Leopold-Franzens University Innsbruck
Faculty of Engineering Sciences
Department of Infrastructure
Unit of Hydraulic Engineering

Bachelor Thesis
November 15, 2018

Comparing a Smoothed Particle Hydrodynamics, a Position Based Dynamics, and a Finite Volume Method on a River Section Captured by Bathymetric LiDAR.

Dipl.-Ing. Marcel Ritter
E-mail: marcel.ritter@student.uibk.ac.at

Supervisor:

Dr. Robert Klar
E-mail: robert.klar@uibk.ac.at
Unit of Hydraulic Engineering, Department of Infrastructure,
University of Innsbruck, Austria

Abstract

Particle-based methods are a widely used technique for fluid-flow simulations nowadays, especially, because GPU hardware became very powerful and methods are very well parallelizable. In this study we apply the smooth particle hydrodynamics (SPH) method, originating from Astrophysics, and the position-based dynamics method (PBD), stemming from Computer Graphics. We compare them to the well-established finite volume method (FVM) used in engineering in the context of a new data source for river-bed geometry. The river geometry was acquired by airborne bathymetric light detection and ranging (LiDAR). As the data from LiDAR scans is point-based, it is naturally compatible with particle-based methods. We analyze the potential of doing hydraulic computations based on particle methods on bathymetric LiDAR data. A $160[m]$ long section of a river is selected for the case-study and hydraulic comparison. Particle simulation codes are extended and computations are tuned to meet a water table, known from the LiDAR scan at a reference cross section. The results show 16% higher velocities using PBD compared to the FVM. With further parameter fine-tuning and extensions this can be improved and the method could be an alternative for mesh based river-flow simulations.

Contents

1 Introduction **4**

2 Methods **6**
- 2.1 Data Acquisition 6
- 2.2 Flow Equations 7
- 2.3 Finite Volumes 10
- 2.4 Smooth Particle Hydrodynamics 11
- 2.5 Position Based Dynamics 16

3 River Section and Data Preparation **20**

4 Computational Setups and Execution **22**

5 Results **27**

6 Conclusion and Future Work **34**

7 Acknowledgements **35**

A Appendix **36**
- A.1 Flow3D Simulation Output Overview 36
- A.2 Particle Simulation Configuration File 38
- A.3 F# Plotting Tool 39

1 Introduction

Problem Statement: Airbone bathymetric LiDAR is a relatively new data source for monitoring water bodies [Steinbacher et al., 2012]. The laser penetrates the water surfaces and records return signals thereof and, additionally, of the wet ground below. The result is a high resolution point cloud, which can be utilized for flood studies, morphology analysis, or, generally, hydraulic computations on coasts, lakes, or rivers. In the case of river hydraulics usually meshes are generated for computation. They are either used as boundary condition or as the computational mesh itself, e.g. using shallow water equations.

Particle methods have become a standard tool for hydraulic simluations in the field of Astrophysics and in the field of visual effects and computer graphics for movies and games. Especially, GPU acceleration of such simluation codes makes the methods attractive. Recent developments have further improved performance for water fluid simulations in means of quality and speed. In this work we focus on two particle simulation methods: predictive-corrective incompressible smooth particle hydrodynamics (PCISPH) and position based dynamics (PBD). PCISPH improved the original formulation of SPH for incompressible fluids, such as water, by an inner prediction process for pressure, allowing to do larger time steps. PBD has a different approach as SPH, by solving on the particle positions by used constraint functions. Its main benefit is its speed and robustness. In an earlier work those two methods have already been prepared and adjusted, especially, for an application in river simulations [Schmidt, 2017]. However, it lacks a ground truth evaluation with a well-established simulation tool, fine tuning, and boundary flow handling. The scaling of the simulations were unrealistic in the verification examples and performance measures. This is caught up in this work and the particle simulation methods are applied in a real world scenario based on a bathymetric LiDAR scan of a river section. Computation results are analyzed and compared to manual engineering calculations and solutions of a well-established 3D flow simulation tool: Flow3D. If a particle method is a considerable alternative the process of mesh generation can be skipped. Also, formulation of sediment transort and coupling with soft and rigid bodies, e.g. driftwood jam, could be simpler. The following questions are targeted in this thesis.

Research Questions: From more general to specific.

- Can a particle simulation be done directly on bathymetric LiDAR?
- Can PBD or PCISPH yield resonable resuls for a river flow?
- Is the mean velocity in a river cross section resonable?
- Can the methods compete in computation time to Flow3D?

Aims: The aims of the thesis in short are:

- Setup a particle simulation tool for river flows using SPH and PBD
- Prepare bathymetric LiDAR data of a river section for computations
- Evaluate the particle methods by comparing to a Gauckler-Mannig-Strickler flow rate estimation and a Flow3D simulation

Contributions: The contributions of the thesis in short are:

- Mesh preparations of the river bed from bathymetric LiDAR
- Setting up and running simulations to match the LiDAR water table
  - in the commercial Flow3D software
  - in a C++ particle simulation tool developed in Visual Studio 2017
- Comparisons of the Flow3D and PBD results at a cross section and a Mannig Strickler estimation of the volumetric flow rate
- Extensions to particle simulation code:
  - Corrections, adjustments and fine tuning of the algorithms
  - Added a particle sink/source
  - Added saving and loading of the full particle simulation state
  - Added obj-mesh import for boundary particles
  - Global size scaling factor
  - Convergence output by statistics of density and velocity
  - 3D view to image file rendering
  - Simulation configuration data structure including save/load
- Development of a $F\#$ interactive cross section plotting tool for a fluid flow state

Outline: First, the applied methods are shortly presented and decribed, including neccesary extensions. A quick start for the related flow equations was added, to make the thesis complete. More details can be found in the referenced literature. Then, river section selection and data preperation is described, followed by the different simulation setups. The result section starts with an engineering calculation to estimate the mean velocity in a reference cross section. Further, results of the simulations are presented and compared. Finally, the thesis closes with a conclusion and future work.

2 Methods

The methods for data acquisition and three different methods used for the numerical computations and are shortly summarized. The numerical methods follow literature and were slightly adjusted, to ease comparability and to reduce content. The subsections related to the flow equations and the finite volumes method are taken from an introductory work by [Versteeg and Malalasekera, 2007], [Macklin and Müller, 2013] and [Solenthaler and Pajarola, 2009] were the main sources for the particle methods. Selected references therefrom were also involved.

2.1 Data Acquisition

Data was acquired using an airborne system running a green laser Riegl VQ880-G [Steinbacher et al., 2012] in a Tecnam 2006T twin engine airplane. The position and rotation of the system is tracked via GPS and IMU. The green laser wavelength is able to penetrate water bodies up to $\sim 13[m]$ in calm or ocean, and up to $\sim 5[m]$ in river waters. Figure 1 illustrates the principal of the data acquisition. Laser pulses are sent at a frequency of 550[kHz] onto the terrain in a circular pattern. For each laser pulse, several echoes can be identified by analyzing local maxima in the response signal and, using the known laser position and rotation, transformed in 3D point

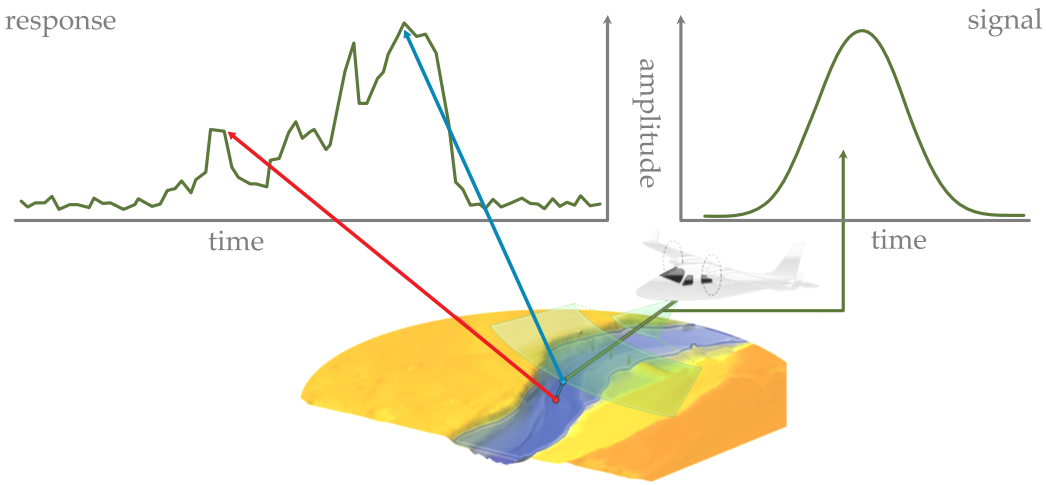

Figure 1: Bathymetric airborne LiDAR scanning penetrates the water surfaces. A sent light impulse (right) is distorted dependent on the targeted object and material. Local maxima of the returned response (left) are identified as echoes and transformed into 3D space, forming a 3D point cloud.

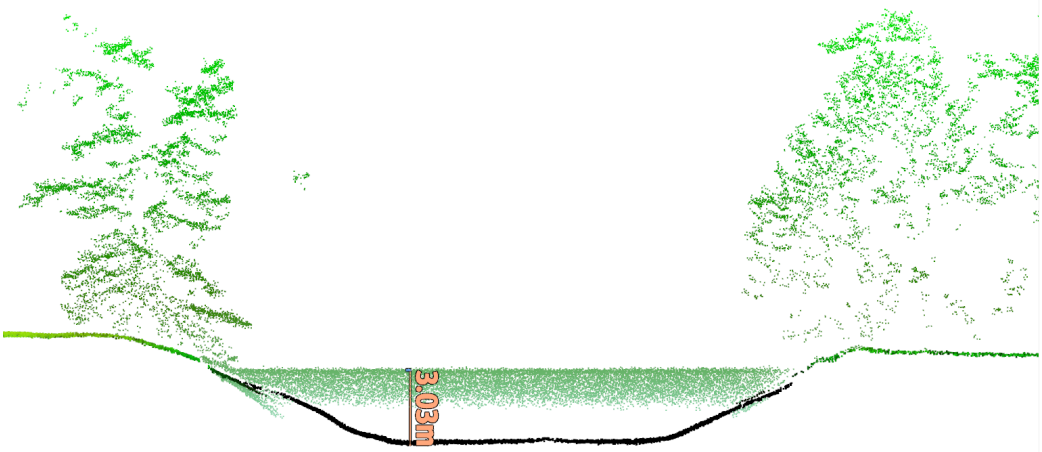

Figure 2: Example of a point cloud stemming from the bathymetric airborne LiDAR laser scanning. A cross section of $1[m]$ depth is shown.

locations. The resulting raw point cloud needs further processing: georeferencing, classification, and refraction correction. Then, it is applicable e.g. for more fine grain classification or for geometric reconstruction, such as generation of a simulation mesh for a hydraulic study. The technique also penetrates vegetation well due to its large footprint. Figure 2 shows a one meter cross section of a classified and refraction-corrected point cloud of a small river.

2.2 Flow Equations

The basic equations describing the dynamics of a fluid are derived from the conservation laws of mechanics, [Fielding, 2018]:

- conservation of mass (continuity equation)
- conservation of momentum (Cauchy equation)
- conservation of energy

The laws are formulated at infintesimal, but still macroscopic scale. Thus, a small fluid element is a very small cell or particle of a fluid continuum, and not modeled on molecular level.
Mass conservation: Here, one relates the rate of a change in mass of a volume V to mass flux crossing the boundary of the element:

$$\text{rate of change of mass in V} := -\frac{d}{dt}\int_V \rho dV = -\int_V \frac{\partial \rho}{\partial t} dV \quad (1)$$

$$\text{rate of mass boundary flux of V} := \oint_{\boldsymbol{S}} \rho \boldsymbol{v} \cdot d\boldsymbol{S} \quad (2)$$

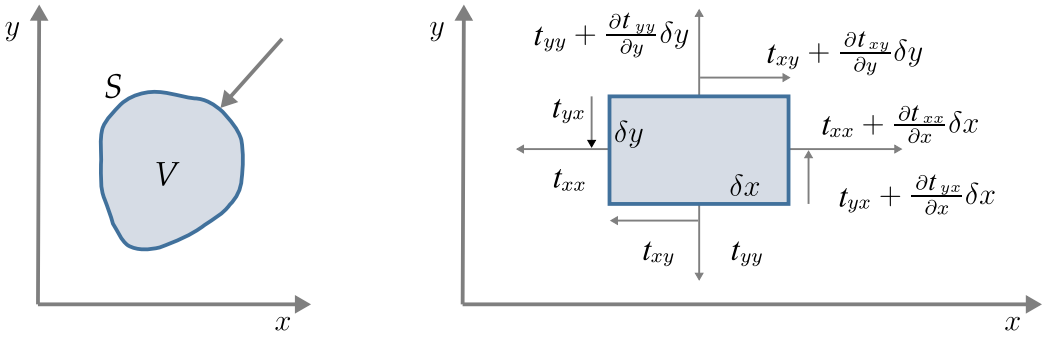

Figure 3: **Left:** Volume and surface of a fluid element. The arrow represents the mass boundary flux. **Right:** Surface forces on a fluid element. Exemplary in $2D$ with t_{ij} the components of the stress tensor.

