structures, energy transportation, heat transfer, etc.

3.1.1 Test Case 1

In order to verify the results of the simulation performed on Test Case 1, observations taken by previous researchers will be used. An experimental visualization technique of Lagrangian coherent structures caused by turbulence in aperiodic flows through a large aspect ratio channel with a surface-piercing rectangular obstacle mounted at a side was developed by Chrisohoides and Sotiropoulos [2]. Thus, the general findings of Test Case 1 are already known. Additionally, computational results for a similar open-channel configuration were already found by Paik and Sotiropoulos [32].

From the work previously done by Chrisohoides and Sotiropoulos [2], the dynamics and general behavior of the flow through the large aspect-ratio channel configuration with a rectangular, surface-piercing obstacle was observed to be as follow: when the flow approaches the rectangular block, it encounters a relatively strong transverse pressure gradient that diverts it around the rectangular block. This transverse pressure gradient is responsible of the formation of a large region of recirculating flow at the upstream connection between the rectangular block and the channel side wall, characterized by multiple large-scale eddies which are apparently quasi-periodic. Also, a larger recirculating region slowly develops at the downstream end of the rectangular block. Finally, a shear layer also develops from the end of the rectangular block, caused by the interaction of the slow moving flow of this region and the flow diverted around the rectangular block.

This project intends to capture these characteristics using a computer-based model, and then use this model on a three-dimensional geometry configuration resembling the actual physical domain of a common fluvial-like flow. The results obtained from the computational model must agree qualitatively and quantitatively to those generated by the simulation of the large aspect-ratio channel implemented by Paik and Sotiropoulos [32], since a similar configuration is being used.
In order to capture the dynamics related to this test case and validate the computational model, the computer-based simulations will be set to match a Reynolds number of $1.21 \times 10^5$, based on the bulk velocity and the hydraulic diameter of the channel. The Froude number of the fluid will be assumed to be 0.4, and the ratio of the obstacle length to the flow level and to the channel width will be set to 13.0 and 0.22, respectively. The computational domain will extend $8L_1$ (where $L_1$ is the length of the rectangular obstacle) upstream and $31L_1$ downstream the rectangular block. The physical geometry of the channel configuration is illustrated in Figure 3.1.

It is essential to declare that for simulation purposes, the Reynolds number will be defined throughout this study as

$$Re = \frac{D_H |\bar{V}|}{\nu}$$  \hspace{1cm} (3.1)

where

$$D_H = \frac{4A}{F}.$$ \hspace{1cm} (3.2)

### 3.1.2 Test Case 2

The configuration of a large aspect ratio channel with multiple cubic obstacles mounted at the upper wall (2 blocks) and lower wall (1 block) is analogous to the interaction of bridges’ supports with water. This is a fast-changing fluid dynamic application that is often not studied in depth by fluvial flow researchers.
Korichi and Oufer [6] performed numerical studies related to the flow through a large aspect-ratio channel with multiple cubic obstacles mounted at the side walls. Using their findings, validation data for a second test case is available. Similar to the previous channel configuration studied, the working fluid flowing through this channel configuration will encounter a series of transverse pressure gradients when approaching the cubic obstacles, which will lead to the formation of several recirculation regions and vortical structures. The formation of shear layers is also expected for this test case.

The physical domain for this simulation consists on a channel with dimensions of 0.25 units depth ($L_2$) and 1.0 unit width. Thus, the characteristic length of the cubic obstacle is $L_2$. The space between the end of an obstacle and the beginning of another obstacle is $2.0L_2$. The Reynolds number for the configuration will be set to $1.0 \times 10^5$, based on the hydraulic diameter of the channel and the bulk velocity. The computational domain extends $16L_2$ upstream the first obstacle, and $56L_2$ downstream the third obstacle. The geometry of this channel is shown in Figure 3.2.

Due to this convenient and still applicable simplification of geometry and mutating geomorphology, this work can be considered as an initial step toward the validation of a computational model capable of describing the nature and behavior of fluvial flows with all its implications.

### 3.2 Laminar Flows Through Large Aspect-Ratio Channels

Laminar flow through large aspect-ratio channels is not nearly as studied as turbulent flows. The turbulent features such as increased mixing, heat transfer, and coherent vortical structures are among the largest motivators for these numerical studies. However, by performing a laminar study the validation of the way in which the Navier-Stokes equations are solved on OpenFOAM and MD_SWMS without the additional differences added by the
turbulence models used on each CFD package.

3.2.1 Test Case 1

CFD simulations using the same channel configurations will be performed using laminar solvers. In order to ensure that a converged solution can be obtained, the Reynolds number of the original turbulent CFD simulations will be decreased. For the large aspect-ratio channel with a single wall-mounted obstacle, the Reynolds number used to perform the laminar simulations will be set to $3.0 \times 10^4$.

3.2.2 Test Case 2

Keeping the same geometry and external features the same (geometry, aspect ratios, etc.), the Reynolds number corresponding to the laminar CFD simulation over the large aspect-ratio channel with the three wall-mounted obstacles will be set to $1.8 \times 10^4$.

3.3 Turbulent Fluvial Flows

The importance of studying fluvial flows to understand its implications is the opportunity to comprehend how a river can produce such a big impact in modern life. Bridge design, environmental planning, and future techniques to determine the optimal growth of towns and cities can be highly improved by fully understanding, modeling, and predicting the impact of fluvial flows.

The first reason why a deeper study on the rich dynamics of fluvial flows is needed is the fact that no computational model is capable of including the effects of the complex topography of the environment and the transportation and deposition of particles on the Navier-Stokes equations. Similarly, to predict the rich dynamics encountered in fluvial flows it is necessary to employ expensive computational power (even using the most powerful computers used to perform CFD simulations, the time necessary to have a sufficient level of detail is very large). Finally, the generation of a computational mesh with a high degree of approximation to the real topography of a river is tedious and computationally expensive.
According to previous studies performed by Bates et al. [35], the attempt to perform numerical studies of fluvial flow is also influenced by the fact that in order to obtain good qualitative and quantitative results from a computer-based simulation, an extremely fine grid must be used to capture the high variability of the driving forces and the turbulent structures. Also, the grid required to model a fluvial environment with fast changes in cross-sectional areas can lead to numerical instability.

All environmental and fluvial systems are formed from the interaction of many other “sub-systems”, which may not be well characterized and/or documented. Also, model validation data may be unavailable due to the difficulty of performing good measurements of all the properties involved in fluvial flows. The time and space scales on which properties are observed on fluvial flows are very different to those used on a computer-based simulation. Due to particle transportation, the local roughness of the bed of fluvial flows is also very difficult to determine with any precision.

In recent years researchers started to focus on more versatile and still realistic computational models capable of achieving the proper characteristics of fluvial flows. Hardy [36] concluded that in order to model fluvial flows, the importance of topography as a factor contributing to the inherent characteristics of the fluvial flow must be included. It is also known that in order to simulate fluvial flows some parameters and flow characteristics are said to be known a priori, when in fact they are a product of the flow itself.

Many computational models incorporate schemes such as roughness parametrisation and depth-averaging. The issues involved with any kind of parametrisation in fluvial flows is that the real parametric function that describes the changes in surface roughness on fluvial flows is near to impossible to obtain. Currently several parametrisation schemes can be observed [35] but none of them prove to be valid for all situations, they show to be dependent on specific applications. Also, due to this dependency, the task to determine which parametrisation scheme performs better is not trivial.

Depth-averaging is a simplification scheme widely used by fluvial flows and geomorphology researchers. This scheme eliminates the uncertainty coming from the parametrisation
procedure. However, there is a common issue that arises from both depth-averaging and parametrisation schemes: what is the accuracy of the model? As mentioned before, the ability to measure any kind of property on fluvial flows so they can be used as validation data can be extremely difficult. However, the accuracy of such schemes must be determined in order to predict flows in fluvial environments.

With the current increase in computational resources, many numerical models and software capable of predicting sediment mobility, perform flood reconstruction, perform bridge-pier analysis, and obtain additional information directly related with the dynamic of the fluvial flow motion have been developed. One of these software applications is MD_SWMS (Multi-Dimensional Surface-Water Modeling System), which is being extensively used by scientists studying surface-water hydraulics.

Moreover, MD_SWMS simplifies the models to be steady-state and two-dimensional (by depth-averaging). This simplification allows MD_SWMS to generate results considerably faster than the time required to obtain results from a three-dimensional unsteady turbulent simulation. Conversely, according to Hardy [36], fluvial flows modeling can be improved by using a three-dimensional physically based approach. Similarly, the nature of fluvial flows indicate that they can only be accurately examined using three-dimensional unsteady models. The reason behind this is that properties can vary spatially in the three dimensions, as well as with respect to time. Thus, a study showing the level of details reached by three-dimensional turbulent simulations versus those obtained from MD_SWMS is required.

To achieve this comparison, a physical fluvial geometry needs to be used. For instance, the geometry presented in Figure 3.3 corresponds to a section of the Strawberry River that flows through Wasatch and Duchesne Counties in the Uinta Basin region of Utah. The data used to obtain this geometry was collected by researchers from the College of Natural Resources at Utah State University. Thus, data to create a computational mesh for a typical fluvial flow can be acquired.

However, even though the data needed to create a meshed river geometry can be obtained, some other computational aspects play a role. Mesh non-orthogonality, open
computational cells, cells aspect ratio, and the orientation of faces are just some of the issues that can be found when trying to mesh a river geometry.

Given that the major advantage of MD_SWMS to its users is the speed in which results are generated, this research intends to give the users an idea of the trade-off of computational time versus level of details obtained. Does increasing the number of nodes reflect a significant change in the results generated by MD_SWMS? This and other questions regarding MD_SWMS versus full three-dimensional, steady-state turbulent simulations will be answered by the end of this research.

For all intents and purposes, the creation of a full three-dimensional model capable of accurately describing fluvial flows is extremely complex. Instead of trying to accurately find the solution to all the situations computational scientists encounter when trying to analyze and predict the behavior of fluvial-like flows, the focus will be made on only a fraction of the problem: a computational model capable of predicting the behavior of the turbulent

Fig. 3.3: Channel geometry for a section of the Strawberry River, Utah.
structures that are found in fluvial flows. Thus, the study includes, but is not limited to, the formation of eddies within the flow, the frequency of appearance of such eddies, pressure gradients, and the development of shear layers.

Consequently, modeling the effects of material transport (such as erosion, formation of deltas, material removal and deposition, changes on roughness, etc.), performing studies of unsteady fluvial flows, and a more efficient grid generation technique will be left for a future study.
Chapter 4

Thesis Project Motivation and Objectives

The motivation to perform the research and complete this project is the urgent necessity to comprehend and learn how to accurately predict the impact and behavior of fluvial flows. More than ten thousand years ago, the migratory activities of man were strongly influenced by fluvial flows; today the tendency of such flows to impact human society remains, but shows a different face.

In the present, due to the significant advance of science and engineering, human beings do not have the necessity to live close by a river to enjoy the privilege of fresh water. Nevertheless, every time bridges supports succumb to scouring and catastrophically fail, the impact that river can potentially have in society is proven to be still large.

When rivers and other kind of Earth-surface water flows go inside of a town, destroying everything they encounter, and every time a hydroelectric power plant needs to be artificially fed from several rivers (making it not sustainable) are also examples of the drastic changes in the way of living of species that can be ignited by such flows.

Thus, increasing the understanding of fluvial flows is a source, undervalued in many occasions, of major improvements for human race. Since the rate at which bridge supports are getting weaker faster that expected due to scouring, transportation is a field in which accurately predicting the behavior of fluvial flows will represent a significant contribution. Based on these facts, the motivation for this project is to analyze the way in which many geophysical researchers are investigating fluvial flows.

Because of the time that is required to perform a numerical simulation that includes the actual physical conditions encountered in fluvial flows with accuracy (three-dimensional domain, the flow is highly turbulent, the geomorphology mutates over time, both the flow and the sediment transportation show high time dependency, multi-phase interactions, heat
transfer, etc.), most researchers opt for an easier way to do it.

Steady-state, two-dimensional, isothermal conditions, and no interaction between air and water are just some of the most popular assumptions used to perform numerical simulations of fluvial flows. However, a comparison between the results of a simulation performed under these assumptions and those of a simulation performed at least assuming that the domain is three-dimensional is not officially documented.

