National Technical University of Athens
School of Mechanical Engineering
Department of Fluids
Laboratory of Hydraulic Turbomachines

Development of a meshfree particle method for the simulation of steady and unsteady free surface flows: application and validation of the method on impulse hydraulic turbines.

Doctoral Dissertation

PHOEVOS-CHARALAMPOS K. KOUKOUVINIS
Mechanical Engineer, M.Sc.

Phd Supervisors:
D. Papantonis, NTUA Professor
K. Giannakoglou, NTUA Professor
J.Anagnostopoulos, NTUA Assistant Professor

Examination Committee:
D. Papantonis, NTUA Professor
K. Giannakoglou, NTUA Professor
S. Tsangaris, NTUA Professor
S. Voutsinas, NTUA Associate Professor
D. Mathioulakis, NTUA Associate Professor
J.Anagnostopoulos, NTUA Assistant Professor
V. Riziotis, NTUA Lecturer

Athens, July 2012

Approval of the doctoral dissertation from the School of Mechanical Engineering of National Technical University of Athens (NTUA) does not imply acceptance of the opinions of the author (Law 5343/1932, Article 202)

στους γονείς μου Κωνσταντίνο και Κατερίνα
και στην αδερφή μου Ίριδα

Everything is both simpler than we can imagine
and more complicated that we can conceive
- Goethe

Summary

The aim of the present thesis is the development of algorithms based on the Smoothed Particle Hydrodynamics (SPH) method for the simulation of free surface flows, focusing mainly on the flow in impulse turbines. During the course of the study, several algorithms were developed, based on different variants of the SPH method and were applied to a wide range of problems including inviscid and viscous, enclosed and free surface flows. The developed SPH algorithms were validated using theoretical solutions, numerical or experimental results. The present work also involves novel features, such as the numerical modeling of a Turgo impulse turbine, the design optimization of a Turgo turbine and the development of a new high order variant for Riemann based SPH solvers.

The SPH method is a Lagrangian meshfree particle method based on the works of Gingold, Lucy and Monaghan. SPH is well suited to describe the violent free surface flows occurring in impulse turbines, since the Lagrangian formalism enables tracking of the free surface, whereas the method is able to describe accurately large deformations, splashing and droplets, without significant loss of accuracy.

The SPH method approximates the described medium as a set of particles which occupy volume, carry mass and all the characteristic properties of the described medium, such as density, velocity etc. The Navier - Stokes equations may be rewritten, using summation interpolations, involving a set of randomly distributed particles and a kernel smoothing function. The kernel smoothing function plays an important role in the validity and accuracy of the SPH interpolations.

For the simulation of incompressible fluids, it is possible to employ the solution of a Poisson equation, but doing so results to a cumbersome and slow algorithm; the assumption of the flow as weakly compressible is used instead. However, the SPH method faces several difficulties in handling boundaries and, due to its nature, pressure oscillations arise when simulating weakly compressible flows. Several corrections were examined, involving density filters, diffusion terms etc. to treat the method's inherent weaknesses. Simulation of viscous flows is possible, however at high Reynolds flows unphysical solutions may be produced, due to distorted particle distributions. Advanced treatments, such as particle redistribution help to regularize the particle arrangement and enable such simulations. With the redistribution technique, combined with traditional turbulence models, it is possible to use the SPH method for high Reynolds flows.

Free surface flows were examined, involving simple benchmark cases, such as water jet impingement on a flat plate under different impingement angles, and more complicated practical cases focusing on impulse turbine components. Simulation of a Pelton turbine nozzle was performed, providing satisfactory results in comparison to experimental and numerical data. Moreover, the method was used for the simulation of an impulse turbine deflector, examining whether it could be used for throttling the turbine load. Also the method was applied in impulse turbine runners, both stationary and moving/rotating and it was used in conjunction with a simpler flow solver for the design optimization of a Turgo impulse turbine.

After identifying the main issues of the SPH method, several modifications were tested. The implementation of the SPH method in conjunction with Riemann solvers was examined, in order to treat the pressure oscillation issues. Further modifications, including the discretization of Euler equations under ALE perspective using SPH approximations and implementation of accurate boundary conditions using partial Riemann problems, were tested in basic benchmark problems, such as shock tube, and more practical applications, such as simplified 2d boat hull / water interaction. High order extensions were developed, based on the MUSCL-Hancock scheme. The results of the SPH modifications are compared to the standard SPH method, proving that Riemann based SPH

variants are superior to the standard SPH method, due to the better handling of boundaries and the improved reproduction of the pressure field. Key issue areas are identified and further research is proposed on the subjects of the SPH interpolation accuracy, the ALE formulation and the parallelization capabilities of the SPH method.

Περίληψη

Στόχος της παρούσας εργασίας είναι η ανάπτυξη υπολογιστικών αλγορίθμων βασισμένων στη μέθοδο Smoothed Particle Hydrodynamics (SPH) για την προσομοίωση ροών με ελεύθερη επιφάνεια, εστιάζοντας στις ροές που διαμορφώνονται σε υδροστροβίλους δράσης. Κατά την εκπόνηση της διατριβής αναπτύχθηκαν αλγόριθμοι βασισμένοι σε διάφορες παραλλαγές της μεθόδου SPH οι οποίοι εφαρμόστηκαν σε ένα ευρύ φάσμα προβλημάτων, περιλαμβάνοντας ροές με ή χωρίς ιξώδες, έγκλειστες ή με ελεύθερη επιφάνεια. Οι αλγόριθμοι που αναπτύχθηκαν πιστοποιήθηκαν με τη χρήση αναλυτικών λύσεων, αριθμητικών και πειραματικών αποτελεσμάτων. Η παρούσα εργασία περιλαμβάνει καινοτόμα χαρακτηριστικά όπως την προσομοίωση και σχεδιασμό υδροστροβίλου Turgo, αλλά και την ανάπτυξη ενός νέου σχήματος υψηλής ακρίβειας για τις μεθόδους SPH που βασίζονται σε επιλύτες Riemann.

Η μέθοδος SPH είναι μια μη-πλεγματική, Lagrangian μέθοδος βασισμένη στην ερευνητική δραστηριότητα των Gingold, Lucy και Monaghan. Αρχικά η μέθοδος είχε διαμορφωθεί για την επίλυση πολύπλοκων αστροφυσικών προβλημάτων, ωστόσο σύντομα επεκτάθηκε καλύπτοντας τη συμπεριφορά των ρευστών και στερεών. Η μέθοδος είναι σε θέση να περιγράψει ικανοποιητικά τις περίπλοκες ροές ελεύθερης επιφάνειας που λαμβάνουν χώρα σε υδροστροβίλους δράσης, καθώς μπορεί να περιγράψει με ακρίβεια μεγάλες παραμορφώσεις και σταγονίδια εύκολα.

Η μέθοδος SPH περιγράφει το περιγραφόμενο μέσο σαν ένα πεπερασμένο σύνολο σωματιδίων, καθένα εκ των οποίων καταλαμβάνει όγκο και φέρει μάζα καθώς και όλα τα χαρακτηριστικά του περιγραφόμενου μέσου. Οι εξισώσεις ροής Navier – Stokes είναι δυνατό να διακριτοποιηθούν χρησιμοποιώντας αθροίσματα προσέγγισης από ένα αυθαίρετα κατανεμημένο σύνολο σωματιδίων σε συνδυασμό με μια συνάρτηση ομαλοποίησης. Η εν λόγω συνάρτηση επηρεάζει δραματικά την ακρίβεια των προσεγγίσεων.

Για την προσομοίωση ασυμπίεστης ροής είναι δυνατό να γίνει η επίλυση της εξίσωσης Poisson, αλλά ο τελικός αλγόριθμος είναι αργός, ενώ υπάρχουν δυσκολίες και στον παραλληλισμό του. Γι' αυτόν τον λόγο γίνεται η παραδοχή ότι το ρευστό είναι ασθενώς συμπιεστό. Η μέθοδος SPH αντιμετωπίζει προβλήματα στην περιγραφή των ορίων και, λόγω της φύσης της μεθόδου, προκαλούνται διαταραχές στο πεδίο της πίεσης όταν προσομοιώνονται ασθενώς συμπιεστές ροές. Διάφορες διορθώσεις δοκιμάστηκαν, όπως φίλτρα πυκνότητας, όροι διάχυσης κλπ. προκειμένου να αντιμετωπίσουν το θέμα των διαταραχών της πίεσης. Η προσομοίωση ροών με συνεκτικότητα είναι δυνατή, αλλά σε ροές υψηλών Reynolds ενδεχομένως να προκύψουν αφύσικες λύσεις, λόγω της ανομοιόμορφης κατανομής στις θέσεις των σωματιδίων. Τεχνικές όπως η ανακατανομή των σωματιδίων είναι δυνατό να αντιμετωπίσουν την ανομοιογενή κατανομή σωματιδίων και σε συνδυασμό με κλασικά μοντέλα τύρβης, μπορεί να γίνει προσομοίωση ροών υψηλού αριθμού Reynolds.

Ροές ελεύθερης επιφάνειας μελετήθηκαν, συμπεριλαμβανομένων απλούστερων δοκιμών όπως η πρόσκρουση δέσμης σε επίπεδη πλάκα για διάφορες γωνίες πρόσκρουσης, αλλά και πιο πολύπλοκων πρακτικών ροών, εστιάζοντας σε ροές σε υδροστροβίλους δράσης. Προσομοιώθηκε η ροή σε ακροφύσιο υδροστροβίλου Pelton, δίνοντας ικανοποιητικά αποτελέσματα σε σχέση με τα πειραματικά αλλά και αριθμητικά δεδομένα. Ακόμα η μέθοδος χρησιμοποιήθηκε για την προσομοίωση του εκτροπέα δέσμης ενός υδροστροβίλου δράσης, με σκοπό τη διερεύνηση του κατά πόσο είναι δυνατή η ρύθμιση του φορτίου του υδροστροβίλου. Τέλος, η μέθοδος χρησιμοποιήθηκε και για την προσομοίωση της ροής σε δρομείς υδροστροβίλων δράσης αλλά και για τη βελτιστοποίηση του δρομέα υδροστροβίλου Turgo σε συνδυασμό με ένα απλούστερο αλγόριθμο προσομοίωσης ροής.

Αφού εντοπίστηκαν οι κύριες αδυναμίες της μεθόδου, δοκιμάστηκαν ορισμένες τροποποιήσεις. Ο συνδυασμός της μεθόδου SPH με επιλύτες Riemann εξετάστηκε, προκειμένου να αντιμετωπιστούν οι διαταραχές της πίεσης. Επιπλέον τροποποιήσεις, όπως η διακριτοποίηση των εξισώσεων Euler με ALE περιγραφή και προσεγγίσεις SPH, καθώς και η αντιμετώπιση των οριακών συνθηκών με την επίλυση του μερικού προβλήματος Riemann δοκιμάστηκαν σε βασικές αλλά και πρακτικές εφαρμογές. Επίσης αναπτύχθηκαν τροποποιήσεις υψηλής ακρίβειας βασισμένες στο αριθμητικό σχήμα MUSCL-Hancock. Τα αποτελέσματα των τροποποιήσεων συγκρίθηκαν σε σχέση με την κλασική μέθοδο SPH, όπου αποδεικνύονται τα πλεονεκτήματα της χρήσης επιλυτών Riemann καθώς είναι δυνατή η αποτελεσματική αντιμετώπιση των ορίων, ενώ βελτιώνεται σημαντικά το πεδίο της πίεσης. Σαν προτάσεις περαιτέρω έρευνας προτείνονται η μελέτη της ακρίβειας των προσεγγίσεων SPH, οι δυνατότητες της περιγραφής κατά ALE και ο αποτελεσματικός παραλληλισμός της μεθόδου.

Acknowledgements

First of all I would like to thank professor D. Papantonis, for the trust shown in me, his support and guidance through the whole work. Also I would like to thank professor J. Anagnostopoulos for his assistance in various numerical aspects I had to deal with, in the course of the study; without his contribution it would be impossible to finish the present work. A particular thank to professor K. Giannakoglou for his useful instructions and his kind comments during the writing of the present thesis.

Moreover I would like to thank professor S.Voutsinas for the fruitful discussions we had regarding particle methods and professor G.Tzabiras for his kind comments and guidance at paper submission.

I would like also to thank my colleagues and staff at the laboratory of hydraulic turbomachines, especially Fotis Stamatelos, Thanasis Nesiadis, Giannis Kassanos, Marios Xrysovergis, Alexandros Panagiotopoulos and Thanasis Michalakopoulos, for our smooth collaboration.

Last but not least, I have to thank my family and my friends for their support and especially my father who inspired me the interest on engineering.

Contents

Chapter 1

Introduction

Since the beginning of the human civilization, mankind's effort was to harness its environment in order to improve the quality of human life. However, in order to do that, it is necessary to understand the underlying behavior of the desired phenomena. One way for doing so is through extensive experimental investigation. Alternatively, it is possible to describe the basic principles of these phenomena in the form of mathematical models, including algebraic, differential or integral equations. By solving the corresponding mathematical models it is possible to obtain valuable results leading to a further understanding of the principles behind these phenomena.

Practically, experiments are necessary as a first step in order to understand and define the mathematical model. However the resulting mathematical model is often rather complicated to be solved analytically. For this reason, in the past, studies were mainly conducted either through experiments or by using approximate theories, based on the mathematical models with considerable simplifications; without those simplifications the mathematical model would be impossible to solve. With the recent increase in the computational power of modern computers it was possible to approximate the mathematical model using numerical approaches. Eventually the concept of numerical simulation was born; the physical problem is described mathematically using a set of equations and, then, solved numerically using approximate numerical methods. Rather than adopting the traditional theoretical practice of constructing layers of assumptions and approximations, the numerical simulation approach directly attacks the original problems, without making too many assumptions, with the help of modern computers. Moreover, it enables the accurate description of complicated problems without having to conduct series of expensive or even dangerous experiments in laboratories or on site. Thus, numerical simulation became an alternative tool for scientific investigation along with experiments and theoretical tools.

1.1. Generally about Computation Fluid Dynamics

In fluid mechanics, the numerical simulation is performed using as governing equations, relations which are established from conservation laws; such laws state that certain quantities, such as mass, momentum and energy must be conserved during the evolution of the described system. These governing equations are expressed as a system of non-linear, partial differential equations, which, through a number of numerical methods, are eventually transformed into a system of simple algebraic equations. The system of algebraic equations is then possible to be solved, in order to obtain a solution. The subject of Computational Fluid Dynamics (CFD) is the development of numerical

methods which enable the solution of the mathematical model representing fluid flows, through numerical operations involving:

- Governing equations
- Boundary and initial conditions
- Domain discretization
- Numerical discretization
- Resulting algebraic equations

As computer computational power increases, more and more refined and advanced methods and computational models are developed in order to describe and solve complicated flow fields and flow patterns. Today CFD has become an integral part in the design process of products, devices, machines or vehicles where fluid flow phenomena play an important factor on the performance. Optimization studies are commonplace in industries, where thousands of virtual designs are evaluated using CFD in order to improve desired characteristics.

1.2. Particle meshfree methods in comparison with Eulerian mesh based methods

The fluid dynamics governing equations may be described using two different approaches; using a Eulerian or Lagrangian frame. The Eulerian description is a spatial description where the computational elements remain motionless throughout a simulation. On the other hand, the Lagrangian description is the material description, where the computational elements move following the flow. The two descriptions lead to totally different sets of PDEs, since in Lagrangian description the convective derivative is included into the total time derivative.

Another way to categorize fluid dynamics methods is depending on whether a computational grid is required or not. The computational grid, also called mesh, is defined as any of the open spaces or interstices between the strands of a net that is formed by connecting computational nodes in a predefined manner [1]. Computational nodes are the locations where field variables are evaluated and their relations are defined by some kind of nodal connectivity. Moreover, accuracy of the numerical approximation is directly related to the mesh size and mesh patterns. Methods based on a computational mesh are called as mesh-based. On the other hand, meshfree methods establish a system of algebraic equations for the whole problem domain without the use of a predefined mesh. Meshfree methods use a set of nodes, scattered within the problem domain as well as on the boundaries of the domain to represent the problem domain and its boundaries. These sets of scattered nodes are called field nodes, and they do not form a mesh; no information is required on the relationship between the nodes for the interpolation or approximation of the unknown functions of field variables [2].

Considering the previous categories, there are four distinct combinations which may be used to derive a numerical method; Eulerian mesh based, Eulerian meshfree, Lagrangian mesh based and Lagrangian meshfree. Eulerian mesh based methods have been widely used in Computational Fluid Dynamics; such methods are the Finite Difference Method (FDM), the Finite Element Method (FEM) and the Finite Volume Method (FVM). The FEM may be also used under Lagrangian description; in this form it is used in Computational Solid Mechanics (CSM).

Lagrangian methods have some advantages in respect to Eulerian methods [3]. Since there is no need to model the convective term, the resulting algorithm is simpler. Moreover, since the computational elements follow the moving material, it is easier to obtain the time history of the field

variables at a material point. Free surfaces and material interfaces are easily handled since computational elements position themselves naturally across these features. Due to the previous property, only the described body is included in the problem domain, since computational elements will move automatically tracking its position in space.

On the other hand, Lagrangian grid based methods suffer in cases where there is a large deformation, since the mesh becomes greatly distorted. This is the main reason why Lagrangian mesh based methods are used primarily for solids and not for fluids. A possible option to enhance the capabilities of a Lagrangian method is by using rezoning [3], where the problem domain is periodically remeshed controlling the mesh quality. However, this procedure is rather tedious and time consuming.

Another possible approach is to adopt a meshfree approach under Lagrangian framework. In this way large deformations, wave breaking, splashes and droplets are easily described [4], without using treatments to handle the mesh distortion. Moreover, a meshfree method tackles the non-trivial part of building a suitable computational mesh.

Meshfree particle methods (MPMs) are a special class of meshfree methods, where a finite set of particles is used to describe the behavior of the simulated medium and record the movement of the system. Each particle may be associated to a distinct physical object or represent a part of the continuum domain. In CFD particle methods, each particle carries a set of field variables such as mass, velocity, density and occupies space. A non-exhaustive list of such methods may be found in Liu's book [3].

1.3. CFD in hydraulic turbines

The precursor of modern turbines was the water wheel which was extensively used in the past thousands years for providing power to mills, manufacturing plants or industries. Its main shortcoming was its size, which limited the water flow rate and water head to be harnessed. Hydraulic turbines have been developed since the 19^{th} century and were primarily used for electrical power generation. The basic underlying principle is that water is properly directed on the turbine blades, exerting force. Since the blades are rotating, useful work is extracted from the fluid and it is transferred through the turbine shaft to the generator attached to the turbine.

Generally, turbines may be divided into two main categories, reaction and impulse turbines. In reaction turbines pressures before and after the turbine runner are different, thus the runner casing should be sealed. Such turbines are Francis, Kaplan and axial flow turbines. On the other hand, in impulse turbines, all pressure drop occurs at the turbine nozzles which direct a high water velocity jet on the runner. Such turbines are the Pelton and Turgo turbines. This enables operation with air presence, in atmospheric environment.

CFD was used in reaction turbines components (rotors, draft tubes) since 1978 [5]; initially simplified 2d and then 3d geometries were simulated under the assumption of potential flow. After 1987, the increase in computer capabilities and the development of more efficient and sophisticated numerical methods enabled the solution of 3D Euler equations in Francis turbine runners [5]. Euler equations enabled a better description of the flow field inside the turbine, by properly reproducing vortices and being in agreement with LDA measurements. During the period between 1990 and 2000, the steady state Reynolds Averaged Navier Stokes equations were mainly used for turbine simulations. Since the RANS equations enabled the modeling of viscosity and turbulent effects, it was possible to study turbine losses, boundary layer effects and separation in the turbines. Moreover,

methods for coupling the flow in stationary and rotating components were used, including mixing plane models, frozen rotor or sliding mesh approaches.

On the other hand, simulation of impulse turbines (and particularly Pelton turbines) was possible only after 2000. This is explained by the complexity of the flow features developing during an impulse turbine operation. Indeed, a reaction turbine is fully enclosed and immersed in a single fluid and its operation could be approximated using a moving reference frame. However, flow in an impulse turbine is rather complicated, since there are two immiscible fluid phases (air and water) with a distinct interface between them. Furthermore the flow is unsteady and involves rotating components. Eventually, after implementing special models, such as the Volume of Fluid method for tracking the phases and their interface in combination with mesh refinement near the free surface, simulation of a Pelton turbine was possible, using traditional Eulerian mesh based methods.

S. Kvicinsky [6] worked on the simulation of free surface flows, involving water jets, with the commercial CFD packages CFX and FIDAP. The first simulations involved simple water jet impingements on flat plates under different angles. After benchmarking the capabilities of the two CFD programs, Kvicinsky eventually used CFX for simulating the flow in a Pelton bucket, replicating the Pelton operation at various bucket angular positions.

H. Matthias et al. [7] performed simulations of the water jet impingement on a flat plate too. However they have used the commercial CFD package Fluent for the solution of the flow field and the Geo-Reconstruct Volume of Fluid scheme for tracking the two phase. Apart from the impingement simulations they are the first to carry out a simplified analysis of a rotating impulse turbine runner, which consisted of flat blades.

Zoppe et al. [8] have used the commercial CFD package Fluent, solving the Reynolds averaged Navier Stokes in combination with the Volume of Fluid PLIC method, to predict the flow patterns in a stationary Pelton bucket impinged by a water jet. The simulations involved various jet diameters and different water impingement angles. The simulation results have been compared with results from their experimental installation showing good agreement.

Perig et al. [9] have used CFX for the simulation of ¼ of a rotating Pelton runner. Their results where compared to experimental results, provided by a scale Pelton model equipped with an onboard acquisition system for measuring the unsteady pressure distribution on the inner surface of the bucket. They also compared the evolution of the free-surface on the bucket between experimental observations and the numerical simulation. The comparison showed that despite some inaccuracies in the free surface evolution, the overall results were accurate enough.

Mack et al. [10] used Eulerian CFD methods for simulating the flow at the distributor of a multijet Pelton turbine. Also they used the volume of fluid model to study the effects of secondary flows, due to pipe bends, on the quality of the jet formed at a Pelton turbine nozzle. Their simulations showed that, whereas the volume of fluid model is able to predict the macroscopic deformation of the water jet, it did not predict droplets and the expected wavy free surface; a much finer computational mesh, than that they have used [10], was required to properly replicate such effects. Furthermore, they performed simulations of a multi-jet Pelton turbine to determine the efficiency drop due to multi-jet operation.

However, despite the ability of simulating Pelton geometries, using appropriate Eulerian techniques, considerable computational resources were required, even for simple flows [8]. Thus, even if the mesh based Eulerian CFD approach was available, several researchers applied semi-analytical methods in order to simplify the flow and estimate approximately Pelton turbine efficiency.

Atthanayake [11] used velocity triangles and the boundary layer equations, assuming that the velocity profile on the Pelton bucket is similar to that of the flow around a cylinder. The analytical model provides a formula with which it is possible to determine the losses due to friction on the surface of a Pelton turbine bucket.

On the other hand, Zhang [12] has developed a method based on the velocity triangles and the forces acting on a fluid element to calculate the flow velocities along the path of the flow, assuming frictionless flow. Also Zhang quantified the influence of Coriolis, impact and centrifugal forces in the energy exchange between the water and the bucket.

Generally, analytical models were able to produce useful results regarding the operation of a Pelton turbine, but they involved several significant simplifications, such as the flow on the Pelton bucket being 2-dimensional [12]. Other researchers, such as Fengqin et al [13], developed a simplified cartoon frame method in order to predict the dynamic performance of a Pelton runner.

Over time it became apparent that a Lagrangian particle method might be preferable for the simulation of the flow in such turbines. Lagrangian methods do not require interface tracking and are able to handle easily rotating components.

J. Anagnostopoulos et al. have performed simulations and optimization of impulse turbines, including Pelton [14, 15] and Turgo [16] impulse turbines, using a simplified Lagrangian model, the FLS algorithm. The FLS algorithm does not solve the Navier – Stokes equations, but rather assumes the water jet as frames of fluid parcels which move in a rotating reference frame, under the influence of the centrifugal and Coriolis forces. However the algorithm involves adjustment of several factors, which affect the jet spreading or losses due to friction, impact etc.

Other researchers have employed the Moving Particle Semi-Implicit method (MPS) [17] to simulate the flow in a Pelton turbine. The MPS method enables the discretization and solution of the N-S equations in a Lagrangian meshless framework. However, apart from a single paper, there are not any other references mentioning the implementation of the MPS method for simulations in impulse turbines.