with ρ density, t time, S the surface of the element, and $\oint$ an integral of a closed surface. By Gauss' divergence theorem the surface integral is converted to a volume integral and equated, yielding the continuity equation:

$$\int_V \nabla \cdot (\rho \boldsymbol{v}) dV = -\int_V \frac{\partial \rho}{\partial t} dV \tag{3}$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}) = 0 \tag{4}$$

with $\nabla := \left(\partial/\partial x, \partial/\partial y, \partial/\partial z\right)$. In case of incompressibility it can be further simplified, as density ρ becomes constant - independent of space and time. Water is modeled as incompressible fluid:

$$\nabla \cdot \boldsymbol{v} = 0 \tag{5}$$

Momentum conservation: Here, one bases on Newton's second law. The rate of change of momentum of the element must equal all forces acting on the element. Forces are separated into external body and surface forces. External body forces act on all fluid elements similarly, such as e.g. gravity. Surface forces are modeled by the stress tensor of the fluid element, gathering all forces at the element's boundary:

$$\text{rate of change of momentum} := \frac{d}{dt}\int_V dV \rho \boldsymbol{v} = \int_V dV \rho \frac{D\boldsymbol{v}}{Dt} \tag{6}$$

$$\text{body force + surface force} := \int_V dV \rho \boldsymbol{g} + \oint_S \underset{\sim}{\boldsymbol{t}} d\boldsymbol{S} \tag{7}$$

$$:= \int_V dV \left(\rho \boldsymbol{g} + \nabla \cdot \underset{\sim}{\boldsymbol{t}}\right) \tag{8}$$

$$\rho \frac{D\boldsymbol{v}}{Dt} = \rho \boldsymbol{g} + \nabla \cdot \underset{\sim}{\boldsymbol{t}} \tag{9}$$

Equation (9) is know as **Cauchy equation** - in Cartesian coordinates:

$$\begin{aligned}
\rho\frac{Du}{Dt} &= \rho g_x + \frac{\partial}{\partial x}t_{xx} + \frac{\partial}{\partial y}t_{xy} + \frac{\partial}{\partial z}t_{xz} \\
\rho\frac{Dv}{Dt} &= \rho g_y + \frac{\partial}{\partial x}t_{yx} + \frac{\partial}{\partial y}t_{yy} + \frac{\partial}{\partial z}t_{yz} \\
\rho\frac{Dw}{Dt} &= \rho g_z + \frac{\partial}{\partial x}t_{zx} + \frac{\partial}{\partial y}t_{zy} + \frac{\partial}{\partial z}t_{zz}
\end{aligned} \tag{10}$$

with $\underset{\sim}{\boldsymbol{t}}$ the symmetric stress tensor. Next, surface stresses are related to pressure and viscous friction, which are the sources for stresses on a fluid element. As the equations relate physical quantities they are called **constitutive equations**, in Cartesian coordinates:

$$\underset{\sim}{\boldsymbol{t}} = \begin{bmatrix} -p\lambda\nabla\cdot\boldsymbol{v} + 2\mu\frac{\partial u}{\partial x} & \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) & \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \\ & -p\lambda\nabla\cdot\boldsymbol{v} + 2\mu\frac{\partial v}{\partial y} & \mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \\ symm. & & -p\lambda\nabla\cdot\boldsymbol{v} + 2\mu\frac{\partial w}{\partial z} \end{bmatrix} \tag{11}$$

Here, μ and λ are the coefficients for dynamic and bulk viscosity. The relation between the velocity gradient and the stress is assumed to be linear and isotropic. Bulk viscosity λ vanishes for incomressible fluids and with a constant density equations (10) can be written as:

$$\rho\frac{D\boldsymbol{v}}{Dt} = \rho\boldsymbol{g} - \nabla p + \mu\Delta\boldsymbol{v} \tag{12}$$

For the scope of the thesis, the analytic introduction stops here. A simplified form of the **Navier-Stokes** equations was derived [Navier, 1821]. Importantly, for solving the flow of water, one has to deal with a set of coupled differential equations. Velocity and pressure are interlinked. More equations and terms are required dependent on the physics or engineering problem, e.g. viscosity, temperature/combustion, turbulance, etc. No analytic solutions are yet known and still among the millennium problems [Clay-Math.-Inst., 2018]. Numerically, the equations can be solved, as discretized initial value problem. Different approached exist for the discretization, e.g. finite differences/volumes, finite elements, SPH and position based fluids. They also require other formulations of the equations. One distinguishes between Eulerian and Langrangian discretization. Eulerian is fixed in space, and the fluid is moving through elements. In contrast, elements are moving along with the fluid using a Langrangian discretization.

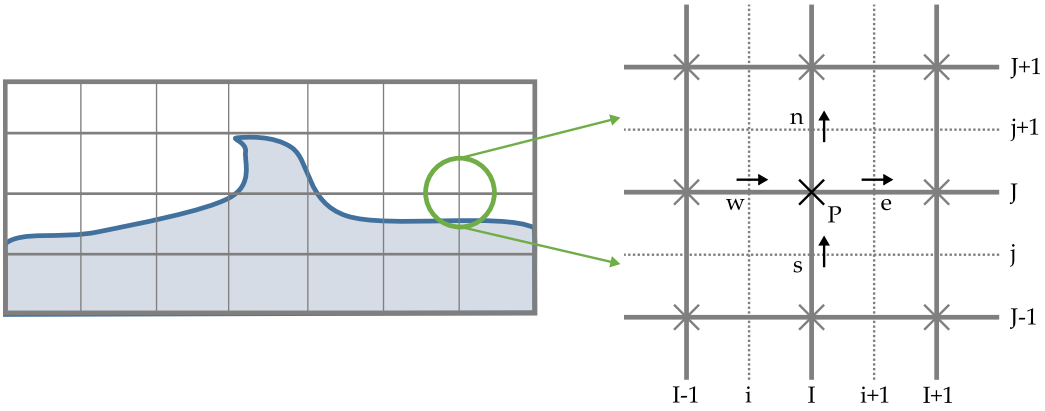

Figure 4: Eulerian discretization example used in the finite volume method. A staggered uniform grid is utilized. Pressure and density are handled on the main grid indexed in upper case letters I,J. Velocities are computed on a halfway shifted grid indexed by lower case letters i,j.

2.3 Finite Volumes

The finite volume method is an Eulerian method operating on a regular grid[1] and based on finite differences. Thus, partial derivatives are expressed as numerical differences to neighbored grid positions. Figure 4 illustrates a $2D$ regular grid and an additional staggered grid inbetween. Scalar variables, e.g. density and pressure, are stored on main grid, but velocity on the staggered grid. This improves for non cancelling velocities in case of fluctuating pressure values. In the finite volume approach, also equations are integrated over the cells and a piece-wise linear variation of the dependent variables is assumed. Based on the integrated quantities, fluxes across boundaries are balanced [Chorin, 1968]. For more indepth introductions refer to e.g. [Moukalled et al., 2016] and [Versteeg and Malalasekera, 2007].

One established industry standard algorithm is known as "SIMPLE": **S**emi-**I**mplicit **M**ethod for **P**ressure-**L**inked **E**quations and its extension "SIMPLER" (**R**evised). It cures a problem stemming from the continuity equation beeing itself independ of pressure. It includes several steps inside one time step, where first some preliminary velocities are estimated followed by pressure and velocity corrections, see Figure 5, [Patankar and Spalding, 1972]. The algorithm is utilized in commercial computational fluid dynamics (CFD) applications, e.g. in *Audodesk CFD* [Autodesk-Inc., 2018] and *Ansys* [Ansys-Inc., 2018].

For a computation by finite volumes the established application Flow3D

[1] each $3D$ cell shares a face to one cell east, west, north, south, up and down

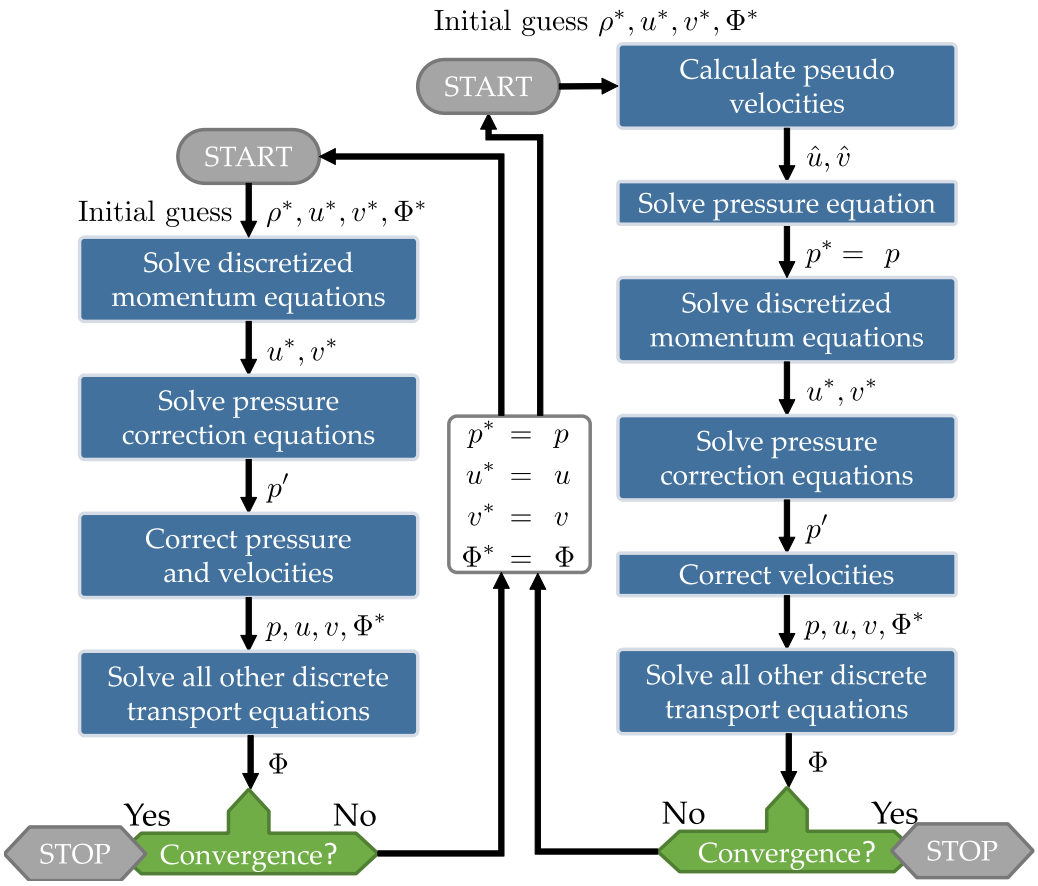

Figure 5: Steps of the SIMPLE (left) and SIMPLER (right) algorithm [PAT, 1980] of the finite volume method (FVM) for solving fluid flows. Both approaches do pressure and/or velocity corrections inside one single time step [Versteeg and Malalasekera, 2007].

[Flow Science Inc., 2018] was chosen to create a ground truth of the fluid flow. The software allows to compute the full 3D solutions and supports advanced features, such as different boundary materials (roughness), erosion, sediment transport, and turbulence. Here, only one single material was setup for the river-bed. The advanced features might become interesting for future comparisons.

2.4 Smooth Particle Hydrodynamics

The SPH method is a Lagrangian method, were particles model the fluid body. The discretization is moving with the flow. Over the Eulerian methods, this has the advantage to not waste unused discretization space, where

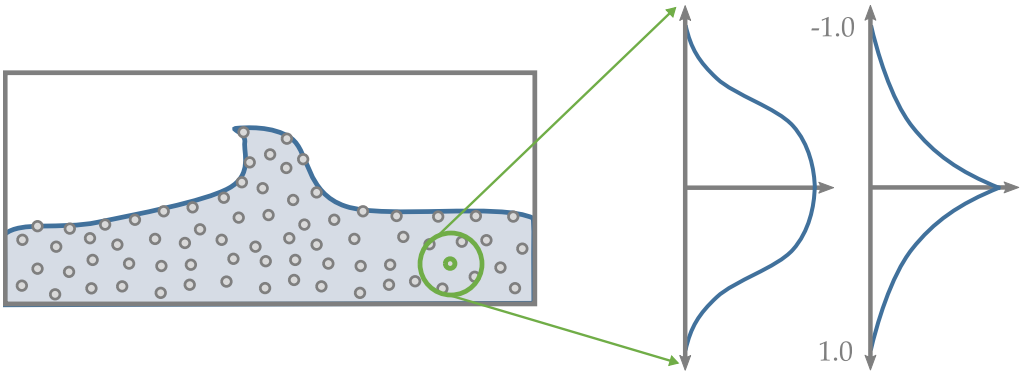

Figure 6: **Left:** Lagrangian methods model the fluid with moving elements, e.g. particles. **Right:** Typical weighting kernels used in a particle neighborhood for integration in the SPH method, with a support radius $h = 1.0$. A smooth kernel is usually used for the pressure field and a spiky kernel for the viscosity field.

no fluid is present. Each particle represents an discrete element of the fluid body, where field values are stored and computed: velocity, position, and pressure. When evaluating field quantities always a small neighborhood is taken into account. Fields are 'smoothed' over the neighborhood by using a radial weighting function $W(r, h)$ called kernel, as illustrated in Figure 6, with r the radius and h a fixed support radius defining the kernel size. A particle itself has a radius and density. But, the density field quantity also depends on the number of, and distance to particles in the neighborhood. Thus, the support radius is defined larger than the particle radius. It is set to 4 times larger in this work. For the computation of the movement of each particle, first the density field is computed, thereon the pressure field, leading to forces and acceleration and, finally, to a velocity. The velocity is then used for a discrete forward time step yielding new particle positions. This procedure is categorized as force-based method.