Moreover, this project intends to document some of the advantages and disadvantages resulting from the implementation of depth-averaged solutions to study and analyze fluvial flows. Also, the suitability of OpenFOAM, an open-source CFD package, to model fluvial-like flows will be studied.
Chapter 5
Formulations and Computational Models

The numerical model used to perform the three-dimensional computer-based simulations is simpleFoam, a steady-state, RANS, incompressible turbulent solver from the OpenFOAM package. To simulate the turbulent motion of the fluid, the standard $k-\varepsilon$ model with wall functions will be used. For this case, all the physical properties of the fluid are assumed to be constant. The fluid is assumed to be Newtonian, viscous, and the effects of buoyancy are assumed to be negligible. The simulation setup includes a structured mesh. It is important to mention that the details of the standard $k-\varepsilon$ model will not be discussed on this dissertation, but it can be found on [27,28,37].

The depth-averaging algorithm being tested corresponds to FaSTMECH, a quasi-3D, steady-state, incompressible turbulent solver from the MD_SWMS package. MD_SWMS is widely used by surface-water researchers, thus a comparison of these two solvers is convenient for the research community.

5.1 Setup of Test Case 1 Using Turbulence Models in OpenFOAM

5.1.1 Computational Mesh and Geometry Considerations

The first step required to complete a CFD simulation for Test Case 1 is to create a computational mesh. Due to the relatively simple geometry being studied, a structured mesh was selected for this case. The three-dimensional computational mesh used for this test case is illustrated in Figure 5.1. The computational mesh is tested to ensure it meets the following requirements: all faces are oriented properly, the non-orthogonality of the cells should not be high, the aspect ratio of the cells is below 2.0, and the meshed domain should be fully closed to ensure that the conservation laws are valid.
The characteristic length used to describe the channel geometry is $L_1 = 1\, m$. In order to ensure that the length of the inlet section of the channel does not affect the results, the upstream face of the obstacle is located at $L_{\text{inlet}} = 6L_1 = 6\, m$. Similarly, to ensure that the outflow has minimum effect on the flow fields, and also that the large recirculation region that will develop downstream of the obstacle is inside the domain, the length from the downstream face of the obstacle to the end of the domain is set to $L_{\text{outlet}} = 28L_1 = 28\, m$. An illustration of this geometry is shown in Figure 5.2.

The final dimensions of the channel ended up being 39.1 m in length (along the x-direction, from $x = -1$ to $x = 38.1$), 4.55 m in width (along the y-direction, from $y = 0$ to $y = 4.55$), and 0.077 m in depth (along the z-direction, from $z = 0$ to $z = 0.077$). Gmsh
was used to create a mesh out of the given geometry, and it contained over 2.22x10^6 cells.\(^1\)

As appreciated in Figure 5.1, the computational mesh is finer around the rectangular obstacle in comparison to the other sections. The geometry was meshed in this way in order to capture the details encountered when the flow approaches the upstream face of the obstacle, without burdening the rest of the computational domain.

5.1.2 Boundary Conditions

After the computational mesh is created, the boundary conditions of the problem are applied. Proper boundary conditions are needed for the velocity field, the thermodynamic pressure, the turbulent kinetic energy (denoted as \(k\)), and the turbulent kinetic energy dissipation rate (denoted as \(\varepsilon\)). An important note to make is that SI units are used to define all variables being used.

The boundary condition for the velocity field at the inlet was set to match the average mean velocity required to give the previously specified Reynolds number. In order to match a Reynolds number of 1.21x10^5 (based on the bulk velocity and hydraulic diameter of the channel) and a Froude number of 0.4, and assuming the working fluid is water at 20°C, the bulk velocity is assumed to be

\[
\mathbf{V}_{\text{inlet}} = (0.8i_x + 0.0i_y + 0.0i_z) \text{ m/s.}
\]  

At the two side walls and the bottom wall, the velocity is set to zero,

\[
\mathbf{V}_{\text{walls}} = (0.0i_x + 0.0i_y + 0.0i_z) \text{ m/s.}
\]

For the outlet boundary conditions, the velocity gradients normal to the outflow face (the y-z plane) are set to zero,

\[\nabla_{\text{outlet}} = (0.0\text{i}_x + 0.0\text{i}_y + 0.0\text{i}_z) \text{ m/s.} \]  

\(^1\)The number of cells used to mesh the geometry was determined by trial and error. The procedure in which the minimum number of cells was selected included running a CFD simulation, calculating the wall-scaled distance \(y^+\), and correcting the number of cells so \(y^+\) is between 30 and 150 for most of the boundary walls.
\[
\left( \frac{\partial V_x}{\partial x} \right)_{\text{outflow}} = 0. \quad (5.3)
\]

At the top of the channel, the slip wall boundary condition gives

\[
\begin{align*}
V_{z,\text{topwall}} &= 0 \\
\left( \frac{\partial V_x}{\partial x} \right)_{\text{topwall}} &= 0 \\
\left( \frac{\partial V_y}{\partial y} \right)_{\text{topwall}} &= 0.
\end{align*} \quad (5.4)
\]

Assuming that the turbulent velocity fluctuations are 15\% the value of the mean velocity, and that the characteristic length used to find the turbulent energy dissipation rate is 5\% of the channel depth, the value of \( k \) used at the inlet of the pre-simulation channel is

\[
k_{\text{inlet}} = 0.0096 \, m^2/s^2. \quad (5.5)
\]

At the side walls and bottom wall, \( k \) is defined to be

\[
k_{\text{walls}} = 0.0 \, m^2/s^2. \quad (5.6)
\]

At the end of the outlet section of the channel, the corresponding boundary condition is

\[
\left( \frac{\partial k}{\partial x} \right)_{\text{outflow}} = 0. \quad (5.7)
\]

And at the top wall, the specified boundary condition for \( k \) is

\[
\left( \frac{\partial k}{\partial z} \right)_{\text{outflow}} = 0. \quad (5.8)
\]

By using the information employed to find \( k \), the prescribed inlet value for \( \varepsilon \) was found to be

\[
\varepsilon_{\text{inlet}} = 0.0231 \, m^2/s^3. \quad (5.9)
\]
At the walls, the Newmann boundary condition is used for the value of $\varepsilon$ at the walls,

\[
\begin{align*}
\left( \frac{\partial \varepsilon}{\partial y} \right)_{\text{sidewalls}} &= 0, \\
\left( \frac{\partial \varepsilon}{\partial z} \right)_{\text{bottomwall}} &= 0.
\end{align*}
\] (5.10)

And at the top wall, the slip boundary condition states that

\[
\left( \frac{\partial \varepsilon}{\partial z} \right)_{\text{topwall}} = 0.
\] (5.11)

Instead of having the absolute pressure, a pressure relative to the inlet will be used. Thus, at the inlet the pressure boundary condition is

\[
\begin{align*}
\mathcal{P}_{\text{inlet}} &= 0.
\end{align*}
\] (5.12)

For the bottom and side walls, the pressure satisfies Eq. (5.13)

\[
\begin{align*}
\left( \frac{\partial \mathcal{P}}{\partial y} \right)_{\text{sidewalls}} &= 0, \\
\left( \frac{\partial \mathcal{P}}{\partial z} \right)_{\text{bottomwall}} &= 0.
\end{align*}
\] (5.13)

In the outflow plane, the pressure boundary condition gives

\[
\left( \frac{\partial \mathcal{P}}{\partial x} \right)_{\text{outflow}} = 0.
\] (5.14)

Finally, the slip wall boundary condition states that at the top wall, the pressure boundary condition is

\[
\left( \frac{\partial \mathcal{P}}{\partial z} \right)_{\text{topwall}} = 0.
\] (5.15)

To summarize the boundary conditions for Test Case 1, all normal gradients go to zero at the outflow plane. The slip wall boundary condition states that if the quantity is a scalar field the normal gradients are set to zero; and if the quantity is a vector field then
the normal components go to zero, and the gradients of the tangential components go to zero. At the walls, Neumann boundary conditions are used for $\varepsilon$, and Dirichlet boundary conditions are used for $\nabla$, and $k$. The inlet values are specified for each individual field variable.

5.2 Setup of Test Case 1 Using Turbulence Models in MD_SWMS

5.2.1 Computational Mesh and Geometry Considerations

The aspect ratios used to define the geometry for Test Case 1 in OpenFOAM were the same used in MD_SWMS. Nevertheless, stability issues may arise when trying to perform a CFD simulation in MD_SWMS using a channel depth of 0.077 m. Thus, the characteristic length used to define Test Case 1 in OpenFOAM was increased from $L_1 = 1$ m to $L_1 = 4$ m. All the aspect ratios of the channel are the same OpenFOAM and MD_SWMS are the same, so dimensional homogeneity is conserved. The mesh created to run the initial simulations for Test Case 1 in MD_SWMS ended up having $5.0 \times 10^4$ cells.\(^2\)

5.2.2 Boundary Conditions

Applying boundary conditions in MD_SWMS is not as demanding as in OpenFOAM. To define a geometry in MD_SWMS, points defining the cross-section of the channel are used. Once the geometry is imported properly, an structured, orthogonal mesh is created. Three valued must be specified to perform CFD simulations using MD_SWMS: inlet discharge, outflow stage, and a value for the isotropic eddy viscosity.

To keep the the Reynolds number a constant value of $1.21 \times 10^5$, the bulk velocity used for this configuration must be decreased by a factor of 4,

$$\nabla_{inlet} = (0.2i_x + 0.0i_y + 0.0i_z) \text{ m/s.} \quad (5.16)$$

Thus, the inlet flow discharge can be defined as

\(^2\)The final number of cells used was found by trial and error.
\[ Q_{inlet} = |\nabla_{inlet}| A = 1.12 \text{ m}^3/\text{s} \]  \hspace{1cm} (5.17)

where \( A \) is the cross-sectional area of the channel.

The outflow stage, \( f_s \) is a measurement of the average depth in the outflow depth of the channel over which the quasi-3D simulation is being performed. Since all the channels included on this study are homogeneous in the \( (z) \)-direction, the flow stage is a constant value,

\[ f_s = 4(0.077 \text{ m}) = 0.308 \text{ m}. \]  \hspace{1cm} (5.18)

To threat turbulence, the Reynolds stresses are related to the shear using the isotropic eddy viscosity. For this simulation, this value was used as

\[ \nu_t = 0.005 \text{ m}^2/\text{s}. \]  \hspace{1cm} (5.19)

### 5.3 Setup of Test Case 1 Using Laminar Models in OpenFOAM

The solver simpleFoam can be used without solving for turbulence in OpenFOAM. To improve the stability of the numerical solutions of these simulations, the Reynolds number was decreased from \( 1.21 \times 10^5 \) to \( 3.0 \times 10^4 \).

#### 5.3.1 Computational Mesh and Geometry Considerations

The same computational meshes used to run simpleFoam with turbulence models were used to run simpleFoam without turbulence models. Thus, the only parameters that changed for these set of simulations when compared to the previously defined turbulent simulations were the boundary conditions.

#### 5.3.2 Boundary Conditions

Since the Reynolds number was decreased, the inlet velocity must be adjusted. To match a Reynolds number of \( 3.0 \times 10^4 \), the initial condition for the inlet velocity must be set
to

\[ \nabla_{inlet} = (0.2i_x + 0.0i_y + 0.0i_z) \text{ m/s}. \]  

(5.20)

The pressure boundary conditions remain the same as those used for the case using turbulence models, and the turbulent quantities \((k \text{ and } \varepsilon)\) can be neglected from the model.

5.4 Setup of Test Case 1 Using Laminar Models in MD_SWMS

By selecting a value of zero for the isotropic eddy viscosity, laminar flow models can be used in MD_SWMS. For this simulation setup, the Reynolds number selected was \(3.0 \times 10^4\) based on the bulk velocity and the hydraulic diameter of the channel.

5.4.1 Computational Mesh and Geometry Considerations

The computational mesh used to run the turbulent simulations of Test Case 1 on MD_SWMS was also used for these simulations. Refer to the previous sections for more information about the computational mesh and geometry considerations for Test Case 1.