A more promising meshfree particle method is the Smoothed Particle Hydrodynamics (hereinafter SPH) method and its variants. The SPH method has been used for the simulation of various hydrodynamics problems, since 1998; e.g. Stamatelos et al. [18] have implemented a simple SPH solver for basic viscous flows. The greatest advantage of the method is the capability of describing large deformations in the free surface and for that reason it has been used extensively in wave breaking [19] and sloshing studies [20]. Marongiu et al. [21, 22, 23] have implemented a variant of the SPH method for the simulation of a Pelton turbine.

1.4. Aim of the present study

In the present work the SPH method, and its more recent variants, are examined. The SPH method was chosen since it is one of the earliest of the meshfree particle methods, thus there is a considerable theoretical background, research and experience behind the method. On the other hand, there is potential for further improvement and research interest on the SPH method, since there are several issues which need to be addressed, such the boundary condition implementation. Another reason why this particular method was chosen is the proven capabilities of the method in simulating free surface flows, since the present study will focus on such cases.

The aim of the present work is the development of algorithms based on the SPH method, along with its variants, to explore its capabilities and to assess its performance. The developed algorithms are used to simulate a wide range of problems including 1D, 2D and 3D problems involving free-surfaces and viscous flows. The main focus is the implementation of the SPH method on impulse turbines and especially in Turgo turbine design.

A novel element of the present work is the numerical simulation of a Turgo impulse turbine with the SPH method. Indeed, as it was discussed in section 1.3, it was not until recently that impulse were able to be modeled using CFD techniques and only one research group has implemented the SPH method for simulation of Pelton impulse turbines [21, 22, 23].

Another novel feature is the design of a Turgo turbine using the SPH method. Very few scientific articles can be found in the literature covering the design of such turbines and only few companies construct this turbine type worldwide. The developed SPH method is used for design optimization, in combination with a simpler flow solver. After the optimization, a Turgo turbine model, based on the optimal blade geometry, was built.

Furthermore, modern SPH implementations and techniques, involving the implementation of Riemann solvers, are discussed and a novel 2^{nd} order variant, which proves capable of preventing oscillations, while also reducing the inherent viscosity of the Godunov scheme, is proposed.

1.5. The structure of the present thesis

The thesis is divided into several parts, with each part discussing an SPH-based method, benchmark tests or applications. Chapter 2 describes the basic characteristics and approximations of all SPH-based methods presented in the present work.

Chapter 3 explains the derivation of the SPH approximations and the corresponding set of Navier-Stokes equations using the SPH method expressions, along with implementation details, such as density filters or kernel gradient corrections.

Chapter 4 presents several basic tests of the SPH method. Each test aims to test characteristic properties of the resulting scheme, such as reproduction of the wave structure of Euler equations, conservation of angular momentum and incompressibility. Moreover tests involving viscosity effects are presented; several academic test cases are examined involving laminar flow, such as the Couette or Poisseuille flow. Also more complicated cases are examined, involving back-step flow at laminar and turbulent regime. Turbulence models and advanced SPH treatments, such as particle redistribution, are also discussed. A simple free-surface flow, involving the impingement of a water jet on a flat plate, is simulated using the SPH method. The results are compared with experimental results from literature.

Chapter 5 focuses on the practical applications of the SPH method, involving the study of various components of impulse turbines, such as the nozzle including the spear valve or the jet deflector. Moreover, the interaction of impulse turbine runner with the water jet is examined for both Turgo and Pelton impulse turbines. An optimization procedure using the SPH method and the FLS algorithm is also presented.

In chapter 6, the implementation of Riemann solvers in combination with the SPH method is presented. The equations of the resulting scheme are explained, along with a novel 2^{nd} order scheme. Various test cases are examined and results of the high order scheme in practical applications are discussed. Moreover, an advanced treatment for enforcing boundary conditions is presented and a new hybrid SPH-based method is analyzed.

In chapter 7, another SPH modification is proposed, employing a totally different perspective from the former methods. Extension to 2^{nd} order accuracy is presented along with advanced numerical techniques. The results of test cases and applications are discussed in comparison with the other methods.

Chapter 8 is a conclusion, summing up all the notable points from the presented methods and presenting various important key areas, proposed for further research.

Appendix A discusses the implementation of various Riemann solvers, approximate and exact, which are used in chapter 4, 8 and 9.

Appendix B is a short description of the structure of the developed algorithms for the present study, along with of block diagrams outlining and implementation details.

Appendix C covers the procedure which was employed for the algorithm parallelization of the presented methods.

During the development of the SPH methods, validation was conducted using theoretical solutions and numerical or experimental results. Analytical solutions are provided for several simple problems, such as the 2d jet impingement (see chapter 3) or the Green-Taylor flow (see chapter 7). Numerical reference solutions were obtained from:

- Literature
- Developed FV algorithms (see appendix A)
- The Fluent® commercial CFD program, based on the Finite Volume method. This program is widely used by researchers around the world for a wide range of problems, including free surface/multiphase flows (see also section 1.3 of the current chapter). Moreover, it is available to the students of NTUA, through Central CloudFront server (cloudfront.central.ntua.gr). For the above reasons it was used to provide reference solutions for various problems. In the present thesis, results produced by the Fluent® software, will be indicated by simply referring to them as *Fluent solution*. Appendix D provides more information about the Fluent software.

Experimental results were obtained from literature sources or the experimental installations of the Laboratory of Hydraulic Turbomachines.

Appendix E has details on the construction of a Turgo turbine runner, based on the optimized blade geometry, produced from the optimization procedure involving the SPH method.

References are located at the end of each chapter.

References

[1] G.R. Liu, “Meshfree Methods: Moving Beyond the Finite Element Method”, CRC Press Second Edition 2009, ISBN-10: 1420082094, ISBN-13: 978-1420082098

[2] G.R. Liu, Y.T. Gu, “An Introduction to Meshfree Methods and Their Programming”, Springer 2005, ISBN-10: 9048168198, ISBN-13: 978-9048168194

[3] G.R. Liu, M. B. Liu, “Smoothed particle hydrodynamics”, World scientific publishing, 2003, ISBN 981-238-456-1

[4] S. Li, W. Liu, “Meshfree Particle Methods”, Springer 2007, ISBN-10: 3540222561, ISBN-13: 978-3540222569

[5] H. Keck, M. Sick, “Thirty years of numerical flow simulation in hydraulic turbomachines”, Acta Mechanica, vol. 201, p. 211-229 (2008), doi: 10.1007/s00707-008-0060-4.

[6] S. Kvicinsky, “Method d’analyse des ecoulements 3D a surface libre: application aux turbines Pelton”, These N^{o}2526, Ecole Polytechnique Federale de Lausanne, 2002.

[7] H.B. Matthias, O. Promper, “Numerical simulation of the free surface flow in Pelton turbines”, Proceedings of the 6th International Conference on Hydraulic Machinery and Hydrodynamics, Timisoara, Romania, October 21-22, 2004.

[8] B. Zoppe, C. Pellone, T. Maitre, P. Leroy, “Flow analysis inside a Pelton turbine bucket”, ASME Journal of Turbomachinery, vol. 128, pages 500-512, 2006.

[9] A. Perrig, F. Avellan, J.-L. Kueny, M. Farhat, E. Parkinson, "Flow in a Pelton turbine bucket: numerical and experimental investigations", Journal of Fluids Engineering, 2005.

[10] R. Mack, W. Rohne, S. Riemann, W. Knapp, R. Schilling, "Using the potential of CFD for Pelton turbine development", 23rd IAHR Symposium Yokohama, October 2006, Japan

[11] I. U. Atthanayake, "Analytical study on flow through a Pelton turbine bucket using boundary layer theory", International Journal of Engineering & Technology, vol. 9, no. 9, p241-245.

[12] Z. Zhang, "Flow dynamics of the free surface flow in the rotating buckets of a Pelton turbine", Proceedings of the institution of Mechanical Engineers, vol. 223, Part A: Journal of Power and Energy

[13] H. Fengqin, X. Yexiang, Z. Jinglin, Z. Ailing, K. Takshi, "Numerical prediction of dynamic performance of Pelton turbine", 23rd IAHR Symposium Yokohama, October 2006, Japan.

[14] J. Anagnostopoulos, D. Papantonis, "Experimental and numerical studies on runner design of Pelton turbines", Hidroenergia 2006, Crieff, Scotland, 7-9 June 2006, United Kingdom.

[15] J. Anagnostopoulos, D. Papantonis, "A numerical methodology for design optimization of Pelton turbine runners", Hydro 2006, Porto Carras, 25-27 September 2006, Greece.

[16] J. Anagnostopoulos, D. Papantonis, "Flow modeling and runner design optimization in Turgo water turbines", International Journal of Applied Science, Engineering and Technology, vol. 4, no. 3, 2007, ISSN: 1307-4318.

[17] Y. Nakanishi, T. Fuji, M. Morinaka and K. Wachi, "Numerical Simulation of the flow in a Pelton bucket by a particle method", 23rd IAHR Symposium Yokohama, October 2006, Japan.

[18] F. Stamatelos, J. Anagnostopoulos, "Sinulation of viscous flows with a gridless particle method", WSEAS Tansactions on Fluid Mechanics, Vol. 3, issue 4, October 2008, ISSN: 1790-5087.

[19] S. Shao, "Incompressible SPH simulation of wave breaking and overtopping with turbulence modelling", International Journal for Numerical Methods in Fluids, vol. 50, p. 597- 621, 2006.

[20] R. Banim, "Some industrial SPH applications undertaken at the BAE systems advanced technology centre", Proceedings of the 2nd Spheric Workshop, Madrid, Spain, 25-25 May 2007.

[21] J.C. Marongiu, F. Leboeuf, E. Parkinson, "Numerical simulation of the flow in a Pelton turbine using the meshless method smoothed particle hydrodynamics: a new simple boundary treatment", Proceedings of the Institution of Mechanical Engineers, vol. 221, Part A: Journal of Power and Energy, 2007.

[22] J. C. Marongiu, E. Parkinson, S. Lais, F. Leboeuf, J. Leduc, "Application of SPH-ALE method to Pelton hydraulic turbines", 5th Spheric Workshop, Manchester, 23-25, June 2010, United Kingdom.

[23] J.C. Marongiu, F. Leboeuf, E. Parkinson, "Riemann solvers and efficient boundary treatments: an hybrid SPH-finite volume numerical method", 3rd Spheric Workshop, Lausanne, 4-6 June 2008, Switzerland.

Chapter 2

SPH method basics

Smoothed Particle Hydrodynamics (SPH) method was invented by Lucy, Gingold and Monaghan in 1977. Initially it was used for the simulation of astrophysical problems in the open three dimensional space. However the potential of the method was recognized, thus it was extended to cover the behavior of solids and fluids. In this chapter, the basic concepts of the SPH method will be introduced. Moreover, the derivation of the SPH approximations will be discussed, along with error estimation of the approximations.

2.1. Basic characteristics of the SPH method

The SPH method was developed for the numerical solution of hydrodynamic problems which can be expressed in the form of *partial differential equations* (*PDEs*) of *field variables* such as velocity density, temperature, etc. The field variables depend on the nature of the described medium and the underlying phenomena. Generally analytical solutions for such problems are not available, apart from several simple cases; thus the main effort is to obtain numerical solutions. In doing so, it is required to discretize the problem domain where the PDEs are defined and then approximate relations between the field variables are needed, for obtaining the PDE operators. These relations result to a set of discretized equations which is possible to be solved in respect to time, using an integration scheme.

The SPH method has the following characteristics [1]:

1. The problem domain is represented using a set of arbitrarily distributed calculation elements, called as *particles*. There is no connectivity between these particles, i.e. particles lack any explicit topology relating them.
2. The PDEs are expressed in *integral form* and the *integral representation* method is used for function approximation (*weak form formulation*). In the SPH method it is called *kernel approximation*. In this way, the SPH method is stabilized, since the integral form has a smoothing effect.
3. The kernel approximation is further approximated using discrete elements, the particles. This is termed in SPH as *particle approximation*. It is done by replacing the integration in the integral representation, using finite summations over the corresponding values of all neighboring particles in a local domain, called the *support domain*. Thus the support domain of each particle is *finite* (or *compact*).
4. The particle approximation is performed every time step and, as a consequence, depends on the local distribution of particles. The SPH method *adapts* to the simulated problem, since the local distribution of particles depends on the nature and the features of the problem. As a result, the SPH method is especially suited for problems with large deformations, such as

breaking waves, offering good resolution in areas of high interest. Since there is no need for a computational mesh, the SPH approximations can be used with any particle distribution. However, in order these approximations to be valid the support domain has to be filled with an adequate number of particles.

5. The particle approximations are performed for all field variables and PDE terms/operators, thus the discretized differential equations are expressed in respect to time only (Lagrangian description)
6. The discretized equations are solved using an explicit integration scheme, enabling efficient parallelization. The only requirement is to determine a time step which would ensure the stability of the algorithm.

Summing up all the above, the SPH method is: a meshfree, explicit, adaptive, Lagrangian method.

2.2. SPH function representation – integral and particle representation

The formulation of the SPH method is divided in two steps: the first step involves the *integral representation* of field functions and variables (or *kernel approximation*) and the second step is the *particle representation*. The first step is the expression of the kernel approximation using the integral of a field function with the *smoothing function* (or *kernel function* – generally this function is represented using *W*). Then this integral can be approximated with the finite sum of all neighboring particles inside the support domain of the kernel function.

Integral representation of a function

The integral representation of a function $f(\mathbf{r})$, using the SPH methodology, is based on the following identity [1]:

$$f(\mathbf{r}) = \int_{\Omega} f(\mathbf{r}')\delta(\mathbf{r}-\mathbf{r}')\,dr' \tag{2.1}$$

In the above relation $f(\mathbf{r})$ is a function of a 3D position vector $\mathbf{r}$ and δ is the δ-Dirac function, defined as:

$$\delta(\mathbf{r}-\mathbf{r}') = \begin{cases} 1 & \mathbf{r} = \mathbf{r}' \\ 0 & \mathbf{r} \neq \mathbf{r}' \end{cases} \tag{2.2}$$

In equation 2.1, Ω is the volume of the integral that contains $\mathbf{r}$. Also eq. 2.1 implies that function f can be expressed in integral form. The identity is valid when the $f(\mathbf{r})$ function is defined and continuous in Ω. If the δ-Dirac function is replaced with a smoothing function $W(\mathbf{r}-\mathbf{r}',h)$, the integral form can be expressed using the following relation:

$$f(\mathbf{r}) \equiv \int_{\Omega} f(\mathbf{r}')W(\mathbf{r}-\mathbf{r}',h)\,dr' \tag{2.3}$$

The $W(\mathbf{r}-\mathbf{r}',h)$ is called as smoothing function or kernel function or kernel or W-function or simply W. The h term is the smoothing length and defines the area of influence of the W kernel function. Since the W-function is not the δ-Dirac function, the previous relation is approximate (from now on the approximate expressions will be denoted using the brackets: $< >$).

The kernel function has to fulfill several requirements in order eq. 2.3 to be valid. The kernel function should [1]:

1. Be normalized in the support domain, i.e.

$$\int_{\Omega} W(\mathbf{r}\text{-}\mathbf{r}', h)\, dr' = 1$$

2. Be compact (local influence), i.e.

$$W(\mathbf{r}-\mathbf{r}', h)=0, \text{ for } \|\mathbf{r}-\mathbf{r}'\| \geq \kappa h$$

3. Be positive for all points inside the support domain
4. Monotonically decreasing with the increase of the distance between two points
5. Tend to the δ-Dirac function, when the smoothing length tends to zero, i.e.

$$\lim_{h\to 0} W(\mathbf{r}-\mathbf{r}', h) = \delta(\mathbf{r}-\mathbf{r}')$$

6. Be even, i.e. $W(\mathbf{r})= W(-\mathbf{r})$
7. Be smooth enough, i.e. have continuous high order derivatives

The kernel functions used by consistency corrected variants of the SPH method may not necessarily satisfy all the above requirements, such as e.g. the normalization condition or the evenness [2]. However in the present work, such SPH variants will not be considered, thus the kernel function will obey to the properties 1-7.

Integral representation of the derivative of a function

The approximation of the space derivative $\nabla f(\mathbf{r})$ is derived by replacing $f(\mathbf{r})$ with $\nabla f(\mathbf{r})$ in equation 2.3:

$$< \nabla f(\mathbf{r}) >= \int_{\Omega} \nabla f(\mathbf{r}') W(\mathbf{r}\text{-}\mathbf{r}', h)\, dr' \tag{2.4}$$

In the previous relation, the space derivative is expressed in respect to the **r'** variable. Since:

$$\nabla f(\mathbf{r}')\, W(\mathbf{r}\text{-}\mathbf{r}', h) = \nabla\left(f(\mathbf{r}') W(\mathbf{r}\text{-}\mathbf{r}', h)\right) - f(\mathbf{r}')\nabla W(\mathbf{r}\text{-}\mathbf{r}', h) \tag{2.5}$$

the result, after replacing in eq. 2.4, is:

$$< \nabla f(\mathbf{r}) >= \int_{\Omega} \nabla\left[f(\mathbf{r}') W(\mathbf{r}\text{-}\mathbf{r}', h)\right] dr' - \int_{\Omega} f(\mathbf{r}')\nabla W(\mathbf{r}\text{-}\mathbf{r}', h)\, dr' \tag{2.6}$$

In the previous equation the first term in the RHS can be written as a surface integral at the boundary *S* of the integration domain *Ω*, using the divergence theorem. Thus:

$$< \nabla f(\mathbf{r}) >= \int_{S} f(\mathbf{r}') W(\mathbf{r}\text{-}\mathbf{r}', h)\, \mathbf{n}\, dS - \int_{\Omega} f(\mathbf{r}')\nabla W(\mathbf{r}\text{-}\mathbf{r}', h)\, dr' \tag{2.7}$$

where **n** is the normal to the infinitesimal surface element *dS*.

Since the kernel function has a compact support, the surface integral is equal to zero, when the boundary of the support domain is within the problem domain. If the support domain extends beyond

the problem domain, then the kernel function is truncated by the boundary and the surface integral cannot be considered zero. In such cases special treatments have to be made, in order to remedy the influence of the boundary effects, if the surface term is assumed zero.

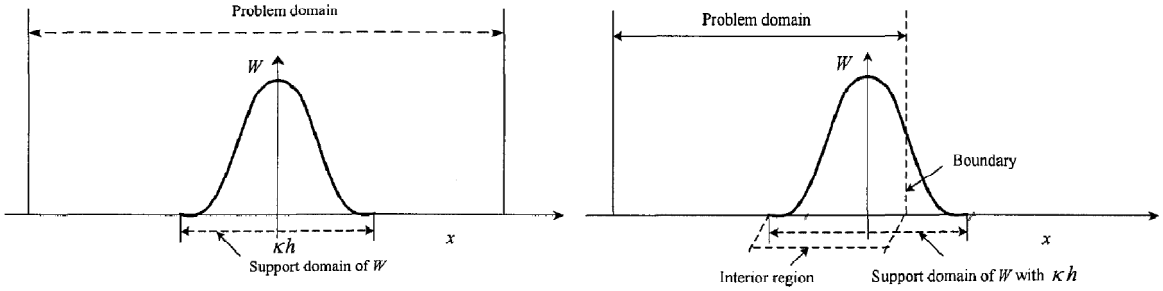


Fig. 2.1. Support domain and problem domain of the kernel function *W*. Left: the support domain lies inside the problem domain, thus the surface integral in eq. 2.7 is equal to zero. Right: truncated support domain near boundary.

Therefore for all points whose support domain is inside the problem domain eq. 2.7 can be written as:

$$< \nabla f(\mathbf{r}) >= -\int_{\Omega} f(\mathbf{r}')\nabla W(\mathbf{r}\text{-}\mathbf{r}', h)\, dr' \tag{2.8}$$

In the previous equation (eq. 2.8) the derivative operator is transmitted from the $f(x)$ function to the smoothing function. Thus the gradient of a function $f(x)$ is calculated using the values of that function and the derivative of the smoothing function W, instead of using the derivatives of the $f(x)$ function. This feature is similar to the weak form methods that reduce the consistency requirement on the assumed field functions and produce stable solutions for PDEs.

Particle representation

In the SPH method the described system is represented using a finite number of particles that occupy space and carry mass and other characteristic properties. The continuous integral representations are expressed as discrete summations for all particles residing inside the support domain of the kernel function.

The infinitesimal volume *dr'* is replaced by the finite volume ΔV_j for particle *j* and is related to the particle mass as follows:

$$m_j = \Delta V_j \rho_j \tag{2.9}$$

where ρ_j is the density of *j* particle. The continuous SPH integral representation is written in the following discrete form [1]:

$$f(\mathbf{r}) = \int_{\Omega} f(\mathbf{r}')W(\mathbf{r}\text{-}\mathbf{r}', h)\, dr' \Rightarrow$$

$$f(\mathbf{r}) \cong \sum_{j=1}^{N} f(\mathbf{r}_j)W(\mathbf{r}\text{-}\mathbf{r}_j, h)\Delta V_j \Rightarrow$$

$$f(\mathbf{r}) \cong \sum_{j=1}^{N} f(\mathbf{r}_j)W(\mathbf{r}\text{-}\mathbf{r}_j, h)\frac{1}{\rho_j}(\rho_j \Delta V_j) \Rightarrow$$

$$f(\mathbf{r}) \cong \sum_{j=1}^{N} \frac{m_j}{\rho_j} f(\mathbf{r}_j) W\left(\mathbf{r}\text{-}\mathbf{r}_j, h\right) \tag{2.10}$$

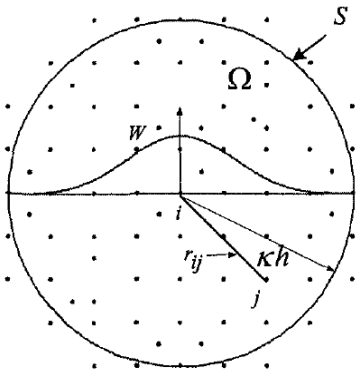


Fig. 2.2. Kernel function support radius (or support domain).

From now on, it will be assumed that $W_{ij} = W\left(\mathbf{r}_i\text{-}\mathbf{r}_j, h\right)$, for simplicity. Thus:

$$< f(\mathbf{r}) >= \sum_{j=1}^{N} \frac{m_j}{\rho_j} f(\mathbf{r}_j) W_{ij} \tag{2.11}$$

$$< \nabla f(\mathbf{r}) >= -\sum_{j=1}^{N} \frac{m_j}{\rho_j} f(\mathbf{r}_j) \nabla W_{ij} \tag{2.12}$$

In the previous relation ∇W_{ij} is expressed in respect to particle *j*. It is possible to rewrite the previous relation, expressing ∇W_{ij} in respect to *i* particle for simplicity, as follows:

$$< \nabla f(\mathbf{r}) >= \sum_{j=1}^{N} \frac{m_j}{\rho_j} f(\mathbf{r}_j) \nabla_i W_{ij} \tag{2.13}$$

$\nabla_i W_{ij}$ is calculated using the chain rule. In 1D:

$$\frac{dW}{dx} = \frac{dW}{dr}\frac{dr}{dx} \tag{2.14}$$

Similarly, in 3D:

$$\nabla_i W_{ij} = \nabla r_{ij} \frac{dW_{ij}}{dr_{ij}} = \left(\frac{dr}{dx}, \frac{dr}{dy}, \frac{dr}{dz}\right) \frac{dW_{ij}}{dr_{ij}} \tag{2.15}$$

where:

$$\frac{dr}{dx} = \frac{d\left(\sqrt{x^2 + y^2 + z^2}\right)}{dx} = \frac{x}{\sqrt{x^2 + y^2 + z^2}}$$

The previous is applied in the same manner for *y* and *z*. Substituting *x* (integral approximation) with $x_{ij}=x_i-x_j$ (particle approximation), leads to:

$$\nabla_i W_{ij} = \left(\frac{x_{ij}}{\|r_{ij}\|}, \frac{y_{ij}}{\|r_{ij}\|}, \frac{z_{ij}}{\|r_{ij}\|} \right) \frac{dW_{ij}}{dr_{ij}} = \left(\frac{x_i - x_j}{\|r_{ij}\|}, \frac{y_i - y_j}{\|r_{ij}\|}, \frac{z_i - z_j}{\|r_{ij}\|} \right) \frac{dW_{ij}}{dr_{ij}} \tag{2.16}$$

Here $\nabla_i W_{ij}$ is expressed in respect to *i* particle and, because of that, the negative sign has been removed. It has to be mentioned that $\|\mathbf{r}_{ij}\|$ is the distance between the *i*, *j* particles, defined as:

$$\|\mathbf{r}_{ij}\| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$$

It is possible, using the following identities:

$$\nabla f(\mathbf{r}) = \frac{1}{\rho}\left[\nabla(\rho f(\mathbf{r})) - f(\mathbf{r})\nabla\rho\right] \text{ and}$$

$$\nabla f(\mathbf{r}) = \rho\left[\nabla\left(\frac{f(\mathbf{r})}{\rho}\right) + \frac{f(\mathbf{r})}{\rho^2}\nabla\rho\right]$$

to derive alternative formulations for the derivative, such as the following:

$$< \nabla f(\mathbf{r}) >= \frac{1}{\rho_i}\sum_{j=1}^{N} m_j \left[f(\mathbf{r}_j) - f(\mathbf{r}_i)\right]\nabla_i W_{ij} \tag{2.17}$$

$$< \nabla f(\mathbf{r}) >= \rho_i \sum_{j=1}^{N} m_j \left[\left(\frac{f(\mathbf{r}_j)}{\rho_j^{\,2}}\right) + \left(\frac{f(\mathbf{r}_i)}{\rho_i^{\,2}}\right)\right]\nabla_i W_{ij} \tag{2.18}$$

The latter approximations, and especially eq. 2.17, tend to perform better than eq. 2.12 in the gradient calculation, according to Colin et al. [3].