One core principle applied in the SPH is the integral interpolation of a field quantity. A field value of field Φ of a fluid domain Ω at certain position x is expressed as an integral:

$$\Phi(x) \quad = \quad \int_{\Omega} \Phi(x')\delta(x - x')d\Omega_{x'} \tag{13}$$

with δ being the Dirac delta function. In the SPH method The Dirac delta function is approximated by a smooth kernel function $W(x, h)$, especially to ensure continuity and differentiability of the field function. In the limit

of the support radius h of the kernel it is becoming the Dirac delta function:

$$\lim_{h\to 0} W(x-x',h) = \delta(x-x') \tag{14}$$

For small values of h the kernel can now be used for approximation:

$$\Phi(x) \approx \int_\Omega \Phi(x')W(x-x',h)d\Omega_{x'} \tag{15}$$

$$\int_\Omega \frac{\Phi(x')}{\rho(x')}\rho(x')W(x-x',h)d\Omega_{x'} \tag{16}$$

$$\Phi(x_i)=\Phi_i \approx \sum_j \frac{\Phi_j}{\rho_j}\underbrace{\rho_j V_j}_{m_j} W(x_i-x_j,h) \tag{17}$$

with j denoting the index of the neighboring particles of particle i and V_i is the volume. As mentioned before, the first step in the SPH method is computing the density field. Here, when inserting ρ as a field quantity (17) simplifies to:

$$\rho_i = \sum_j W(x_i-x_j,h)m_j \tag{18}$$

Therefrom, the pressure field is computed. In our case we focus on the predictive-corrective incompressible SPH (PCISPH) method and utilize the pressure update from [Solenthaler and Pajarola, 2009]:

$$p_i = \frac{\rho_0}{7}\left(\left(\frac{\rho_i}{\rho_0}\right)^7 - 1\right), \tag{19}$$

known as the equation of state (EOS), originally from [Batchelor, 1967] with the tuning parameter γ set to 7. To get partial derivatives of a field quantity, in the approximation (17), just the kernel function has to be derived:

$$\frac{\partial \Phi}{\partial x} = \sum_j \frac{\Phi_j}{\rho_j} m_j \frac{\partial W}{\partial x} \tag{20}$$

Dropping viscosity from (12), the acceleration can be expressed by:

$$\frac{dv}{dt} = -\frac{1}{\rho}\nabla p + g \tag{21}$$

$$\tag{22}$$

To write the acceleration equation in a conserving form for linear and angular momentum, it can be written as [Monaghan, 2005]:

$$\frac{\nabla p}{\rho} = \nabla\left(\frac{p}{\rho}\right) + \frac{p}{\rho^2}\nabla\rho \tag{23}$$

Both additive terms can be approximated by using (17) and (20):

$$\nabla\left(\frac{p}{\rho}\right) = \sum_j m_k \frac{p_j}{\rho_j^2} \nabla W(x - x_j, h) \tag{24}$$

$$\nabla\rho = \sum_j m_j \nabla W(x - x_j, h) \tag{25}$$

$$\frac{dv_i}{dt} = -\sum_j m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2}\right) \nabla \underbrace{W(r_i - r_j, h)}_{W_{ij}} \tag{26}$$

Now, yielding a discrete formula for the pressure computation. The new position for each particle is then computed by:

$$x_i(t) = x_i(t-1) + \underbrace{v_i(t-1)\Delta t}_{\Delta x_i(t-1)} \tag{27}$$

This is the SPH method in a very simple form. We need to extend the method further for the purpose of river simulations. The handling of boundaries and the property of the incompressibility of water are required. The work of [Solenthaler and Pajarola, 2009] extends for incompressibility in an efficient way. First, a normal SPH step is computed as a prediction. Via the velocities v_i^* and positions x_i^* predicted densities ρ_i^* are estimated. By looking at its derivation from ρ_0 a pressure correction can be obtained replacing the pressure used for the time integration step and preserving incompressibility. The method is iterative and requires a few steps for the pressure correction. Still, it enables to keep the time step size for the main position step large, compared to earlier methods. A positional update is now formulated analytically by the forces F_i:

$$\Delta x_i = \Delta t^2 \frac{F_i}{m} \tag{28}$$

A simplification is introduced: neighbors have equal pressure $\tilde{p}_i$ and that the density is the rest density ρ_0. To reach incompressibility it is required that two neighboring particles reach the same density and, finally, ρ_0, from (26) and (28):

$$F_i = -\sum_j m_j \left(\frac{\tilde{p}_i}{\rho_0^2} + \frac{\tilde{p}_i}{\rho_0^2}\right) \nabla W_{ij} = -m^2 \frac{2\tilde{p}_i}{\rho_0^2} \sum_j \nabla W_{ij} \tag{29}$$

$$\Delta x_i = -\Delta t^2 m \frac{2\tilde{p}_i}{\rho^2} \sum_j \nabla W_{ij} \tag{30}$$

When looking at one single pair of particles: particle j is influenced by the pressure of i:

$$\Delta x_{j|i} = \Delta t^2 \underbrace{m \frac{2\tilde{p}_i}{\rho_0^2} \nabla W_{ij}}_{F_{j|i}} \tag{31}$$

According to [Solenthaler and Pajarola, 2009], this can be used to correct the pressures by inserting (30) and (31) into:

$$\Delta\rho_i(t) = m\big(\Delta x_i(t)\big) \sum_j \nabla W_{ij} - \sum_i \nabla W_{ij} \Delta x_j(t)\big) \tag{32}$$

$$p_i^* = \frac{\Delta\rho_i(t)}{\underbrace{\Delta t^2 m^2 \frac{2}{\rho_0^2}}_{\beta} \big(- \sum_j \nabla W_{ij} \sum_j \nabla W_{ij} - \sum_j (\nabla W_{ij} \nabla W_{ij})} \tag{33}$$

Equation (33) is problematic for dividing by zero in empty neighborhoods. To achieve a final and stable pressure update, the following term is computed only once for a reference configuration within a full neighborhood:

$$\delta = \frac{-1}{\beta\big(- \sum_j \nabla W_{ij} \sum_j \nabla W_{ij} - \sum_j (\nabla W_{ij} \nabla W_{ij})} \tag{34}$$

$$\tilde{p}_i = \delta \underbrace{(\rho_i^* - \rho_0)}_{\rho_{err}^*} \tag{35}$$

$$p_i \mathrel{+}= \tilde{p}_i \tag{36}$$

with $\tilde{p}$ the pressure in each prediction-correction step und p the accumulated pressure, when the incompressibility condition is reached.

Finally, boundary particles are introduced, such that a river bed can be included into the SPH simulation by static particles. Again, first a density has to be computed. But, a summation is only done over the boundary particles. For a boundary particle ℓ and neighborhood boundary particles k boundary volume element V^b is defined by:

$$V_\ell^b = \frac{m_\ell^b}{m_\ell^b \sum_k W_{\ell k}} \tag{37}$$

This enables the support for non-uniformly placed boundary particles, as is natural for particles stemming from a LiDAR measurement. Taken from

the method of [Akinci et al., 2012] and adjusted yields the density for a fluid particle i taking boundary neighborhood particles into account:

$$\Psi_\ell^b(\rho_0) = \rho_0 V_\ell^b \tag{38}$$

$$\rho_i = m_i \sum_j W_{ij} + \sum_k \Psi_k^b(\rho_0) W_{ik} \tag{39}$$

Another term is required for the forces and again two particles must have similar densities and pressures in incompressible fluids, simplifying (26):

$$F_{ij} = -m_i m_j \frac{p_x}{\rho_x^2} \nabla W_{ij} \tag{40}$$

$$F_{ik} = -m_i \Psi_k^b \frac{p_i}{\rho_i^2} \nabla W_{ik} \tag{41}$$

Here, the pressure of a fluid particle i is inserted. Thus, the reaction force at the boundary depends on the pressure of the fluid particle. The higher the pressure, the higher the reaction at the boundary pushing the particle back. Finally, the force per particle i is the sum of the forces of fluid particles j and boundary particles k, used in the inner correction loop of the PCISPH:

$$F_{i_total} = \sum_j F_{ij} + \sum_k F_{ik} \tag{42}$$

2.5 Position Based Dynamics

The concepts of PBD were developed within computer graphics, games and real-time simulation. Here, performance is more important than high physical accuracy. In a modern high quality game physically based simulations for different kinds of objects are of interest, and their interactions: e.g. cloth, rigid bodies, soft bodies, and fluids. PBD allows to simulate fluids [Macklin and Müller, 2013], but can capture all those aspects of mechanical simulations [Macklin et al., 2014]. Besides its versatileness, it is also numerically very robust - it is unconditionally stable. This three features (efficiency, versitaleness, and robustness) made it attractive to be a choice for an industry implementation directly into the PhysX engine of NVIDIA via a code called FleX [NVIDIA, 2018].

In PBD, the solver works in a different way as compared to SPH. In SPH the numerical solution is based on forces - in PBD it is based on positions. Variables such as acceleration, velocity, and forces can be derived from the computed positions. The important variables for a fluid simulation are velocity, mass, external forces, density and pressure.

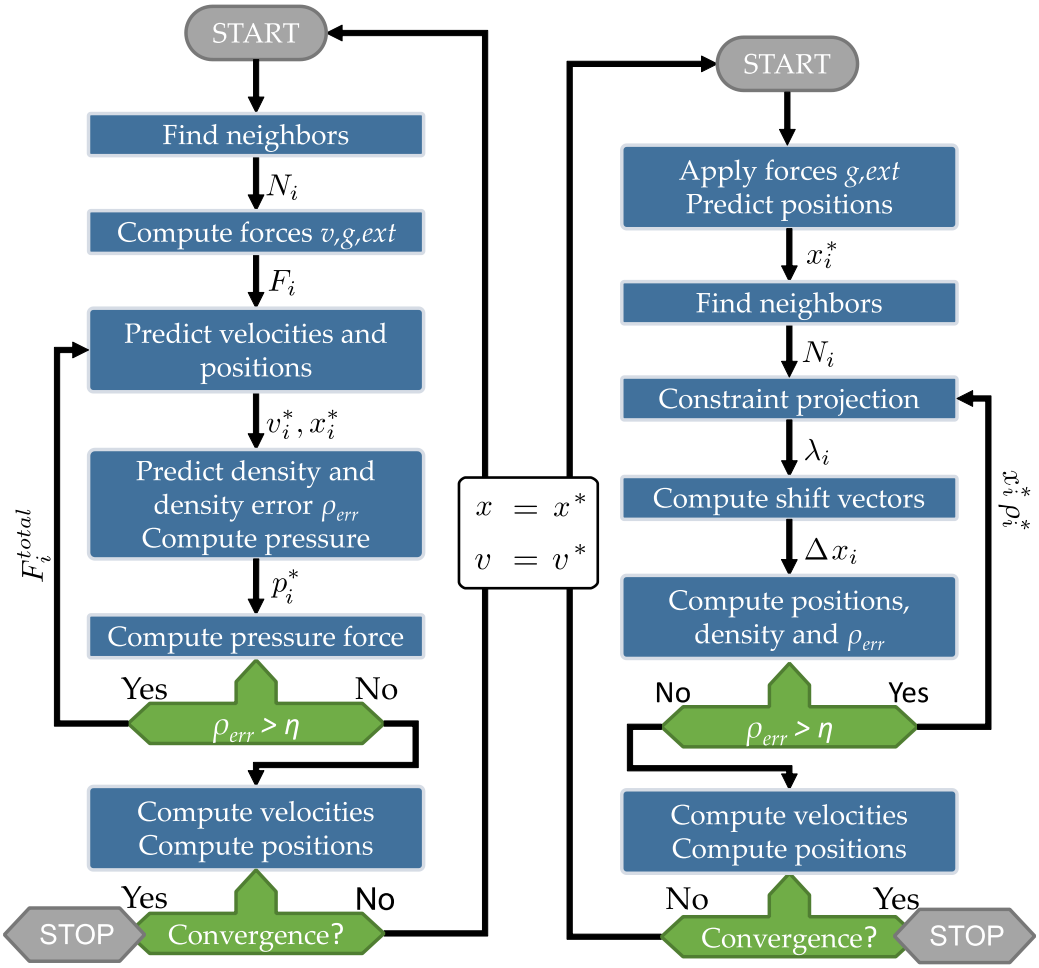

Figure 7: Particle methods. **Left:** PCISPH uses an inner loop per time step for the prediction of pressure and velocity. **Right:** PBD uses an inner loop to recompute constraint vialation and position shift vectors Δx_i.