5.4.2 Boundary Conditions

To match a Reynolds number of \(3.0 \times 10^4\), and taking into account that the configuration used in MD_SWMS is 4 times larger than this for OpenFOAM, the initial condition for the inlet velocity must be set to

\[ \nabla_{inlet} = (0.05i_x + 0.0i_y + 0.0i_z) \text{ m/s}. \]  

(5.21)

Thus, the inlet flow discharge can be defined as

\[ Q_{inlet} = |\nabla_{inlet}| \ A = 0.28 \text{ m}^3/\text{s}. \]  

(5.22)

The outflow stage remains intangible, since is a measurement of the average depth in the outflow depth of the channel over which the quasi-3D simulation is being performed.
5.5 Setup of Test Case 2 Using Turbulence Models in OpenFOAM

5.5.1 Computational Mesh and Geometry Considerations

For the completion of the computer-based simulation on the large-aspect ratio channel previously described and denoted as Test Case 2, a structured mesh was used to discretize the computational domain. The computational mesh was tested to ensure the minimum quality requirements of non-orthogonality, orientation of the faces in the three-dimensional space, cell aspect ratio, and openness of the domain are all meet. The three-dimensional computational mesh used for this test case is illustrated in Figure 5.3.

Due to the increased number of obstacles with respect to the geometry studied in Test Case 1, a total of $1.87 \times 10^6$ cells were required to mesh the channel geometry (considerable larger number of cells than those required to mesh Test Case 1).

For this case, the characteristic length used to define the geometry is $L_2 = 0.25 \, \text{m}$. As done on the previous case, minimum inlet and outlet section lengths $L_{\text{inletB}}$ and $L_{\text{outletB}}$, respectively, were set to $16L_2$ (measured from the inlet of the channel to the upstream face of the first cubic obstacle) and $44L_2$ (measured from the downstream face of the third obstacle to the end of the outlet section). Again, this is to ensure that the inlet and the outlet sections do not affect the results, and also to include the recirculation regions in the computational domain. The considerations described above are shown in Figure 5.4. The final dimensions of the channel ended up being $19.75 \, \text{m}$ in length (along the $x$-direction, from $x = 0$ to $x = 19.75$), $1 \, \text{m}$ in width (along the $y$-direction, from $y = 0$ to $y = 1$), and $0.1 \, \text{m}$ in depth (along the $z$-direction, from $z = 0$ to $z = 0.1$).

5.5.2 Boundary Conditions

The boundary conditions used for Test Case 2 are essentially identical to those used for Test Case 1. SI units are used to define all the flow fields and geometric variables.

---

3The number of cells used to mesh the geometry for Test Case 2 was determined by trial and error. The procedure in which the minimum number of cells was selected included running a complete CFD simulation, calculating the wall-scaled distance $y^+$, and correcting the number of cells so $y^+$ is between 30 and 150 for most of the boundary walls.
To match a Reynolds number of \(1 \times 10^5\) (based on the hydraulic diameter of the channel), and assuming the working fluid is water at 20°C, the bulk velocity is approximated to be \((0.55, 0.0, 0.0)\) m/s. Assuming that the turbulent velocity fluctuations are 15% the value of the mean velocity, and that the characteristic length used to find the turbulent energy dissipation rate is 5% of the channel depth, the values of \(k\) and \(\varepsilon\) at the inlet are found to be \(0.003\) \(m^2/s^2\) and \(0.009\) \(m^2/s^3\), respectively.

The pressure is set to zero at the inlet, and the gradients are set to zero at the walls. At the outlet, all gradients go to zero. Also, the slip wall condition is applied at the top wall of the channel to all flow variables. Finally, the boundary conditions for the sides and bottom walls are treated as for Test Case 1.
5.6 Setup of Test Case 2 Using Turbulence Models in MD_SWMS

5.6.1 Computational Mesh and Geometry Considerations

The aspect ratios used to define the geometry for Test Case 2 in OpenFOAM were also the same used in MD_SWMS. However, the characteristic length was increased from $L_2 = 0.25 \, m$ to $L_2 = 0.75 \, m$ (increased by a factor of 3) in order ensure stability of the computational results. The mesh created to run the initial simulations for Test Case 2 in MD_SWMS ended up having 5.0x10$^4$ cells$^4$.

5.6.2 Boundary Conditions

Analogous to the procedure used to complete the MD_SWMS simulations for Test Case 1, a set of points defining the cross-sectional area of the channel were used. Once the geometry is imported properly, an structured, orthogonal mesh is created. Three valued must be specified to perform CFD simulations using MD_SWMS: inlet discharge, outflow stage, and a value for the isotropic eddy viscosity.

To keep the Reynolds number a constant value of 1.0x10$^5$, the bulk velocity used for this configuration must be decreased by a factor of 3,

$$\nabla_{inlet} = (0.183i_x + 0.0i_y + 0.0i_z) \, m/s. \quad (5.23)$$

The inlet flow discharge was defined as

$$Q_{inlet} = |\nabla_{inlet}| \, A = 0.165 \, m^3/s. \quad (5.24)$$

The outflow stage for this setup was given by

$$fs = 3(0.1 \, m) = 0.3 \, m. \quad (5.25)$$

To threat turbulence, the value of the isotropic eddy viscosity used was

$^4$The final number of cells used was found by trial and error.
\[ \nu_t = 0.005 \, m^2/s. \] (5.26)

5.7 Setup of Test Case 2 Using Laminar Models in OpenFOAM

5.7.1 Computational Mesh and Geometry Considerations

The computational mesh and geometry considerations for this portion of the study are the same as those for the simulations performed using turbulence models in OpenFOAM. Refer to the previous sections for more information about the surface boundaries and computational meshes used for the present simulations.

5.7.2 Boundary Conditions

The Reynolds number used for this portion of the study was selected to be 1.81x10^4. Thus, the mean velocity necessary to match this Reynolds number is \( (0.1i_x + 0.0i_y + 0.0i_z) \) m/s. The boundary conditions for pressure remain unaltered.

5.8 Setup of Test Case 2 Using Laminar Models in MD_SWMS

5.8.1 Computational Mesh and Geometry Considerations

The computational mesh and geometry considerations for this portion of the study are the same as those for the simulations performed using turbulence models in MD_SWMS. For further explanations refer to the previous sections.

5.8.2 Boundary Conditions

By changing the Reynolds number from 1.0x10^5 to 1.81x10^4 the mean inlet velocity changed from \( (0.55i_x + 0.0i_y + 0.0i_z) \) m/s to \( (0.1i_x + 0.0i_y + 0.0i_z) \) m/s. This velocity sets the inlet flow discharge to 0.03 m^3/s. The outflow stage remains static, and the isotropic eddy viscosity is set to zero in order to use laminar models.
5.9 Validation of the Numerical Model

To determine if the three-dimensional, steady-state turbulent simulations used for this project are generating conclusive profiles, two well-known test cases will be studied before attempting to obtain results from MD-SWMS. Given that the current research is spotlighting the turbulent structures developing in fluvial flows, the problem was simplified to the analysis of the flow through large aspect ratio channels. In order to include the turbulent structures developing on fluvial flows and induced by their complicated geometries, rectangular obstacles will be mounted at the side walls of the channels.

As observed by Chrisohoides and Sotiropoulos [2] and later by verified by Paik and Sotiropoulos [32], the coherent vortical structures developing on a large aspect ratio channel due to a long, rectangular, surface-piercing obstacle mounted at a side wall are qualitative analogous to those encountered on fluvial flows.

Furthermore, the turbulent structures and multiple recirculation regions generated inside of a large-aspect ratio channel with multiple cubic, surface-piercing obstacles mounted at the side walls are in many aspects homogeneous to the turbulent structures found in bridges built on rivers. Further descriptions of channel flows relevant to this validation process can be found in [5, 6, 16–18, 20, 31, 33, 34, 38, 39].

With the two cases described above, validation of the numerical models used to perform the CFD simulations can be acquired.
Chapter 6

Results

CFD simulations using simpleFoam were completed for the geometries specified on Test Case 1 and Test Case 2. The results of these simulations, as well as the results obtained from MD SWMS (FaSTMECH) are summarized in this chapter.

6.1 Test Case 1: OpenFOAM using Turbulent Models

6.1.1 Mesh Quality

The software package OpenFOAM contains many post- and pre-processing utilities that were used to complete this work. One of these is OpenFOAM’s capability to check the quality of a given mesh (called checkMesh) before running a CFD simulation. After using this pre-processing utility, could be seen that the mesh non-orthogonality test passed, the face orientation test passed, the boundary openness test passed, and the cell aspect ratio test passed.

Still, a value of 31 for the maximum aspect ratio of a cell seems high, but an extremely fine mesh would be required to take this number down. The reason is because the region showing cells with relatively large aspect ratio are found near the rectangular obstacle (where the mesh was constructed to be very fine in the xy-plane) and near the walls (where the mesh was constructed to be very fine in the y-direction). To fix this issue, the mesh will need to be refined in the x- and z-directions, and this would dramatically increase the total number of cells on the discretized geometry, and due to the present hardware limitations this option was not feasible. Since most of the flow is moving in the x-direction downstream the obstacle, the results will be still valid if the length $\Delta x$ is several times larger than the lengths $\Delta y$ and $\Delta z$. This issue can be improved by mesh refinement.
6.1.2 Initial Results

The velocity distribution of the depth-averaged, converged solution from simpleFoam can be observed in Figure 6.1. The distribution of the velocity magnitude showed, as expected, typical behavior of turbulent steady-state incompressible flows. As observed by [2, 32], large vortical structures would have formed when the flow approaches the upstream wall of the obstacle, if the flow is studied as an unsteady problem. Since steady-state assumptions were made for the development of this work, visualization of vortical structures and other unsteady flow properties will be impossible to achieve. Nevertheless, the flow follows the expected behavior.

The velocity distributions are scaled by the mean velocity in all cases, so that the solutions from OpenFOAM and MD_SWMS can be compared side-by-side. The small region of slow-moving flow upstream the obstacle, and the relatively large region of slow-moving flow downstream the obstacle can be clearly observed. These recirculation regions also agree with the work done by [2,32]. A close-up view of the two recirculation regions is shown in Figure 6.2.
In order to perform the simulations and decrease the number of cells required to generate a valid solution, wall functions were used as the methodology to solve the flow equations near the walls. The other option that can be used to solve the flow equations near the walls is to directly integrate down the walls. However, most researchers claim that in order to obtain a valid CFD solution the computational mesh should have at least 10 points/cells below $y^+ = 5.0$ (the laminar sublayer) and the first point/cell should be at $y^+ < 1.0$, which may require a very fine mesh.

Instead of directly solving the flow equations down to the walls, the usage of wall function is an alternative that does not require a computational mesh as fine as that required to directly solve the flow equations down to the walls. Thus, the turbulence model selected to fulfil this purpose ended up being the standard $k-\varepsilon$ model with wall functions. When using wall functions, the flow equations are forced to follow the Law of the Wall, and the first point/cell out from the walls needs to be anywhere between $y^+ = 30$ and $y^+ = 120$ a solution to be considered valid.
Figure 6.3 illustrates the values of $y^+$ at the bottom wall for Test Case 1. As appreciated, all the values of $y^+$ are within the specified range for the law of the wall. It can be also observed that the portion of the computational mesh which contains the highest values of $y^+$ is the region where the velocity increases, making the value of $y^+$ close to 95. Such a value of $y^+$ is on the ideal range for turbulence models with wall functions. To obtain the value of $y^+$ in the proper range for the side walls is not nearly as challenging as for the bottom wall, so only the bottom wall is showed in this study.

6.1.3 Mesh Refinement

In order to ensure that the solutions can be considered as valid, a mesh refinement (or grid refinement) test was completed (which involved the completion of three simulations). As can be seen from Figure 6.4, the velocity profiles at different $x$-locations follow the same patterns. Moreover, using a finer mesh improves the overall shape of the fully developed turbulent profile. However, the impact of the mesh refinement can be better seen on the values of $y^+$. Thus, the velocity profiles can be considered as grid converged.

Figure 6.5 presents the values of $y^+$ for two simulation setups: the initial simulation setup with $2.22 \times 10^6$ cells and a the refined case with $4.1 \times 10^6$ cells. When comparing the two plots, the main difference can be perceived immediately. The majority of the computational cells for the mesh-refined case shows values of $y^+$ between 50 and 90, which is the ideal scenario for the implementation of turbulence models with wall functions.