Similar formulations may be derived for the second derivative (or laplacian, denoted with Δ) of a function [4]:

$$< \Delta f(\mathbf{r}) >=< \nabla^2 f(\mathbf{r}) >= 2\sum_j \frac{m_j}{\rho_j}\left[f(\mathbf{r}_i) - f(\mathbf{r}_j)\right]\frac{\mathbf{r}_{ij}\cdot\nabla_i W_{ij}}{\|\mathbf{r}_{ij}\|^2} \tag{2.19}$$

For two arbitrary functions, the following rules may be applied, in conjunction with the SPH approximations:

$$< f_1 + f_2 >=< f_1 > + < f_2 >$$

$$< f_1 f_2 >=< f_1 >< f_2 >$$

$$< c f_1 >= c < f_1 >, \text{ for a constant } c$$

2.3. Error estimation in the SPH approximations

By observing the procedure of the SPH approximations derivation, one may notice that there are two distinct approximations. The first approximation is made in eq. 2.3, when the δ-Dirac function is replaced by the smoothing function and is called *integral approximation*. The second approximation is the calculation of the integral using finite summations of particle contributions (eq. 2.11-2.12) and

is called *particle approximation*. Errors of the SPH expressions are attributed to the contribution of both aforementioned expressions.

Integral approximation

The integral approximation has, at least, second order of accuracy (h^2) [1, 2, 5]. This can be proved by considering the error of a Taylor expansion of the $f(x')$ function around x:

$$< f(x) >= \int_{\Omega} \left[f(x) + f'(x)(x'-x) + + f''(x)(x'-x)^2 + ... \right] W(x-x',h)dx' \Rightarrow$$

$$< f(x) >= f(x)\int_{\Omega} W(x-x',h)dx' + f'(x)\int_{\Omega} (x'-x)W(x-x',h)dx' + O(h^2) \tag{2.20}$$

In the previous relation the $O(h^n)$ term denotes the error (and n the rank of the error). Since the W-function is even in respect to x, then the function $(x- x')W(x - x',h)$ is odd, thus its integral in the support domain is zero:

$$f'(x)\int_{\Omega} (x'-x)W(x-x',h)dx' = 0$$

Eventually:

$$< f(x) >= f(x) + O(h^2) \tag{2.21}$$

The residual error term may be expressed in the following way:

$$O(h^2) = \frac{\sigma h^2}{2} f''(x) + O(h^4) \tag{2.22}$$

The σ is a constant which depends on the kernel function used. It is possible to construct a kernel function, for which the σ constant is zero. In that case the error of the integral approximation would be of fourth order, since the $O(h^3)$ term is zero, due to the fact that the $(x-x')^3 W(x-x',h)$ function is odd. Generally, in order to obtain n-th order approximations for the value of a function, it is proved [1] that the following conditions have to be satisfied:

$$\begin{cases} A_o = \int_{\Omega} W(x-x',h)dx' = 1 \\ A_1 = -h\int_{\Omega} \left(\frac{x-x'}{h} \right) W(x-x',h)dx' = 0 \\ A_2 = \frac{h^2}{2}\int_{\Omega} \left(\frac{x-x'}{h} \right)^2 W(x-x',h)dx' = 0 \\ \cdots \\ A_n = \frac{(-1)^n h^n}{n!}\int_{\Omega} \left(\frac{x-x'}{h} \right)^n W(x-x',h)dx' = 0 \end{cases} \tag{2.23}$$

In the previous relations:

- A_o is equal to unity, due to the kernel normalization condition.

- A_1 is equal to zero, since the function inside the integral is odd.
- A_2 is equal to $\frac{\sigma h^2}{2}$, which, for specially designed kernel functions, may be zero

Similar conditions can be expressed for the *n*-th order integral approximation of the first and second derivative of a function. To summarize; if a smoothing function satisfies the following conditions:

$$\left\{\begin{array}{c} M_o = \int_\Omega W(x\text{-}x',h)dx' = 1 \\ M_1 = \int_\Omega (x-x')W(x\text{-}x',h)dx' = 0 \\ M_2 = \int_\Omega (x-x')^2 W(x\text{-}x',h)dx' = 0 \\ \cdots \\ M_n = \int_\Omega (x-x')^n W(x\text{-}x',h)dx' = 0 \end{array}\right\} \tag{2.24}$$

and

$$\left\{\begin{array}{l} W(x\text{-}x',h)\big|_S = 0 \\ W'(x\text{-}x',h)\big|_S = 0 \\ \cdots \\ W^{(k-1)}(x\text{-}x',h)\big|_S = 0 \end{array}\right\} \tag{2.25}$$

then it is able to reproduce functions and the *k*-th order derivative of a function with *n*-th order of accuracy [1]. The first requirement has to do with the accurate reproduction of polynomial functions. The second requirement defines the surface values of the smoothing function and its derivatives at the boundary of the support domain. It is highlighted that the second condition is a requirement of the compact support of the smoothing function.

Using the above conditions, it is possible to construct high order smoothing functions. Monaghan [5] has developed a technique for the calculation of high order smoothing functions. However, such smoothing functions may become negative and as a result may produce unphysical solutions, such as negative density (and mass) or energy. Actually G.R. Liu [1] proved that it is impossible to construct high order functions, which are also positive in the whole support domain. This can be understood if one considers the relations in eq. 2.24; for *n* being even (n=2, 4, 6,…), then $(x-x')^n \geq 0$, thus in order the integral to be zero, the smoothing function ***has*** to be negative at a region of the support domain.

All the above are valid when the integral relations are applied on the complete support domain, in the opposite case the error would be larger.

Particle approximation

Generally the calculation of the particle approximation errors is not an easy task, since the particle positions change over time, due to their motion (Lagrangian representation). Particle approximation

errors depend mainly on the distribution of particles inside the support domain; errors are greatly affected by the uniformity or irregularity of particle distribution and the distribution depends on the flow patterns and the dynamics of the simulated phenomena. A first approach of estimating the errors would be by placing particles on a regular Cartesian grid and moving them by a small, relatively to the grid spacing, random perturbation. However, errors calculated in this way were found to be overestimated [5]. The main reason why this happens is because, in practice, errors are much smaller than the aforementioned estimate, since the induced disorder would not happen in a real simulation. In other words, particle distributions in the SPH method may exhibit irregularities, but they do so in *an orderly way*, following the motion of the described medium.

For that reason the errors will be estimated for a set of uniformly distributed particles. The error estimation of the particle approximation is possible through the Poisson summation formula [2, 5]. Monaghan [5] proved that:

- The particle approximation error depends on the smoothness of the kernel function. Smoother kernel functions tend to reduce errors.
- The summation approximation for a function (eq. 2.11) has negligible error when the smoothing length is greater that the interparticle distance Δx (grid spacing).
- The summation approximation for the derivative of a function shows the same behavior with the function approximation, i.e. error is small when $h>\Delta x$, but generally the error is larger than the error of the function approximation.

However, even if the particle approximation error is small/negligible for a uniformly set of particles, such a type of particle distribution may not be advisable. The reason is the idiosyncratic behavior of particles in SPH method, which, in the course of a simulation, tend to gather at characteristic lines (for example in directional flows which appear in jets) and remain aligned even if the result is unphysical [6]. Eventually particles tend to get compressed at one direction, while they expand at another direction. Thus, at later stages of a simulation, the particle approximation errors may dramatically increase due to the non-uniformity of the distribution and the large voids occurring between particles. Also, it is very likely the final distribution of particles and, consequently, field variable distributions to be unphysical. The previously discussed issue occurs especially when particles are placed on uniform grids, since they are already positioned aligned. In the course of the present work several examples of this issue will be shown, in cases of jet impingements, near stagnation points, or in other cases (for example shear cavity flow).

To sum-up, the particle approximation is greatly affected by the particle distribution, but using a regular particle distribution may not always be a good idea, since at later time steps it will result to clumped particles. Sometimes a tradeoff has to be made and a random perturbation may be needed on a regular particle distribution, in order to keep errors at a minimum, while ensuring that particles will not align at later time steps.

2.4. The kernel function

From the above it is obvious that the kernel function plays a crucial role in the SPH approximations. In the next part, several important kernel functions, widely mentioned in literature, will be mentioned.

The general form of the kernel function is the following:

$$W(\mathbf{r}_{ij}) = A(h^d) f\left(\frac{\|\mathbf{r}_{ij}\|}{h}\right) \tag{2.26}$$

The kernel function may be expressed as the product of a function *f*, which depends on the ratio of the interparticle distance $\|\mathbf{r}_{ij}\|$ to the smoothing length *h*, and a parameter *A* which depends on the smoothing length raised to the problem dimension *d* (1 for 1-dimensional, 2 for 2-dimensional and 3 for 3-dimensional problems). Function *f* may be polynomial or exponential, but it has to be monotonically decreasing when the interparticle distance is increasing. The parameter *A* is used in order the kernel function to be normalized. Its value can be calculated from the normalization condition:

$$\int_{\Omega} W(\mathbf{r}) dr = 1 \Rightarrow$$

$$\int_{\Omega} \left[A(h^d) f\left(\frac{\|\mathbf{r}\|}{h}\right) \right] dr = 1 \Rightarrow$$

$$A(h^d) = \frac{1}{\int_{\Omega} \left[f\left(\frac{\|\mathbf{r}\|}{h}\right) \right] dr} \tag{2.27}$$

Lucy (1977) initially used the following bell-shaped function:

$$W(\mathbf{r}) = A(h^d) \begin{cases} \left(1 + 3\frac{\|\mathbf{r}\|}{h}\right)\left(1 - \frac{\|\mathbf{r}\|}{h}\right)^3 & \frac{\|\mathbf{r}\|}{h} \le 1 \\ 0 & \frac{\|\mathbf{r}\|}{h} > 0 \end{cases} \tag{2.28}$$

The constant parameter $A(h^d)$ can be calculated as follows:

- For the 1D problem, the following definite integral has to be calculated (obviously there is no point integrating beyond *h* since the kernel function has a value of zero):

$$\int_{-h}^{+h} W dr = \int_{-h}^{h} \left(1 + 3\frac{|r|}{h}\right)\left(1 - \frac{|r|}{h}\right)^3 dr = \frac{4h}{5}$$

Thus $A(h^1) = \frac{5}{4h}$

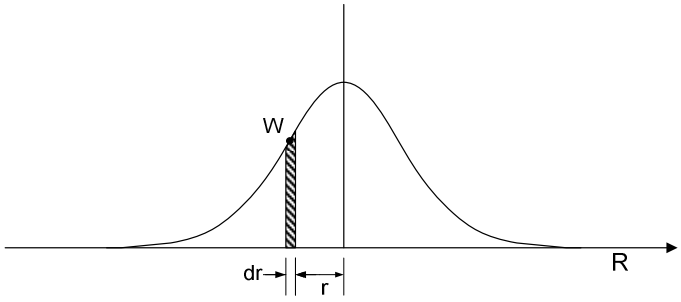


Fig. 2.3. Integration of the kernel function in 1D. Integration is performed in respect to the infinitesimal *dr*.

- For the 2D problem:

$$\int_S W dS = \int_0^{2\pi}\int_0^h \left[\left(1+3\frac{r}{h}\right)\left(1-\frac{r}{h}\right)^3 r\right] dr d\phi = \frac{h^2\pi}{5}$$

Thus $A(h^2) = \frac{5}{h^2\pi}$

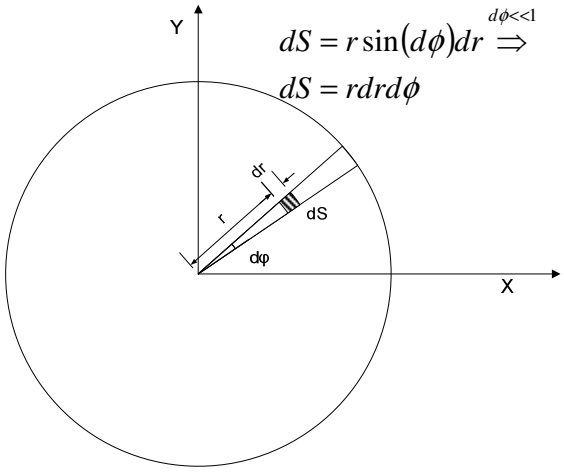


Fig. 2.4. Integration of the kernel function in 2D. Integration is performed in respect to the infinitesimal surface element *dS*.

- For the 3D problem:

$$\int_V W dV = \int_0^h \left[\left(1+3\frac{r}{h}\right)\left(1-\frac{r}{h}\right)^3 4\pi r^2\right] dr = \frac{16h^3\pi}{105}$$

Thus $A(h^3) = \frac{105}{16h^3\pi}$

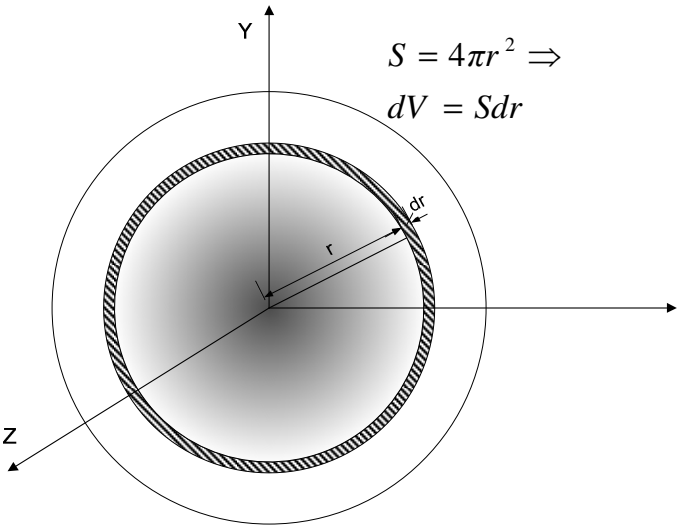


Fig. 2.5. Integration of the kernel function in 3D. Integration is performed in respect to the infinitesimal volume shell *dV*.

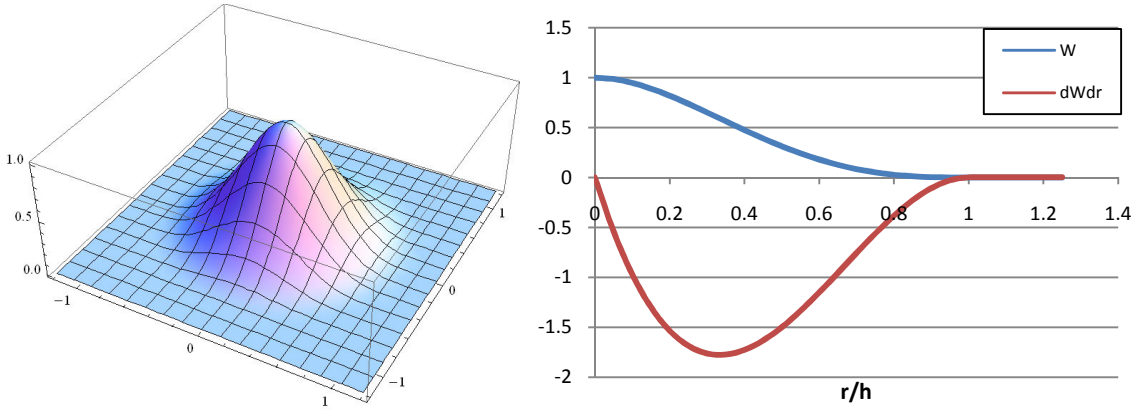


Fig. 2.6. Lucy's kernel function. Left: 3D view. Right: Plot of the kernel function and its derivative

Monaghan (1992) [3, 4] devised a kernel function based on the Gaussian function. The main advantage of this kernel function was the smoothness, even on high order derivatives, since the Gaussian is differentiable infinite times, with continuous derivatives. This results to increased stability and accuracy of the approximations. On the other hand, the Gaussian does not have a compact support, but tends to zero at infinity. Its value approaches zero very fast numerically, but it takes longer distance for its derivatives to approach zero. The previous facts lead to a large support domain, with the inclusion of more particles and increased computational cost.
The Gaussian kernel function is the following:

$$W(\mathbf{r}) = A(h^d)e^{-\left(\frac{\|\mathbf{r}\|}{h}\right)^2} \tag{2.29}$$

where $A(h^1) = \frac{1}{\sqrt{\pi}h}$, $A(h^2) = \frac{1}{\pi h^2}$ and $A(h^3) = \frac{1}{\pi^{3/2}h^3}$, calculated exactly in the same manner as above.

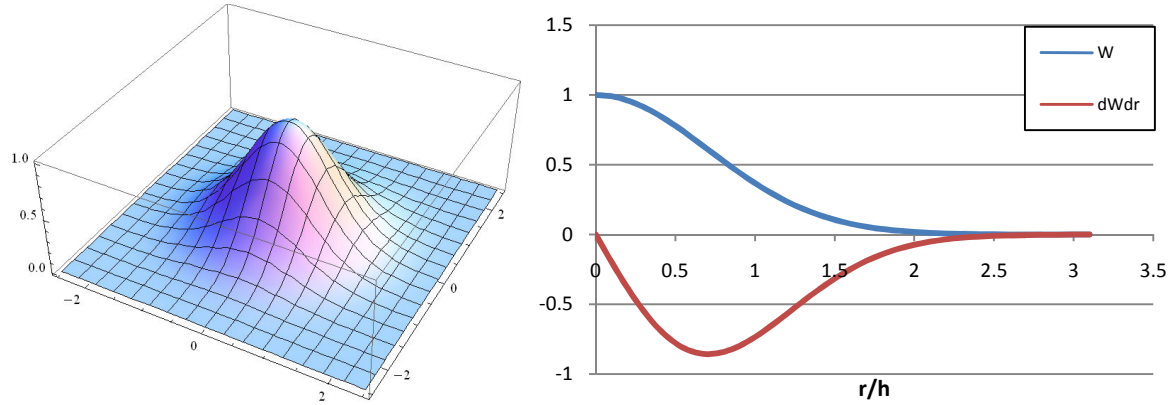


Fig 2.7. Gaussian kernel function. Left: 3D view. Right: Plot of the kernel function and its derivative

Monaghan and Lattanzio (1985) formulated the cubic spline kernel function. The cubic spline had the advantage of resembling the Gaussian function, while also having compact support. However the second derivative of the cubic spline is a piecewise function, thus not differentiable in areas of the support domain, causing instabilities.

$$W(\mathbf{r}) = A(h^d)\begin{cases} \frac{2}{3} - q^2 + \frac{1}{2}q^3 & 0 \le q < 1 \\ \frac{1}{6}(2-q)^3 & 1 \le q < 2 \\ 0 & q \ge 2 \end{cases} \tag{2.30}$$

where $q = \frac{\|\mathbf{r}\|}{h}$ and $A(h^1) = \frac{1}{h}$, $A(h^2) = \frac{15}{7\pi h^2}$, $A(h^3) = \frac{3}{2\pi h^3}$

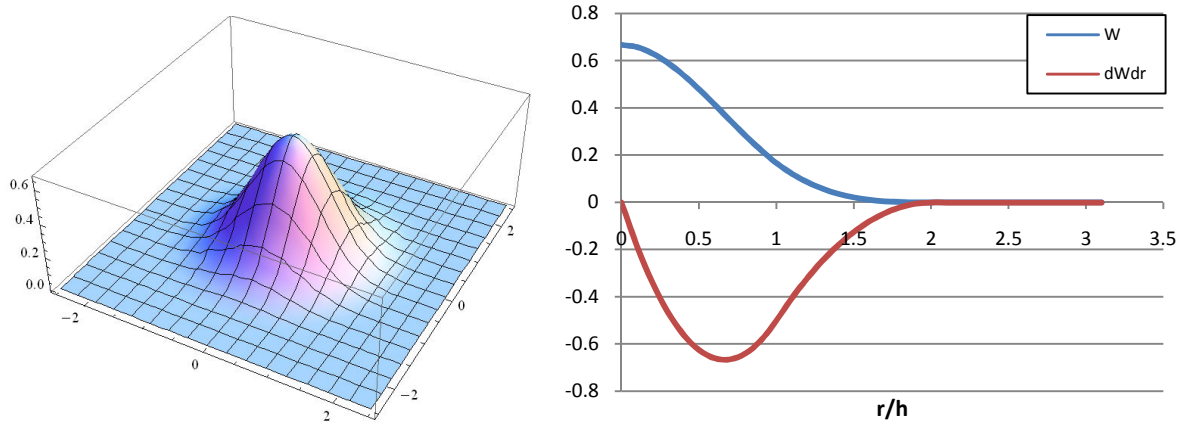


Fig. 2.8. Cubic spline kernel function. Left: 3D view. Right: Plot of the kernel function and its derivative

Morris (1994, 1996) devised higher order spline kernel functions, based on the 4th and 5th order spline. These kernel functions resemble the Gaussian function, have a finite support domain, are smoother than the cubic spline kernel function and, as a result, are more stable.

- 4th order Spline kernel function (quartic spline):

$$W(\mathbf{r}) = A(h^d)\begin{cases} (q+2.5)^4 - 5(q+1.5)^4 + 10(q+0.5)^4 & 0 \le q < 0.5 \\ (2.5-q)^4 - 5(1.5-q)^4 & 0.5 \le q < 1.5 \\ (2.5-q)^4 & 1.5 \le q < 2.5 \\ 0 & 2.5 \le q \end{cases} \tag{2.31}$$

where $A(h^1) = \frac{1}{24h}$, $A(h^2) = \frac{96}{1199\pi h^2}$, $A(h^3) = \frac{1}{20\pi h^3}$

- 5th order Spline kernel function (quintic spline):

$$W(\mathbf{r}) = A(h^d)\begin{cases} (3-q)^5 - 6(2-q)^5 + 15(1-q)^5 & 0 \le q < 1 \\ (3-q)^5 - 6(2-q)^5 & 1 \le q < 2 \\ (3-q)^5 & 2 \le q < 3 \\ 0 & 3 \le q \end{cases} \tag{2.32}$$

with $q = \frac{\|\mathbf{r}\|}{h}$, $A(h^1) = \frac{120}{h}$, $A(h^2) = \frac{7}{478\pi h^2}$, $A(h^3) = \frac{3}{359\pi h^3}$

In the present work the 4th order spline kernel function was chosen and finally used, mainly because of smoothness and stability. Alternatively the 5th order spline might have been used, however no notable difference was observed in results obtained with the 4th and 5th order spline kernel functions. Moreover the 5th order spline has increased support domain radius (3 smoothing lengths in contrast to 2.5 smoothing lengths for the 4th order spline), which means increased computational cost.