The key concept in PBD is the use of so called constraints. All defined constraints have to be satisfied by the simulation state to be valid. Constraints are realized as functions and are applied within a particle neighborhood. Two different kinds of constraints are distinguished:

$$C_j(x_{i_1}, ..., x_{i_n}) = 0 \quad ... \, equality \tag{43}$$

$$C_j(x_{i_1}, ..., x_{i_n}) \geq 0 \quad ... \, inequality \tag{44}$$

with $j \in [1...M]$ the index of M different constraints, n the number of neighborhood particles around x_i, and $i \in [1...N]$ the index of N particles of a dynamic object. A constraint is satisfied if C holds for all n particles. In

each time step of a PBD computation, for every particle, a new position x_i is estimated based on a gravitational force, and a preliminary velocity derived from the previous time step. Then, the estimated positions are iteratively corrected, such that all the involved constraints are satisfied. An iterative process is necessary, since a constraint correction at one position, might introduce a violation at a different location or of a different constraint. From the final corrected positions of the new time step the velocities are computed, retrospectivly. For fluid simuation, the constraints have to be formulated such that they model its physical behavior. Here, the driving constraint is a density constraint and a boundary interaction constraint. In the correction step positions are corrected in direction of the gradients of the constraints. This step is called constraint projection.

In practice, the method works as follows, as described in [Schmidt, 2017] and [Macklin and Müller, 2013]. First, a predicted velocity and a predicted position is computed:

$$v_i^*(t) = v_i(t-1) + a_i(t-1)\Delta t \tag{45}$$
$$x_i^*(t) = x_i(t-1) + v_i^*(t)\Delta t \tag{46}$$

with * denoting the predicted variables, particle velocities v_i, accelerations a_i and time t. Using the preliminary particle positions x^*, a density constraint has to be met:

$$C_{dens}(x_{i_1}^*, ..., x_{i_n}^*) = \rho_i/\rho_0 - 1 \tag{47}$$
$$C_i(x_1^*, ..., x_n^*) = \rho_i/\rho_0 - 1 \tag{48}$$

with ρ_0 the rest density of water and ρ_i. Sub indices can be avoided by using the notation in (48). The index of C now denotes the particle location of the constraint. Since just one constraint is used here no constraint index is required. The density at particle i is formulated via the SPH density estimator (18). For the case of the river simluation, the density equality constraint was changed into an inequality constraint. Only high densities are corrected. Low density regions, such as particles at the free surface, have only a little impact on the overall flow behaviour at that scale.

In PBD one is solving for the correction of positions. Thus, the position x^* and its position correction Δx are separated:

$$C(x^* + \Delta x) \leq 0 \tag{49}$$

With a series of Newton steps along the constraint gradient:

$$\Delta x = \nabla C(x^*)\lambda \tag{50}$$
$$C(x^* + \Delta x) \approx C(x^*) + \nabla C(x^*)^T \Delta x \leq 0 \tag{51}$$
$$C(x^* + \Delta x) \approx C(x^*) + \nabla C(x^*)^T \nabla C(x^*)\lambda \leq 0 \tag{52}$$

The gradient ∇C can be estimated by borrowing from the SPH method [Monaghan, 1992]. The pressure gradient with respect to a particle k is:

$$\nabla_{x_k} C_i = \frac{1}{\rho_0} \sum_j \nabla_{x_k} W_{ij} \tag{53}$$

Depending if k is a neighboring particle or not (53) has two cases:

$$\nabla_{x_k} C_i = \frac{1}{\rho_0} \begin{cases} \frac{m}{\rho_0} \sum_j \nabla_{x_k} W_{ij}, & \text{for } k = i \text{ (at particle)} \\ -\frac{m}{\rho_0} \nabla_{x_k} W_{ij}, & \text{for } k = j \text{ (in neighborhood)} \end{cases} \tag{54}$$

Inserting into Equation (52) and solving for λ:

$$\lambda_i = -\frac{C_i(x_1^*, ..., x^n)}{\sum_j |\nabla_j C_i|^2} \tag{55}$$

Because of the non-linear constraint function and its vanishing gradient at the kernel boundary, this equations causes instabilities when particles are about to separate. This can be cured by introducing an additional term, or by using a pre-computed correction scale based on a reference particle configuration in a filled neighborhood, which was done in this work. The position correction at particle i, including also corrections from neighboring particle density constraints (λ_j) is now given as:

$$\Delta x_i = \frac{1}{\rho_0} = \sum_j (\lambda_i + \lambda_j) \nabla W_{ij} \tag{56}$$

$$x_i^*(t) = x_i(t-1) + \Delta x_i \tag{57}$$

$$v_i^*(t) = \Delta x_i / \Delta t \tag{58}$$

The new x^* is checked by the constraint again, and if valid for all particles the actual time step is executed ($x_i = x_i^*$). Boundaries are also handled via the density constraint by utilizing Equation (54) and replacing m with the boundary density dependent parameter Ψ_b, see (38):

$$\nabla_k C_i = \frac{\Psi_b}{\rho_0} \sum_b \nabla W_{ib} \tag{59}$$

In contrast to the original PBD method each constraint is solved independently in a Jacobi fashion. Neighborhoods are updated once in a time step and not within the iterative constraint projection and correction loop. Figure 7 illustrates the algorithm compared to PCISPH.

3 River Section and Data Preparation

River Section: A small river section was selected from a surveying project and chosen for the comparison. For a start, it should fulfill several conditions to ease computations and simulations:

- stable and mostly non-turbulent flow conditions
- quite straight river axis
- trapezoidal cross-section geometry
- good coverage by LiDAR echoes of the wet ground and water surface
- high density of LiDAR echoes

Therefore, a river section downstream of a backwater region and a curve was chosen. The length of the chosen section has a quite straight river-axis and a length of about $500[m]$, see Figure 8. The full surveying project is of about $25[km]$ length, a rather small project.

Data Preparation: Ground points have been already separated from the rest. A hydraulic mesh generated from the LiDAR ground points was also already available from HydroVish [Airborne Hydromapping GmbH, 2018]. All the availabe data was then cut and exported using the software, while treating the point cloud and the mesh separately. The ground mesh was converted into a OBJ file for further editing.

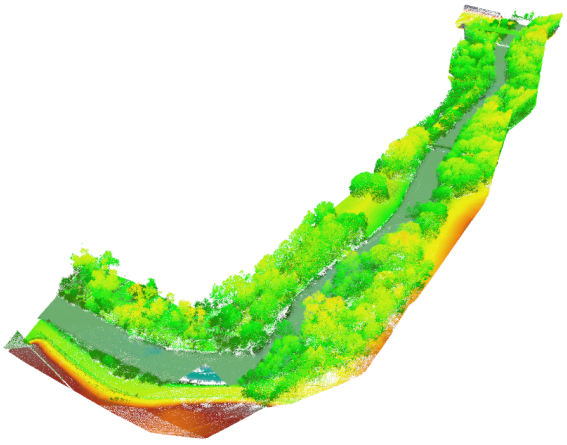

Figure 8: Selected river section. A section after a curve was chosen flowing along a straight pathway. Flow conditions are simple with minimal turbulence effects. Also, the data acquisition has a very good coverage and, thus, many laser echoes on the wet ground and water surface.

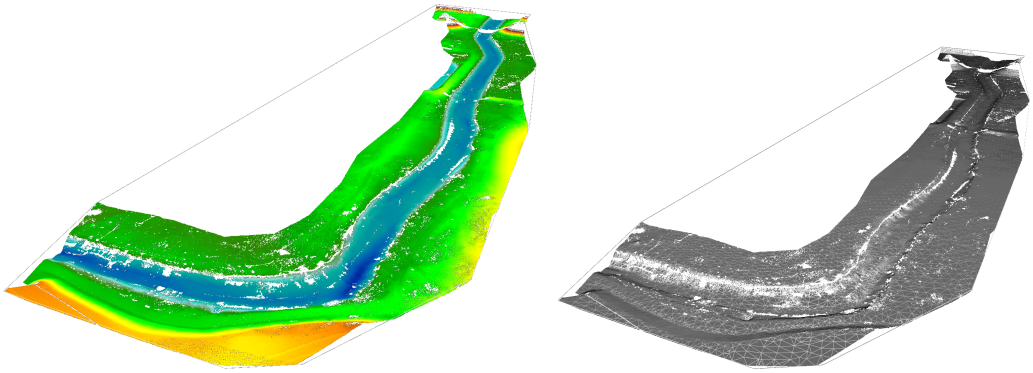

Figure 9: Separated ground points and generated mesh. The software HydroVISH [Airborne Hydromapping GmbH, 2018] was used for LiDAR preparations and data export. The mesh (right) is a triangular adaptive mesh, adjusting to finer details in the river-bed.

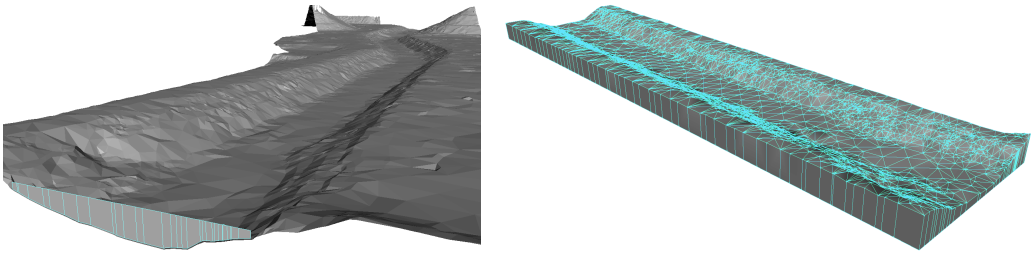

Figure 10: Manual mesh operations in Maya. A cuboid section was axis-aligned and closed by border-edge polygon-extrusion and vertex-welding.

Maya 8.5 [Autodesk Inc., 2018] was used for manual mesh operations. The triangular mesh exported from HydroVISH was loaded as an OBJ file into the modeling software. To further ease the simulation workflow and setting boundary conditions, the mesh was translated to the coordinate origin and rotated, such that the river-axes are aligned to the Cartesian coordinate system axes. Further, a rectangular region was cut out. Two different versions of the mesh have been prepared. An open surface mesh capturing the river bed, and a closed mesh. The closed mesh was generated by extruding the open border edges downwards, flattening, and closing the hole, see Figure 10. The meshes are used as solid boundary conditions in the simulations. The closed mesh is required by Flow3D and the open mesh was used for the particle methods. They were exported from Maya as OBJ files and further converted to STL, to enable the data import into Flow3D.

The particle code requires boundary particles and could not load a mesh

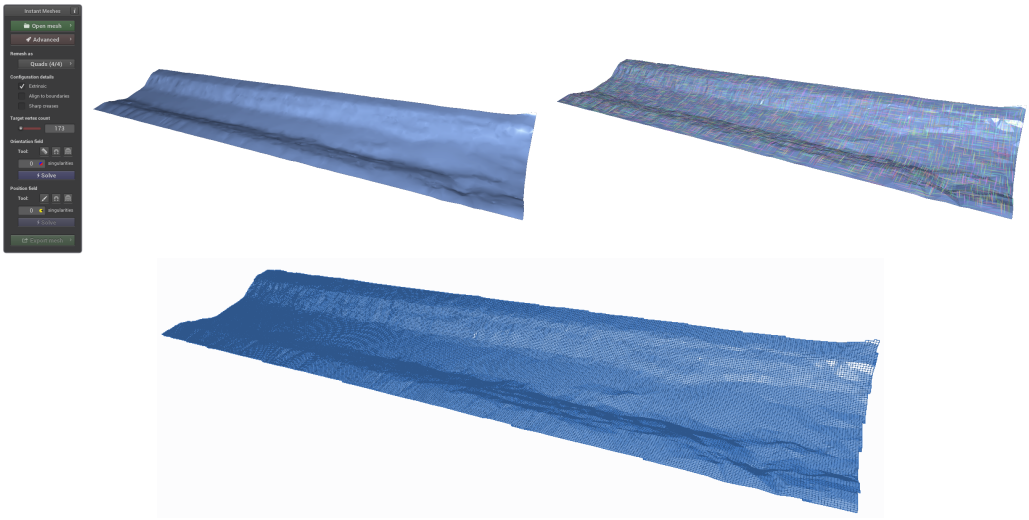

Figure 11: Retopology of the adaptive triangular mesh. A quad mesh was generated using the method and tool of [Jakob et al., 2015]. **Top:** input mesh and directional field analysis. **Bottom:** uniformly distributed result.

file directly. To create a comparable scene for the particle simulation, first, a finer resolution mesh was computed from the mesh used in the Flow3D simulation. A retopology to a quad based mesh was done using the tool[2] and method by Jabok Wenzel [Jakob et al., 2015]. The face count was increased from 2500 to $25k$ and $50k$ yielding almost rectangular grid cells of about $0.3[m]$ and $0.2[m]$ size. Figure 11 shows the process of mesh refinement using the field aligned generator. The particle code was extended to load an OBJ file and create a boundary particle for every vertex.

4 Computational Setups and Execution

Finite Volume Simulation: For the finite volume simulation the software Flow3D Version 11.1.4.2 win64 was applied. It is a well-established simulation software for hydraulics and environmental engineering. Over a hundret technical publications utilizing Flow3D and about fifty already published in 2018 can be found on their web-page [Flow Science Inc., 2018].