Furthermore, the converged velocity profiles for the three CFD simulations corresponding to Test Case 1 are shown in Figure 6.6. As appreciated, high velocity regions, recirculation regions, and velocity magnitudes behave in the same way for both cases. Thus, based on the velocity distribution, the velocity profiles, and the values of $y^+$, the simulation setup with $2.22 \times 10^6$ cells is said to be a mesh-converged solution. It is important to mention that the computational meshes used for the OpenFOAM simulations were obtained using Gmsh [40,41].
Fig. 6.3: Pseudocolor plot: values of $y^+$ at the bottom wall of the converged CFD simulation for Test Case 1, using OpenFOAM with turbulence models.

6.1.4 Computational Uncertainties

Even though a CFD simulation was performed to draw all the conclusions for this work, a discussion on computational uncertainty may be possible. Errors in the model (due to modeling assumptions and approximations), numerical errors (due to the numerical solutions of the PDEs), and input errors (errors in the final results due to the input parameters of the simulation) can be addressed.

Errors in the model refer to those that arise from uncertainty in the formulation of the model. For this work, modeling the turbulent structure will always be a model error (because turbulence is still an unsolved problem, and is not thoroughly understood). Also, the parameters $k$ and $ε$ contain uncertainty. Finally, the steady-state assumption does not correspond to the phenomena observed in reality by researchers and thus, another source of model uncertainty takes play.

Numerical errors are the most commonly discussed errors in CFD. The uncertainty added from round-off errors (which develop with the representation of floating point numbers on the computer and the accuracy at which numbers are processed and stored), iterative convergence (which arises from the fact that when running the CFD simulation, converge...
is said to be achieved when the residuals of velocity, pressure, \( k \) and \( \varepsilon \) between two iteration falls below \( 1 \times 10^{-6} \) so there are still differences small differences between the solutions from two consecutive iterations, and discretization errors (which are the errors that occur from the representation of the governing equations of fluid flows into algebraic equations in a discretized domain). The numerical schemes used to perform this work are said to be second-order accurate, which are included as part of the discretization errors of the CFD simulation.

**Input errors** are often underestimated. The main impact of input errors is that since most of the time they are underrated, solutions can be successfully converged but the conclusions arrived from the simulation can be erroneous. The prescribed inlet velocity profile is one of the entries that can be prescribed wrong as an input parameter. Also, the usage of boundary conditions, and the setup of the turbulence model are other important sources of input errors.

It is important to mention that this work does not intend to quantify the overall uncertainty of the CFD simulations. Further discussion on uncertainty and validation of simulations was documented by Coleman and Steele [42].

### 6.2 Test Case 1: MD\_SWMS using Turbulent Models

In order to compare the results obtained from the CFD simulations from OpenFOAM, a second set of computational experiments was completed. A computer-based simulation using FaSTMECH (Flow and Sediment Transport and Morphological Evolution of Channels), one of the computational models available on the MD\_SWMS package, was successfully completed. FaSTMECH is a two-dimensional, depth-averaged and quasi-3D steady-state flow solver. As previously recognized, the original depth of 0.077 m of Test Case 1 will result in an unstable solution. To get around the issue, the channel was scaled up by a factor of 4. Furthermore, since the Reynolds number used to set up this simulation used the channel depth as the characteristic length, the magnitude of the mean inlet velocity was decreased by a factor of 4.
6.2.1 Initial Results

Figure 6.7 illustrates the velocity distribution through the complete computational domain. As appreciated, the velocity distribution of the solution from FaSTMECH agrees well with that obtained with simpleFoam (Figure 6.1). Furthermore, the regions of small-moving flows were predicted qualitatively on the same zones by both numerical solvers.

Fig. 6.4: Two-dimensional velocity profiles of the depth-averaged, converged CFD simulation for Test Case 1 as a function of distance $y$. As presented; a) corresponds to a simulation setup with $4.1 \times 10^6$ cells, b) corresponds to a simulation setup with $3.2 \times 10^6$ cells, and plot c) corresponds to a simulation setup with $2.22 \times 10^6$ cells. Sub-figure A) is the velocity profile at $x = 10L_1$, B) is the velocity profile at $x = 15L_1$, and c) is the velocity profile at $x = 18L_1$. For all cases, the simpleFoam solver with turbulence models was used.
However, simpleFOAM shows more precise predictions of the recirculation regions. The recirculation regions can be better seen in Figure 6.8.

Fig. 6.5: Pseudocolor plot: values of $y^+$ at the bottom wall of the converged results from the mesh-refined CFD simulation for Test Case 1, using OpenFOAM with turbulence models. As presented, a) correspond to a computational mesh with $2.22 \times 10^6$ cells and b) corresponds to a computational mesh with $4.1 \times 10^6$ cells.
Fig. 6.6: Pseudocolor plot: velocity magnitudes scaled by the channel mean velocity for Test Case 1, obtained using OpenFOAM with turbulence models. As presented, a) corresponds to a meshed geometry with $2.22 \times 10^6$ cells, b) corresponds to a meshed geometry with $3.2 \times 10^6$, and c) corresponds to a meshed geometry with $4.1 \times 10^6$ cells.
6.2.2 Mesh Refinement

A grid refinement study on the solutions generated by FaSTMECH is not intuitive. The developers recommend to refine the computational mesh and repeat the calculations in order to ensure that the obtained results and consistent. Thus, a total of three simulations were completed and the velocity distributions always showed an overall similar behavior (i.e. the velocity magnitude scaled by the mean channel velocity was between 1.803 and 1.950, the velocity distributions showed similar patterns, etc.) Figure 6.9 illustrates partial results of the grid refinement analysis.

To have a more accurate perspective of the mesh-converged results, velocity profile lines were taken at different locations downstream the channel. Based on this information, solutions generated on a computational mesh larger than $7.0 \times 10^4$ cells are said to be mesh-converged. Figure 6.10 illustrates the velocity profile lines used to perform the analysis.

6.3 Test Case 1: OpenFOAM using Laminar Models

The laminar solution for the geometry prescribed by Test Case 1 converged to satisfactory solutions. Figure 6.11 illustrates the scaled velocity distribution.
Similarly, the mesh refinement study for this configuration revealed that the computational results can be considered mesh-converged after using a mesh size of $2.22 \times 10^6$ cells, as can be seen in Figure 6.12 and Figure 6.13.

### 6.4 Test Case 1: MD SWMS using Laminar Models

The successful completion of this simulation indicated the ability of MD SWMS to correctly predict the velocity distribution for laminar flows. Once the mesh refinement analysis was completed, MD SWMS showed consistency on the estimated results. Figure 6.14 and Figure 6.15 are used to support this statement. To verify that the results were mesh converged, the scaled velocity profile lines at two different location for two different mesh sizes is shown in Figure 6.16.

Fig. 6.8: Velocity vector plots showing the main recirculation regions upstream and downstream the obstacle, obtained using the turbulent MD SWMS solution.

### 6.5 Test Case 2: OpenFOAM using Turbulent Models

#### 6.5.1 Mesh Quality

The pre-processing utility used to verify the quality of the computational mesh available through OpenFOAM was utilized in Test Case 2. The mesh non-orthogonality test passed, the face orientation test passed, the boundary openness test passed, and the cell aspect
ratio test passed. An important keypoint is that the non-orthogonality of this mesh is lower than this for Test Case 1, but the skewness is slightly higher. Also, the cell aspect ratio for the mesh used to discretize the geometry for Test Case 2 is also lower than that for Test Case 1 (5.92). Later will be shown how the cell aspect ratio can be lowered through mesh refinement.

6.5.2 Initial Results

The basic flow features of the three-dimensional results obtained with simpleFoam agree with those found by Korichi and Oufer [6], but since their work was obtained using unsteady turbulence models, full agreement was not achieved. Several recirculation regions can be appreciated upstream and downstream of each cubic obstacle. Due to the adverse pressure gradients, the recirculation regions downstream the obstacles are generally larger than those originated upstream. Since the CFD simulation used to generate these results was completed under the steady-state assumption, vortical structures and several other turbulent phenomena will not be visualized. Nevertheless, visualization of the main recirculation regions is obtained.

Figure 6.17 is a vector plot of the depth-averaged velocity, and shows a total of six recirculation regions: three small regions upstream each cubic obstacle, and three larger regions downstream each obstacle. The magnitude of the scaled velocity field (using the channel mean velocity as the scaling parameter) can be observed in Figure 6.18. As expected, the velocity profile follows the typical behavior of a viscous, incompressible, steady-state fluid flow.

The values of $y^+$ were monitored and reported, in order to ensure that the computational mesh follows the guidelines required to use turbulence models with wall functions. Wall functions for the left wall and the right wall were easily enforced, since the velocity profiles do not change dramatically. However, keeping the value of $y^+$ in the proper range for the bottom wall was not as straightforward. Due to the cubic obstacles and their effect on the velocity field, a fine computational mesh was required when attempting to keep the value of $y^+$ below 200. As can be seen in Figure 6.19, at the regions where the velocity if
high, the values of $y^+$ are slightly above 170 in some areas. Such values are still considered good enough to use turbulence models with wall functions and get a valid solution, but are a somewhat far from the ideal range of $y^+$ previously described on this chapter. The validity of a turbulence model with wall functions when the value of $y^+$ goes up to 200 will be further discussed.

6.5.3 Mesh Refinement

As part of the general procedure to perform and report CFD simulation, a mesh refinement study was completed. The two-dimensional scaled-velocity profiles of the solutions generated by computational meshes containing $1.87 \times 10^6$, $2.10 \times 10^6$, and $2.80 \times 10^6$ cells are compared in Figure 6.20 for different $x$-locations. As observed, the three solutions follow closely the same flow patterns. The converged scaled, depth-averaged velocity magnitudes for the coarser- and finer-mesh solutions can be found on Figure 6.21.

Using the refined computational mesh also helped reduce the values of $y^+$, particularly on the bottom wall. The range of $y^+$ values for the refined computational mesh is from 11.44 to 107.85 on the right wall, from 10.25 to 131.02 on the left wall, and from 1.78 to 148.1 on the bottom wall. These ranges are ideal for turbulence models with wall functions. Refer to Figure 6.22 for a post-processed version of the information given about $y^+$.

Finally, the overall quality of the mesh was also improved by mesh refinement. As expected, the skewness ended up lower, the non-orthogonality was quantized slightly lower, and the maximum cell aspect ratio also ended up slightly lower than this for the original mesh (from 5.92 to 5.48).

6.5.4 Computational Uncertainties

The sources of computational uncertainties that apply to Test Case 1 also apply to Test Case 2. Refer to Coleman and Steele [42] for insides on how to quantize uncertainties for numerical simulations.
Fig. 6.9: Velocity distributions for different mesh sizes at different locations, obtained using the turbulent MD-SWMS solution. As shown, a) corresponds to a computational mesh with $5.0 \times 10^4$ cells, b) corresponds to a computational mesh with $7.0 \times 10^4$ cells, and c) corresponds to a computational mesh with $9.0 \times 10^4$ cells.
Fig. 6.10: Velocity profile lines of the converged solutions, obtained using the turbulent MD_SWMS solution. As shown, A) corresponds to a location $x = 10L_1$, B) corresponds to a location $x = 15L_1$ and C) corresponds to a location $x = 18L_1$. The legend a) represents a computational mesh with $9.0 \times 10^4$ cells, b) represents a computational cell with $7.0 \times 10^4$ cells, and c) represents a computational mesh with $5.0 \times 10^4$ cells.
Fig. 6.11: Scaled velocity distribution obtained using the laminar simpleFoam solution for Test Case 1.
Fig. 6.12: Scaled velocity distribution obtained using the laminar simpleFoam solution for different mesh sizes. As shown, a) correspond to a mesh size of $2.22 \times 10^6$ cells, and b) corresponds to a mesh size of $4.1 \times 10^6$ cells.
Fig. 6.13: Scaled velocity profile lines obtained using the laminar simpleFoam solution for different mesh sizes. As shown, a) correspond to a mesh size of $2.22 \times 10^6$ cells, and b) corresponds to a mesh size of $4.1 \times 10^6$ cells. As shown, A) corresponds to a location $x = 10L_1$, B) corresponds to a location $x = 22.5L_1$, and C) corresponds to a location $x = 25L_1$. 
Fig. 6.14: Scaled velocity distribution obtained using the laminar FaSTMECH solution for Test Case 1.
Fig. 6.15: Scaled velocity distribution obtained using the laminar FaSTMECH solution for different mesh sizes used to study Test Case 1. As shown, a) correspond to a mesh size of 5.0x10^4 cells, and b) corresponds to a mesh size of 9.0x10^4 cells.
Fig. 6.16: Scaled velocity profile lines obtained using the laminar FaSTMECH solution for different mesh sizes (Test Case 1). As shown, a) correspond to a mesh size of $5 \times 10^4$ cells, and b) corresponds to a mesh size of $9 \times 10^4$ cells. As shown, A) corresponds to a location $x = 22.5L_1$, and B) corresponds to a location $x = 25L_1$.