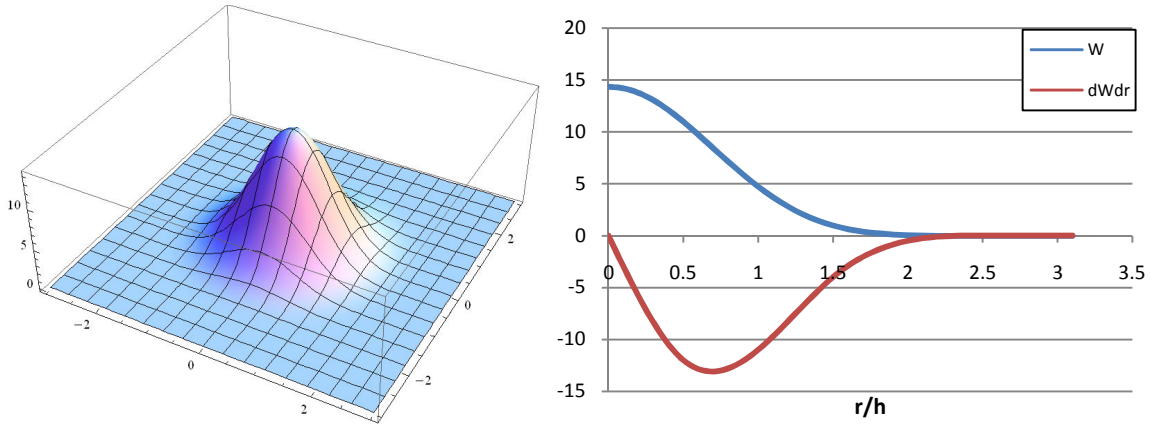


Fig. 2.9. Quartic kernel function. Left: 3D view. Right: Plot of the kernel function and its derivative

References

[1] G.R. Liu, M. B. Liu, "Smoothed particle hydrodynamics", World scientific publishing, 2003, ISBN 981-238-456-1

[2] Quinlan, N. J., Basa, M., Lastiwka, M., "Truncation error in mesh-free particle methods", International Journal of Numerical Methods in Engineering, vol. 66, p. 2064–2085, 2006, DOI: 10.1002/nme.1617

[3] F. Colin, R. Egli, F. Y. Lin, "Computing a null divergence velocity field using smoothed particle hydrodynamics, Journal of Computational Physics", Volume 217 Issue 2, 20 September 2006

[4] D.J. Price, "Magnetic fields in astrophysics", Phd thesis, University of Cambridge, UK, 2004.

[5] J.J. Monaghan, "Smoothed Particle Hydrodynamics", Reports on progress in physics, vol. 68, p. 1703-1759, 2005, DOI: 10.1088/0034-4885/68/8/R01

[6] P.J. Reichl, P. Morris, S.A.T. Stoneman, M.C. Thompson, K. Hourigan, "Smooth particle hydrodynamics modeling of vertical jet impingement", Proceedings of the 13th Australian Fluid Mechanics Conference, Monash University, Melbourne, Australia, 13-18 December 1998.

Chapter 3

The standard SPH method for flow simulation

In this part, the standard SPH method will be described, based on the works of Monaghan [1] and G.R. Liu [2]. Moreover, corrections and further developments, formulated by Violeau [3], Morris [4] and other researchers, will be presented. The SPH formulation which is used in the present study is based on the weakly compressible assumption, where pressure is linked to density through a stiff equation of state. Truly incompressible SPH is possible, but, as it is explained in section 3.5, it involves increased complexity and is prone to instabilities. The SPH model described here is capable of including the influence of viscous effects of Newtonian and non-Newtonian flows.

3.1. The Navier-Stokes equations

The equations of fluid flow are based on the following principles:

- Conservation of mass
- Conservation of momentum
- Conservation of energy

The flow may be described using Lagrangian or Eulerian framework. The basic difference between the two ways of description is the motion of the computational elements; using Eulerian description the computational elements remain still, whereas in Lagragian description the computational elements move following the flow. The mathematic consequence of the two different descriptions is that the Lagrangian time derivative is the sum of the Eulerian time derivative and the convection derivative. To be more specific the Lagrangian derivative of a scalar variable f is equal to:

$$\frac{Df}{Dt} = \frac{\partial f}{\partial t} + \mathbf{u} \cdot \nabla f = \frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} + v\frac{\partial f}{\partial y} + w\frac{\partial f}{\partial z} \tag{3.1}$$

where:

- $\frac{Df}{Dt}$ the Lagrangian time derivative (or else material derivative)
- $\frac{\partial f}{\partial t}$ the Eulerian time derivative
- $\mathbf{u} = (u, v, w)$ the velocity vector of the flow
- $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$, $\frac{\partial f}{\partial z}$ the space derivative of the scalar variable f in respect to x, y, z respectively

The fact that the SPH method is a particle Lagrangian method means that the calculation of the convection derivative is not required, because it is included in the Lagrangian time derivative. This is an advantage of the method, since the procedure of calculating the convection derivative, required for Eulerian methods, introduces additional numerical dissipation. In the Lagrangian description the control volume moves following the flow, thus the mass enclosed inside the control volume does not change; the control volume, however, may deform.

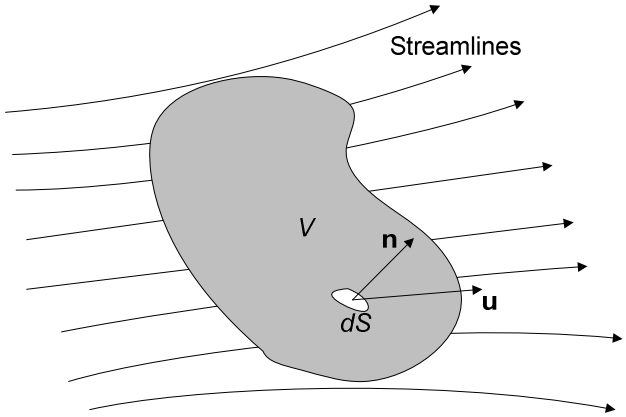


Fig. 3.1. A Lagrangian control volume.

The total volume change ΔV of a Lagrangian control volume inside a velocity field $\mathbf{u}=(u,v,w)$ may be calculated as the surface integral:

$$\Delta V = \int_S \mathbf{u}\cdot\mathbf{n}\Delta t dS \tag{3.2}$$

Using the divergence theorem (or Gauss theorem), the surface integral is written as a volume integral:

$$\frac{\Delta V}{\Delta t} = \int_V (\nabla\cdot\mathbf{u}) dV \tag{3.3}$$

Assuming that the control volume is small enough, so that all properties of the medium enclosed are the same throughout the volume, then:

$$\frac{D(\delta V)}{Dt} = (\nabla\cdot\mathbf{u})\int_V d(\delta V) = (\nabla\cdot\mathbf{u})\delta V \Rightarrow$$

$$\frac{1}{\delta V}\frac{D(\delta V)}{Dt} = \nabla\cdot\mathbf{u} \tag{3.4}$$

Continuity equation

The continuity equation is based on the conservation of mass. Considering that $\delta m = \rho\delta V$:

$$\frac{D(\delta m)}{Dt} = \frac{D(\rho\delta V)}{Dt} \tag{3.5}$$

However $\frac{D(\delta m)}{Dt}$=0, since the enclosed mass in the control volume remains unchanged through time.

Thus:

$$\delta V \frac{D\rho}{Dt} + \rho \frac{D(\delta V)}{Dt} = 0 \Rightarrow$$
$$\frac{D\rho}{Dt} + \rho \frac{1}{\delta V} \frac{D(\delta V)}{Dt} = 0 \Rightarrow$$
$$\frac{D\rho}{Dt} = -\rho(\nabla \cdot \mathbf{u}) \tag{3.6}$$

Momentum equation

The momentum equation expresses the 2nd Newton's law, i.e. the net force on the Lagrangian volume is equals to its mass multiplying the acceleration of that fluid cell.

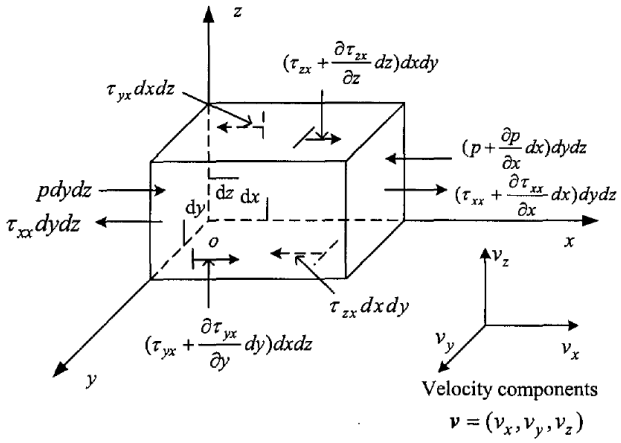


Fig. 3.2. Forces acting on a Lagrangian cell

The forces acting on the fluid cell are body forces and surface forces. Gravity, magnetic forces or other forces affecting the volume of a fluid cell can be considered as body forces. The surface forces are:

- Pressure, which is imposed by fluid elements surrounding the control volume
- Shear and normal stresses, which result to shear deformation and volume change

In the *x*-direction the forces which act upon the Lagrangian control volume are:

$$-\left[\left(p + \frac{\partial p}{\partial x} dx\right) - p\right] dydz + \left[\left(\tau_{xx} + \frac{\partial \tau_{xx}}{\partial x} dx\right) - \tau_{xx}\right] dydz$$
$$+\left[\left(\tau_{yx} + \frac{\partial \tau_{yx}}{\partial y} dy\right) - \tau_{yx}\right] dxdz + \left[\left(\tau_{zx} + \frac{\partial \tau_{zx}}{\partial z} dz\right) - \tau_{zx}\right] dxdy \tag{3.7}$$
$$= -\frac{\partial p}{\partial x} dxdydz + \frac{\partial \tau_{xx}}{\partial x} dxdydz + \frac{\partial \tau_{yx}}{\partial y} dxdydz + \frac{\partial \tau_{zx}}{\partial z} dxdydz$$

where p is pressure, τ_{ij} is the stress at j direction acting on the surface whose normal lies on i direction. Assuming the body force F_x, expressed per unit mass, on the x-axis:

$$m\frac{dv_x}{dt} = \rho dxdydz\frac{dv_x}{dt} =$$
$$-\frac{\partial p}{\partial x}dxdydz + \frac{\partial \tau_{xx}}{\partial x}dxdydz + \frac{\partial \tau_{yx}}{\partial y}dxdydz + \frac{\partial \tau_{zx}}{\partial z}dxdydz + F_x\left(\rho dxdydz\right) \tag{3.8}$$

Thus finally the momentum equation is written as:

$$\rho\frac{Du}{Dt} = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + \rho F_x \tag{3.9}$$

$$\rho\frac{Dv}{Dt} = -\frac{\partial p}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + \rho F_y \tag{3.10}$$

$$\rho\frac{Dw}{Dt} = -\frac{\partial p}{\partial z} + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} + \rho F_z \tag{3.11}$$

The stress is proportional to the strain rate through the dynamic viscosity:

$$\tau_{ab} = \mu\,\varepsilon_{ab} \tag{3.12}$$

where ε_{ab} is the strain rate (considering compressibility effects):

$$\varepsilon_{ab} = \frac{\partial u_b}{\partial x_a} + \frac{\partial u_a}{\partial x_b} - \frac{2}{3}(\nabla\cdot\mathbf{u})\delta_{ab} \tag{3.13}$$

In the previous relation, δ_{ab} is δ-Dirac function (unity when *a*=*b* and zero in all other cases) and *a*, *b* denote spatial dimension *x*, *y*, *z*.

All previous equations may be written in tensor form using the notation below:

$$\rho\frac{D\mathbf{u}}{Dt} = \nabla\cdot\mathbf{P} + \rho\mathbf{f}_{body} \tag{3.14}$$

$$\mathbf{P} = \begin{bmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} \end{bmatrix} = \begin{bmatrix} \tau_{xx}-p & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \tau_{yy}-p & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz}-p \end{bmatrix} = -p\cdot\mathbf{I} + \mathbf{T} \tag{3.15}$$

Energy equation

The energy equation expresses the conservation of energy. The rate of change of energy for a fluid element is equal to the total heat flux through the boundary from/towards the fluid element and the rate of work done by forces (body and surface forces) acting on the fluid element. If the heat flux is omitted, then the rate of change of the internal energy of the infinitesimal fluid cell consists of the:

- Pressure work due to volumetric strain
- Energy dissipation due to viscosity

Thus:

$$\begin{aligned}\rho\frac{De}{Dt} &= -p\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}\right)\\ &+\tau_{xx}\frac{\partial u}{\partial x}+\tau_{xy}\frac{\partial u}{\partial y}+\tau_{xz}\frac{\partial u}{\partial z}\\ &+\tau_{yx}\frac{\partial v}{\partial x}+\tau_{yy}\frac{\partial v}{\partial y}+\tau_{yz}\frac{\partial v}{\partial z}\\ &+\tau_{zx}\frac{\partial w}{\partial x}+\tau_{zy}\frac{\partial w}{\partial y}+\tau_{zz}\frac{\partial w}{\partial z}\end{aligned} \tag{3.16}$$

3.2. The Navier – Stokes equations expressed using the SPH method

Relations for density calculation / continuity equation

The density approximation is very important for the SPH method, since it determines the particle distribution and, depending on the implementation, the smoothing length evolution. Density may be evaluated using two different approaches. The first approach is the *summation density*, which directly applies the SPH approximations on the density field. Direct use of equation 2.11 for the density field, leads to the following relation:

$$\rho_i = \sum_{j=1}^{N} m_j W_{ij} \tag{3.17}$$

Here the kernel function, W_{ij} , acts as the inverse volume. Eq. 3.17 states that the density of a particle may be approximated using the weighted average of the density of all particles lying inside the support domain of the particle being examined.

The second approach is the *continuity density* which is based on calculating density evolution using the continuity equation, written in SPH form, plus some transformations. Beginning from the approximation 2.11, applied for the continuity equation (eq. 3.6) leads to:

$$\frac{D\rho_i}{Dt} = -\rho_i \sum_{j=1}^{N} \frac{m_j}{\rho_j}\mathbf{u}_j \cdot \nabla_i \mathbf{W}_{ij} \tag{3.18}$$

$$\text{i.e.: } \frac{D\rho_i}{Dt} = -\rho_i \sum_{j=1}^{N} \frac{m_j}{\rho_j}\left(u_j\frac{\partial W_{ij}}{\partial x}+v_j\frac{\partial W_{ij}}{\partial y}+w_j\frac{\partial W_{ij}}{\partial z}\right)$$

Assuming the particle approximation for the gradient of unity:

$$\nabla 1 = \sum_{j=1}^{N} \frac{m_j}{\rho_j} 1 \nabla W_{ij} = 0$$

and by multiplying both equation sides with $\rho_i \mathbf{u}_i$, one derives the following relation:

$$0 = \rho_i \mathbf{u}_i \sum_{j=1}^{N} \frac{m_j}{\rho_j} 1 \nabla W_{ij} = \rho_i \sum_{j=1}^{N} \frac{m_j}{\rho_j} \mathbf{u}_i \nabla W_{ij} \tag{3.19}$$

After adding eq. 3.18 and eq. 3.19, the resulting equation is:

$$\frac{D\rho_i}{Dt} = -\rho_i \sum_{j=1}^{N} \frac{m_j}{\rho_j} \left(\mathbf{u}_j - \mathbf{u}_i \right) \cdot \nabla_i W_{ij} = \rho_i \sum_{j=1}^{N} \frac{m_j}{\rho_j} \mathbf{u}_{ij} \cdot \nabla_i W_{ij} \tag{3.20}$$

where $\mathbf{u}_{ij} = \mathbf{u}_i - \mathbf{u}_j$ (generally φ_{ij} is defined as $\varphi_i - \varphi_j$)

The previous expression of the continuity equation uses relative particle velocities, instead of particle velocities themselves. It is preferred in respect to eq. 3.18, since using relative velocities reduces errors, especially in case there are few particles within the support radius.
A more popular form of the continuity equation is the following:

$$\frac{D\rho_i}{Dt} = \sum_{j=1}^{N} m_j \mathbf{u}_{ij} \cdot \nabla_i W_{ij} \tag{3.21}$$

Both approaches offer advantages and have disadvantages. The summation approximation for density evaluation is simple and it is able to conserve mass exactly, since the density integration for the entire problem domain is exactly the total mass of particles; Monaghan and Fulk proved that the continuity approach does not. However, density summation approach has the disadvantage of greatly underestimating density near boundaries (edge effect – will be further discussed in boundary conditions), or near interfaces between particles representing material of different densities (for example air and water). Another disadvantage is that the density evaluation has to be performed before any other calculation, thus leading to an extra calculation loop. The density change from the continuity equation may be calculated at the same time with other variables, such as the acceleration from the momentum equation, making the algorithm less computationally intensive and easier parallelizable. Also the continuity equation can be used successfully in cases of free surface flows, strong shocks and discontinuities.

A way of improving the accuracy of the summation approach is to normalize the RHS of eq. 3.17 with the SPH summation of the smoothing function itself:

$$\rho_i = \sum_j m_j \tilde{W}_{ij} \tag{3.22}$$

where:

$$\tilde{W}_{ij} = \frac{W_{ij}}{\sum_j \frac{m_j}{\rho_j} W_{ij}} \tag{3.23}$$

If one observes the relation for $\tilde{W}_{ij}$, it is obvious that the denominator will be unity for a regular set of particles of the same density (the denominator is actually eq. 2.11 where $f(x)$ is equal to unity). This approach is a more refined weighted average for the density calculation, which behaves better near boundaries and material interfaces. In the present work, it will be used as a density filter, as it will be mentioned later on.

Momentum equation

The derivation of the momentum equation is similar to the continuity density approach. Direct application of eq. 2.12 to the momentum equation yields, written for the *a*-velocity component (*a*, *b* represent the dimension):

$$\frac{Du_i^a}{Dt} = \frac{1}{\rho_i}\sum_j m_j \frac{\sigma_j^{ab}}{\rho_j}\frac{\partial W_{ij}}{\partial x^b} \tag{3.24}$$

Eq. 3.24 follows Einstein notation in respect to the *b* superscript (denoting dimension *x*, *y*, *z*), i.e.:

$$\frac{\sigma_j^{ab}}{\rho_j}\frac{\partial W_{ij}}{\partial x^b} = \sum_b \frac{\sigma_j^{ab}}{\rho_j}\frac{\partial W_{ij}}{\partial x^b} = \frac{\sigma_j^{ax}}{\rho_j}\frac{\partial W_{ij}}{\partial x} + \frac{\sigma_j^{ay}}{\rho_j}\frac{\partial W_{ij}}{\partial y} + \frac{\sigma_j^{az}}{\rho_j}\frac{\partial W_{ij}}{\partial z}$$

Thus for the *x*-velocity component the previous equation would be written as:

$$\frac{Du_i}{Dt} = \frac{1}{\rho_i}\sum_j \frac{m_j}{\rho_j}\left[\left(\tau_j^{xx} - p_j\right)\frac{\partial W}{\partial x} + \left(\tau_j^{xy}\right)\frac{\partial W}{\partial y} + \left(\tau_j^{xz}\right)\frac{\partial W}{\partial z}\right] \tag{3.25}$$

The same notation is adopted for the rest of the equations too.
Eq. 3.24 may be rewritten, using the following identity:

$$\frac{\sigma_i^{ab}}{\rho_i}\sum_j \frac{m_j}{\rho_j}\frac{\partial W_{ij}}{\partial x^b} = \sum_j m_j \frac{\sigma_i^{ab}}{\rho_i\rho_j}\frac{\partial W_{ij}}{\partial x^b} = 0$$

as:

$$\frac{Du_i^a}{Dt} = \sum_j m_j \frac{\sigma_i^{ab} + \sigma_j^{ab}}{\rho_i\rho_j}\frac{\partial W_{ij}}{\partial x^b} \tag{3.26}$$

As with the continuity equation, the symmetrized form of the momentum equation helps in minimizing errors.

An alternative and more popular formulation may be derived [2] using:

$$\frac{1}{\rho}\frac{\partial \sigma^{ab}}{\partial x^b} = \frac{\partial}{\partial x^b}\left(\frac{\sigma^{ab}}{\rho}\right) + \frac{\sigma^{ab}}{\rho^2}\frac{\partial \rho}{\partial x^b} \tag{3.27}$$

leading eventually to:

$$\frac{Du_i^a}{Dt} = \sum_j m_j \left(\frac{\sigma_i^{ab}}{\rho_i^2} + \frac{\sigma_j^{ab}}{\rho_j^2}\right)\frac{\partial W_{ij}}{\partial x^b} \tag{3.28}$$

The latter formulation will be used in the present work, which may be written in a general form as:

$$\frac{Du_i^a}{Dt} = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\frac{\partial W_{ij}}{\partial x^a} + \sum_j m_j \left(\frac{\mu_i \varepsilon_i^{ab}}{\rho_i^2} + \frac{\mu_j \varepsilon_j^{ab}}{\rho_j^2}\right)\frac{\partial W_{ij}}{\partial x^b} + f_{body}^a \tag{3.29}$$

where f_{body}^a is any acceleration due to body forces (such as gravity) at the *a*-direction.

One popular SPH approximation for the strain rate is:

$$\varepsilon_i^{ab} = \sum_{j=1}^{N} \frac{m_j}{\rho_j} u_{ji}^{a} \frac{\partial W_{ij}}{\partial x^b} + \sum_{j=1}^{N} \frac{m_j}{\rho_j} u_{ji}^{b} \frac{\partial W_{ij}}{\partial x^a} - \frac{2}{3}\left(\sum_{j=1}^{N} \frac{m_j}{\rho_j} \mathbf{u}_{ji} \cdot \nabla W_{ij} \right) \delta^{ab} \tag{3.30}$$

Finally the *x*-momentum equation is written as:

$$\begin{aligned} \frac{Du_i}{Dt} = & -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \frac{\partial W_{ij}}{\partial x} \\ & + \sum_j m_j \left(\frac{\mu_i \varepsilon_i^{xx}}{\rho_i^2} + \frac{\mu_j \varepsilon_j^{xx}}{\rho_j^2} \right) \frac{\partial W_{ij}}{\partial x} + \sum_j m_j \left(\frac{\mu_i \varepsilon_i^{xy}}{\rho_i^2} + \frac{\mu_j \varepsilon_j^{xy}}{\rho_j^2} \right) \frac{\partial W_{ij}}{\partial y} + \sum_j m_j \left(\frac{\mu_i \varepsilon_i^{xz}}{\rho_i^2} + \frac{\mu_j \varepsilon_j^{xz}}{\rho_j^2} \right) \frac{\partial W_{ij}}{\partial z} \\ & + f_x \end{aligned} \tag{3.31}$$

and the strain rates:

$$\varepsilon_i^{xx} = \sum_{j=1}^{N} \frac{m_j}{\rho_j} u_{ji} \frac{\partial W_{ij}}{\partial x} + \sum_{j=1}^{N} \frac{m_j}{\rho_j} u_{ji} \frac{\partial W_{ij}}{\partial x} - \frac{2}{3}\left[\sum_{j=1}^{N} \frac{m_j}{\rho_j} \left(u_{ji} \frac{\partial W_{ij}}{\partial x} + v_{ji} \frac{\partial W_{ij}}{\partial y} + w_{ji} \frac{\partial W_{ij}}{\partial z} \right) \right] \tag{3.32}$$

$$\varepsilon_i^{xy} = \sum_{j=1}^{N} \frac{m_j}{\rho_j} u_{ji} \frac{\partial W_{ij}}{\partial y} + \sum_{j=1}^{N} \frac{m_j}{\rho_j} v_{ji} \frac{\partial W_{ij}}{\partial x} \tag{3.33}$$

$$\varepsilon_i^{xz} = \sum_{j=1}^{N} \frac{m_j}{\rho_j} u_{ji} \frac{\partial W_{ij}}{\partial z} + \sum_{j=1}^{N} \frac{m_j}{\rho_j} w_{ji} \frac{\partial W_{ij}}{\partial x} \tag{3.34}$$

Energy equation

In a similar manner as the momentum and the continuity equation, it is possible to derive the SPH equation for the internal energy evolution. The two forms of the energy equation are the following:

$$\frac{de_i}{dt} = \frac{1}{2} \sum_j \left[m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \mathbf{u}_{ij} \cdot \nabla W_{ij} \right] + \frac{\mu_i}{2\rho_i} \boldsymbol{\varepsilon}_i : \boldsymbol{\varepsilon}_i \tag{3.35}$$

$$\frac{de_i}{dt} = \frac{1}{2} \sum_j \left[m_j \left(\frac{p_i + p_j}{\rho_i \rho_j} \right) \mathbf{u}_{ij} \cdot \nabla W_{ij} \right] + \frac{\mu_i}{2\rho_i} \boldsymbol{\varepsilon}_i : \boldsymbol{\varepsilon}_i \tag{3.36}$$

The expanded form of the eq. 3.35, for 3D, would be the following:

$$\begin{aligned} \frac{de_i}{dt} = & \frac{1}{2} \sum_j \left[m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \left(u_{ij} \frac{\partial W_{ij}}{\partial x} + v_{ij} \frac{\partial W_{ij}}{\partial y} + w_{ij} \frac{\partial W_{ij}}{\partial z} \right) \right] \\ & + \frac{\mu_i}{2\rho_i} \left(\varepsilon_i^{xx} \varepsilon_i^{xx} + 2\varepsilon_i^{xy} \varepsilon_i^{xy} + 2\varepsilon_i^{xz} \varepsilon_i^{xz} + \varepsilon_i^{yy} \varepsilon_i^{yy} + 2\varepsilon_i^{yz} \varepsilon_i^{yz} + \varepsilon_i^{zz} \varepsilon_i^{zz} \right) \end{aligned} \tag{3.37}$$

3.3. Alternative viscosity treatments

In the previous part, the momentum equation was expressed considering the viscous contributions using stress and strain relations. This way of estimating the viscous forces requires nested approximations first for the strain rate (eq. 3.30) and then on the stresses (eq. 3.31), which generally are rather computationally intensive; at a first step the strain rate has to be calculated looping all particles and then the viscous contribution is estimated after looping all particles again. Moreover the application of stress and strain relations in wall bounded flows would require the extrapolation of additional fields (the stresses) on the wall particles in order to properly estimate viscous contributions at the near wall area. In complicated geometries this may be problematic.