For the setup, first, a new simulation project was created and the STL mesh of the river bed loaded as a solid object component. The STL file needs to be a closed mesh. An open mesh will be displayed but ignored as a simulation boundary. A material of coarse sand was assigned to the

[2] https://github.com/wjakob/instant-meshes

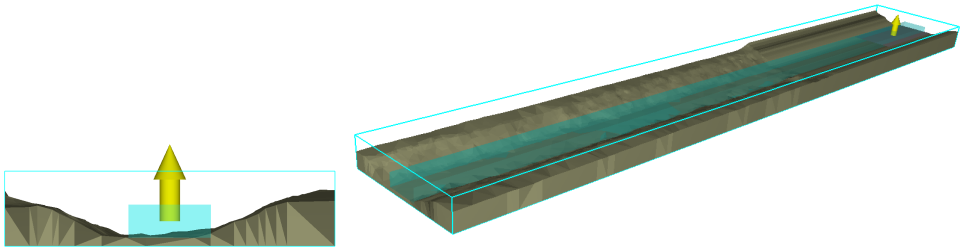

Figure 12: Simulation setup in Flow3D: river bed boundary (grey), initial fluid region (transparent blue box), rectangular volume source (yellow arrow and grey plane), and computation grid region (cyan out lines).

river bed. Additionally, a mean grain diameter of $0.01[m]$ was configured in the simulation pre-check. The computational grid was adjusted to the geometry defined by the bounding box from $(-20, -100, -1)$ to $(20, 160, 8)$ and 200000 cells for mesh generation. Flow3D finally divided the space into $53 \times 311 \times 13$ cells, yielding 214279 cells of size $0.75[m] \times 0.69[m] \times 0.83[m]$. Boundary conditions on the grid were chosen as wall or open border:

Xmin:	**w**all	Xmax	**w**all
Ymin:	**w**all	Ymax	**o**pen
Zmin:	**o**pen	Zmax	**o**pen

The fluid was set as water at $20°C$. In the initial condition configuration a fluid region within $(-6, -100, 0)$ to $(6, 160, 4)$ was set up as a breaking dam to quickly fill the river bed with some water. A fluid source with a constant volume rate was added upstream as a rectangle facing upwards at position $(-6, 0, -100)$ with length $39[m]$ and width $20[m]$.

The software modules for gravity and mass source were activated. The numerical solver was configured to use the default implicit method. A maximum time for the solution was set to 100 and an additional breaking condition was enabled to stop if a steady state in the flow simulation is reached. The first setups of the simulation setup took mostly over 4 hours, since a higher grid resolution was used and, most importantly, no initial fluid region was created. Thus, the river bed had to fill up from the mass source alone. By adding the initial fluid region, the simulation became more stable and steady states were reached much faster, in about 20 minutes. The parameters of the simulation, mainly the rate of the mass source, were tuned manually until the water table at the cross section $y = 140[m]$ matched the reference value of $3.03[m]$ from the LiDAR measurement. About 30 simulation runs were executed to reach that goal. Finally, the simulation output provides results at five different time values: 0.0,

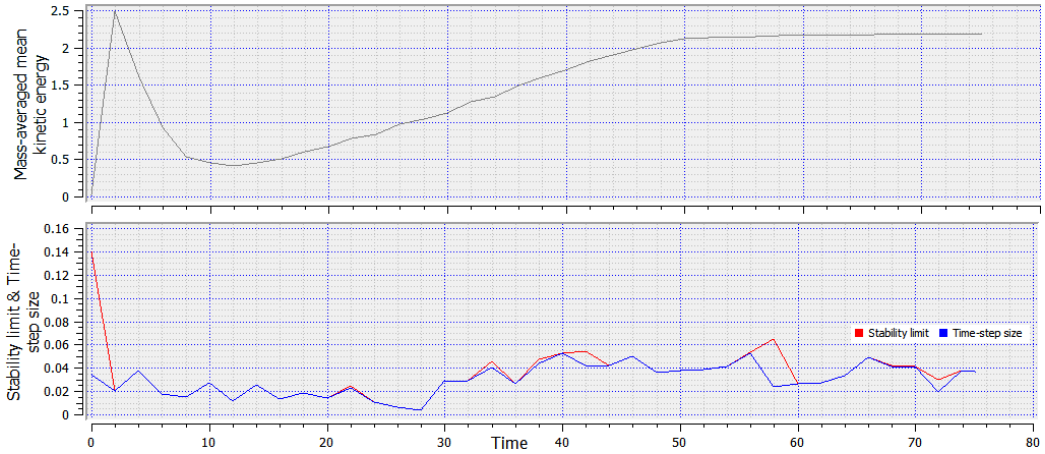

Figure 13: Kinetic energy, time step size, and stability limit plotted over simulation time. **Top:** the velocity of the fluid starts to converge to a steady state at time ~ 50. **Bottom:** the time steps size varies around ~ 0.03.

20.0, 40.0, 60.0, and 75.1.

Figure 12 shows plots over simulation time. The plot on top illustrates the kinetic energy, thus, velocity over time. Velocity is high at the beginning, when the initial fluid region is evening out. Thereafter, the fluid slows down and starts to accelerate due to the slope, the gravity and the mass source placed up-streams. At a simulation time of about $50[s]$ it is about to reach stability. The simulation finally stops at $75[s]$. The lower plot shows the chosen time step sizes and a stability limit of the numerical solver. The time step size is adaptively chosen between 0.005 and 0.055.

The computation was run on a Windows 8.1 64 bit system with six core Intel Xeon $X56650$ @ $2.67GHz$ and $24GB$ RAM under the hood. The log of the simulation is provided in Appendix A.1. Flow3D used *one* of the available 12 hyper-threads for its numerical computation, which took 13 minutes and 29 seconds.

Particle Simulations: For the particle simulations a custom C++ code was used and extended within the Visual Studio 17 integrated development environment. The original codes were developed by [Bender, 2018b] and [Bender, 2018a], and adjusted for river flow related numerical experiments in [Schmidt, 2017]. A new fluid simulation program was implemented for this work, along with some new features required for the study. A source/sink process was added, which moves particles flowing out of the simulation bounding box back into a fluid source bounding box, therein, at a randomized position. Any particle flowing out of the simulation bounds and, thus, also out at the river down-stream cross section at $z = -160$, is

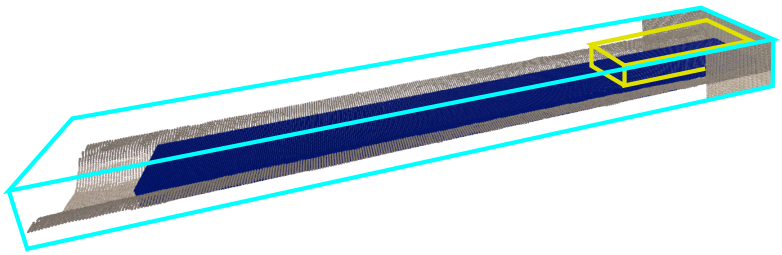

Figure 14: Simulation setup for SPH/PBD: river bed (grey), simulation bounds (cyan), particle source bounds (yellow), and initial particles (blue).

moved back into the simulation space as a 'rain drop' in the source bound.

Note that the coordinate system was adjusted in comparison to Flow3D. In the Flow3D setup gravity was in $-z$ and in the particle code it is now $-y$. Accordingly, the river axis, and flow direction, changed from $+y$ in Flow3D to $-z$ in the particle simulation. Further, a data structure to capture the most important simulation configurations was created, together with a save and load features via an ASCII file: 'start.cfg'. Parameters, such as, a simulation name, the particle radius, the source bounds, the simulation bounds, the simulation mode, the time step size, etc. can be configured without compiling the source code. An example is shown in Appendix A.2. To load the prepared geometry file of the river bed an OBJ import was implemented. The original code used a much smaller scale in the simulations. This was adjusted to meet meters as the overall unit scale. The scaling was introduced as a variable, that now allows to scale the whole simulation scene with one value. The possibility to store and load a full fluid state was implemented via ASCII files. For particle variables CSV files were chosen, and separated into a boundary particle and a fluid particles file. They can be directly opened and inspected, e.g., in a text editor or a spreadsheet application. Meta information is stored in a different file as key-value pairs, similar to the configuration file. Loading a previously computed fluid states reduces waiting times in case of parameter variation, or debugging the simulation code. To enable creation of animated movies of the simulation a numbered image output rendered into PNG files was added. Finally, the computational loop was equipped with an output of gathered information per simulation time step into an ASCII CSV file: mean value and standard deviation of particle density, pressure, and velocity are stored. Figure 15 mid and left shows theses values of a final simulation run via line plots.

The simulation setup is shown in Figure 14. The simulation bounds are set to $(-20, -1, -160)$ to $(20, 6, 0)$ and the bounds of the fluid source to

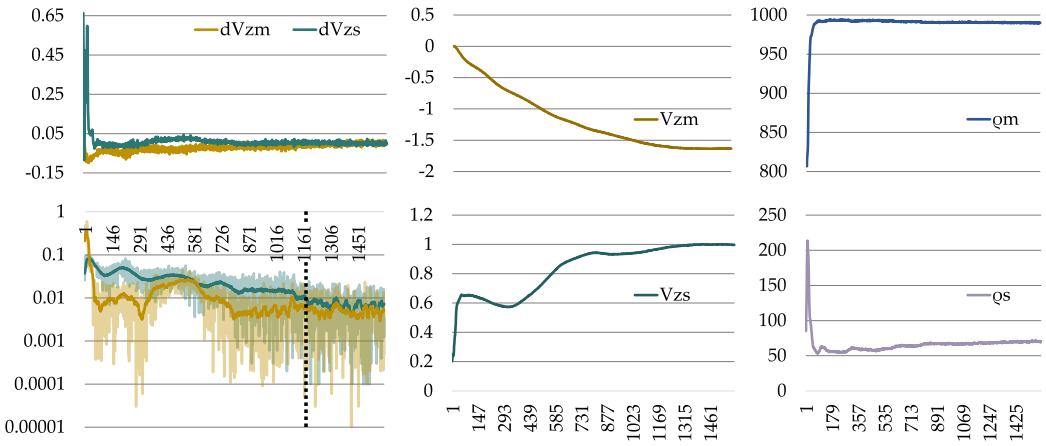

Figure 15: Mean (m) and standard deviation (s) plots per time step of the PBD simulation. The velocity component v_z and the density ρ are illustrated. **Right:** ρ becomes stable at ~ 800 steps. **Mid:** V_z becomes stable at ~ 1200 steps. **Left:** The first derivative of the mean and the standard deviation of V_z. In the logarithmic plot the derivatives go below 0.01 at ~ 1200. The curves were smoothed using a median and a local averaging filter.

$(-8, 4, -40)$ to $(8, 10, -1)$. Boundary particles do not change position and have been loaded via the retopo OBJ file. Wall boundaries are set up in the upstream area at $z = 0$ and on the sides downstream for $20[m]$. Initial water particles are uniformly placed as a breaking dam within $(-10, 1, -143)$ and $(14, 5, -1)$. Some of the water particles are initially placed below the river bed, but after a few simulation steps they go out of range and are moved into the source box. Particles were placed quite low on the surface to start with a low potential energy in the fluid.

PBD Simulation: The PBD simulation was run using different steps sizes and two different particle radii. Several tries were necessary for manual parameters variation. Finally, the water table height of $\sim 3.03[m]$ at the reference cross section $z = -140$ in a steady state was reached. Figure 15 illustrates the mean and standard deviations for density, pressure, and z-velocity of the particles over simulation time steps. The mean density is used to analyze the state of the simulation. It should be stable and close to the water density of $1000[kg/m^3]$. A steady flow state, however requires the velocity field to be stable. The mid column shows the mean and standard deviation of the velocity z component. The z-axis is aligned with the river axis. This relates to the kinetic energy plot of Figure 13. Here, the convergence is smoother. To find a criterion for a possible steadyness detection, the first derivatives of the velocity component's mean and standard

derivation were computed by central differences (top left). A logarithmic scale of its absolute values reveals how the changes in velocity are decreasing (bottom left). The mean and the standard deviation go down to below 0.01 at ~ 1200 time steps - corresponding to a simulation time of $60.0[s]$. The values in the logarithmic plot were smoothed by a median filter of size 5 and a local averaging of radius 10 samples. The steady simulation time is similar compared to the Flow3D simulation.

The computation was done on the same hardware system. But the particle code used *all* of the hyper-threads for its numerical computation, which took about 60 to 100 minutes, dependent on where stability is considered. Note that, the particle method is especially suited for GPU implementation which creates a speedup of one or even two orders of magnitude.

SPH Simulation: The PCISPH simulation code required major corrections and rewrites, especially, in the pressure correction (PC) loop. At the time of writing, numerical stability was achieved with corrected scales, at large time stepping of 0.01 and a manually configured $\delta = 1000$, see (34). In free fall 3 iterations of the PC loop was reached. Using $200k$ particles iterations can go up to 500 when colliding with the ground. Computation time per time step ranges from $0.7[s]$ to $113.2[s]$. More fine tuning and optimization is required here, but by means to the thesis' scope left for future work.