Fig. 6.17: Vector plot of the converged velocity field showing the different recirculation regions of Test Case 2, using solution obtained with simpleFoam with turbulence models.
Fig. 6.18: Pseudocolor plot: velocity magnitude scaled by the channel mean velocity of the converged solution for Test Case 2, using simpleFoam with turbulence models.

Fig. 6.19: Pseudocolor plot: values of $y^+$ at the bottom wall of the converged CFD simulation for Test Case 2, using simpleFoam with turbulence models.
Fig. 6.20: Two-dimensional velocity profiles of the averaged, converged CFD simulation for Test Case 2 as a function of distance $y$, obtained using simpleFoam with turbulence models. As shown, the label a) corresponds to a computational mesh containing $2.8 \times 10^6$ cells, b) corresponds to a computational mesh containing $2.1 \times 10^6$ cells, and c) corresponds to a computational mesh with $1.87 \times 10^6$ cells. Sub-figure A) is the velocity profile at $x = 13.32L_2$, B) is the velocity profile at $x = 17.8L_2$, and C) is the velocity profile at $x = 30L_2$. 
Fig. 6.21: Pseudocolor plot: scaled velocity magnitudes of the depth-averaged converged solution for Test Case 2, using simpleFoam with turbulence models. As shown, a) performed on a computational mesh with $1.87 \times 10^6$ cells, b) performed on a computational mesh with $2.1 \times 10^6$, and c) performed on a computational mesh with $2.8 \times 10^6$ cells.
Fig. 6.22: Pseudocolor plot: values of $y^+$ at the bottom wall of the converged solution from simpleFoam. As shown, a) performed on a computational mesh with $1.87 \times 10^6$ cells, b) performed on a computational mesh with $2.1 \times 10^6$, and c) performed on a computational mesh with $2.8 \times 10^6$ cells.
6.6 Test Case 2: MD_SWMS using Turbulent Models

6.6.1 Initial Results

The results from the turbulent MD_SWMS simulations for Test Case 2 were satisfactory. The converged velocity distribution can be found in Figure 6.23. As appreciated, Figure 6.23 is comparable to Figure 6.18 in most aspects.

![Velocity distribution of the turbulent solution obtained from FaSTMECH, converged on a mesh size of 5.0x10^4 cells.](image)

Fig. 6.23: Velocity distribution of the turbulent solution obtained from FaSTMECH, converged on a mesh size of 5.0x10^4 cells.

6.6.2 Mesh Refinement

The scaled-velocity distribution utilized for the mesh refinement analysis is shown in Figure 6.24. As appreciated, the results agree qualitative well with those from the simpleFoam solutions in Figure 6.21. Similarly, scaled velocity profile lines were obtained to determine of the solutions are mesh-converged more accurately. As can be seen from Figure 6.25, the solutions are mesh converged when using at least 5.0x10^4 cells.

Not as clearly as for the simulation performed with simpleFoam, but the slow-moving and recirculation regions can be also found on the results from FaSTMECH. One more time, SimpleFoam shows to be superior to FaSTMECH when predicting the accurate location and
size of the recirculation regions, but they both agree in a general sense. The recirculation regions can be seen in Figure 6.26.

Fig. 6.24: Velocity distributions of the turbulent solution obtained from FaSTMECH. As shown, a) corresponds to a solution converged on a computational mesh with $5 \times 10^4$ cells, and b) corresponds to a solution converged on a computational mesh with $8.2 \times 10^4$ cells.
Fig. 6.25: Velocity profile lines of the turbulent solution obtained from FaSTMECH. The legend a) corresponds to a solution converged on a computational mesh with 8.2x10^4 cells, and b) corresponds to a solution converged on a computational mesh with 5.0x10^4 cells. As shown, sub-figure A) is the velocity profile at \(x = 13.32L_2\), B) is the velocity profile at \(x = 17.8L_2\), and C) is the velocity profile at \(x = 30L_2\).

Fig. 6.26: Velocity vector plot showing the recirculation regions from the FaSTMECH solution.
6.7 Test Case 2: OpenFOAM using Laminar Models

The laminar study for Test Case 2 using simpleFoam showed a similar velocity distribution as the turbulent simulation completed using simpleFoam, as shown in Figure 6.27. Through the mesh refinement analysis was proven that solutions obtained using a computational mesh with $1.87 \times 10^6$ cells were accurate enough to capture all the flow properties of interest, as can be deduced from Figure 6.29, Figure 6.28, and Figure 6.30.

![Scaled velocity distribution](image1)

**Fig. 6.27:** Scaled velocity distribution obtained using the laminar simpleFoam solution for Test Case 2.

![Velocity vector plot](image2)

**Fig. 6.28:** Velocity vector plot showing the recirculation regions from the laminar simpleFoam solution for Test Case 2.
Fig. 6.29: Scaled velocity distribution obtained using the laminar simpleFoam solution for different mesh sizes. As shown, a) correspond to a mesh size of $1.87 \times 10^6$ cells, and b) corresponds to a mesh size of $2.8 \times 10^6$ cells.
Fig. 6.30: Velocity profile lines of the laminar solution obtained from simplFoam. The legend a) corresponds to a solution converged on a computational mesh with $2.8 \times 10^6$ cells, and b) corresponds to a solution converged on a computational mesh with $1.87 \times 10^6$ cells. As shown, sub-figure A) is the velocity profile at $x = 13.32L_2$, and B) is the velocity profile at $x = 48L_2$. 
6.8 Test Case 2: MD_SWMS using Laminar Models

As expected, the laminar solutions from MD_SWMS agree with those from simpleFoam in a general sense. However, the velocity profiles are consistently different from the two solvers. This issue will be further discussed. Figure 6.31 shows the dimensionless velocity distribution for two different meshes. Not only the solutions are about the same but also the major and minor recirculation regions are formed in the same place. A vector plot is also provided in Figure 6.32 to facilitate the visualization of the recirculation regions.

![Scaled velocity distribution obtained using the laminar FaSTMECH solution for different mesh sizes. As shown, a) correspond to a mesh size of 5x10^4 cells, and b) corresponds to a mesh size of 9x10^4 cells.](image)
6.9 Comparison of Turbulence Simulations: OpenFOAM and MD_SWMS

6.9.1 Test Case 1

Overall, the results obtained when using the turbulent, depth-averaging scheme available through FaSTMECH generated not very similar results to those obtained by running a three-dimensional simulation in simpleFoam using turbulence models with wall functions and then performing the depth-average. In addition, when the velocity is scaled by the mean velocity for each simulation, the resulting dimensionless velocity was found to be approximately within the same range of values. This fact can be observed in Figure 6.33, Figure 6.34, and Figure 6.35. Besides the differences of the velocity profiles obtained by the two CFD solvers, FaSTMECH also tends to under-predict the values of velocity.

6.9.2 Test Case 2

Besides all turbulent structures being better described by simpleFoam, the tendency of the turbulence model implemented in FaSTMECH to under-predict the velocities can also be observed on Test Case 2. Also, the velocity distributions were significantly different for MD_SWMS, specially downstream the last cubic obstacle. Thus, MD_SWMS revealed a weakness when trying to predict the velocity profiles for flows with high geometry-induced turbulent structures.

Fig. 6.32: Velocity vector plot showing the recirculation regions from the laminar FaSTMECH solution for Test Case 2.
Fig. 6.33: Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 1, using turbulence models, plotted against the scaled dimensionless distance $y/y_{\text{max}}$, corresponding to a location $x = 10L_1$.

The difference in velocity profiles can be also seen in Figure 6.36, Figure 6.37, and Figure 6.38.
Fig. 6.34: Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 1, using turbulence models, plotted against the scaled dimensionless distance $y/y_{max}$, corresponding to a location $x = 15L_1$. 
Fig. 6.35: Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 1, using turbulence models, plotted against the scaled dimensionless distance $y/y_{max}$, corresponding to a location $x = 18L_1$. 
6.10 Comparison of Laminar Simulations: OpenFOAM and MD_SWMS

6.10.1 Test Case 1

When testing this configuration, FaSTMECH showed to overpredict the general velocity profiles far from the rectangular obstacle. However, the velocity values in the vicinity of the obstacle are under-predicted. The velocity profile lines for MD_SWMS and OpenFOAM are shown in Figure 6.39 and Figure 6.40.

6.10.2 Test Case 2

The analysis of the effect of multiple surface-piercing obstacles mounted at the sides of a channel using laminar models revealed, again, one of the major weakness of FaSTMECH: difficulty to correctly predict velocity distributions in the presence of multiple geometry-induced recirculation regions. The magnitude of the velocity distribution decreases sooner after passing the last cubic obstacle when using laminar models with FaSTMECH, a behavior also observed on the results predicted by FaSTMECH with turbulence models. The velocity profile lines for MD_SWMS and OpenFOAM are shown in Figure 6.41 and Figure 6.42.
Fig. 6.36: Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 2, using turbulence models, plotted against the scaled dimensionless distance $y/y_{max}$, corresponding to a location $x = 13.32L_2$. 
Fig. 6.37: Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 2, using turbulence models, plotted against the scaled dimensionless distance $y/y_{\text{max}}$, corresponding to a location $x = 17.8L_2$. 
Fig. 6.38: Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 2, using turbulence models, plotted against the scaled dimensionless distance $y/y_{\text{max}}$, corresponding to a location $x = 30L_2$. 
Fig. 6.39: Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 1, using laminar models, plotted against the scaled dimensionless distance $y/y_{max}$, corresponding to a location $x = 10L_1$. 
Fig. 6.40: Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 1, using laminar models, plotted against the scaled dimensionless distance $y/y_{max}$, corresponding to a location $x = 22.5L_1$. 

Fig. 6.41: Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 2, using laminar models, plotted against the scaled dimensionless distance $y/y_{\text{max}}$, corresponding to a location $x = 13.32L_2$. 
Fig. 6.42: Scaled velocity profile lines of the converged solutions (finest computational meshes) for Test Case 2, using laminar models, plotted against the scaled dimensionless distance $y/y_{\text{max}}$, corresponding to a location $x = 48L_2$. 
Chapter 7

Conclusions and Future Work

In order to accurately predict the behavior of fluvial flows, important pieces of information, not easily available and most of the times tedious to obtain, are needed. Velocity measurements, shear stresses, sediment transportation, and habitat analysis are some of the properties needed to perform model validation studies. However, sustainable conclusions can be made after the termination of this study.

Moreover, based on a stronger agreement of simpleFoam on observations performed by Chrisohoides and Sotiropoulos [2], the claim of its superiority when compared to FaSTMECH will be made. For mesh-converged solutions, the velocity distributions of both numerical solvers agree qualitatively (i.e. the velocity distribution profile plots) and quantitatively (i.e. the ratio of the largest velocity magnitude to the average mean inlet velocity). However, the recirculation regions predicted by FaSTMECH do not agree with the experimental observations as strongly as those predicted by simpleFoam. It was found that no matter how fine the mesh, the results from FaSTMECH will never be as accurate as those from simpleFoam. In addition, FaSTMECH tends to underpredict the velocity in situation where turbulent structures are highly induced by the channel geometry.