Thus, most SPH practitioners [1, 2, 3, 4, 5, 6] do not use the stress-strain relations, but other simplified equations which originate from the SPH approximation for the second derivative of the velocity (see also eq. 2.19 for the SPH approximation of an arbitrary function), such as the following [4]:

$$\mathbf{\Pi}_{ij} = \frac{\mu_i + \mu_j}{\rho_i \rho_j} \frac{\mathbf{u}_{ij}}{\left\| \mathbf{r}_{ij} \right\|^2} \mathbf{r}_{ij} \cdot \nabla W_{ij} \tag{3.38}$$

and the momentum equation is written as:

$$\frac{D\mathbf{u}_i}{Dt} = -\sum_j m_j \left[\left(\frac{p_i}{\rho_i^{\,2}} + \frac{p_j}{\rho_j^{\,2}} \right) \nabla W_{ij} - \mathbf{\Pi}_{ij} \right] + \mathbf{f}_{body} \tag{3.39}$$

This simplified relation for accounting the viscosity term is much simpler than the strain – stress relations and it is able to handle materials with different viscosities, non-Newtonian flows and turbulence effects. Also it is simpler to program and can be calculated at the same time with the pressure contribution of the momentum equation, thus leading to only one loop needed to estimate the momentum and continuity contributions for all particles. Moreover its results are very close to those obtained by the stress strain relations as it will be shown later on.

The previous viscosity treatments had to do with the modeling of physical viscosity. There is another form of viscosity which has to be considered when simulating problems of hydrodynamics, in order to model shock waves, without unphysical oscillations near the shock wave. A shock wave is not a true discontinuity, but a transition zone whose thickness is in the order of a few molecular mean free paths. Application of the mass, momentum and energy conservation across a shock wave requires the simulation of a mechanism transforming kinetic energy to heat energy. A way to represent this transformation is with the use of an artificial form of dissipation. The Monaghan type artificial viscosity Π_{ij} is the most widely artificial viscosity term used in the SPH literature for modeling shock waves. It provides the necessary dissipation to convert kinetic energy into heat at the shock front and also prevents the unphysical penetration of particles approaching each other. The detailed formulation is as follows:

$$\Pi_{ij} = \begin{cases} \dfrac{-\alpha_{\Pi} \bar{c}_{ij} \varphi_{ij} + \beta_{\Pi} \varphi_{ij}}{\bar{\rho}_{ij}} & \mathbf{u}_{ij} \cdot \mathbf{r}_{ij} < 0 \\ 0 & \mathbf{u}_{ij} \cdot \mathbf{r}_{ij} < 0 \end{cases} \tag{3.40}$$

where:

$$\varphi_{ij} = \frac{h_{ij}\mathbf{u}_{ij} \cdot \mathbf{r}_{ij}}{\left\|\mathbf{r}_{ij}\right\|^2 + (0.1h_{ij})^2} \tag{3.41}$$

The symbols $\overline{\rho}_{ij}$, $\overline{c}_{ij}$ represent the average density and average speed of sound respectively. h_{ij} is the average smoothing length, in case variable smoothing length is used. The term $(0.1h_{ij})^2$ is added at the denominator to avoid singularities when two particles are approaching each other.

$$\overline{\rho}_{ij} = \frac{1}{2}\left(\rho_i + \rho_j\right)$$

$$\overline{c}_{ij} = \frac{1}{2}\left(c_i + c_j\right)$$

$$c_i = \sqrt{\frac{\partial p}{\partial \rho}}^{Tait} = c_0\left(\frac{\rho}{\rho_0}\right)^{\frac{\gamma-1}{2}}$$

$$h_{ij} = \frac{1}{2}\left(h_i + h_j\right)$$

$$\mathbf{u}_{ij} = \mathbf{u}_i - \mathbf{u}_j \qquad \mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$$

Here c_0 is the speed of sound at the reference density ρ_0 , when the Tait equation of state is used. If the ideal gas equation of state is used, then the appropriate formulation for the speed of sound should be adopted.

The parameters α_{Π}, β_{Π} are constants set to ~1. Viscosity associated with α_{Π} produces bulk viscosity, while β_{Π} is intended to suppress particle interpenetration.

3.4. Equation of state – Tait equation

When solving compressible flows, the continuity, momentum and energy equations are solved, while pressure is estimated through the ideal gas equation of state, linking pressure to the internal energy and density. In the case of incompressible flows, a divergence free velocity field has to be calculated. However the resulting set of equations is rather cumbersome due to the nature of the SPH method (see the next section covering the incompressibility in SPH). On the other hand, the fact that every incompressible flow is actually compressible, at least to a small extent, led many SPH researchers to use the concept of weak compressibility, i.e. assuming that the described medium is weakly compressible, using an appropriate equation of state. Often the Tait equation of state is used [7]:

$$p = B\left[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right] \tag{3.42}$$

where, for nearly incompressible flows, γ is a constant set to 7, ρ_0 is the reference density and B is the stiffness parameter equal to $\rho_0 c_0^{\,2} \big/ \gamma$.

The c_0 parameter in the stiffness parameter B represents the speed of sound at the reference density ρ_0. Generally it is not computationally efficient to use the actual speed of sound of the simulated medium, since its value would result to prohibitive time steps (for example the speed of sound for water at standard temperature and pressure is 1480m/s). The value of the speed of sound directly affects the integration time step of the method. A compromise has to be made; on the one hand, using a higher speed of sound leads to increased fluid stiffness, but also a reduction of the integration time step is required to ensure stability. On the other hand lower speed of sound values lead to increased time step and faster time-marching, but the fluid may not have a realistic behavior. The value of the numerical speed of sound has to be selected properly in order to simulate the fluid as nearly incompressible, while ensuring the density variation at a minimum. Monaghan [1] showed that:

$$\frac{|\delta\rho|}{\rho} \sim \frac{\|\mathbf{u}\|_{\max}^{\ 2}}{c^2} \tag{3.43}$$

Using a numerical speed of sound at least ten times the maximum velocity appearing in the simulation would result to density variation less or equal to one percent, since:

$$\frac{|\delta\rho|}{\rho} \sim \frac{\|\mathbf{u}\|_{\max}^{\ 2}}{\left(10 \cdot \|\mathbf{u}\|_{\max}\right)^2} \Rightarrow |\delta\rho| \sim 0.01\rho \tag{3.44}$$

The time step can be estimated, using the following relation [3]:

$$dt = \min\left(0.4\frac{h}{c_0}, 0.25\min\left(\sqrt{\frac{h}{\gamma_i}}\right), 0.125\min\left(\frac{\rho_i h^2}{\mu_i}\right)\right) \tag{3.45}$$

where γ_i is the modulus of acceleration and μ_i the dynamic viscosity of particle i. The first term in eq. 3.45 is the CFL condition, the second term is an additional constrains due to particle acceleration and the final term is due to viscous diffusion.

Here it has to be highlighted that, even when using the time step given by the previous relation and an appropriate speed of sound, pressure and density distributions tend to exhibit oscillations and numerical noise. This is a side effect from the equation of state used; the specific equation of state creates large pressures for small density variations [6]. In order to smooth the oscillations in the density (and consequently pressure) field a density filter is used. An alternative way would be to enforce incompressibility as will be shown in the next section. Another way, which will be considered later on, is to use Riemann solvers for the solution of the inter-particle contributions.

3.5. Truly incompressible SPH

As it was mentioned in the previous paragraph, it is possible to solve the truly incompressible Navier Stokes equations using the SPH method. For doing so there are two ways; the first uses the projection method and the second the pressure correction method.

Projection method

The projection method was invented by Chorin [8] for solving the incompressible Navier-Stokes equations. The key advantage of the method is that pressure and velocity are decoupled. The main idea of the method is to calculate an intermediate velocity field, which does not satisfy incompressibility and then calculate a pressure field to project the intermediate velocity field to the divergence free space.

Starting from the NS equations (in Lagrangian perspective):

$$\left.\begin{aligned} \nabla \cdot \mathbf{u} &= 0 \\ \frac{D\mathbf{u}}{Dt} &= -\frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2 \mathbf{u} + \mathbf{f} \end{aligned}\right\} \tag{3.46}$$

The time discretization of the momentum equation is split in two parts:

- First an intermediate velocity field is calculated using the body forces and the viscosity contribution (*n* represents the time step):

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{dt} = \frac{\mu}{\rho}\nabla^2 \mathbf{u}^n + \mathbf{f}^n \tag{3.47}$$

- This intermediate velocity field $\mathbf{u}^*$ is not divergence free. A correction is required to obtain the final velocity field $\mathbf{u}^{n+1}$ which satisfies $\nabla \mathbf{u}^{n+1} = 0$:

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{dt} = -\frac{1}{\rho}\nabla p^{n+1} \tag{3.48}$$

Applying the divergence operator to the previous equation, leads to the pressure Poisson equation:

$$\Delta p^{n+1} = \frac{\rho}{\Delta t}\nabla \cdot \mathbf{u}^* \tag{3.49}$$

The previous equation actually is a linear system of equations. Solving this linear system of equations results to the pressure field at the time step *n+1* and then the pressure field is used to correct the intermediate velocity field (eq. 3.48).

The described procedure is applicable in the SPH framework, following Moulinec et al. [9]. Moulinec et al. used the following approximation for the Laplacian of pressure:

$$\Delta p = \frac{2}{\rho_i}\sum_j m_j \left(p_i - p_j\right)\frac{\mathbf{r}_{ij} \cdot \nabla_i W_{ij}}{\left\|\mathbf{r}_{ij}\right\|^2} \tag{3.50}$$

The intermediate velocity field is calculated without considering the pressure contribution:

$$\mathbf{u}_i^* = \mathbf{u}_i^n + dt\left(\sum_j m_j \frac{\mu_i + \mu_j}{\rho_i \rho_j}\frac{\mathbf{u}_{ij}}{\left\|\mathbf{r}_{ij}\right\|^2}\mathbf{r}_{ij} \cdot \nabla_i W_{ij} + \mathbf{f}_{body}\right) \tag{3.51}$$

The pressure Poisson equation is expressed in the SPH form as follows:

$$\sum_j m_j p_{ij}^{n+1} \frac{\mathbf{r}_{ij} \cdot \nabla_i W_{ij}}{\|\mathbf{r}_{ij}\|^2} = -\frac{\rho_i}{2\Delta t} \sum_j m_j \mathbf{u}_{ij}^* \cdot \nabla_i W_{ij} \tag{3.52}$$

After calculation of the pressure field the final velocity field is calculated:

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^* - \frac{dt}{\rho_i} \left(\sum_j \frac{m_j}{\rho_j} p_{ji}^{n+1} \nabla_i W_{ij} \right) \tag{3.53}$$

By observing eq. 3.52, it is possible to rewrite it in matrix form. By setting:

$$A_{ij} = m_j \frac{\mathbf{r}_{ij} \cdot \nabla_i W_{ij}}{\|\mathbf{r}_{ij}\|^2} \tag{3.54}$$

then the LHS of equation 3.52 for *i* particle, is written as:

$$\sum_j A_{ij} p_{ij} = A_{i1}(p_i - p_1) + A_{i2}(p_i - p_2) + \ldots + A_{iN}(p_i - p_N) \Rightarrow$$

$$\sum_j A_{ij} p_{ij} = -A_{i1} p_1 - A_{i2} p_2 + \ldots + p_i \sum_j (A_{ij}) - \ldots - p_N A_{iN} \tag{3.55}$$

Note that the A_{ij} term is negative, since:

$$A_{ij} = m_j \frac{\mathbf{r}_{ij} \cdot \nabla_i W_{ij}}{\|\mathbf{r}_{ij}\|^2} = \frac{m_j}{\|\mathbf{r}_{ij}\|^2} (x_{ij}, y_{ij}, z_{ij}) \cdot \left(\frac{x_{ij}}{\|\mathbf{r}_{ij}\|}, \frac{y_{ij}}{\|\mathbf{r}_{ij}\|}, \frac{z_{ij}}{\|\mathbf{r}_{ij}\|} \right) \frac{dW_{ij}}{dr_{ij}} = \frac{m_j}{\|\mathbf{r}_{ij}\|} \frac{dW_{ij}}{dr_{ij}} \tag{3.56}$$

In the above equation all terms are positive, apart from the kernel function derivative (*dW/dr*) which is always negative, since the kernel function has to be monotonically decreasing (see also kernel function properties, section 2.2).

Using this approach, eq. 3.52 can be written in matrix form $\mathbf{A}\,\mathbf{P} = \mathbf{B}$, where **A** is a sparse symmetric *N*x*N* matrix (*N* is the number of particles involved in the simulation) and **B** is a *N*x1 vector:

$$\mathbf{A} = \begin{bmatrix} \sum_{j=1}^{n} A_{1j} & -A_{12} & -A_{13} & .... & -A_{1N} \\ -A_{21} & \sum_{j=1}^{n} A_{2j} & -A_{23} & .... & -A_{2N} \\ ... & ... & .... & .... & .... \\ ... & ... & ... & .... & .... \\ -A_{N1} & A_{N2} & ... & ... & \sum_{j=1}^{n} A_{Nj} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} -\frac{\rho_1}{2\Delta t} \sum_j m_j \mathbf{u}_{1j}^* \cdot \nabla_1 W_{1j} \\ -\frac{\rho_2}{2\Delta t} \sum_j m_j \mathbf{u}_{2j}^* \cdot \nabla_2 W_{2j} \\ ... \\ ... \\ -\frac{\rho_N}{2\Delta t} \sum_j m_j \mathbf{u}_{Nj}^* \cdot \nabla_N W_{Nj} \end{bmatrix}$$

Since the **A** matrix is sparse, special treatments have to be used in order to be stored efficiently. Only non-zero elements have to be stored, reducing memory requirements.

As an example of the specific method, the following test case is presented: a square patch of fluid is assumed, with an initial velocity field which does not satisfy the divergence free condition (fig. 3.5). The dimensions of the square patch is 40x40 particles, the total number of particles is 1600. The size of the **A** matrix is 1600x1600, but there are only 55100 non-zero elements. An indicative plot of

the matrix pattern is shown in fig. 3.3. It has to be highlighted that the matrix is not band diagonal, since not all consecutive diagonals are non-zero. Thus, simple linear equation system solvers, such as tridiagonal etc. cannot be used.

The linear system of equations is solved with an iterative method (Bi-Conjugate Gradients – fortran code of the algorithm may be found in [10]). A comparison is also made (see fig. 3.4) with the iterative Gauss-Seidel method (may be found from [11, 12]). Obviously the BiCG method is able to solve the system of equations much faster, both in respect to CPU time and iterations, without oscillations. In both cases the velocity field is resolved using the presented method and the final velocity and pressure fields are shown in fig. 3.5.

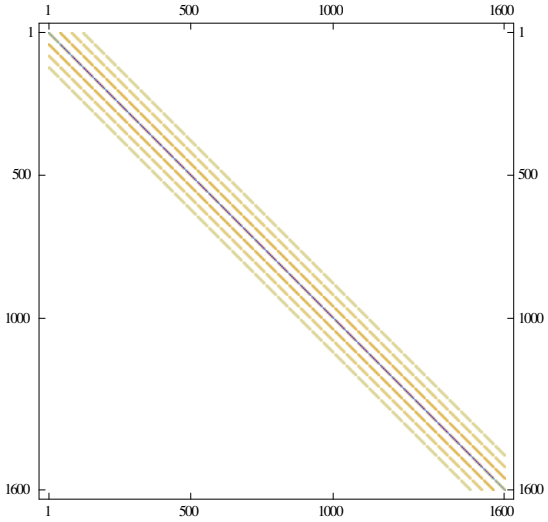


Fig. 3.3. Plot of the elements of matrix **A**. Non-zero elements are displayed only. Negative elements are shown with blue, positive elements with other colors

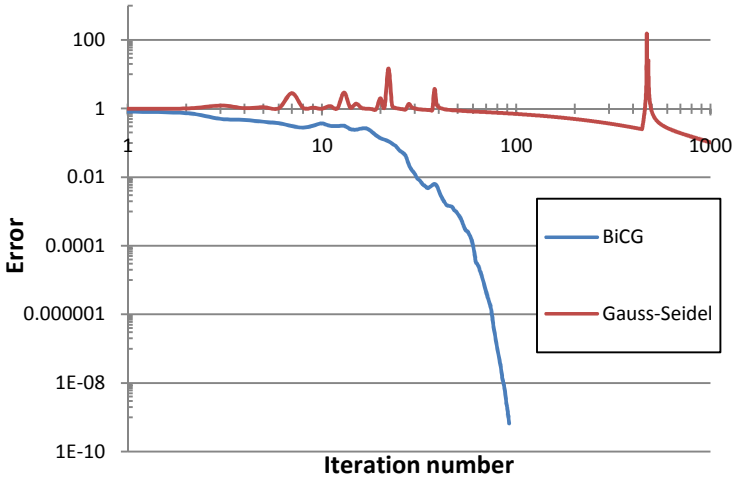


Fig. 3.4. Convergence history for the BiCG and Gauss-Seidel iterative methods

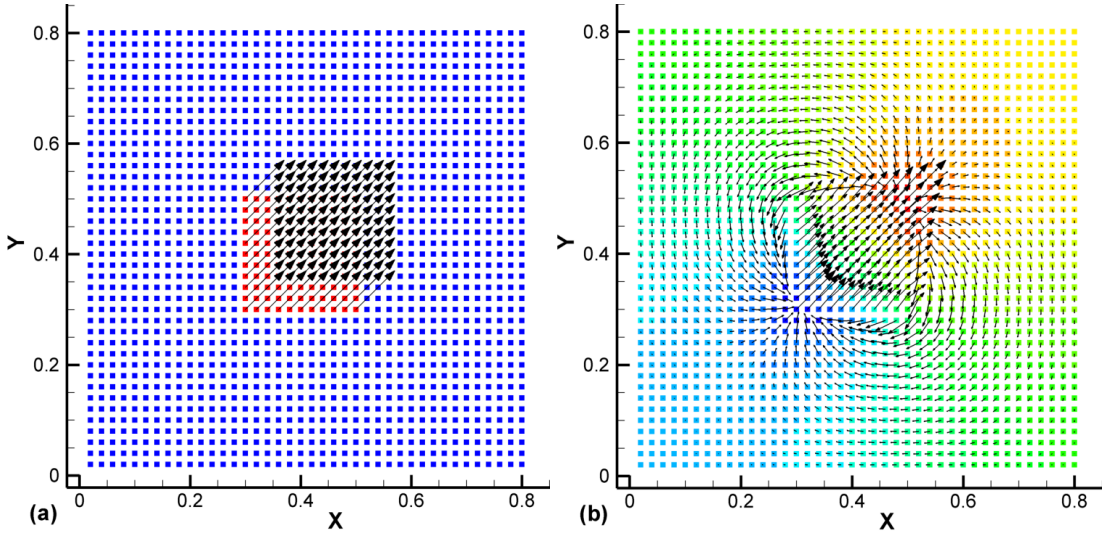


Fig. 3.5. Projection method test case: (a) Indicative velocity field before solving the Poisson equation and (b) Pressure field and velocity vectors after solving the Poisson equation and correcting the intermediate velocity field

The main issue of the presented method is the fact that, since particles change location throughout a simulation, their relative positions change and the coefficients of the **A** matrix do not remain constant. In other words, at one time step an element A_{ij} may be zero but at a later time step it may be not. In the finite difference method the coefficients of the equivalent matrix do not change, since the topology of the computational grid does not change also, thus the coefficient matrix is assembled only once. Actually the coefficients in the matrix represent the connectivity of the computational elements, or, better adapted to the SPH method, which particle interacts with the other and the weights of interactions. Since particles move, different particles will interact with different sets of neighbors each time. All the above render the assembling of the matrix a difficult and time consuming task for the SPH method.

Additionally, it is difficult to incorporate boundary conditions in the presented scheme, apart from simple cases where mirror particles are used. In more complex geometries particle redistribution might be required for smoothing the solution [9]. Moreover, in cases with free-surface, pressure has to be fixed to zero at the free surface particles. This means that at each time step the free surface has to be tracked, increasing the computational cost, while the results may not always be good; researchers have reported spurious pressure gradients near the free surface close to boundaries [13]. Finally, the parallelization of the described algorithm is not straightforward [14].

Pressure correction method

The pressure correction method [15, 16] avoids the difficulties which arise when trying to solve the pressure Poisson equation, by linking pressure to density and correcting iteratively the pressure and velocity fields, instead of solving the linear system of equations. The method could be considered as a hybrid of the SPH using an equation of state and the SPH solving the pressure Poisson equation [15]. First the momentum equation is solved giving an initial velocity field $\mathbf{u}^*$ [15]:

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{dt} = -\frac{1}{\rho}\nabla p^n + \frac{\mu}{\rho}\nabla^2 \mathbf{u}^n + \mathbf{f} \tag{3.57}$$

The initial velocity field $\mathbf{u}^*$ does not necessarily satisfy the divergence free constraint. Then, using the continuity equation:

$$\frac{D\rho}{Dt} = -\rho\left(\nabla \cdot \mathbf{u}^*\right) = \sum_{j=1}^{N} m_j \mathbf{u}_{ij}^{\ *} \cdot \nabla_i W_{ij} \tag{3.58}$$

it is possible to link the density change to a pressure change (correction). Using the equation of state:

$$p = B\left[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right] \Rightarrow \rho = \rho_0 \sqrt[\gamma]{\frac{p+B}{B}} \tag{3.59}$$

Replacing density in the continuity equation from eq. 3.59 yields:

$$\frac{D\left(\rho_0 \sqrt[\gamma]{\frac{p_i + B}{B}}\right)}{Dt} = \sum_{j=1}^{N} m_j \mathbf{u}_{ij}^{\ *} \cdot \nabla_i W_{ij} \Rightarrow$$

$$\frac{1}{\left(\frac{p_i + B}{B}\right)^{\frac{1}{\gamma}-1}} \frac{Dp_i}{Dt} = \frac{1}{\rho_0} \sum_{j=1}^{N} m_j \mathbf{u}_{ij}^{\ *} \cdot \nabla_i W_{ij} \Rightarrow$$

$$\delta p_i^{\ n} = dt \frac{\left(\frac{p_i + B}{B}\right)^{\frac{1}{\gamma}-1}}{\rho_0} \sum_{j=1}^{N} m_j \mathbf{u}_{ij}^{\ *} \cdot \nabla_i W_{ij} \tag{3.60}$$

Eq. 3.60 gives the pressure correction. Then pressure for *i* particle should be corrected as following:

$$p_i^{\ n+1} = p_i^{\ n} + \omega \delta p_i^{\ n} \tag{3.61}$$

and velocity should be corrected too, using the gradient of pressure correction:

$$\frac{d\mathbf{u}_i}{dt} = -\frac{1}{\rho_i} \nabla \delta p_i^{\ n} \tag{3.62}$$

which, by applying the SPH gradient approximation (eq. 2.17), results to:

$$\mathbf{u}_i^{\ n+1} = \mathbf{u}^* - \Omega \frac{dt}{\rho_i^{\ 2}} \sum_j m_j \left(\delta p_j^{\ n} - \delta p_i^{\ n}\right) \nabla_i W_{ij} \tag{3.63}$$

At the end of this step, $\mathbf{u}_i^{\ n+1}$ is considered as $\mathbf{u}^*$ and the procedure goes on, from eq. 3.57 to eq. 3.63, until the velocity divergence calculated from the continuity equation becomes less than a threshold value. The relaxation parameters ω and Ω play a crucial role in the corrective procedure. Large values may lead to instability and numerical oscillations. Low values will lead to very small corrections and many iterations to achieve zero (practically a small) velocity divergence. If one observes the whole procedure, it becomes apparent that the idea of this method is to create a pressure

disturbance and propagate it throughout the whole problem domain, within the current time step, correcting both velocity and pressure fields.