5 Results

Volumetric Flow Rate Estimation: The flow and mean velocity at the reference cross section of the river section was first estimated using an analytic hydraulic equation. The Gauckler-Mannig-Strickler equation (GMS) was used for that purpose [Manning, 1891]:

$$Q = Av_m = A \cdot k_{St,m} \cdot R_{hyd}^{2/3} \cdot I_0^{1/2} \tag{60}$$

$$R_{hyd} = \frac{A}{U} \tag{61}$$

$$k_{St} = \frac{26}{d_{90}^{1/6}} \tag{62}$$

$$k_{St,m} = \left(\frac{U}{\sum \frac{U_i}{k_{St,i}^{3/2}}} \right)^{2/3} \tag{63}$$

It is based on the continuity equation and valid for flowing water in open cross sections. It expresses the mean velocity by the roughness coefficient $k_{St,m}$, the hydraulic radius R_{Hyd}, and the slope I_0 of the flume. For the

Position [m]	B	H	b1	h1	b2	h2	n1	n2
120	12.15	0.10	7.90	4.30	6.60	3.30	1.84	2.00
130	11.85	0.05	8.15	4.05	6.55	3.35	2.01	1.96
140	11.45	0.05	9.55	3.80	6.00	3.40	2.51	1.76
150	11.00	0.05	8.20	4.45	7.65	4.45	1.84	1.72
160	11.75	0.05	6.50	2.95	5.85	2.95	2.20	1.98
mean	11.64						2.08	1.88

Table 1: Cross section dimensions in numbers yielding mean width B and mean inclinations $n1$ and $n2$.

chosen river section, I_0 and R_{Hyd} can be both estimated from the geometry captured by the bathymetric LiDAR. Five cross sections around the reference cross section at $z = 140$ were chosen: 120, 130, 140, 150, and 160. A trapezoid was fitted manually to the cross sections, see Figure 16. The widths and inclinations of the embankment are collected in Table 1. The mean width B and the mean inclinations $n1$ and $n2$ were computed by the arithmetic mean. With the known water table height from the measurement, the area, the circumference are computed and yield the hydraulic radius R_{Hyd}, see (67).

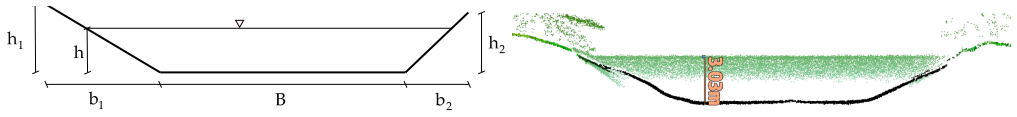

Figure 16: A trapezoid was chosen to represent the measurement.

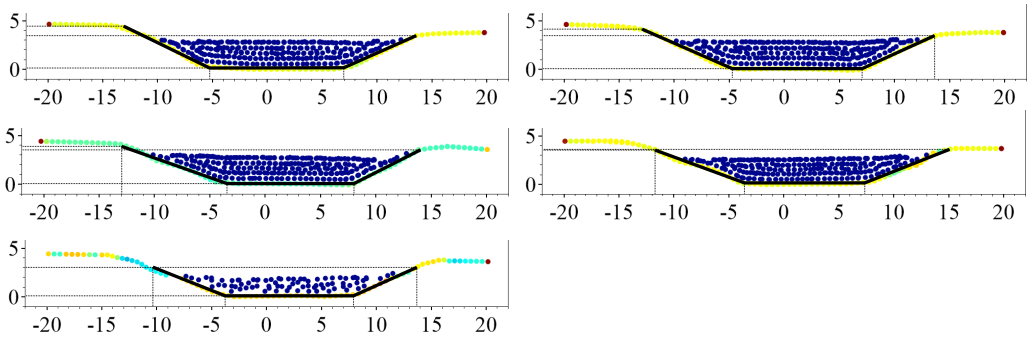

Figure 17: Trapezoids (black line) were manually fitted to five selected cross sections around the reference at $z = 140[m]$.

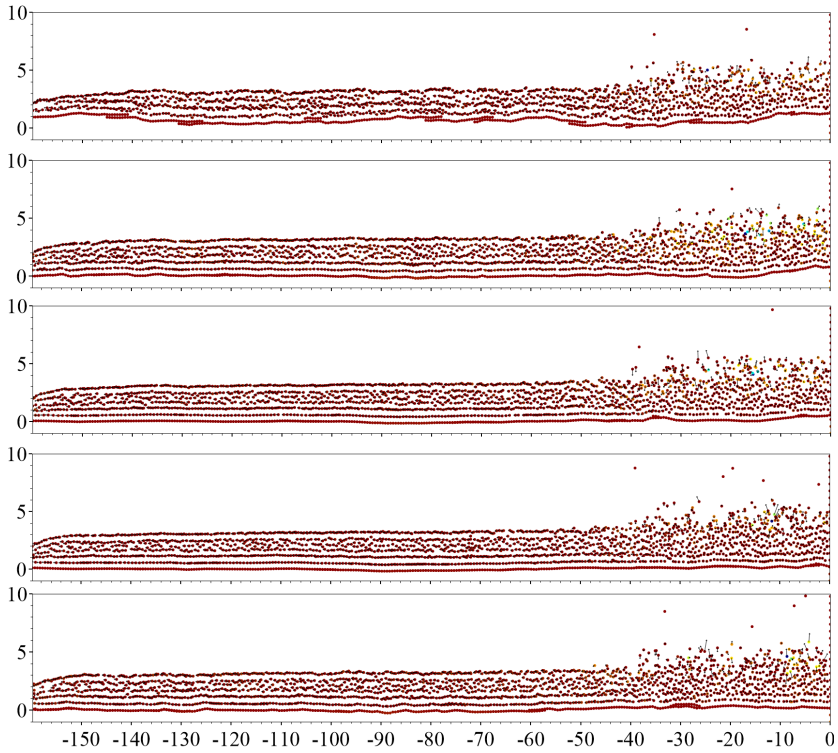

Figure 18: Five longitudinal sections of the river bed x: -6, -3, 0, 3, 6.

$$
\begin{aligned}
h &= 3.03[m] && (64)\\
A &= B \cdot h + h^2 \cdot n_1/2 + h^2 \cdot n_2/2 = 53.45[m^2] && (65)\\
U &= B + h\sqrt{1+n_1^2} + h\sqrt{1+n_2^2} = 25.09[m^2] && (66)\\
R_{Hyd} &= 53.45/25.09 = 2.13 && (67)
\end{aligned}
$$

To estimate the slope of the flume longitudonal sections were made. The slope was computed as the mean of five sections ($x = -6, -3, 0, 3, 6$) around the river axis using the y values at $z = 0$ and $z = 160$:

$$
\begin{aligned}
I_6 &= (0.2 - 0.07)/160 = 0.0008\\
I_3 &= (0.24 - 0.13)/160 = 0.0007\\
I_0 &= (0.57 - 0.078)/160 = 0.0013\\
I_{-3} &= (0.8 - 0.02)/160 = 0.0049\\
I_{-6} &= (1.29 - 0.07)/160 = 0.002\\
I_{avg} &= 0.00194 && (68)
\end{aligned}
$$

Now, all parameters except of the roughness of the river bed are fixed. The roughness can be estimated by the diameter of the sediment. No soil

measurements or other estimatations were available, thus a variation and rough guess was done by looking at some photographs of the river close to the chosen section. No larger grain seems to be present and thus rather small diameters for d_{90} were used for a parameter variation. The grain diameters for the inclination was estimated as half the size of the bed. Therefrom, the Strickler coefficient for the inclination and bed were computed, yielding the mean coefficient $K_{St,m}$ weighted by the circumferences, see (63). The results of the variation are gathered in Table 2. The mean velocity v_m varies from about 1.8 and 2.9 $[m/s]$ for the chosen diameters. This defines a first reference and order of magnitude to be excepted from the numerical computations.

Flow3D Simulation: The Flow3D (F3D) simulation was run locally with a maximum simulation time $t = 100[s]$ and an early break condition in case of reaching a steady state. The textual simulation output is added to appendix A.1. It reached steady a state at $t = 75[s]$ with a computation time of about $14[mins]$. Figure 19 shows the results of different time steps at the reference cross section. To reach a water table of about $3[m]$ several simulations runs have been performed and, finally, a volume source rate of 108 chosen as the best fitting solution. The river-axis velocity component is in the interval from $0.96[m/s]$ to $2.14[m/s]$. To compute the mean velocity v_m a text-file output of the cross section was created including the result fields: positions x, y, z, fraction of fluid, and velocities u, v, w. Taking the fraction of fluid f per cell i into account:

$$v_m = \frac{\sum_i f_i v_i}{\sum_i f_i} = \frac{229.4}{110.9} = 2.07[m/s] \tag{69}$$

$$A = vol_{cell} \sum_i f_i = 0.43 \cdot 110.9 = 47.64[m^2] \tag{70}$$

$$Q = A v_m = 98.61[m^3/s], \tag{71}$$

d_{90S} $[mm]$	d_{90B} $[mm]$	$k_{St,S}$ $[m^{\frac{1}{3}}/s]$	$k_{St,S}$ $[m^{\frac{1}{3}}/s]$	$k_{St,m}$ $[m^{\frac{1}{3}}/s]$	R_{hyd} $[1/m]$	I $[\%]$	Q $[m^3/s]$	v_m $[m/s]$
1.00	2.00	26.00	23.16	24.38	2.13	0.10	68.22	1.28
1.00	2.00	26.00	23.16	24.38	2.13	0.20	96.48	1.81
1.00	2.00	26.00	23.16	24.38	2.13	0.19	95.03	1.78
0.50	1.00	29.18	26.00	27.36	2.13	0.19	106.66	2.00
0.01	0.25	53.97	32.76	39.49	2.13	0.19	153.95	2.88

Table 2: Mean velocity and flow rate computed by the Gauckler-Mannig-Strickler equation (GMS) with varying, rather small, grain diameters.

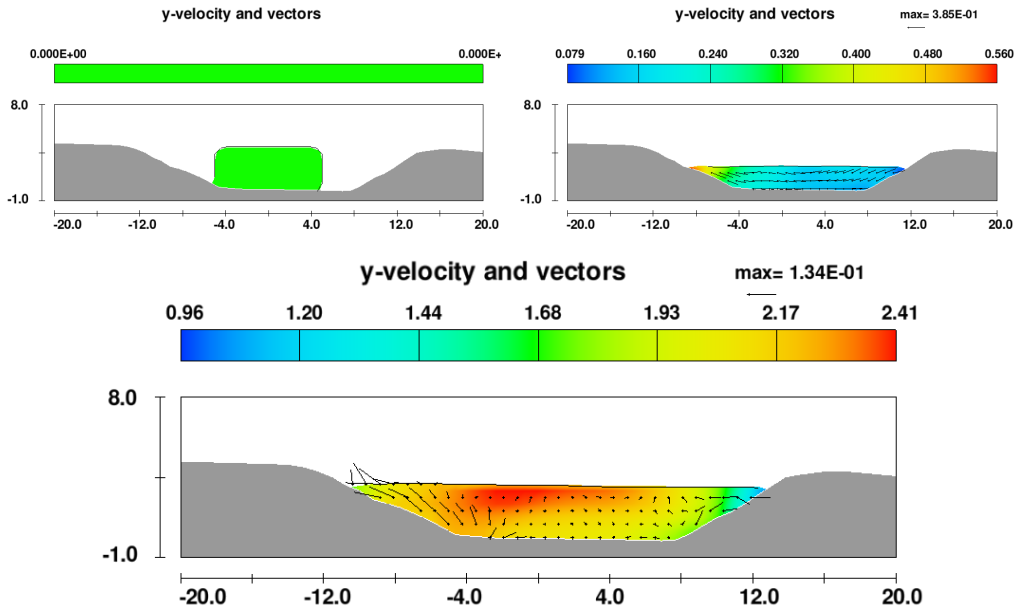

Figure 19: Results of a final Flow3D simulation at different time steps and the reference cross-section at $y = 140[m]$. The simulation was fine tuned to meet the water table height given by the LiDAR measurement. Color shows the velocity component in river-axis direction and the black lines illustrate velocity components in-plane.

with $vol_{cell} = 0.75 \cdot 0.83 \cdot 0.69$.

PBD Simulation: The PBD simulation was performed locally on 12 cores and the mean velocity and density trace over time. When a steady state was reached, the state of the simulation was stored as ASCII CSV result files. Figure 20 shows different time steps from the simulation. Similar to the Flow3D setup the initial fluid is defined in a cuboid region.