Generally, the velocity distributions obtained using FaSTMECH reveal fluctuations. Most of the time, the velocity distributions are under-predicted by FaSTMECH, specially in the presence of more than one geometry-induced recirculation region. Furthermore, the number of turbulence models that can be used for OpenFOAM applications is far superior to those that can be used with MD_SWMS models. Similarly, geometry boundaries can be precisely meshed using Gmsh and OpenFOAM. The geometries imported to MD_SWMS models are limited to simple, relatively small meshed surfaces. All these tools make the OpenFOAM package a powerful tool for fluid flow modeling.
Nevertheless, for a general study of fluvial flows, FaSTMECH is considerably faster and easier to use. The reasons are mainly two (even though they are closely related): time and computational resources. To perform valid three-dimensional simulations using simpleFoam, the following steps were followed:

1. Set up and run simulation on complete channel geometries (5 hours using parallel computing and 64 processors per case, total CPU hours = 1280 to run two laminar and two turbulent simulations).

2. Post-process results to verify that $y^+$ is inside the proper range (total CPU time = 30 minutes).

3. If $y^+$ is not inside the proper range, refine computational mesh and repeat steps 1-2.

4. If $y^+$ is inside the proper range, post-process results to perform depth-average and visualize flow properties (total CPU time = 30 minutes).

As can be seen, the CPU time needed to complete the two cases ascends to at least 1281 CPU hours. In order to use this approach is essential to use parallel computing. Also, it is important to know that a relatively large computer cluster is needed to speed up the calculations required for a converged three-dimensional solution using simpleFoam, or any other turbulent solver from the OpenFOAM package. This is not the case for FaSTMECH.

The time required to perform the simulations needed to generate results for Test Case 1 and Test Case 2 would not exceed 48 hours (using a single processor, as MD_SWMS does not supports parallel computing), including the time required to learn the software and a mesh refinement analysis. Such a significant decrease in the amount of time required to generate results when compared to the three-dimensional solvers of the OpenFOAM package contribute to the motivate researchers to choose FaSTMECH over simpleFoam or other similar CFD package when studying fluvial flows.

MD_SWMS also includes post-processing capabilities so the results can be visualized without the usage of another software, which is not the case for OpenFOAM. Table 7.1 contains a side-by-side comparison of MD_SWMS and OpenFOAM.
Furthermore, three different versions of OpenFOAM were used to complete this project (even though only two are required), making the process to obtain solutions even more burdensome. Table 7.2 contains information about the usage of each version of OpenFOAM.

With all this said, the fact is that the low accuracy of FaSTMECH, specially to solve flows with high geometry-induced turbulent structures, is an issue. Some of the factors contributing to the limitations of MD_SWMS as an accurate CFD solver for fluvial flows found through this research include, but are not limited to:

1. Level of detail of the geometries that can be imported and meshed.

2. The validation data used to validate the results from FaSTMECH was obtained using 1D models, which limits the accuracy of FaSTMECH to this of a 1D model.

3. The characterization of the velocity and channel geometry to incorporate the depth-averaging scheme.

4. The model used to solve the turbulent closure problem is an algebraic model with limited accuracy.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>OpenFOAM</th>
<th>MD_SWMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difficulty to setup</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>CPU hours required to obtain solutions</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Complexity of meshed geometries</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Boundary conditions used</td>
<td>Flexible</td>
<td>Fixed</td>
</tr>
<tr>
<td>Requires additional software</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Implementation of diverse turbulence models</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Open-source</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>License</td>
<td>Free</td>
<td>Free</td>
</tr>
<tr>
<td>Accuracy of final results</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Available official documentation</td>
<td>Unsatisfactory</td>
<td>Satisfactory</td>
</tr>
<tr>
<td>Handling parallel computations</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

In the future, to improve the computational accuracy of the MD_SWMS package, a more effective depth-average technique would be desirable. The ideal solution is to modify the solvers in the OpenFOAM package and include a depth-averaged version of solvers.
such as simpleFoam. Given the significant advantage of required time and computational resources of the approach followed by MD_SWMS, a similar application with parallel computing support would represent a significant advance in fluvial flow modeling. Thus, even though the OpenFOAM package is robust, it does not present itself as the optimal way to quickly study fluvial flows. Moreover, additional data (such as velocity measurements, shear stress measurements, etc.) to perform validation studies of the numerical solvers used to complete fluvial flows simulations would need to be considered in order to definitively choose the superior software package. Also, calibration of FaSTMECH with 2D and/or 3D models will be highly beneficial.

In addition, the literature available about large-aspect ratio channels suggests that Low-Re turbulence models, such as Launder-Sharma $k - \varepsilon$ model or the Lam-Bremhorst $k - \varepsilon$ should be used. Comparing the results from these two models with those from the standard $k - \varepsilon$ model with wall functions will be important to document as well.

The main weakness of FaSTMECH and MD_SWMS may arise from either the depth-averaging scheme of from the turbulence model being used to solve the turbulent closure problem. The question of which one of these factors corresponds to the larger contributor to the lack of accuracy of FaSTMECH is left for a future study.

---

Table 7.2: Usage of the Different OpenFOAM Versions on the Current Study

<table>
<thead>
<tr>
<th>Function</th>
<th>Yes</th>
<th>No</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating a mesh with no renumbered cells for inlet mapping</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Build the meshes with the boundary definitions</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Final inlet mapping</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Use simpleFoam or similar numerical solver</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Obtain $y^+$</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Post-process results for visualization</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Capability to depth-average results</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
References


Appendices
Appendix A

Gmsh files used to create mesh for Test Case 1 and Test Case 2

A.1 Gmsh file for Test Case 1

// Variable Bank (SI Units used) cl = 0.05; // Characteristic length for grid
L = 1.0; // Obstacle length
D = 0.077; // Fluid depth
B = 4.55*L; // Channel width
w = 0.1; // Obstacle width
n1 = 55; // Number of points per meter, coarse grid
n2 = 65; // Number of points per meter, fine grid
n = 14; // Number of points per meter, second grid
pr1 = 1.01;
pr2 = 1.03;
pr3 = 1.01;

// Points to define geometry
Point(1) = {-1.*L,0.0,cl};
Point(2) = {-1.*L,2*L,0,cl};
Point(3) = {-1.*L,B,0,cl};
Point(4) = {5.*L,0.0,cl};
Point(5) = {5.*L,2*L,0,cl};
Point(6) = {5.*L,B,0,cl};
Point(7) = {7.*L,0.0,cl};
Point(8) = {7.*L,B,0,cl};
Point(9) = {7.*L + w,0.0,cl};
Point(10) = {7.*L + w,L,0,cl};
Point(11) = {9.*L + w,0.0,cl};
Point(12) = {9.*L + w,2*L,0,cl};
Point(13) = {9.*L + w,B,0,cl};
Point(14) = {35.*L + w,0.0,cl};
Point(15) = {35.*L + w,2*L,0,cl};
Point(16) = {35.*L + w,B,0,cl};

// Lines to define the surface boundaries
Line(1) = {1,2};
Line(2) = {2, 3};
Line(3) = {3, 6};
Line(4) = {6, 13};
Line(5) = {13, 16};
Line(6) = {16, 15};
Line(7) = {14, 15};
Line(8) = {11, 14};
Line(9) = {9, 11};
Line(10) = {9, 10};
Line(11) = {10, 8};
Line(12) = {7, 8};
Line(13) = {7, 4};
Line(14) = {4, 1};
Line(15) = {2, 5};
Line(16) = {4, 5};
Line(17) = {5, 6};
Line(18) = {8, 5};
Line(19) = {5, 12};
Line(20) = {10, 12};
Line(21) = {11, 12};
Line(22) = {12, 13};
Line(23) = {12, 15};

// Line loops
Line Loop(1) = {1, 15, -16, 14};
Line Loop(2) = {2, 3, -17, -15};
Line Loop(3) = {16, -18, -12, 13};
Line Loop(4) = {-18, -11, 20, -19};
Line Loop(5) = {17, 4, -22, -19};
Line Loop(6) = {10, 20, -21, -9};
Line Loop(7) = {21, 23, -7, -8};
Line Loop(8) = {22, 5, 6, -23};
Ruled Surface(1) = {1};
Ruled Surface(2) = {2};
Ruled Surface(3) = {3};
Ruled Surface(4) = {4};
Ruled Surface(5) = {5};
Ruled Surface(6) = {6};
Ruled Surface(7) = {7};
Ruled Surface(8) = {8};

// Surface 1
Transfinite Line(1) = 1.75*L*n1 Using Progression 1.01;
Transfinite Line(15) = 9*L*n;
Transfinite Line(16) = 1.75*L*n1 Using Progression 1.01;
Transfinite Line(14) = 9*L*n;
Transfinite Surface\{1\} = \{1,2,5,4\};
Recombine Surface\{1\};

// Surface 2
Transfinite Line(2) = 1.25*L*n*2*2;
Transfinite Line(3) = 9.*L*n; // Using Bump 0.8;
Transfinite Line(17) = 1.25*L*n*2*2;
Transfinite Surface\{2\} = \{2,3,6,5\};
Recombine Surface\{2\};

// Surface 3
Transfinite Line(18) = 2.0*L*n*2 Using Progression 1.02;
Transfinite Line(12) = 1.75*L*n1 Using Bump 0.061;
Transfinite Line(13) = 2.0*L*n2 Using Progression 1.02;
Transfinite Surface\{3\} = \{4,5,8,7\};
Recombine Surface\{3\};

// Surface 4
Transfinite Line(19) = (4.0*L + w)*n;
Transfinite Line(20) = 2.0*L*n2 Using Progression 1.02;
Transfinite Line(11) = (4.0*L + w)*n Using Bump 1.5;
Transfinite Surface\{4\} = \{8,5,12,10\};
Recombine Surface\{4\};

// Surface 5
Transfinite Line(4) = (4.0*L + w)*n; // Using Bump 0.55;
Transfinite Line(22) = 1.25*n*L*2*2; // Using Bump 0.4;
Transfinite Surface\{5\} = \{5,6,13,12\};
Recombine Surface\{5\};

// Surface 6
Transfinite Line(10) = 1.75*L*n1 Using Bump 0.061;
Transfinite Line(21) = 1.75*L*n1 Using Progression 1.01;
Transfinite Line(9) = 2.0*L*n2 Using Progression 1.02;
Transfinite Surface\{6\} = \{9,10,12,11\};
Recombine Surface\{6\};

// Surface 7
Transfinite Line(23) = 8.0*L*n*2*2; // Using Progression 1.01;
Transfinite Line(7) = 1.75*L*n1 Using Progression 1.01;
Transfinite Line(8) = 8.0*n*L*2*2; // Using Progression 1.01;
Transfinite Surface\{7\} = \{11,12,15,14\};
Recombine Surface\{7\};

// Surface 8
Transfinite Line(5) = 8.0*n*L*2*2;
Transfinite Line(6) = 1.25*n*L*2*2;
Transfinite Surface{8} = {12,13,16,15};
Recombine Surface{8};

// Extrude final mesh
Extrude {0,0,D} {Surface{1:8}; Layers{24}; Recombine;}

// PHYSICAL GROUPS

Physical Surface('inletfluid') = {32,54};
Physical Surface('outflow') = {172,194};
Physical Surface('channelbed') = {1,2,3,4,5,6,7,8};
Physical Surface('atmosphere') = {155,177,199,133,111,89,67,45};
Physical Surface('leftwall') = {58,124,190};
Physical Surface('rightwall') = {176,154,88,44,84,142,102};
Physical Volume('inside') = {1:8};

A.2 Gmsh file for Test Case 2

// Variable Bank (SI Units used)
cl = 0.05; // Characteristic length for grid
W = 1.0; // Channel width
D = W/10.0; // Channel depth
Lin = 4*W; // Inlet length
Lout = 11*W; // Outlet length
lw = 0.25*W; // Obstacle length
n = 50;
pr1 = 1.025;
pr2 = 1.05;