To assess the capability of the described method, the same test case, as in the previous section, is used; a square fluid patch with an initial divergent velocity field. In fig. 3.6 several instances are shown, during successive iterations. Also in fig. 3.7 the convergence history is shown.

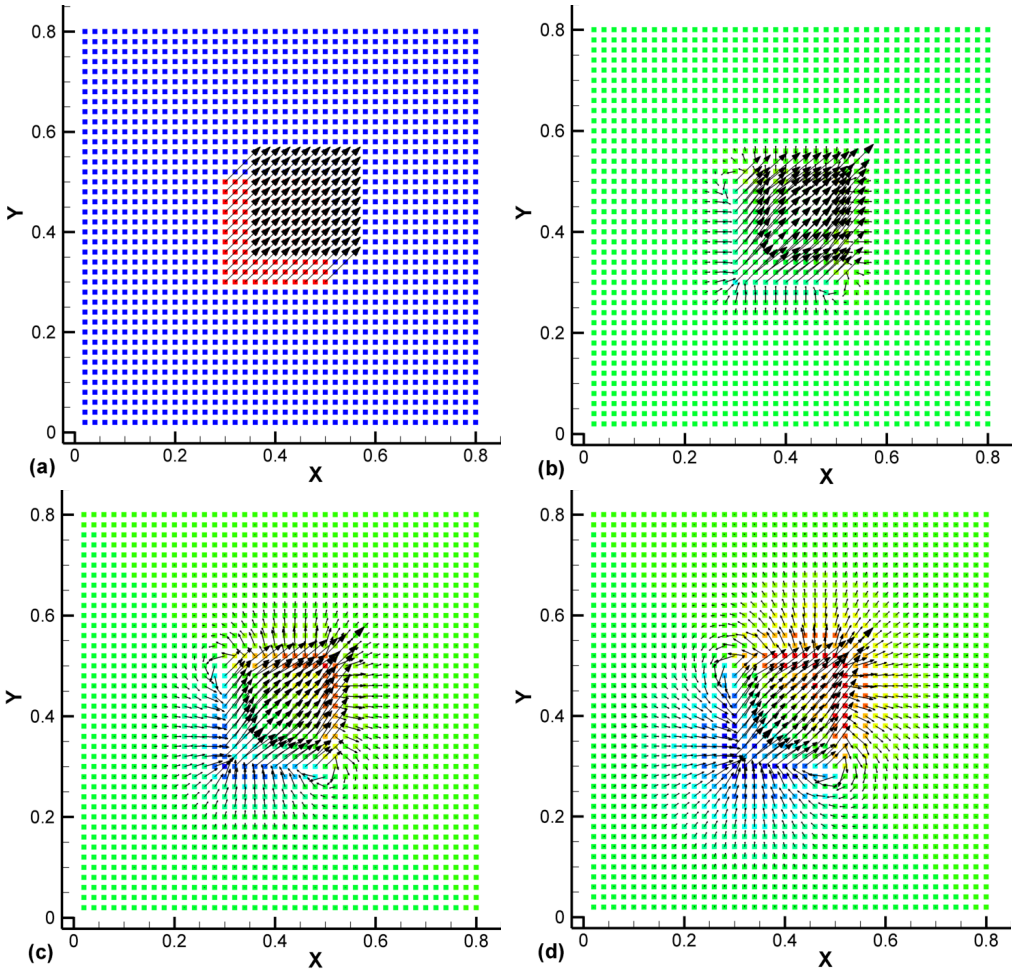


Fig. 3.6. Pressure correction test case: (a) Initial velocity field.
Pressure and velocity vectors at: (b) 1st (c) 5th and (d) 20th corrective iteration

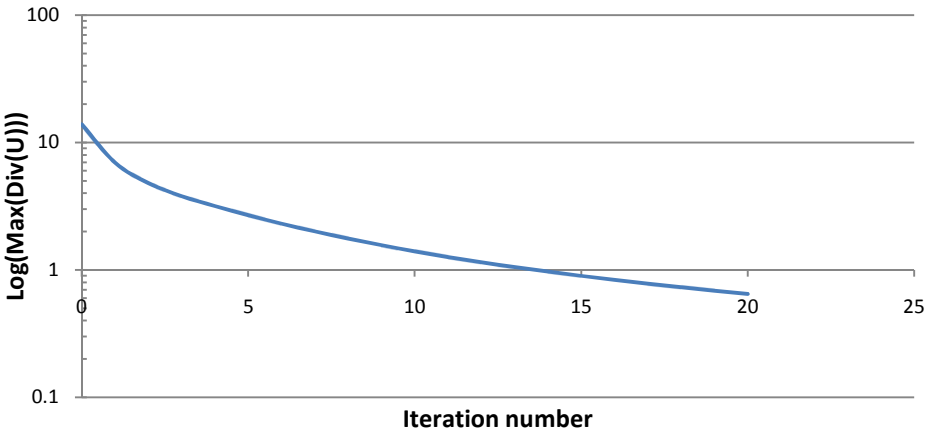


Fig. 3.7. Maximum divergence in respect to iteration number

The disadvantages of this method are:

- Difficult selection of the relaxation parameters, since their values are problem dependent. Careful selection is necessary to avoid numerical instability and convergence within few iterations. Kai Bao et al. [15] proposed using a different equation of state which is less stiff than the Tait equation of state, limiting numerical instabilities. The proposed equation of state is:

$$p = c^2(\rho - \rho_0) \tag{3.64}$$

  and the resulting pressure correction:

$$\delta p_i^{\,n} = c^2 dt \sum_{j=1}^{N} m_j \mathbf{u}_{ij}^{\,*} \cdot \nabla_i W_{ij} \tag{3.65}$$

  Joseph Ha [16] proposed a more general form for the pressure correction in respect to velocity divergence:

$$\delta p = -\gamma(\nabla \cdot \mathbf{u}^*), \quad \text{with } 0 \le \gamma \le dx^2/(4dt) \tag{3.66}$$

- Since the method is based on the propagation of a pressure disturbance throughout the computational domain, this means that the larger the computational domain the more iterations will be required to propagate the disturbance throughout the domain. This is especially problematic in cases where the problem domain has large aspect ratios (one problem dimension is much larger than the others, as for example the flow inside a pipe).
- The pressure correction method does not require the solution of a linear equation system, but it comes at a very high computational price. First a loop must involve all particles to find the intermediate velocity field $\mathbf{u}^*$. Then several iterations have to be made until velocity divergence drops below the specified tolerance; each iteration involves looping all particles to calculate to calculate the pressure correction through velocity divergence (eq. 3.60 or eq. 3.65) and then another loop to calculate the velocity correction (eq. 3.63). In combination with the fact that in realistic cases, geometries are much more complicated than the test case used, this means many iterations and many loops calculating pressure and velocity corrections.
- After the initial iterations, divergence drops very slowly. The quality of the pressure field is inferior to that obtained by solving the pressure Poisson equation (compare fig. 3.5 and fig. 3.6), while requiring much more time to achieve the specified divergence tolerance. Even if the pressure correction procedure can be easily parallelized, its practical application in simulations with many particles does not seem feasible.
- The treatment of boundaries is still problematic, but the free surface does not need any special boundary conditions, due to the influence of the equation of state.

To sum up, truly incompressibility in SPH is possible to be enforced but requires increased computational cost and some of its aspects are not yet fully explored. Thus, due to the issues encountered in both the incompressible treatments for the SPH method, in the rest work the weakly compressible approach will be used.

3.6. Pressure / density field smoothing – XSPH correction

In section 3.4 it was already mentioned that using the Tait equation of state results to the introduction of numerical oscillations in the density and pressure fields. Many SPH practitioners use some sort of filter, in order to smooth out these oscillations [6, 17]. In the present work two different filters were used:

- 0^{th} order (Shepard filter): The Shepard filter is a straightforward correction of the density field which is periodically reinitialized using the weighted average summation density approximation (see eq. 3.22):

$$\rho_i = \sum_j m_j \tilde{W}_{ij} \tag{3.67}$$

where:

$$\tilde{W}_{ij} = \frac{W_{ij}}{\sum_j \frac{m_j}{\rho_j} W_{ij}} \tag{3.68}$$

- 1^{st} order (MLS filter) [18, 19]: The MLS filter was proposed by Dilts, is based on the Moving-Least-Squares method and it is capable of reproducing linear variations of the density field. Similarly with the Shepard filter, density is calculated using a corrected kernel function, as follows:

$$\rho_i = \sum_j m_j \tilde{W}_{MLS} \tag{3.69}$$

In 3D the MLS corrected kernel function is calculated from:

$$\tilde{W}_{ij}^{MLS} = \left[\beta_0 + \beta_x (x_i - x_j) + \beta_y (y_i - y_j) + \beta_z (z_i - z_j)\right] W_{ij} \tag{3.70}$$

Where the $\beta_0, \beta_x, \beta_y, \beta_z$ correction coefficients are calculated by:

$$\begin{bmatrix} \beta_0 \\ \beta_x \\ \beta_y \\ \beta_z \end{bmatrix} = \mathbf{A}^{-1} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.71}$$

The matrix **A** is defined as:

$$\mathbf{A} = \sum_j \frac{m_j}{\rho_j} W_{ij} \tilde{\mathbf{A}} \tag{3.72}$$

$$\tilde{\mathbf{A}} = \begin{bmatrix} 1 & (x_i - x_j) & (y_i - y_j) & (z_i - z_j) \\ (x_i - x_j) & (x_i - x_j)^2 & (x_i - x_j)(y_i - y_j) & (x_i - x_j)(z_i - z_j) \\ (y_i - y_j) & (x_i - x_j)(y_i - y_j) & (y_i - y_j)^2 & (y_i - y_j)(z_i - z_j) \\ (z_i - z_j) & (x_i - x_j)(z_i - z_j) & (y_i - y_j)(z_i - z_j) & (z_i - z_j)^2 \end{bmatrix} \tag{3.73}$$

The inversion of the **A** matrix is relatively simple, since the matrix is 4x4 in 3D, enabling the use of closed form relations for its inversion. However, care has to be taken when a particle is remote (i.e. has few or none neighbors), since in that case the *A* would be very close to zero (singular matrix). Inversion of such a matrix would lead to unphysical corrected kernel values. The disadvantage of that type of filter is the fact that it has increased memory requirements, since the four correction coefficients (β_0, β_x, β_y, β_z) must be stored for all particles prior to applying the filter. Furthermore, in practical applications it does not give significantly better quality in the density field than the Shepard filter.

In weakly compressible applications it is generally beneficial to smooth also the particle motion. This is done using the XSPH correction proposed by Monaghan [1]:

$$\frac{d\mathbf{r}_i}{dt} = \mathbf{u}_i - e\sum_j \frac{m_j}{\rho_j}\mathbf{u}_{ij}W_{ij} \tag{3.74}$$

For nearly incompressible flows the value of the *e* parameter is ~0.3. By observing the previous equation, it is understood that particles moving under the influence of the XSPH correction scheme take into account the contribution of neighboring particles, thus making the particle to move in a velocity closer to the average velocity of the neighboring particles. This technique helps in moving particles more orderly [2], especially in weakly compressible flows, where the stiffness of the equation of state may deteriorate particle distribution.

Another possible option for dealing with the noisy pressure field is to use a diffusion term in the density equation. Such methods have been developed by Antuono et al. [14], using the laplacian of the density field (eq. 2.19) and by Ferrari et al. [20, 21] using a monotone upwind flux, for the density equation. Both methods are similar, however the density diffusion approach by Antuono et al. requires the artificial viscosity term, along with tuning of the artificial parameters, whereas the monotone upwind flux by Ferrari et al. does not require any additional terms or tuning of ad hoc parameters. Here the method of Ferrari et al. will be briefly described.

The key idea is to introduce an extra term in the continuity equation, which will damp the high frequency numerical noise of the pressure and density fields. The additional term was inspired by the Advection Upstream Splitting Method (AUSM) [20]. Eventually the new continuity equation is written as:

$$\frac{D\rho_i}{Dt} = \sum_{j=1}^{N} m_j \left(\mathbf{u}_{ij}\cdot\nabla_i W_{ij} + \mathbf{n}_{ij}\cdot\nabla_i W_{ij}\frac{c_{ij}}{\rho_j}\left(\rho_i - \rho_j\right)\right) \tag{3.75}$$

where:

- $\mathbf{n}_{ij}$ is the unity vector pointing from particle *j* to particle *i,* i.e. $\mathbf{n}_{ij} = \dfrac{\mathbf{r}_{ij}}{\left\|\mathbf{r}_{ij}\right\|}$.

- c_{ij} is the maximum celerity of particle *i* and *j*. Celerity of particle *i* is defined as:

$$c_i = c_0\sqrt{\left(\frac{\rho}{\rho_0}\right)^{\gamma-1}} \tag{3.76}$$

In fig. 3.8, a simple test case is presented, in order to compare the effects of the density smoothing methods already discussed. The test case is a 2D jet impingement on a flat plate under 60° angle. In fig. 3.8 the results of the pressure coefficient distribution ($C_p = 2p/\left(\rho u^2\right)$) are shown for: the standard SPH method without any corrections, the SPH method with the celerity correction (Ferrari et

al. method) and SPH with shepard density filter. The boundary is simulated as a free slip wall, by mirroring nearby particles (boundary conditions will be further discussed later).

From the results (fig. 3.8) it is obvious that, without any corrections, the SPH method is unable to calculate a reasonable pressure field. Density smoothing and the diffusion term are able to provide reasonable pressure fields. Diffusion term formulation produces some scattering on the pressure distribution. On the other hand the density smoothing predicts minor pressure fluctuations on the jet before impingement.

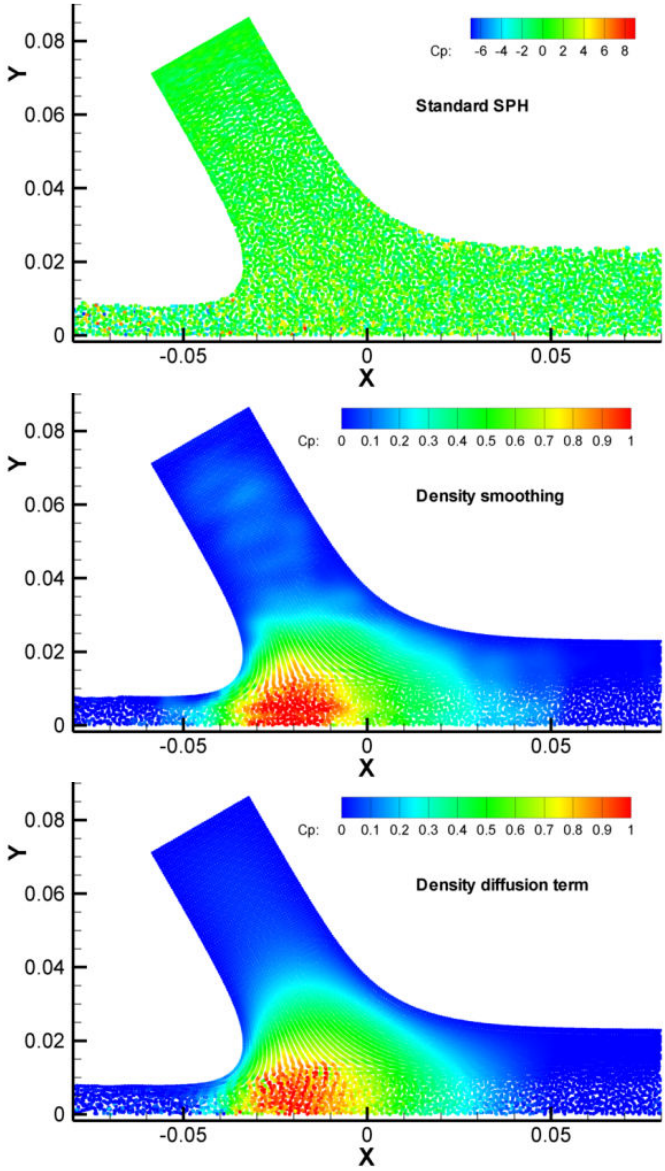


Fig. 3.8. Pressure coefficient distribution for the 2D jet impingement (impingement angle 60^{o}).

It is possible to obtain an exact solution of the pressure distribution on the wall for the described test case. Following Taylor [22]:

$$x=\frac{h}{\pi}\left[\ln\left(\frac{1-q}{1+q}\right)-\cos a\ln\left(\frac{1+q^2-2q\cos a}{1-q^2}\right)-2\sin a\tan^{-1}\left(\frac{q\sin a}{1-q\cos a}\right)\right] \qquad (3.77)$$
$$C_p=(1-q^2)$$

Where:

- x is the x-coordinate on the plate surface
- h is the jet width (here 0.03m)
- a is the impingement angle (here 60^o)
- q is a parameter to be eliminated from eq. 3.77.

Thus, in the following figure the instantaneous pressure distribution of each method is compared to the exact solution.

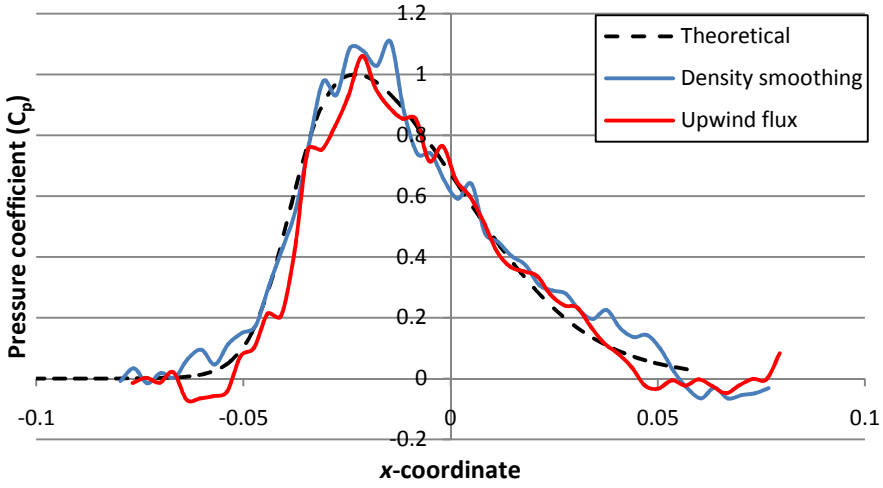


Fig. 3.9. 2d jet impingement, 60^o jet angle. Instantaneous pressure coefficient distribution.

In the present work the 0^{th} order (Shepard) density filter was used for the all SPH simulations. However the density diffusion term is an attractive alternative which is used in the SPH/SPH-R hybrid (see chapter 6).

3.7. Tensile instability / Kernel correction

It is known [23, 24] that the SPH method suffers from an instability that arises in the tensile regime. This problem was first discovered when simulating the behavior of elastic solids. When the particles are under tensile stress state, their motion becomes unstable, eventually leading either to a blow up of the simulation, or to the creation of unphysical particle clumping. Swegle et al. [23] examined this type of instability and proved that it is independent of artificial viscosity or of the time integration scheme and time step, but rather it is caused by a combination of the stress acting on particles and the sign of the second derivative of the kernel function W. Swegle proved using one dimensional Neumann analysis, that the condition of the tensile instability is (see also fig. 3.10):

$$W''\sigma^{aa} > 0$$

Note that the above condition shows that instability may arise even in cases of compression, if the second derivative of the kernel function has the appropriate sign. However, it is more common this instability to appear in cases of tension, since the range at which the instability in compression occurs is often less than the inter-particle distance, depending on the kernel function. This is the reason why this issue is encountered more in cases with tension and that is why it was called tensile instability.

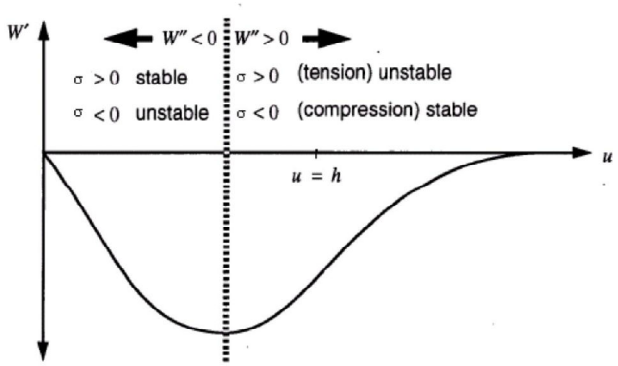


Fig. 3.10. Stability and instability regions for tension and compression.

Tensile instability may appear even in gas/fluid dynamics, even when pressure is positive [24, 25]. In order to treat the tensile instability there are several ways:

- Use of appropriate kernel functions. Using a kernel function with a second derivative equal to zero would result to stability in both compression and tension. Unfortunately such a kernel function cannot go to zero smoothly enough, thus being very sensitive to particle disorder.
- Incorporation of a small repulsive force inside the momentum equation [24]. Monaghan proposed adding a contribution in the pressure gradient of the momentum equation as follows:

$$\frac{D\mathbf{u}_i}{Dt} = -\sum_j m_j \left[\left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} + R\, f_{ij} \right) \nabla W_{ij} \right] + \mathbf{f}_{body} \tag{3.78}$$

$$\text{where: } R = R_i + R_j$$

$$\text{and } R_i = \max\left(-\varepsilon \frac{p_i}{\rho_i^2}, 0 \right)$$

The parameter ε is usually set to ~0.2. The f_{ij} parameter is calculated as the fraction of the kernel function W between particle i and particle j and the kernel function value for a distance equal to the particle spacing dx:

$$f_{ij} = \frac{W_{ij}}{W(dx)} \tag{3.79}$$

A similar treatment may treat particle clumping at characteristic directions. As it was already discussed in the chapter covering the basics of SPH (chapter 2), regarding the accuracy of the SPH particle approximations, particles tend to move in an aligned manner, compressing in one direction and expanding in another. A possible solution is to modify eq. 3.78, in case pressure is positive. In that case:

$$R = 0.01 \left[\max\left(0, \frac{p_i}{\rho_i^2} \right) + \max\left(0, \frac{p_j}{\rho_j^2} \right) \right] \tag{3.80}$$

The small repulsive force induced in this way may help to avoid clumping in some situations, without adding a considerable unphysical contribution in the simulation (according to Monaghan [24], the influence of this force is less than 1%).