To analyze the CSV files a mini application was developed to visualize cross sections in a plotting style, similar to the 2D output module of Flow3D. The whole state of one time step is read, filtered to cut out cross section particles and plotted. By choosing $F\#$ along with the FSharp-Data and OxyPlot libraries, a stand alone multi-platform tool can be developed in a few lines of code. The main function including e.g. data filtering and mouse controls is added to the Appendix A.3 and shows the compactness of the language. Columns of the CSV files are selected for plotting by theirs names. Circles are plotted for each particle using a similar color-map as Flow3D. Arrows visualize the in-plane velocity components. The velocity component in river-axis is shown by the colors, and mean value is computed and added to the title. An interactive mouse control allows to walk

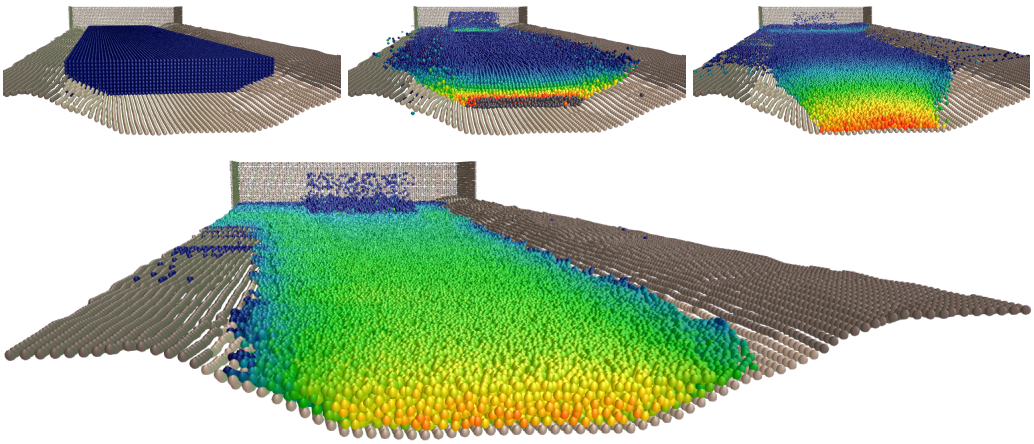

Figure 20: Results of a final PBD setup at different time steps. The simulation was fine tuned to meet the water table height given by the LiDAR measurement at $z = -140[m]$. Color represents the velocity in $-z$ direction. **Bottom:** Final simulation state at simulation time $t = 60[s]$.

through the results along the z-axis, here, corresponding to the river-axis.

Figure 21 illustrates such a plot at the reference cross-section at $z = -140$ of a final PDB simulation. The river-axis velocity component is in the interval from $0.5[m/s]$ to $3.4[m/s]$. The cross section depth is the diameter of a particles. With N, the number of cross section particles and w the velocity component in z direction:

$$v_m = \frac{\sum_i w_i}{N} = 2.42[m/s] \tag{72}$$

$$A = Nr^2\pi = 397 \cdot 0.2^2 \cdot \pi = 49.89[m^2] \tag{73}$$

$$Q = Av_m = 120.74[m^3/s], \tag{74}$$

The smallest velocities are at the boundaries and highest at the water table. **Comparison:** The results of three different methods are gathered in Table 3: GMS, FVM, and PBD. Mean velocity, flow rate, water table height, number of discretization elements, element size, and computation time are listed. Overall, the velocities have the same order of magnitude and are in the interval from 2.0 to 2.42. GMS and F3D coincide better. The PBD is about 16% higher compared to the F3D for the velocity and 22% to the flow rate. When comparing the PBD-runs 10 and 13, where the time step size was halved, one notices almost no difference. Generally, the time step size of F3D and PBD are quite similar. Note that time step sizes for other SPH methods are usually located around 0.001, but PBD allowed for larger steps. When looking at PBD-runs 10 and 20, where the particle radius was

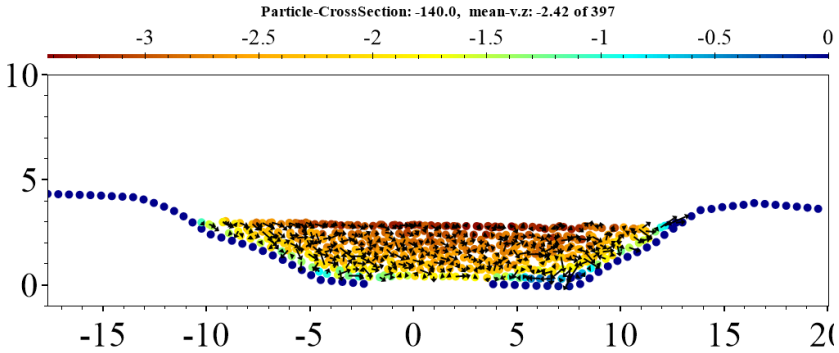

Figure 21: Cross section of a PBD-run 20 at $z = -140[m]$. The depth is equal to the particle diameter. The velocity is illustrated as color-map in river-axis component and via black arrows for the in-plane components.

Method-Run	Time Step	v_m $[m/s]$	A $[m^2]$	Q_m $[m^3/s]$	H_w $[m]$	#Elems	Size $[m]$	Time $[min]$
GMS-1	-	2.00	53.45	106.66	3.03	-	-	240
F3D-30	0.03	2.07	47.64	98.61	3.0	133/214k	0.5	13.5
PBD-10	0.05	2.75	52.3	143.25	3.18	185/60k	0.3	37
PBD-13	0.025	2.75	52.6	144.65	3.18	186/60k	0.3	37
PBD-14	0.05	2.58	38.80	100.18	2.7	309/198k	0.2	60
PBD-15	0.05	2.51	49.51	124.27	3.3	394/216k	0.2	70
PBD-20	0.05	2.42	49.89	120.74	3.0	397/216k	0.2	60

Table 3: Simulation parameters and results at the reference cross section of all computational methods. The two values in the $\#Elems$ column represent the number of elements within the reference cross section followed by the total number of elements used in the simulation.

reduced from $0.3[m]$ to $0.2[m]$, the fluid slows down, coming closer to the F3D and GMS values. Further decreasing the particle size could improve the results. This comes at cost in computation time. But, computation time did not increase linearly - elements size was doubled but computation time increased only by a factor of 1.6. The final PBD-run 20 takes about 4.44 times longer than the F3D simulation. Note that the F3D simulation was executed on a single core and the PBD on 12 cores. However, PBD is suited very well for GPU computation.

6 Conclusion and Future Work

Conclusion: A bathymetric LiDAR dataset of a river was chosen and a short river section selected, which fulfills conditions easing manual and numerical computations thereon. The section was cut out containing the separated ground and water points and a hydraulic triangular mesh. The mesh was further prepared by aligning the river axis to the Cartesian coordinate axes. A uniformly refined mesh was created by retopology of the adaptive original. A reference cross section was chosen $20[m]$ up-stream of the end of the river section and its water table height was determined from the point cloud.

Numerical simulations were prepared. A simulation in Flow3D was setup for results based on the finite volumes method. Several simulation runs were conducted and tuned manually, such that the water table as the reference cross section matches the measured value. A simulation setup in a custom developed *C++* code was set up for results based on the position based dynamics and the smooth particle hydrodynamics methods. The code was extended by features to enable and ease river flow based simulations, e.g. save/load of the fluid state, source/sink procedures, and convergence variables output. A manual calculation and estimation of mean velocity and flow rate was done utilizing the Gauckler-Mannig-Strickler formula for free surface flows in a trapezoid shaped channel. This serves as a plausibility reference for the numerical computations.

Finally, the results are compared. All results are located in the same order of magnitude and are located within a variation of 22%. Also, there are still some uncertainties involved. Especially, the grain diameters (roughness) are just estimated. There is no roughness model in the PBD, but maybe it even is not required, when the LiDAR resolution is high enough. One has to keep in mind, that the implicit finite difference method used within Flow3D has been fine tuned and optimized for more than a decade. The PBD code is new and optimization will improve the early results.

Future Work: Firstly, a flow simulation using the LiDAR points directly as boundary condition can now be done - everything is prepared therefore. Generally, more river sections capturing different flow features should then be selected for investigation: curves, widenings, weirs, falls, bifurcations, etc. Further work can improve the PBD/SPH computations. Solving within the simulation using both methods in combination is expected to have better results in critical (e.g. turbulent) flow regions by PCISPH and a higher computational efficiency by the PBD in calm flow regions. PBD/PCISPH could be extended to include a roughness parameter on the boundary, based on geometry and LiDAR signal parameters. Maybe, this is not even

necessary and naturally captured by the geometry of laser scan with high density. In a river flow simulation, one usually is interested in reaching a steady flow state of the fluid. Therefore, an adaptive step size control can be applied to faster converge to a steady solution. E.g. after the first initial phase, time step size could be enlarged to speed up the acceleration process of the fluid body, and in the final steps shrinked to improve quality and capture finer details of the flow, [Goswami and Pajarola, 2011]. Also, methods could be switched over time. Particles need not have to have the same radii. In calm flow regions particles could be merged, further, reducing computational complexity. SPH/PBD methods are especially suitable for porting to the GPU. In a final step such a port can produce a speed-up of up to two orders of magnitude in computation time, which would then enable a mesh free flow analysis of large river sections. Finally, the fluid code could be extended for an automated parameter variation until it meets the water table at reference cross sections or minimize the error for the whole measured water surface. Grain and thus sediment transport could be directly modeled by introducing grain particles on the river bed having a higher density. Maybe, the fluid simulation can also be used to improve the LiDAR data, by solving an inverse problem. In case of data holes, preliminary boundary particles could be placed and their positions shifted and optimized such that the flow parameters and the water surface are matched best. This would decrease the uncertainty for data fill-ins.

7 Acknowledgements

I thank AHM - Airborne Hydromapping GmbH, for providing the anonymized data-set, especially, the team of data acquisition and data processing, and the CEO Frank Steinbacher. I thank Daniel Schiffner for fruitful sessions and discussions, and getting a first work of our ideas started at the Professorship for Graphical Data Processing, Goethe University of Frankfurt via the master thesis by Johannes Schmidt. Special thanks go to Robert Klar, Katharina Schneider, and Bernd Steidl of the Unit for Hydraulic Engineering, Univerity of Innsbruck, who encouraged and supported me, and gave me the opportunity to reduce my carryovers from the past. Finally, many thanks to Prof. Matthias Harders and the team at the Interactive Graphics and Simulation Group (IGS) for providing an inspiring supportive work environment and maintaining the spirit of high quality standards and, especially, Fernando Zorrilla for last minute support.

A Appendix

A.1 Flow3D Simulation Output Overview

```
Preprocessor Starting

 processing options and properties
 processing mesh
 processing geometry components
 processing non-moving component     1 in mesh block    1
   processing cad data for subcomponent    1
   total, fluid and solid sub-domain cell counts:
    254774    182910     45547
 processing initial conditions
   processing fluid initialization regions
 processing baffles
 setting up remaining array data
 evaluating mesh boundary conditions for block   1
 initializing fluid boundary conditions
 processing graphics and output requests
 processing particle data
 processing user-defined plot requests
 producing preprocessor plot data
 successful completion of preprocessor

Preprocessor Done

Solver starting

trying to check out a serial token

 License token checked out for simulation:  hydr3d
 Number of core license tokens checked out:   0

 program title : FLOW-3D
 program version :   hydr3d   version  11.1.4.2  win64 2016
 version id : double
 DOUBLE precision version
   process identification number for this job=     1

 job name:     vkqb
 problem date: 10/25/2018
 problem time: 13:13:29
```

```
 Title

*** estimated uncompressed solver output file size (flsgrf):   79 mb  ***
*** running serial code ***
***
 restart and spatial data available at t= 0.00000E+00
***
```

progress		time step		pressure		fluid #1			performance			
sim_time	cycle	delt	dt_stbl/code	iter	res/epsi	volume	%loss	frac	el_time	%PE	clk_time	est_rem_time
0.00000E+00	0	1.40E-01	1.40E-01/fs	0	0.00E+00	9.6878E+03	+0.00E+00	0.16	00:00:07	100	13:13:30	...

convective flux exceeded stability limit
 at t= 1.4032E-01 cycle= 1 iter= 4 delt= 1.4032E-01 mesh block 1
 restarting cycle with smaller time step

2.00965E+00	115	2.00E-02	2.04E-02/cy	1	3.24E-01	9.7045E+03	+5.01E-01	0.16	00:00:50	69	13:14:13	01:11:19
4.01054E+00	179	3.72E-02	3.72E-02/cx	2	8.34E-01	9.8743E+03	+4.34E-01	0.16	00:01:18	66	13:14:41	00:57:49
6.01503E+00	268	1.76E-02	1.76E-02/cz	1	3.08E-01	1.0134E+04	+2.29E-01	0.17	00:01:58	88	13:15:22	01:00:02
8.02251E+00	413	1.45E-02	1.45E-02/cz	1	2.54E-01	1.0342E+04	+1.48E-01	0.17	00:02:59	76	13:16:22	01:08:38
1.00471E+01	562	2.68E-02	2.68E-02/cx	1	4.53E-01	1.0522E+04	+1.16E-01	0.17	00:03:59	100	13:17:22	01:13:12
1.20540E+01	672	1.13E-02	1.13E-02/cx	1	1.02E-01	1.0710E+04	+1.04E-01	0.18	00:04:44	69	13:18:07	01:12:04
1.40699E+01	817	2.42E-02	2.42E-02/cz	1	1.73E-01	1.0887E+04	+1.01E-01	0.18	00:05:42	59	13:19:05	01:13:47
1.60723E+01	933	1.31E-02	1.31E-02/cx	1	1.97E-02	1.1082E+04	+9.85E-02	0.18	00:06:24	92	13:19:47	01:11:57
1.80728E+01	1087	1.97E-02	1.97E-02/cz	1	1.01E-01	1.1267E+04	+9.82E-02	0.19	00:06:56	98	13:20:19	01:08:37