// Points to define geometry
Point(1) = {0,0,0,cl};
Point(3) = {0,0.5*W,0,cl};
Point(5) = {0,W,0,cl};
Point(6) = {Lin,0,0,cl};
Point(7) = {Lin,lw,0,cl};
Point(11) = {Lin + lw,0,0,cl};
Point(12) = {Lin + lw,lw,0,cl};
Point(19) = {Lin + 3*lw,lw + 0.5*W,0,cl};
Point(20) = {Lin + 3*lw,W,0,cl};
Point(24) = {Lin + 4*lw,lw + 0.5*W,0,cl};
Point(25) = {Lin + 4*lw,W,0,cl};
Point(26) = {Lin + 6*lw,0,0,cl};
Point(27) = {Lin + 6*lw,lw,0,cl};
Point(31) = \{Lin + 7*lw,0,0,cl\};
Point(32) = \{Lin + 7*lw,lw,0,cl\};
Point(33) = \{Lin + 7*lw + lw,0.5*W,0,cl\};
Point(34) = \{Lin + 7*lw + lw,0,0,cl\};
Point(35) = \{Lin + 7*lw + lw,W,0,cl\};
Point(36) = \{Lin + 7*lw + Lout,0,0,cl\};
Point(37) = \{Lin + 7*lw + Lout,0.5*W,0,cl\};
Point(38) = \{Lin + 7*lw + Lout,W,0,cl\};
Point(39) = \{Lin - lw,0,0,cl\};
Point(40) = \{Lin - lw,0.5*W,0,cl\};
Point(41) = \{Lin - lw,W,0,cl\};
Point(43) = \{Lin + 2*lw,0,0,cl\};
Point(44) = \{Lin + 2*lw,0.5*W,0,cl\};
Point(45) = \{Lin + 2*lw,W,0,cl\};
Point(46) = \{Lin + 5*lw,0,0,cl\};
Point(47) = \{Lin + 5*lw,0.5*W,0,cl\};
Point(48) = \{Lin + 5*lw,W,0,cl\};

// Lines to define boundaries
Line(1) = \{1,3\};
Line(2) = \{45,43\};
Line(3) = \{5,3\};
Line(4) = \{34,33\};
Line(5) = \{6,7\};
Line(6) = \{35,33\};
Line(7) = \{33,53\};
Line(8) = \{27,53\};
Line(9) = \{11,12\};
Line(10) = \{32,33\};
Line(11) = \{3,43\};
Line(12) = \{33,38\};
Line(13) = \{20,19\};
Line(14) = \{25,24\};
Line(15) = \{26,27\};
Line(16) = \{31,32\};
Line(17) = \{36,38\};
Line(18) = \{40,38\};
Line(19) = \{1,41\};
Line(20) = \{7,12\};
Line(21) = \{27,32\};
Line(22) = \{34,36\};
Line(23) = \{19,24\};
Line(24) = \{25,55\};
Line(25) = \{55,35\};
Line(26) = \{35,40\};
Line(27) = \{7,43\};
Line(28) = \{41,43\};
Line(29) = \{12,48\};
Line(30) = \{46,48\};
Line(31) = \{43,48\};
Line(32) = \{19,48\};
Line(33) = \{24,53\};
Line(34) = \{48,53\};
Line(35) = \{50,48\};
Line(36) = \{55,53\};
Line(37) = \{5,45\};
Line(38) = \{45,50\};
Line(39) = \{20,50\};
Line(40) = \{11,46\};
Line(41) = \{46,51\};
Line(42) = \{26,51\};
Line(43) = \{51,53\};
Line(44) = \{6,41\};
Line(45) = \{31,34\};

// Line loops and ruled surfaces
Line Loop(1) = \{1,11,-28,-19\};
Line Loop(2) = \{-3,37,2,-11\};
Line Loop(3) = \{28,-27,-5,44\};
Line Loop(4) = \{27,31,-29,-20\};
Line Loop(5) = \{29,-30,-40,9\};
Line Loop(6) = \{-2,38,35,-31\};
Line Loop(7) = \{30,34,-43,-41\};
Line Loop(8) = \{-35,-39,13,32\};
Line Loop(9) = \{-32,23,33,-34\};
Line Loop(10) = \{-14,24,36,-33\};
Line Loop(11) = \{43,-8,-15,42\};
Line Loop(12) = \{8,-7,-10,-21\};
Line Loop(13) = \{10,-4,-45,16\};
Line Loop(14) = \{-36,25,67\};
Line Loop(15) = \{4,12,-17,-22\};
Line Loop(16) = \{-6,26,18,-12\};

Ruled Surface(1) = \{1\};
Ruled Surface(2) = \{2\};
Ruled Surface(3) = \{3\};
Ruled Surface(4) = \{4\};
Ruled Surface(5) = \{5\};
Ruled Surface(6) = \{6\};
Ruled Surface(7) = \{7\};
Ruled Surface(8) = \{8\};
Ruled Surface(9) = \{9\};
Ruled Surface(10) = \{10\};
Ruled Surface(11) = \{11\};
Ruled Surface(12) = \{12\};
Ruled Surface(13) = \{13\};
Ruled Surface(14) = \{14\};
Ruled Surface(15) = \{15\};
Ruled Surface(16) = \{16\};

// Surface 1
Transfinite Line(1) = n*W*1.4 Using Progression pr1;
Transfinite Line(11) = n*Lin*1.2;
Transfinite Line(28) = n*W*1.4 Using Progression pr1;
Transfinite Line(19) = n*Lin*1.2;
Transfinite Surface\{1\} = \{1,3,43,41\};
Recombine Surface\{1\};

// Surface 2
Transfinite Line(3) = n*W*1.4 Using Progression pr1;
Transfinite Line(2) = n*W*1.4 Using Progression pr1;
Transfinite Line(37) = n*Lin*1.2;
Transfinite Surface\{2\} = \{3,5,45,43\};
Recombine Surface\{2\};

// Surface 3
Transfinite Line(5) = n*W*1.4 Using Bump 0.5;
Transfinite Line(27) = 1.2*n Using Progression pr2;
Transfinite Line(44) = 1.2*n Using Progression pr2;
Transfinite Surface\{3\} = \{41,43,7,6\};
Recombine Surface\{3\};

// Surface 4
Transfinite Line(29) = 1.2*n Using Progression pr2;
Transfinite Line(31) = 3*lw*1.3*n;
Transfinite Line(20) = 3*lw*1.3*n;
Transfinite Surface\{4\} = \{43,48,12,7\};
Recombine Surface\{4\};

// Surface 5
Transfinite Line(30) = n*W*1.4 Using Progression pr1;
Transfinite Line(40) = 1.2*n Using Progression pr2;
Transfinite Line(9) = n*W*1.4 Using Bump 0.5;
Transfinite Surface\{5\} = \{12,48,46,11\};
Recombine Surface\{5\};

// Surface 6
Transfinite Line(35) = n*W*1.4 Using Progression pr1;
Transfinite Line(38) = 3*lw*1.3*n;
Transfinite Surface\{6\} = \{43,45,50,48\};
Recombine Surface\{6\};

// Surface 7
Transfinite Line(34) = 3*lw*1.3*n;
Transfinite Line(43) = n*W*1.4 Using Progression pr1;
Transfinite Line(41) = 3*lw*1.3*n;
Transfinite Surface{7} = {46,48,53,51};
Recombine Surface{7};

// Surface 8
Transfinite Line(39) = 1.2*n Using Progression pr2;
Transfinite Line(13) = n*W*1.4 Using Bump 0.5;
Transfinite Line(32) = 1.2*n Using Progression pr2;
Transfinite Surface{8} = {50,20,19,48};
Recombine Surface{8};

// Surface 9
Transfinite Line(23) = 3*lw*1.3*n;
Transfinite Line(33) = 1.2*n Using Progression pr2;
Transfinite Surface{9} = {19,24,53,48};
Recombine Surface{9};

// Surface 10
Transfinite Line(14) = n*W*1.4 Using Bump 0.5;
Transfinite Line(24) = 1.2*n Using Progression pr2;
Transfinite Line(36) = n*W*1.4 Using Progression pr1;
Transfinite Surface{10} = {25,55,53,24};
Recombine Surface{10};

// Surface 11
Transfinite Line(8) = 1.2*n Using Progression pr2;
Transfinite Line(15) = n*W*1.4 Using Bump 0.5;
Transfinite Line(42) = 1.2*n Using Progression pr2;
Transfinite Surface{11} = {53,27,26,51};
Recombine Surface{11};

// Surface 12
Transfinite Line(7) = 3*lw*1.3*n;
Transfinite Line(10) = 1.2*n Using Progression pr2;
Transfinite Line(21) = 3*lw*1.3*n;
Transfinite Surface{12} = {53,33,32,27};
Recombine Surface{12};

// Surface 13
Transfinite Line(4) = n*W*1.4 Using Progression pr1;
Transfinite Line(45) = 1.2*n Using Progression pr2;
Transfinite Line(16) = n*W*1.4 Using Bump 0.5;
Transfinite Surface{13} = {32,33,34,31};
Recombine Surface{13};
// Surface 14
Transfinite Line(25) = 3*lw*1.3*n;
Transfinite Line(6) = n*W*1.4 Using Progression pr1;
Transfinite Surface{14} = {55,35,33,53};
Recombine Surface{14};

// Surface 15
Transfinite Line(12) = n*Lout*1.3;
Transfinite Line(17) = n*W*1.4 Using Progression pr1;
Transfinite Line(22) = n*Lout*1.3;
Transfinite Surface{15} = {34,33,38,36};
Recombine Surface{15};

// Surface 16
Transfinite Line(26) = n*Lout*1.3;
Transfinite Line(18) = n*W*1.4 Using Progression pr1;
Transfinite Surface{16} = {35,40,38,33};
Recombine Surface{16};

// Extrude final mesh
Extrude {0,0,D} {Surface{1:16}; Layers{50}; Recombine;}

// Physical groups
Physical Surface('inlet') = {54,76};
Physical Surface('outflow') = {370,392};
Physical Surface('channelbed') = {1:16};
Physical Surface('atmosphere') = {67,89,111,133,155,177,199,221,243,265,353,287,309,331,375,397};
Physical Surface('rwall') = {66,110,132,150,198,286,308,326,374,154,106,282,330};
Physical Surface('lwall') = {388,344,256,212,168,80,234,252,216};
Physical Volume('inside') = {1:16};
Appendix B
OpenFOAM post-processing utility used to obtain the average in a homogeneous direction

B.1 sampledAveragePlane.C

This utility was used with the knowledge and permission of its author and developer, Frédéric Collonval.

#include "sampledAveragePlane.H"
#include "dictionary.H"
#include "polyMesh.H"

---

Technical University of Munich - Thermodynamics department, Boltzmannstrasse 15, 85748 Garching, collonval@td.mw.tum.de
```cpp
#include "volFields.H"
#include "addToRunTimeSelectionTable.H"

#define TOL 1e-6

// * * * * * * * * * * * * * * Static Data Members * * * * * * * * * * * * * //

namespace Foam
{
    defineTypeNameAndDebug(sampledAveragePlane, 0);
    addNamedToRunTimeSelectionTable(sampledSurface, sampledAveragePlane, word, averagePlane);
}

// * * * * * * * * * * * * * * * * Constructors * * * * * * * * * * * * * * //

Foam::sampledAveragePlane::sampledAveragePlane
(
    const word& name,
    const polyMesh& mesh,
    const plane& planeDesc,
    scalar endOfDomain,
    // label nPoints,
    const word& zoneName
)
:
    sampledSurface(name, mesh),
    cuttingPlane(planeDesc),
    zoneName_(zoneName),
    needsUpdate_(true)
{
    end_ = endOfDomain;
    // nPoints_ = nPoints;

    if (debug && zoneName_.size())
    {
        if (mesh.cellZones().findZoneID(zoneName_) < 0)
        {
            Info<< "cellZone """ << zoneName_ << "\" not found - using entire mesh" << endl;
        }
    }

    const vector& normal = planeDesc.normal();
    if (abs(abs(normal.x())+abs(normal.y())+abs(normal.z())-1.0) > TOL)
    {
```
FatalErrorIn
(   "Foam::sampledAveragePlane::sampledAveragePlane"
) << "The plane normal is not one of the coordinate axis"
   << exit(FatalError);
}

if(abs(abs(normal.x())-1.0) < TOL)
{
   axis_ = "X";
}
else if(abs(abs(normal.y())-1.0) < TOL)
{
   axis_ = "Y";
}
else if(abs(abs(normal.z())-1.0) < TOL)
{
   axis_ = "Z";
}