- Kernel gradient correction. Another available possibility to treat the inherent tensile instability of the SPH method, without the addition of unnatural forces as Monaghan's solution, is the correction of the kernel function itself. The correction proposed here is based on the renormalization procedure which is quite popular among many SPH researchers [6, 26, 27, 28]. A detailed description of the renormalization procedure is in the work of Oger et al. [26]. The main idea is the formulation of SPH discrete approximations which will be capable of reproducing exact gradient interpolations of linear fields. To do so, one must enforce the following condition (in 3D):

$$\begin{pmatrix} \nabla x \\ \nabla y \\ \nabla z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.81}$$

This condition, using SPH approximation, can be written as:

$$\sum_j \left(\mathbf{r}_j - \mathbf{r}_i\right) \otimes \nabla W_{ij} \frac{m_j}{\rho_j} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.82}$$

The kernel function gradient ∇W_{ij} is replaced by the corrected value $\nabla \tilde{W}_{ij}$ calculated as follows:

$$\nabla \tilde{W}_{ij} = \frac{1}{2}\left(\mathbf{L}_i + \mathbf{L}_j\right) \nabla W_{ij} \text{ , or} \tag{3.83}$$

$$\begin{bmatrix} d\tilde{W}/dx \\ d\tilde{W}/dy \\ d\tilde{W}/dz \end{bmatrix} = \begin{bmatrix} \bar{L}_{11} & \bar{L}_{12} & \bar{L}_{13} \\ \bar{L}_{21} & \bar{L}_{22} & \bar{L}_{23} \\ \bar{L}_{31} & \bar{L}_{32} & \bar{L}_{33} \end{bmatrix} \begin{bmatrix} dW/dx \\ dW/dy \\ dW/dz \end{bmatrix}$$

where:

$$\mathbf{L}_i = \mathbf{M}^{-1} \tag{3.84}$$

$$\mathbf{M} = \begin{bmatrix} \sum_j \frac{m_j}{\rho_j}\left(x_j - x_i\right)\frac{dW}{dx} & \sum_j \frac{m_j}{\rho_j}\left(x_j - x_i\right)\frac{dW}{dy} & \sum_j \frac{m_j}{\rho_j}\left(x_j - x_i\right)\frac{dW}{dz} \\ \sum_j \frac{m_j}{\rho_j}\left(y_j - y_i\right)\frac{dW}{dx} & \sum_j \frac{m_j}{\rho_j}\left(y_j - y_i\right)\frac{dW}{dy} & \sum_j \frac{m_j}{\rho_j}\left(y_j - y_i\right)\frac{dW}{dz} \\ \sum_j \frac{m_j}{\rho_j}\left(z_j - z_i\right)\frac{dW}{dx} & \sum_j \frac{m_j}{\rho_j}\left(z_j - z_i\right)\frac{dW}{dy} & \sum_j \frac{m_j}{\rho_j}\left(z_j - z_i\right)\frac{dW}{dz} \end{bmatrix} \tag{3.85}$$

The **M** matrix is symmetric and can be easily inverted using closed form relations. As with the MLS density filter, care must be taken in case particles are remote, since in that case the **M** matrix would be singular. Another important characteristic of the correction matrix is that in the interior of a regular particle lattice, the corrected gradient and the gradient would be equal (i.e. the **L** correction matrix would be equal to the identity matrix). However when a particle is located near the free surface, or near a boundary, or when the particle distribution is not uniform, then the appropriate correction is applied (see fig. 3.11). It is highlighted here that in order to preserve symmetric interactions between particles, the average value of the correction matrix of both particles is used to correct the gradient (eq. 3.83). The kernel correction described above corrects the tensile instability and also offers greater accuracy in cases of non-uniform particle distribution.

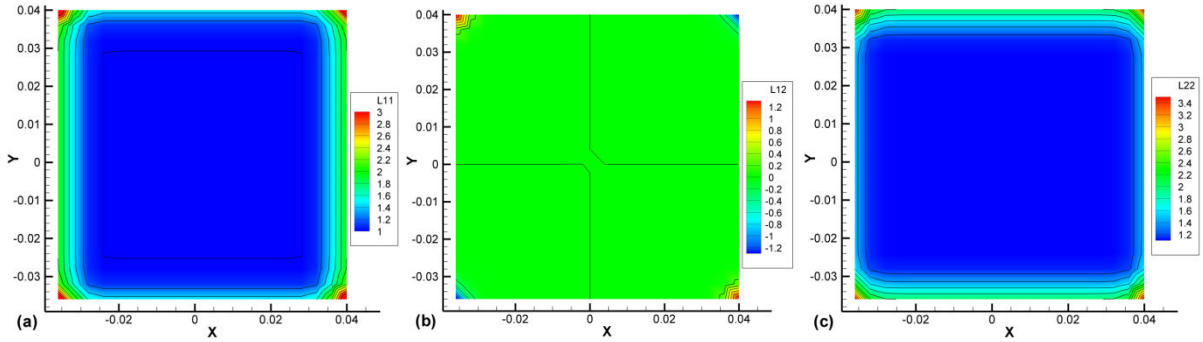


Fig. 3.11. Distribution of the corrective matrix. Elements: (a) L_{11} (b) L_{12}, L_{21}, (c) L_{22}. Note that inside the computational domain L_{11}=L_{22}=1 and L_{12}=L_{21}=0.

3.8. Boundary conditions

The main weakness of the SPH method is the implementation of boundary conditions. The reason is that initially the SPH method was developed for the simulation of unbounded systems. The problem that arises when dealing with boundaries is better understood if one considers fig. 3.12; the particle *i* is located near the boundary surface *S*. The particle interacts with particles within the kernel function support radius from inside the problem domain *Ω*. However, a part of the kernel function support domain is truncated at the boundary – which in the figure is the highlighted part. For particles near or on the boundary, only particles from the inside of the computational domain contribute to the SPH summation approximations and no contribution comes from outside since there are no particles beyond the boundary. This greatly affects the accuracy of the SPH approximations.

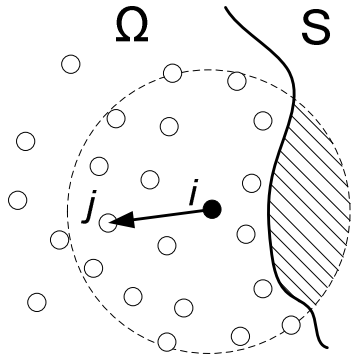


Fig. 3.12. Boundary deficiency problem

In the SPH literature there are various proposed ways to compensate the described problem, some of them will be presented here, focusing mainly on the case of wall boundary conditions (see also fig. 3.13):

- Mirror particles: these particles are created by mirroring the actual particles from within the problem domain across the boundary surface. In the case of describing walls, the mirrored and the actual particles have opposite normal, to the boundary, velocity and equal parallel velocity (in case the wall is treated as free-slip – for no slip case the parallel velocity should be the opposite). Other boundary conditions may be treated by adjusting properly the velocity of the mirror particles and the other field variables values. The main issue of this treatment is that it

is impractical for complicated geometries. It is also highlighted that when using the standard SPH this treatment may not be enough; for example describing a wall with mirror particles only, might not ensure wall impermeability, thus requiring additional treatments such as boundary forces. This boundary type is mainly used for defining symmetry boundary conditions.

- Dynamic particles: the particles are positioned forming layers across boundaries and remain fixed in space, or move according to a prescribed manner. However, their density and pressure change in time according to the continuity equation [6]. Particles may be placed in various configurations: in aligned layers, as in fig. 3.13, or in a staggered manner [6]. This particular boundary ensures impermeability of walls. It has the disadvantage that fluid particles tend to stick on the wall, rather than draining of completely [29] (see also fig. 3.19). Another possible variation of this particular type of boundary condition is to mirror the dynamic particles inside the computational domain and use the mirrors to interpolate field variables. Then, the interpolated values are used to impose the values of the field variables on the dynamic particles (fig. 3.14).

- Boundary forces: this boundary type is only applicable for walls. It was firstly proposed by Monaghan [1], to place a layer of particles on a boundary. These particles would exert a repulsive force to any fluid particle moving close enough to the boundary. This boundary force would be added in the RHS of the momentum equation as a body force. Initially Monaghan proposed to model this force using a Lennard-Jones potential [30]. Later on the approach of boundary forces was further refined, using a denser particle distribution for boundary particles to make the repulsive field smoother [31, 32]. The Lennard-Jones boundary force, acting from a boundary particle *j* towards a fluid particle *i*, is calculated as:

$$\mathbf{f}_i \ \left[m/s^2\right] = \begin{cases} D\left[\left(\dfrac{r_0}{\left\|\mathbf{r}_{ij}\right\|}\right)^{12} - \left(\dfrac{r_0}{\left\|\mathbf{r}_{ij}\right\|}\right)^{6}\right]\dfrac{\mathbf{r}_{ij}}{\left\|\mathbf{r}_{ij}\right\|^2} & \dfrac{r_0}{\left\|\mathbf{r}_{ij}\right\|} \le 1 \\ 0 & \dfrac{r_0}{\left\|\mathbf{r}_{ij}\right\|} > 1 \end{cases} \tag{3.86}$$

D is a problem dependent parameter, which is usually set at the same order of magnitude of the square of the maximum velocity appearing in the problem. Also, r_0 is the repulsive force cut-off distance. It is usually set equal to the particle spacing *dx*. Using the previous relation, it is possible to obtain the forces acting on a boundary particle, summing all the particle reactions on the boundary particle:

$$\mathbf{f}_i = -\sum_j m_j D\left[\left(\frac{r_0}{\left\|\mathbf{r}_{ij}\right\|}\right)^{12} - \left(\frac{r_0}{\left\|\mathbf{r}_{ij}\right\|}\right)^{6}\right]\frac{\mathbf{r}_{ij}}{\left\|\mathbf{r}_{ij}\right\|^2} \quad [N] \tag{3.87}$$

Here *i* denotes the boundary particle and *j* the fluid particles interacting with the specific boundary particle *i*. By summing the forces acting on all boundary particles, it is possible to obtain the total force or torque acting on a boundary:

$$\mathbf{F}_{boundary} = \sum_{i \in boundary} \mathbf{f}_i \quad [N] \tag{3.88}$$

$$\mathbf{T}_{boundary} = \sum_{i \in boundary} \mathbf{r} \times \mathbf{f}_i \quad [Nm] \tag{3.89}$$

The advantage of this particular boundary type is that it is simple to implement, robust, since it does not permit wall penetration and it is also able to describe complex geometries. The main disadvantage is that the calculated forces exhibit oscillations, as it will be shown later. Also, SPH interpolations are not so accurate near the boundary.

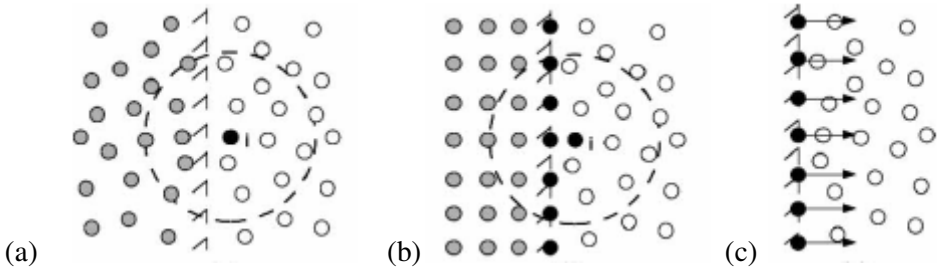


Fig. 3.13. Boundary conditions: (a) mirror particles, (b) dynamic particles, (c) boundary forces

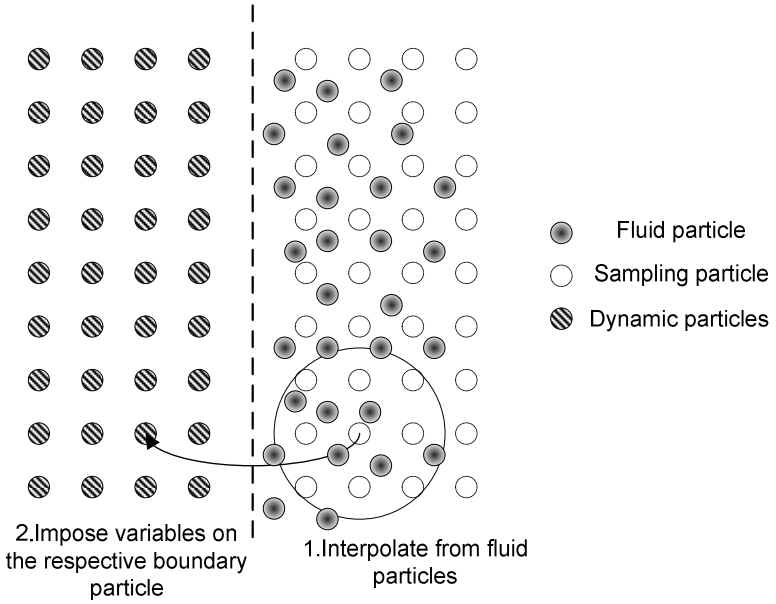


Fig. 3.14. Dynamic boundary particles using interpolation

To conclude this part, a small test case is presented: this test case involves a very popular benchmark widely used among the SPH practitioners, the dam break. The dam break test case consists of a water column which collapses under the influence of gravity inside a tank (a simple schematic is shown in fig. 3.15). In this test, results will only concern the propagation of the generated wave and the height of the water column in respect to time and will be compared with the experimental results of Koshizuka et al. [33] and the SPH literature [3]. The three different boundary types were used, comparing the results and showing their advantages / disadvantages. In fig. 3.16 and 3.17 the results shown are from the mirror particles boundary conditions; however the rest boundary treatments yield identical results.

From the results, it is obvious that all boundary treatments give the same general flow field; all cases agree with the experimental results and the results in SPH literature. Mirror particles are the most accurate way for simulating boundaries, but are especially difficult to implement in areas such as corners. Dynamic boundary conditions have issues due to their sticky behavior (fig. 3.19), but ensure impermeability. The boundary forces are easy to implement and do not have the issues encountered by the dynamic boundaries.

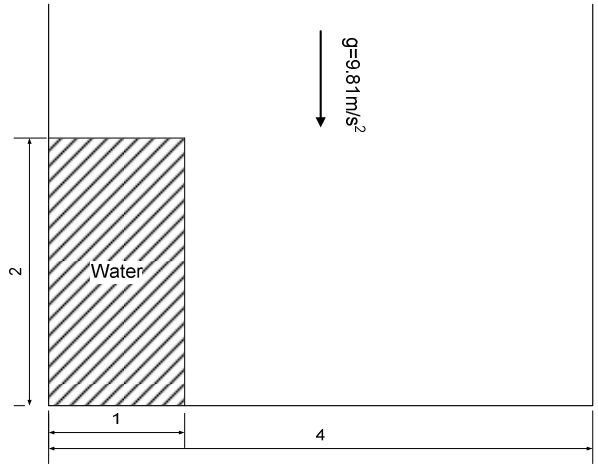


Fig. 3.15 Dam break case schematic. All dimensions are in meters.

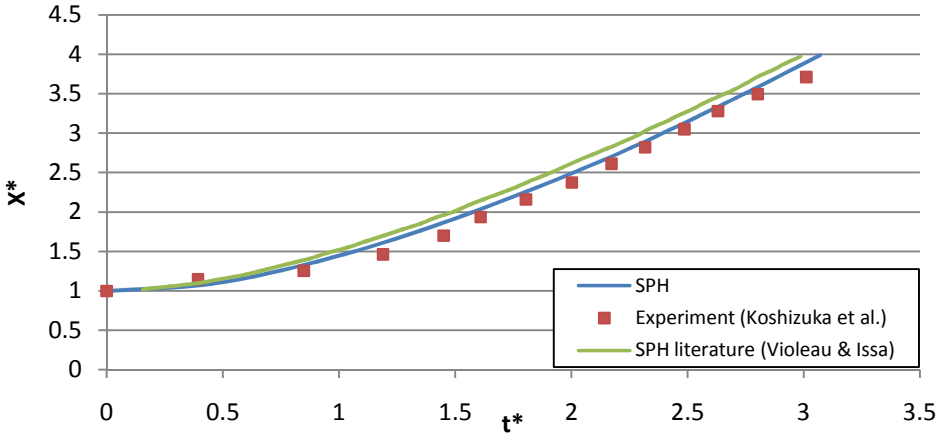


Fig. 3.16. Dam break problem. Non-dimensional wave front propagation in respect to time

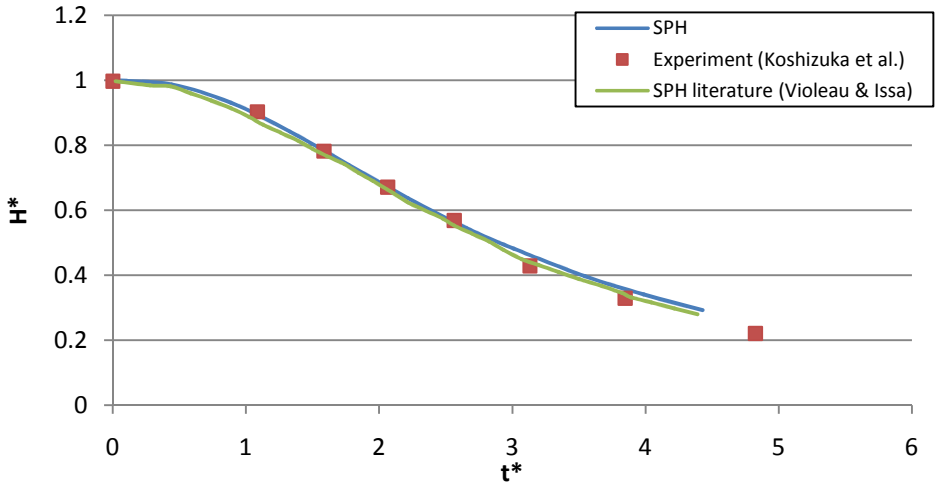


Fig. 3.17. Dam break problem. Non-dimensional column height evolution in respect to time

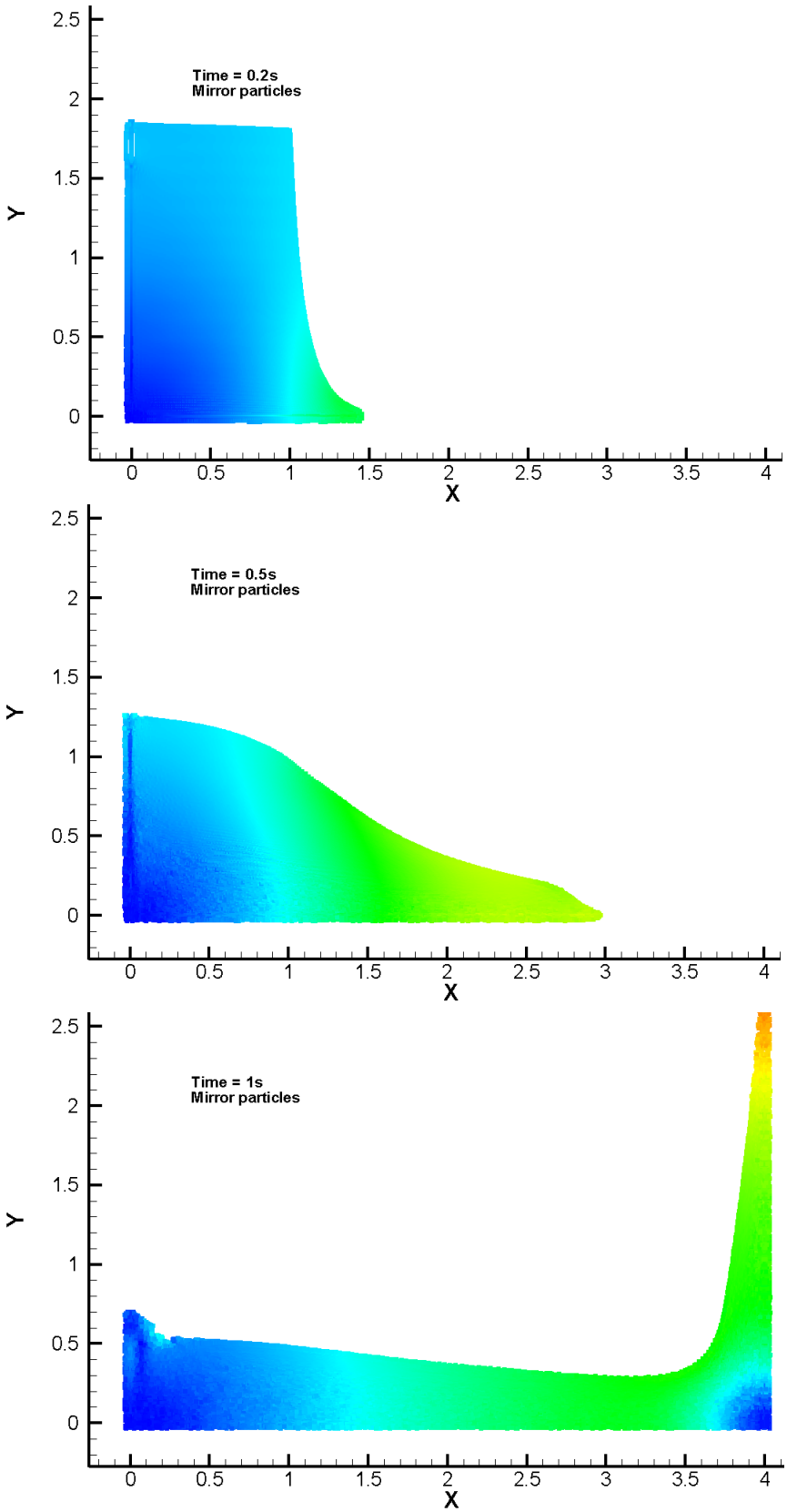


Fig. 3.18. Dam break case simulated with mirror particles.

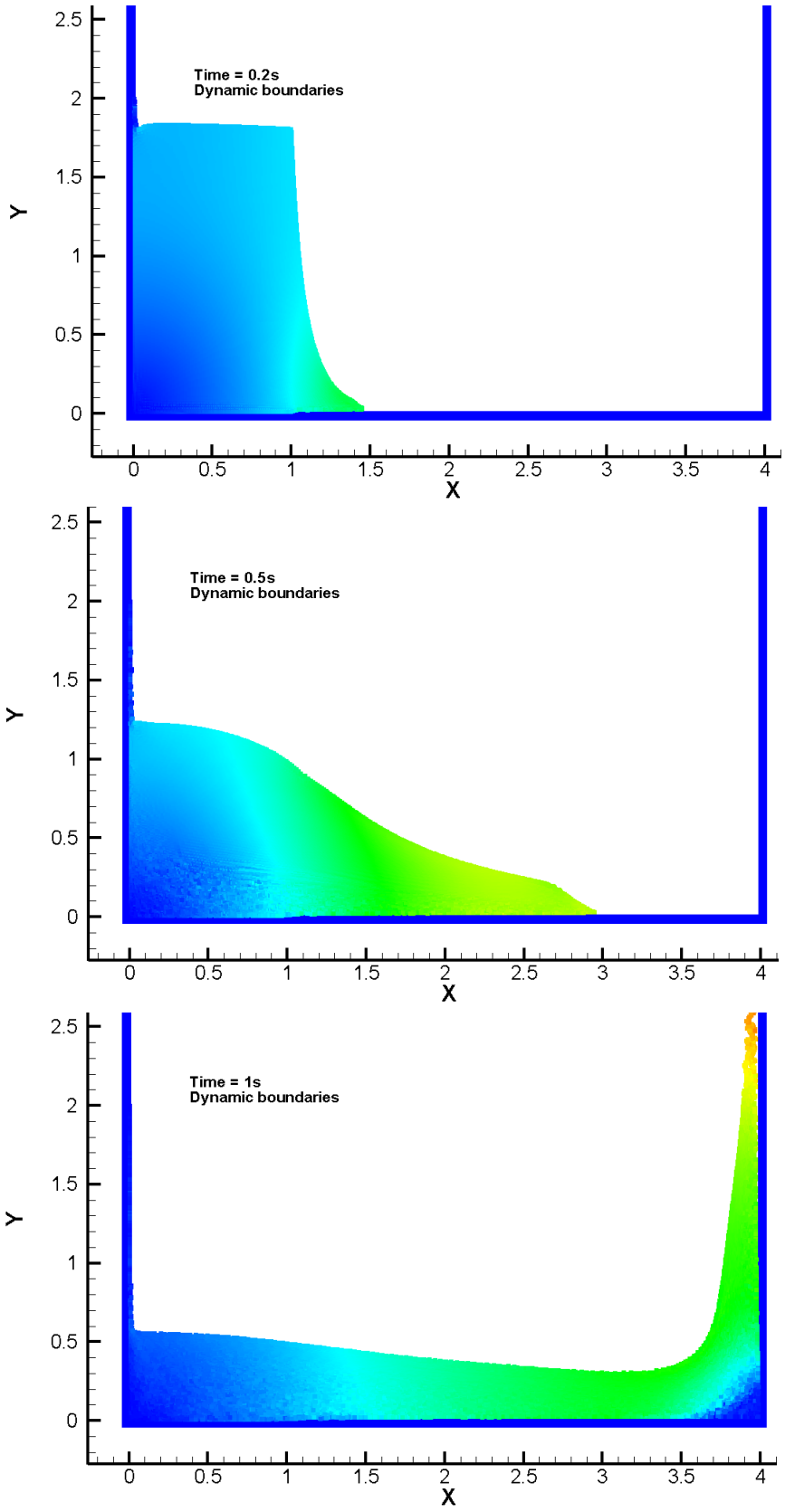


Fig. 3.19. Dam break case simulated with dynamic boundaries.