***
 restart and spatial data available at t= 2.00044E+01
***

2.00044E+01	1201	1.39E-02	1.39E-02/cz	1	1.99E-02	1.1444E+04	+9.62E-02	0.19	00:07:19	100	13:20:43	01:04:53
2.00742E+01	1206	1.40E-02	1.40E-02/cz	1	2.16E-02	1.1450E+04	+9.62E-02	0.19	00:07:21	100	13:20:44	01:04:49
2.20961E+01	1323	2.68E-02	2.68E-02/cz	1	1.64E-01	1.1643E+04	+9.46E-02	0.19	00:07:45	100	13:21:08	01:01:28
2.41016E+01	1443	1.01E-02	1.01E-02/cx	1	3.94E-02	1.1833E+04	+9.68E-02	0.20	00:08:10	100	13:21:34	00:58:49
2.61024E+01	1695	6.28E-03	6.28E-03/cx	1	8.23E-03	1.2024E+04	+9.68E-02	0.20	00:09:02	100	13:22:26	00:59:27
2.81051E+01	2122	3.70E-03	3.70E-03/cx	1	3.99E-04	1.2222E+04	+9.76E-02	0.20	00:10:28	91	13:23:52	01:03:22
3.01240E+01	2252	2.72E-02	2.72E-02/cx	1	2.45E-01	1.2414E+04	+9.47E-02	0.21	00:10:56	100	13:24:19	01:00:60
3.21297E+01	2340	2.82E-02	2.82E-02/cx	1	1.81E-01	1.2609E+04	+9.19E-02	0.21	00:11:14	100	13:24:38	00:58:08
3.41304E+01	2410	4.14E-02	4.14E-02/cz	1	5.53E-01	1.2803E+04	+8.84E-02	0.21	00:11:29	100	13:24:52	00:55:15
3.61313E+01	2478	2.40E-02	2.40E-02/cz	1	8.17E-02	1.3005E+04	+9.14E-02	0.22	00:11:43	100	13:25:06	00:52:37
3.81772E+01	2537	4.72E-02	4.72E-02/cy	1	7.99E-01	1.3216E+04	+9.10E-02	0.22	00:11:55	96	13:25:19	00:50:03

***
 restart and spatial data available at t= 3.99743E+01
***

3.99743E+01	2581	5.28E-02	5.28E-02/cz	1	5.94E-01	1.3384E+04	+9.63E-02	0.22	00:12:04	100	13:25:28	00:47:53
4.01915E+01	2586	3.36E-02	3.36E-02/cz	1	4.69E-01	1.3403E+04	+9.71E-02	0.22	00:12:05	100	13:25:29	00:47:38

progress		time step		pressure		fluid #1			performance			
sim_time	cycle	delt	dt_stbl/code	iter	res/epsi	volume	%loss	frac	el_time	%PE	clk_time	est_rem_time
4.22207E+01	2636	4.90E-02	4.90E-02/cy	2	3.99E-02	1.3588E+04	+1.03E-01	0.23	00:12:16	97	13:25:39	00:45:25
4.42455E+01	2685	3.96E-02	3.96E-02/cy	1	5.50E-01	1.3784E+04	+1.07E-01	0.23	00:12:26	99	13:25:49	00:43:22
4.62850E+01	2742	4.79E-02	4.80E-02/cy	1	8.60E-01	1.3988E+04	+1.15E-01	0.23	00:12:38	100	13:26:01	00:41:35
4.83283E+01	2790	4.44E-02	4.44E-02/cz	1	4.91E-01	1.4189E+04	+1.18E-01	0.24	00:12:48	100	13:26:12	00:39:50
5.03297E+01	2844	3.33E-02	3.33E-02/cx	1	1.26E-01	1.4267E+04	+1.19E-01	0.24	00:12:59	100	13:26:23	00:38:18
5.23663E+01	2896	3.92E-02	3.92E-02/cy	1	2.32E-01	1.4257E+04	+1.17E-01	0.24	00:13:10	100	13:26:34	00:36:49
5.44133E+01	2947	4.89E-02	5.00E-02/cz	1	1.49E-01	1.4229E+04	+1.14E-01	0.24	00:13:21	100	13:26:45	00:35:26
5.64670E+01	2991	5.46E-02	5.46E-02/cy	1	2.31E-01	1.4209E+04	+1.12E-01	0.24	00:13:30	98	13:26:54	00:34:03
5.84728E+01	3067	4.59E-02	6.98E-02/cy	1	1.36E-01	1.4193E+04	+1.10E-01	0.24	00:13:47	100	13:27:10	00:33:04

***
 restart and spatial data available at t= 6.00063E+01
***

6.00063E+01	3113	2.64E-02	2.64E-02/cz	1	5.45E-02	1.4180E+04	+1.08E-01	0.24	00:13:56	96	13:27:20	00:32:15
6.04916E+01	3132	2.51E-02	2.51E-02/cz	1	6.33E-02	1.4176E+04	+1.08E-01	0.24	00:14:00	99	13:27:24	00:32:02
6.25012E+01	3208	2.74E-02	2.74E-02/cz	1	3.06E-02	1.4153E+04	+1.06E-01	0.23	00:14:16	100	13:27:40	00:31:09
6.45220E+01	3277	3.89E-02	3.89E-02/cz	1	8.37E-02	1.4127E+04	+1.04E-01	0.23	00:14:31	99	13:27:54	00:30:14
6.65458E+01	3326	4.03E-02	4.37E-02/cy	1	1.04E-01	1.4107E+04	+1.02E-01	0.23	00:14:41	100	13:28:04	00:29:13
6.85695E+01	3382	4.54E-02	4.54E-02/cy	1	1.04E-01	1.4091E+04	+9.98E-02	0.23	00:14:52	97	13:28:16	00:28:17
7.05827E+01	3434	2.04E-02	2.04E-02/cx	1	1.93E-02	1.4070E+04	+9.80E-02	0.23	00:15:03	100	13:28:26	00:27:23
7.25913E+01	3520	3.20E-02	3.20E-02/cy	1	2.13E-02	1.4055E+04	+9.62E-02	0.23	00:15:21	99	13:28:44	00:26:44
7.46196E+01	3588	3.53E-02	3.53E-02/cy	1	4.22E-02	1.4039E+04	+9.47E-02	0.23	00:15:35	99	13:28:58	00:25:59

***
 restart and spatial data available at t= 7.50595E+01
***

7.50595E+01	3600	3.66E-02	3.66E-02/cy	1	8.52E-02	1.4035E+04	+9.43E-02	0.23	00:15:37	96	13:29:01	00:25:49

```
    end of calculation    at t=    7.50595E+01      cycle =    3600
    total fluid mass has become steady

    elapsed time =    9.37401E+02 seconds, or
                      0 days :  0 hours : 15 minutes : 37 seconds

    cpu =    8.71109E+02 seconds

    date of completion = 10/25/2018
    time               = 13:29:01

Postprocessor starting
   reading general data catalogue
   reading mesh block data catalogue  1
   reading particle data catalogue
   constructing time edit index
   processing plot requests
Postprocessor Done

Simulation run complete
```

A.2 Particle Simulation Configuration File

```
runNr=0
scale=1
particleRadius=0.2
boundMin=-20, -1, -160
boundMax=20, 6, 0
sourceMin=-8, 4, -40
sourceMax=8, 10, -1
sourceVel=0, 0.1, -0.1
lookAt=-10, -10, -80
observer=10, 10, -200
damWidth=60
damDepth=360
damHeight=10
damStart=-10, 1, -1
sphMode=0
logFlag=0
tss=0.05
name=PBD01
```

A.3 F# Plotting Tool

```
let main argv =
    let path = argv.[0]

    let mutable zPos  = -140.0
    let mutable title = sprintf "CrossSection: %f" zPos
    let mutable model = new PlotModel()

    let data =
        readPoints path "model.part.csv"
            "m_x.x" "m_x.y" "m_x.z" "m_v.x" "m_v.y" "m_v.z"
        |> Seq.append
            (readPoints path "model.bound.csv"
                "m_boundX.x" "m_boundX.y" "m_boundX.z"
                "m_boundX.y" "m_boundX.y" "m_boundX.y")
        |> Seq.toArray

    let updateModel( zPos ) =
        let w = 0.4
        let miMax = ( zPos - w/2., zPos + w/2.)
        let filtered =
            data
            |> Array.filter( fun x ->
                (x.z >= (min miMax) && x.z <= (max miMax)))

        create filtered (sprintf "CrossSection: %.1f" zPos)

    let model = updateModel( zPos )

    let plot, win = showChartAndRun title model

    let updatePlot() =
        plot.Model <- updateModel( zPos )
        plot.Show()

    plot.MouseWheel.Add (fun ev ->
        if ev.Delta > 0 then
            zPos <- zPos + 1.0
            updatePlot()
        else if ev.Delta < 0 then
            zPos <- zPos - 1.0
            updatePlot()
        else () )

    win.ShowDialog() |> ignore

    0
```

References

[PAT, 1980] (1980). *Numerical Heat Transfer and Fluid Flow*. McGraw-Hill, New York.

[Airborne Hydromapping GmbH, 2018] Airborne Hydromapping GmbH (2018). HydroVish. http://ahm.co.at/software.

[Akinci et al., 2012] Akinci, N., Ihmsen, M., Akinci, G., Solenthaler, B., and Teschner, M. (2012). Versatile rigid-fluid coupling for incompressible sph. *ACM Trans. Graph.*, 31(4):62:1–62:8.

[Ansys-Inc., 2018] Ansys-Inc. (visited 2018). ANSYS. https://www.ansys.com/.

[Autodesk Inc., 2018] Autodesk Inc. (2018). Maya - Computer Animation and Modeling Software. https://www.autodesk.eu/products/maya.

[Autodesk-Inc., 2018] Autodesk-Inc. (visited 2018). AUTODESK CFD. https://www.autodesk.com/products/cfd.

[Batchelor, 1967] Batchelor, G. K. (1967). *An Introduction to Fluid Dynamics*. Cambridge University Press.

[Bender, 2018a] Bender, J. (2018a). Positionbaseddynamics. github.com/InteractiveComputerGraphics/PositionBasedDynamics.

[Bender, 2018b] Bender, J. (2018b). Splishsplash. github.com/InteractiveComputerGraphics/SPlisHSPlasH.

[Chorin, 1968] Chorin, A. J. (1968). Numerical solution of the Navier-Stokes equations. *AMS Mathematics of Computations*, 22:745–762.

[Clay-Math.-Inst., 2018] Clay-Math.-Inst. (visited 2018). Navier-Stokes Equation. http://www.claymath.org/millennium-problems/navier%E2%80%93stokes-equation.

[Fielding, 2018] Fielding, S. (visited 2018). Lecture notes - laminar boundary layer theory. http://community.dur.ac.uk/suzanne.fielding/teaching.html.

[Flow Science Inc., 2018] Flow Science Inc. (2018). FLOW3D - web-page. https://www.flow3d.com/resources/bibliography/water-environmental-bibliography.

[Goswami and Pajarola, 2011] Goswami, P. and Pajarola, R. (2011). Time Adaptive Approximate SPH. In Bender, J., Erleben, K., and Galin, E., editors, *Workshop in Virtual Reality Interactions and Physical Simulation "VRIPHYS" (2011)*. The Eurographics Association.

[Jakob et al., 2015] Jakob, W., Tarini, M., Panozzo, D., and Sorkine-Hornung, O. (2015). Instant field-aligned meshes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH ASIA)*, 34(6).

[Macklin and Müller, 2013] Macklin, M. and Müller, M. (2013). Position based fluids. *ACM Trans. Graph.*, 32(4):104:1–104:12.

[Macklin et al., 2014] Macklin, M., Müller, M., Chentanez, N., and Kim, T.-Y. (2014). Unified particle physics for real-time applications. *ACM Trans. Graph.*, 33:153:1–153:12.

[Manning, 1891] Manning, R. (1891). On the flow of water in open channels and pipes. *Transactions of the Institution of Civil Engin. of Ireland*, 20.

[Monaghan, 1992] Monaghan, J. J. (1992). Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics*, 30(1):543–574.

[Monaghan, 2005] Monaghan, J. J. (2005). Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68(8):1703.

[Moukalled et al., 2016] Moukalled, F., Mangani, L., and Darwish, M. (2016). *The Finite Volume Method in ComputationalFluid Dynamics*. Springer International Publishing Switzerland.

[Navier, 1821] Navier, C.-L. (1821). Sur les lois des mouvements des fluides, en ayant égard à l'adhésion des molecules. *Annales de Chimie et de Physique*, XIX:224.

[NVIDIA, 2018] NVIDIA (visited 2018). Flex. https://developer.nvidia.com/flex.

[Patankar and Spalding, 1972] Patankar, S. and Spalding, D. (1972). A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15(10):1787 – 1806.

[Schmidt, 2017] Schmidt, J. (2017). Animation eines Gewässers basierend auf Punktdaten. Master Thesis, Goethe Univerity of Frankfurt, Germany.

[Solenthaler and Pajarola, 2009] Solenthaler, B. and Pajarola, R. (2009). Predictive-corrective incompressible sph. *ACM Trans. Graph.*, 28(3):40:1–40:6.

[Steinbacher et al., 2012] Steinbacher, F., Pfennigbauer, M., Aufleger, M., and Ullrich, A. (2012). High Resolution Airborne Shallow Water Mapping. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIX-B1:55–60.

[Versteeg and Malalasekera, 2007] Versteeg, H. K. and Malalasekera, W. (2007). *An Introduction to Computational Fluid Dynamics - The Finite Volume Method*. Pearson Education Limited, 2nd edition.