Foam::sampledAveragePlane::sampledAveragePlane
(   const word& name,
   const polyMesh& mesh,
   const dictionary& dict
)
:
   sampledSurface(name, mesh, dict),
   cuttingPlane(plane(dict.lookup("basePoint"), dict.lookup("normalVector"))),
   zoneName_(word::null),
   needsUpdate_(true)
{
   // make plane relative to the coordinateSystem (Cartesian)
   // allow lookup from global coordinate systems
   if (dict.found("coordinateSystem"))
   {
      coordinateSystem cs(dict, mesh);

      point   base = cs.globalPosition(planeDesc().refPoint());
      vector  norm = cs.globalVector(planeDesc().normal());

      // assign the plane description
      static_cast<plane&>(*this) = plane(base, norm);
   }

dict.readIfPresent("zone", zoneName_);

if (debug && zoneName_.size())
{
    if (mesh.cellZones().findZoneID(zoneName_) < 0)
    {
        Info<< "cellZone "$" "size: " < zoneName_ 
            " not found - using entire mesh" << endl;
    }
}

const vector& normal = planeDesc().normal();
if (abs(abs(normal.x())+abs(normal.y())+abs(normal.z())-1.0) > TOL)
{
    FatalErrorIn
    (   Foam::sampledAveragePlane::sampledAveragePlane"
        << "The plane normal is not one of the coordinate axis"
        << exit(FatalError);
}
if(abs(abs(normal.x())-1.0) < TOL)
{
    axis_ = "x";
}
else if(abs(abs(normal.y())-1.0) < TOL)
{
    axis_ = "y";
}
else if(abs(abs(normal.z())-1.0) < TOL)
{
    axis_ = "z";
}
dict.lookup("endOfDomain") >> end_;}
bool Foam::sampledAveragePlane::needsUpdate() const
{
    return needsUpdate_;}

bool Foam::sampledAveragePlane::expire()
{
    // already marked as expired
    if (needsUpdate_)
    {
        return false;
    }

    sampledSurface::clearGeom();

    needsUpdate_ = true;
    return true;
}

bool Foam::sampledAveragePlane::update()
{
    if (!needsUpdate_)
    {
        return false;
    }

    sampledSurface::clearGeom();

    label zoneId = -1;
    if (zoneName_.size())
    {
        zoneId = mesh().cellZones().findZoneID(zoneName_);
    }

    if (zoneId < 0)
    {
        reCut(mesh());
    }
    else
    {
        reCut(mesh(), mesh().cellZones()[zoneId]);
    }

    if (debug)
    {
        print(Pout);
Pout << endl;
}

needsUpdate_ = false;
return true;
}

Foam::tmp<Foam::scalarField> Foam::sampledAveragePlane::sample
(
    const volScalarField& vField
) const
{
    return sampleField(vField);
}

Foam::tmp<Foam::vectorField> Foam::sampledAveragePlane::sample
(
    const volVectorField& vField
) const
{
    return sampleField(vField);
}

Foam::tmp<Foam::sphericalTensorField> Foam::sampledAveragePlane::sample
(
    const volSphericalTensorField& vField
) const
{
    return sampleField(vField);
}

Foam::tmp<Foam::symmTensorField> Foam::sampledAveragePlane::sample
(
    const volSymmTensorField& vField
) const
{
    return sampleField(vField);
}

Foam::tmp<Foam::tensorField> Foam::sampledAveragePlane::sample
(
const volTensorField& vField ) const {
    return sampleField(vField);
}

Foam::tmp<Foam::scalarField>
Foam::sampledAveragePlane::interpolate
(    const interpolation<scalar>& interpolator
) const
{
    return interpolateField(interpolator);
}

Foam::tmp<Foam::vectorField>
Foam::sampledAveragePlane::interpolate
(    const interpolation<vector>& interpolator
) const
{
    return interpolateField(interpolator);
}

Foam::tmp<Foam::sphericalTensorField>
Foam::sampledAveragePlane::interpolate
(    const interpolation<sphericalTensor>& interpolator
) const
{
    return interpolateField(interpolator);
}

Foam::tmp<Foam::symmTensorField>
Foam::sampledAveragePlane::interpolate
(    const interpolation<symmTensor>& interpolator
) const
{
    return interpolateField(interpolator);
}

Foam::tmp<Foam::tensorField>
Foam::sampledAveragePlane::interpolate
(    const interpolation<tensor>& interpolator
) const
{
    return interpolateField(interpolator);
}
) const
{
    return interpolateField(interpolator);
}

void Foam::sampledAveragePlane::print(Ostream& os) const
{
    os << "sampledAveragePlane: " << name() << " :" << " base:" << refPoint()
    << " normal:" << normal()
    << " faces:" << faces().size()
    << " points:" << points().size();
}

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

B.2  sampledAveragePlaneTemplates.C

This utility was used with the knowledge and permission of its author and developer,
Frédéric Collonval.²

Licensed
This file is part of OpenFOAM.

OpenFOAM is free software; you can redistribute it and/or modify it
under the terms of the GNU General Public License as published by the
Free Software Foundation; either version 2 of the License, or (at your
option) any later version.

OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
for more details.

You should have received a copy of the GNU General Public License

²Technical University of Munich - Thermodynamics department, Boltzmannstrasse 15, 85748 Garching,
collonval@td.mw.tum.de
along with OpenFOAM; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

#include "sampledAveragePlane.H"
#include "midPointSet.H"
// #include "uniformSet.H"
#include "meshSearch.H"
/* * * * * * * * * * * * * Private Member Functions * * * * * * * * * * * */

template <class Type>
Foam::tmp<Foam::Field<Type> >
Foam::sampledAveragePlane::sampleField
(
    const GeometricField<Type, fvPatchField, volMesh>& vField
) const
{
    FatalErrorIn
    ("Foam::averagePlan::sampleField(const GeometricField<Type, fvPatchField,
    volMesh>& vField)"
     << "No interpolation scheme specified"
     << exit(FatalError);

    return tmp<Field<Type> >(new Field<Type>(vField, meshCells()));
}

template <class Type>
Foam::tmp<Foam::Field<Type> >
Foam::sampledAveragePlane::interpolateField
(
    const interpolation<Type>& interpolator
) const
{
    // One value per point
    tmp<Field<Type> > tvalues(new Field<Type>(points().size()));
    Field<Type>& values = tvalues();

    // Mesh search engine
    meshSearch searchEngine(mesh(), true);

    boolList pointDone(points().size(), false);

    forAll(faces(), cutFaceI)
const face& f = faces()[cutFaceI];

forAll(f, faceVertI)
{
    label pointI = f[faceVertI];

    if (!pointDone[pointI])
    {
        // values[pointI] = interpolator.interpolate
        // (points()[pointI],
        //  meshCells()[cutFaceI]
        // );

        // Creation of the homogeneous line that start from points()[pointI]
        // on the plane
        // to endPt
        point endPt = points()[pointI];

        if (axis_ == "x")
        {
            endPt.x() = end_;
        }
        else if (axis_ == "y")
        {
            endPt.y() = end_;
        }
        else if (axis_ == "z")
        {
            endPt.z() = end_;
        }

        if (debug)
        {
            Info << "Start point : " << (points()[pointI]).x() << " " << (points()[pointI]).y() << " " << (points()[pointI]).z() << nl;
            Info << "End point : " << endPt.x() << " " << endPt.y() << " " << endPt.z() << nl;
            Info << "Normal : " << axis_ << nl;
            Info << "End : " << end_ << endl;
        }

        // Extract the line in the homogeneous direction
        midPointSet line("homogeneousLine", mesh(), searchEngine, axis_, points()[pointI],
                          endPt); //, nPoints_);

        tmp<Field<Type> > tlinevalues(new Field<Type>(line.size()));
Field<Type>& linevalues = tlinevalues();

    // Interpolate the values of the fields in each point of the line
    forAll(line, lineI)
    {
        linevalues[lineI] = interpolator.interpolate
            (line[lineI],
             line.cells()[lineI],
             line.faces()[lineI]);
    }
    // Compute the average of the line
    values[pointI] = Foam::average(tlinevalues);

    if(debug)
    {
        Info << "Average value : " << values[pointI] << endl;
    }

    // Register that the point was analyzed
    pointDone[pointI] = true;
}
return tvalues;

// ************************************************************************* //
B.3 sampledAveragePlane.H

This utility was used with the knowledge and permission of its author and developer,
Frédéric Collonval.3

3Technical University of Munich - Thermodynamics department, Boltzmannstrasse 15, 85748 Garching,
collonval@td.mw.tum.de
License

This file is part of OpenFOAM.

OpenFOAM is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

OpenFOAM is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with OpenFOAM; if not, write to the Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Class

Foam::sampledAveragePlane

Description

A sampledSurface defined by a cuttingPlane. Always triangulated. This cutting plane is the basic plane on which the average field will be projected. So this plane have to be normal to the homogeneous direction.

Two additional parameters are required the endOfDomain and the nPoints. The first one is the maximal value of the coordinate normal to the plane. The second is the number of points used to compute the average in the homogeneous direction.

Definition of the dictionary

@author Frederic Collonval
@email fcollonv@umd.edu
@version 08172009

SourceFiles

sampledAveragePlane.C

*>-------------------------------------------------------------------------*/

 ifndef sampledAveragePlane_H
 define sampledAveragePlane_H
 include "sampledSurface.H"
 include "cuttingPlane.H"
namespace Foam
{

class sampledAveragePlane :
    public sampledSurface,
    public cuttingPlane
{

    // Private data

    // zone name (if restricted to zones)
    word zoneName_;

    // Track if the surface needs an update
    mutable bool needsUpdate_;

    // Private Member Functions

    // sample field on faces
    template <class Type>
    tmp<Field<Type> > sampleField
    (
        const GeometricField<Type, fvPatchField, volMesh>& vField
    ) const;

    template <class Type>
    tmp<Field<Type> >
    interpolateField(const interpolation<Type>&) const;

    // End of domain from the plane
    scalar end_; 

    // Direction normal to the plane
    word axis_; 

    // Number of points used to compute the average
    // label nPoints_;
public:

    // Runtime type information
    TypeName("sampledAveragePlane");

    // Constructors

    // Construct from components
    sampledAveragePlane
    (const word& name,
     const polyMesh& mesh,
     const plane& planeDesc,
     scalar endOfDomain,
     // label nPoints,
     const word& zoneName = word::null
    );

    // Construct from dictionary
    sampledAveragePlane
    (const word& name,
     const polyMesh& mesh,
     const dictionary& dict
    );

    // Destructor

    virtual ~sampledAveragePlane();

    // Member Functions

    // Does the surface need an update?
    virtual bool needsUpdate() const;

    // Mark the surface as needing an update.
    // May also free up unneeded data.
    // Return false if surface was already marked as expired.
    virtual bool expire();

    // Update the surface as required.
    // Do nothing (and return false) if no update was needed
    virtual bool update();

    // Points of surface
virtual const pointField& points() const
{
    return cuttingPlane::points();
}

//- Faces of surface
virtual const faceList& faces() const
{
    return cuttingPlane::faces();
}

//- For every face original cell in mesh
const labelList& meshCells() const
{
    return cuttingPlane::cutCells();
}

//- sample field on surface
virtual tmp<scalarField> sample
(
    const volScalarField&
) const;

//- sample field on surface
virtual tmp<vectorField> sample
(
    const volVectorField&
) const;

//- sample field on surface
virtual tmp<sphericalTensorField> sample
(
    const volSphericalTensorField&
) const;

//- sample field on surface
virtual tmp<symmTensorField> sample
(
    const volSymmTensorField&
) const;

//- sample field on surface
virtual tmp<tensorField> sample
(
    const volTensorField&
) const;
const;

virtual tmp<scalarField> interpolate
(const interpolation<scalar>&
) const;

virtual tmp<vectorField> interpolate
(const interpolation<vector>&
) const;

virtual tmp<sphericalTensorField> interpolate
(const interpolation<sphericalTensor>&
) const;

virtual tmp<symmTensorField> interpolate
(const interpolation<symmTensor>&
) const;

virtual tmp<tensorField> interpolate
(const interpolation<tensor>&
) const;

virtual void print(Ostream&) const;

};

} // End namespace Foam

#ifndef NoRepository
#include "sampledAveragePlaneTemplates.C"
#endif
# endif

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

# endif

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

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

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