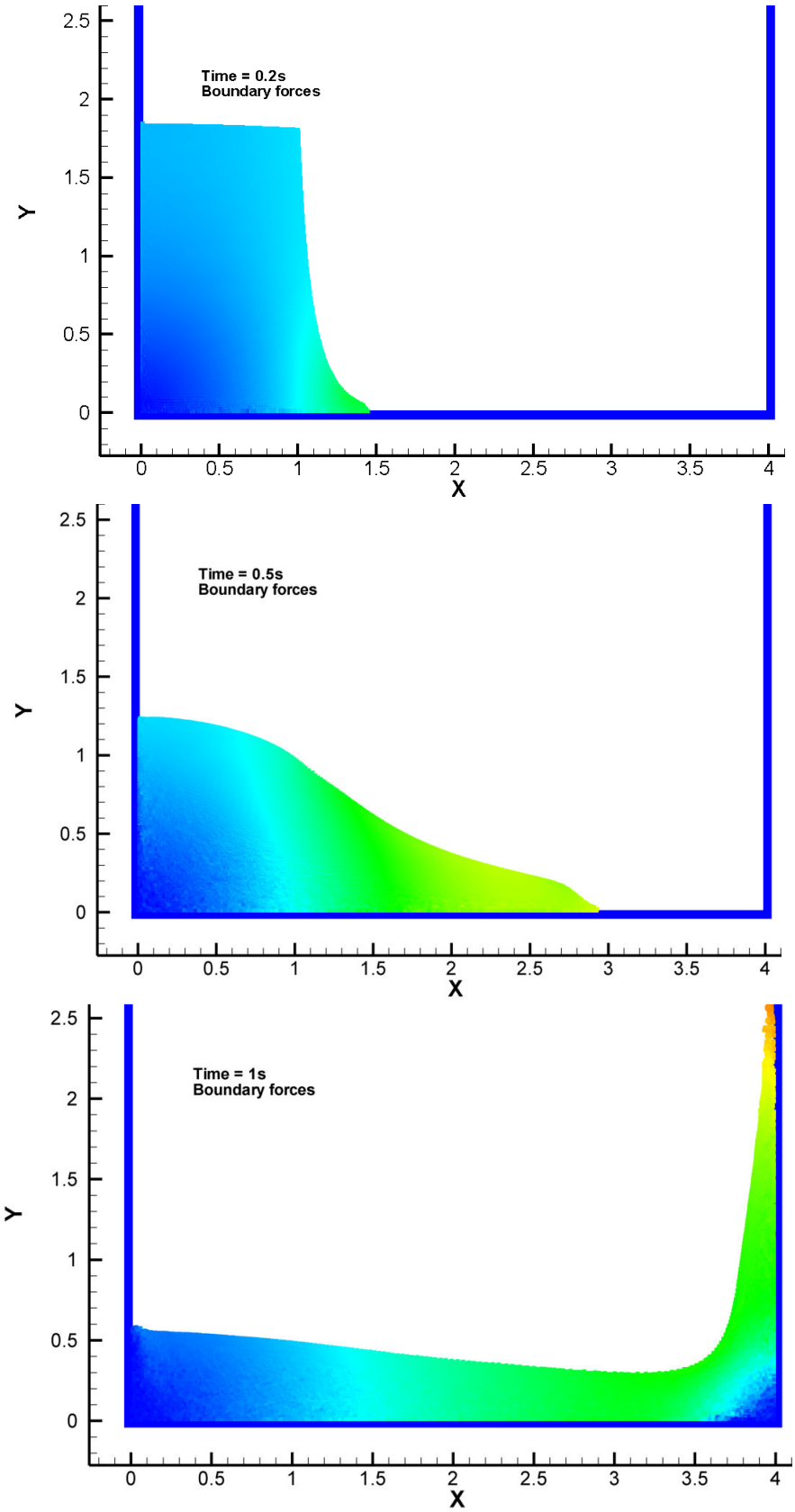


Fig. 3.20. Dam break case simulated with boundary forces.

Here it is mentioned that:

- Non-dimensional wave front: $X^* = X/W_c$, W_c being the initial water column width (in this case 1m)
- Non-dimensional column height: $H^* = h/H_c$, H_c is the initial column height (which is 2m)
- Non-dimensional time: $t^* = t\sqrt{2g/W_c}$ [33]

The subject of boundary conditions for the SPH method is rather subtle and complicated; indeed from what was described before, it is obvious that there is not a universal treatment which is effective for all boundary conditions. There is on-going research on the subject by many SPH researchers [34, 35, 36]. Vila has proposed a change in the perspective of the SPH method which would allow a much better boundary treatment. This will be further analyzed on the chapter discussing SPH-ALE.

3.9. Neighbor search algorithm

From the particle SPH approximations it is obvious that in order to estimate the summations, one must calculate all particle interactions. A straightforward way to do this would be to directly consider all particle interactions, i.e. if a simulation involves *N* particles then in order to find the SPH approximations for the *i* particle then all *N* particles should be considered. Thus, for each time step a total of *N*x*N* particle estimations should be evaluated. This method, despite of being simple and easy to program, is rather computationally inefficient since it considers many interactions which have zero influence due to the kernel function being compact.

An alternative way of finding the particle interactions would be by exploiting the compact support of the kernel function and considering the interactions only in a close neighborhood of the examined particle *i*. To do this, the computational domain is divided into cells and particles are assigned to each cell. It is highlighted here that these cells are used only for indexing purposes and not for calculations. A neighbor list is created linking each cell to the particles which it contains and also linking the particles to the cell to which they belong. The cell size is selected to be equal to the support radius of the kernel function. Thus, if it is desired to find the neighbor contributions of particle *i*, then it is sufficient to search the neighbors only at the adjacent cells to particle *i* cell (see also fig. 3.21 the highlighted cells). In this way, it is possible to reduce the computational load from *N*x*N* to *N*xlog(*N*). In 2D, one should search for neighbors in 9 cells totally and in 3D in 27 cells. The neighbor list algorithm is based on the static matrix algorithm [37]. In the SPH literature, there are also other more refined/complicated ways of forming the neighbor list [37] and other methods for finding neighbors (for example tree search algorithms [2]).

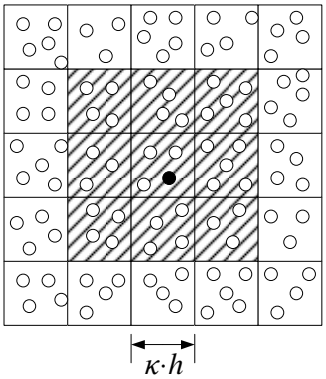


Fig. 3.21. Background grid required for the neighbor list search algorithm.

3.10. Parallelization

The SPH method is able to simulate problems with free-surfaces, deformable boundaries, moving interfaces, wave propagation and solid deformation. However all these application require excessive computational time, if a three dimensional simulation has to be carried out. In practical applications millions of particles may be required to properly describe the underlying phenomena, resulting to poor performance when the algorithm is implemented to execute on a single CPU.

One interesting feature of the SPH method is the fact that the particle interactions are independent for each particle. If properly programmed it is possible to calculate at the same time the particle interactions of several particles. The SPH method is said to be "embarrassingly parallel" due to the aforementioned feature. Implementation of the SPH method using parallelization techniques is a subject of ongoing research, regarding the best implementation techniques, in order to increase parallel efficiency [38].

There are three main ways to parallelize the SPH algorithm:

- *Shared memory parallel using OpenMP* (*Open Multi-Processing*). OpenMP is an API (*Application Programming Interface*) that supports shared memory multiprocessing [39]. The algorithm is divided in serial and parallel regions. In serial regions, one thread (the master thread) is created for the algorithm execution. When it reaches a parallel region, the process spawns more threads, according to the OpenMP directives and all processes are executed simultaneously on different CPU cores, sharing the same memory. Due to the shared memory, it is relatively easy to transform a serial SPH algorithm to parallel. However, care must be taken in order to avoid data race conditions; such cases occur when different threads access the same memory positions. Due to its nature, OpenMP may be only used for programs running on a single multicore machine. This type of parallelization was employed for the present SPH implementations and more information may be found at Appendix C.

- *Distributed parallel using MPI* (*Message Passing Interface*) system, such as OpenMPI. Message Passing Interface is a standardized and portable interface to facilitate communication between processes running on different computers [40]. MPI enables point-to-point or collective communication and process synchronization for organizing parallel processing. This type of parallel processing is called as Distributed Memory Multiprocessing, since each process has its own private memory which is inaccessible by the other processes. Processes

may be spawned on different machines and communication is done through a network (Ethernet, Myrinet, Infiniband), or on a single machine utilizing the available CPU cores; in that case MPI is not as efficient as OpenMP. MPI may be used in conjunction to OpenMP; MPI would facilitate communication between the processes spawned on different machines and OpenMP is used to create threads for each process, in order to cover the number of available cores on each machine. Efficient parallelization with MPI is done by decomposing the problem domain and distributing it on the processes. Information of particles near the boundaries of the distributed domains must be transferred between processes. Transformation of a serial code into distributed parallel using MPI, requires considerable changes, in order to organize the communication of processes and the decomposition. These are still a subject of research, since different decomposition methods and different ways of communication (blocking, non-blocking) may achieve greater parallel efficiencies. However, MPI does not have any limit on how many computers and how many CPU cores will be used, enabling the simulation of difficult and computationally intensive problems; Moulinec et al. have managed to perform simulations involving over a hundred million particles using 8192 CPU cores with MPI [41].

- Utilization of *GPGPUs* (*General Purpose Graphics Processing Units*) with or without CPU collaboration. General Purpose GPUs are able to be programmed to perform tasks beyond computer graphics, involving tasks which are traditionally handled by CPUs. GPUs may be considered as stream processors, i.e. processors which operate in parallel by running a single kernel on many records in a stream at once. GPGPUs are much cheaper than cluster computers, linked using MPI, and may be more efficient, since they do not suffer the data transfer latencies due to network traffic. The parallel philosophy of these devices enables to achieve 100x or even larger speed-ups comparing to the execution on a multicore CPU [42]. However, writing an algorithm which is suitable and efficient for execution on a GPU is not a trivial task and requires an entirely different programming philosophy from programming CPUs. The main idea is to transfer data between CPU (called also as host) and the GPU (called also as compute device [43]) and use the GPU for the computationally intensive areas. This programming layout suffers from the latency due to data transfer, which is considerable, even if transfers are much faster than in the case of MPI, due to the computer internal bus speed. Efforts are mainly focusing on developing SPH algorithms running entirely on the GPU, thus eliminating or minimizing data transfers. Generally GPUs are not as much flexible as CPUs and they have their own limitations. Nevertheless, recent GPUs have different features and more capabilities from older models.

References

[1] J.J. Monaghan, “Smoothed Particle Hydrodynamics”, Reports on progress in physics, vol. 68, p. 1703-1759, 2005, doi: 10.1088/0034-4885/68/8/R01

[2] G.R. Liu, M. B. Liu, “Smoothed particle hydrodynamics”, World scientific publishing, 2003, ISBN 981-238-456-1

[3] D. Violeau, R. Issa, “Numerical modeling of complex turbulent free surface flows with the SPH method: an overview”, International Journal of Numerical Methods in Fluids, vol. 53, p. 277-304, 6 July 2006.

[4] J. P. Morris, P. J. Fox, Y. Zhu, “Modeling low Reynolds number incompressible flows using SPH”, Journal of Computational Physics, vol. 136, p. 214-226, 1997.

[5] F. Jiang, M.S.A. Oliveira, A.C.M. Sousa, "SPH simulation of low Reynolds number planar shear flow and heat convection", Materials science and engineering technology, vol. 36, issue 10, p. 613-619, October 2005, DOI: 10.1002/mawe.200500921.

[6] M. Gomez-Gesteira, B.D. Rogers, R.A. Dalrymple, A.J.C. Crespo, M. Narayanaswamy "Users guide for the SPHysics code", http://wiki.manchester.ac.uk/sphysics/

[7] M.J. Ivings, D.M. Causon, E.F.Toro, "On Riemann solvers for compressible liquids", International Journal for Numerical Methods in Fluids, vol. 28, p.395 – 418, 1998.

[8] A.J. Chorin, "The numerical Solution of the Navier-Stokes Equations", Mathematics of computation, vol. 22, p. 745–762, Oct. 1968

[9] G. Ramos-Becerra, C. Moulinec, D.R. Emerson, X.J. Gu, "Inlet-outlet boundary conditions and truly incompressible SPH", Proceedings of the 4th Spheric workshop, Nantes, France 2009.

[10] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, "Numerical recipes in Fortran 77", Cambridge university press, Volume 1.

[11] K. Giannakoglou, J. Anagnostopoulos, G. Bergeles, "Numerical analysis for engineers", 3rd edition, NTUA publishing, Athens 2003.

[12] Lectures web server for the school of mechanical engineering: http://courseware.mech.ntua.gr/ml25273/

[13] O. Shipilova, A. Bockmann, G. Skeie, P. Bergan, "Assesment of incompressible and weakly compressible SPH for marine applications", Proceedings of the 4th Spheric workshop, Nantes, France 2009.

[14] M. Antuono, A. Colagrossi, b, S. Marrone, D. Molteni, "Free-surface flows solved by means of SPH schemes with numerical diffusive terms", Comput. Phys. Commun., Vol. 181, Issue 3, March 2010, Pages 532-549.

[15] K.Bao, H. Zhang, L. Zheng, E. Wu, "Pressure corrected SPH for fluid animation", Computer animation and virtual worlds 2009, vol. 20, p. 311-320, 13 May 2009, DOI: 10.1002/cav.299.

[16] J. Ha, "Applications of generalized Smoothed Particle Hydrodynamics to benchmark CFD problems", 2nd Spheric workshop, Madrid, Spain, 2007.

[17] J. P. Hughes, C. Carne, P. James, D. Graham, "SPH modeling of viscous and non-Newtonian sloshing", Proceedings of the 4th Spheric workshop, Nantes, France 2009.

[18] G. Dilts, "Moving-Least-Squares particle hydrodynamics – I. consistency and stability", International Journal for Numerical Methods in Engineering, Volume 44, Issue 8, 15 Feb 1999.

[19] A. Colagrossi, M. Landrini, "Numerical simulation of interfacial flows by smoothed particle hydrodynamics", Journal of Computational Physics, Volume 191, p 448-475, 2003.

[20] A. Ferrari, M. Dumbser, E.F. Toro and A. Armanini, "A New 3D Parallel SPH scheme for Free Surface Flows" Computers & Fluids, vol. 38, issue 6, pages 1203-1217, 2009.

[21] A. Ferrari, L. Fraccarollo, M. Dumbser, E.F. Toro and A. Armanini, "Three-Dimensional Flow Evolution after a Dambreak" Journal of Fluid Mechanics, vol. 663, pages 456-477, 2010.

[22] Taylor G., "Oblique Impact of a Jet on a Plane Surface", Philosophical transactions for the Royal society of London. Series A, Mathematical and Physical sciences, vol. 260 (1966), issue 1110, p. 96-100.

[23] J.W. Swegle, D.L. Hicks, S.W. Attaway, "Smoothed particle hydrodynamics stability analysis", Journal of Computational Physics, vol. 116, p. 123-134, 1995.

[24] J.J. Monaghan, "SPH without a tensile instability", Journal of Computational Physics, vol. 159, p. 290-311, 2000.

[25] Y. Melean, L. Sigalotti, A. Hasmy, “On the SPH tensile instability in forming viscous liquid drops”, Computer physics communications, vol. 157, p.191-200, 2004.

[26] G. Oger, M. Doring, B. Alessandrini, P. Ferrant, “An improved SPH method: towards higher order convergence”, Journal of Computational Physics, vol. 225, p. 1472-1492, 2007.

[27] N. Grenier, D. Le Touze, P. Ferrant, J.-P. Vila, “Two phase simulations using a volume fraction SPH scheme with a Riemann solver”, 3rd Spheric workshop, Lausanne, Switzerland 2008.

[28] J. P. Vila, “On weighted methods and smooth particle hydrodynamics”, Mathematical models and methods in applied sciences, vol. 9, issue 2, p. 161-209, March 1999, DOI: 10.1142/S0218202599000117

[29] R.A. Dalrymple, “Smooth Particle Hydrodynamics for Surf Zone Waves”, Annual report John Hopkins University Baltimore MD Department of civil engineering, 2007.

[30] J. J. Monaghan, “Simulating free surface flows with SPH”, Journal of Computational Physics, vol. 110, p.399-406, 1994.

[31] J.J. Monaghan, J. Kajtar “SPH boundary forces”, Proceedings of the 4th Spheric workshop, Nantes, France 2009.

[32] J.J. Monaghan, J. Kajtar, “SPH simulations of fish-like swimers”, Proceedings of the 5th Spheric workshop, Manchester, UK 2010.

[33] M. Kondo, S. Koshizuka, “Improvement of stability in moving particle semi implicit method”, International Journal of Numerical Methods in Fluids, vol. 65, p. 638-654, 2010 DOI: 10.1002/fld.2207.

[34] A. Lenhart, F. Fleissner, P. Eberhand, “An alternative approach to modeling complex Smoothed Particle Hydrodynamics boundaries”, Proceedings of the 5th Spheric Workshop, Manchester, UK, 2010.

[35] M. Ferrand, D. Laurence, B.D. Rogers, D. Violeau, “Improved time scheme integration approach for dealing with semi analytical boundary conditions in SPARTACUS2D”, Proceedings of the 5th Spheric Workshop, Manchester, UK, 2010.

[36] M. Shadloo, A. Zainali, M. Yildiz, “Improved solid boundary treatment method for the solution of flow over an airfoil and square obstacle by SPH method”, Proceedings of the 5th Spheric Workshop, Manchester, UK, 2010.

[37] J. M. Dominiguez, A. Crespo, “Improvements on SPH neighbour list”, Proceedings of the 4th Spheric workshop, Nantes, France, 2009.

[38] C. Ulirch, T. Rung, “Massively-Parallel SPH-simulations of viscous flows”, Proceedings of the 4th Spheric workshop, Nantes, France, 2009.

[39] B. Chapman, G. Jost, R. Pas, “Using OpenMP : Portable Shared Memory Parallel Programming”, MIT Press, ISBN-10:0-262-53302-2, ISBN-13: 978-0-262-53302-7

[40] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, J. Dongarra, “MPI: the complete reference”, MIT Press, ISBN-10: 0262692163, ISBN-13: 978-0262692168

[41] C. Moulinec, R. Issa, D.R. Emmerson, X.J. Gu, “High-Performance computing SPH:towards a hundred million particle simulation”, 3rd Spheric workshop, Lausanne, Switzerland 2008.

[42] A.J.C. Crespo, E. Parkinson, J.C. Marongiu, M. Gomez-Gesteira, “High performance of SPH codes: Best approaches for efficient parallelization on GPU computing”, Proceedings of the 4th Spheric workshop, Nantes, France 2009.

[43] “NVIDIA CUDA: Compute Unified Device Architecture programming guide”, version 1, http://developer.download.nvidia.com/compute/cuda/1_1/NVIDIA_CUDA_Programming_Guide_1.1.pdf

Chapter 4

Validation of the SPH method

4.1. Basic tests

This chapter covers some basic numerical tests done using the standard SPH. The tests which will be presented, involve simulating the basic properties of the Euler equations and conserving basic properties, such as angular momentum. The basic tests are the following:

- Shock tube test, direct application of the hyperbolic nature of Euler equations [1, 2, 3]
- Rotation of a square patch, conservation of angular momentum [4, 5, 6]
- Deformation of a circular patch of fluid, conservation of volume (for weakly compressible) [5, 7, 8]

Shock tube test

The Shock tube test involves replication of the wave formation, inherent in hyperbolic conservation laws such as Euler equations, caused by a discontinuity. The shock tube is a long straight tube filled with gas which is separated by a membrane into two parts under different conditions of density and pressure [1]. The gas at each chamber is initially at equilibrium. Then, suddenly, the diaphragm is removed and three characteristic waves are formed, depending on the conditions. A *shock wave* moves towards the low pressure region, a *rarefaction wave* moves towards the high pressure region. These waves are *non-linear* waves [2, 3] and define a region (called also as *star* region) of different pressure and velocity, than the two initial regions. There is also a *linearly degenerate contact discontinuity* which moves with the local flow velocity and is always between the two non-linear waves.

The shock tube test may be applied for the weakly compressible SPH equations, using the Tait equation of state, instead of the ideal gas equation of state. This can be used as a test to check if the developed SPH method is able to replicate the described wave structure. The solution of the SPH method will be compared to the exact solution of the Riemann problem (for more information. see appendix A). The initial conditions are:

$$\begin{cases} \rho_L = 1100, \ u_L = 100 & x < 0 \\ \rho_R = 1000, \ u_R = 0 & x > 0 \end{cases}$$

These conditions will lead to the formation of a shock wave and a contact discontinuity moving to the right and a rarefaction wave moving to left. In order to model the realistic wave structure, the

correct speed of sound, for modeling water, is used for the equation of state, thus a value of 1450m/s is used for c_0 and the parameter γ is set equal to 7. The problem is treated as one dimensional, thus the independent variables are density ρ and u velocity component – pressure is only dependent on density through the Tait equation of state. Particles are distributed on the x-axis with a non-uniform density; particle distribution is denser in the high density region (spacing dx=0.018m) and sparser at the low density region (spacing dx=0.02m). A total of 200 particles are involved in the simulation. It is highlighted here that in order to get a correct result with the SPH method, it is necessary to introduce the artificial viscosity, presented in chapter 3, else numerical oscillations will appear which will eventually ruin the solution.

In fig. 4.1 the results are shown comparing the SPH solution and the exact solution of the Riemann problem for the same conditions. It is obvious that the SPH method predicts correctly the velocities of the non-linear waves and the characteristics of the star region (u_* and ρ_*). However there is an oscillation in the area of the contact discontinuity (fig. 4.1 right). Similar observations have been made in shock tube test cases by other researchers too (for example G.R. Liu [1]).

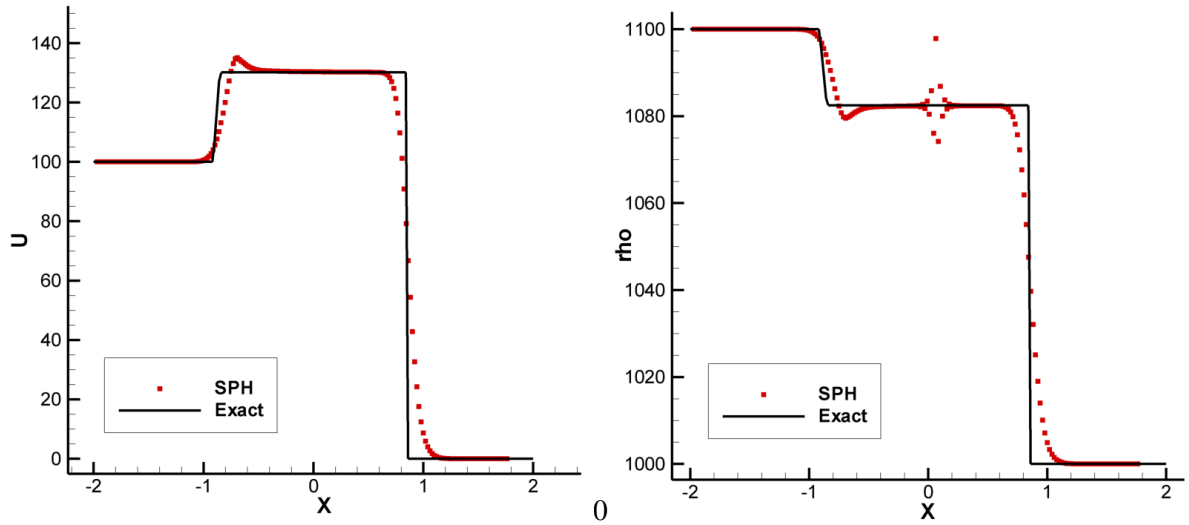


Fig. 4.1. Shock tube simulated with SPH (red dots). Left: x-velocity. Right: density. Comparison with the exact solution of the Riemann problem (continuous line).

Rotation of a square patch - Conservation of angular momentum

This test involves the rotation of a square patch of fluid with side L (equal to 0.4m), surrounded by void. The square patch is initially given a velocity field which will cause rotation. The initial velocity field is:

$$\left.\begin{aligned} u(t=0) &= \omega \left\| \mathbf{r}_i - \mathbf{r}_C \right\| \frac{y_i - y_C}{\left\| \mathbf{r}_i - \mathbf{r}_C \right\|} \\ v(t=0) &= -\omega \left\| \mathbf{r}_i - \mathbf{r}_C \right\| \frac{x_i - x_C}{\left\| \mathbf{r}_i - \mathbf{r}_C \right\|} \end{aligned}\right\} \tag{4.1}$$

where ω is the angular velocity of the rotation, equal to 1rad/s, $\mathbf{r}=(x,y)$ is the position vector and the indexes i and c represent particle i and the center of rotation respectively. The fluid is assumed to have density of 1.23kg/m^3 and the sound speed c_0 is equal to 20m/s. The particle discretization used is 0.004m, leading to a total of 10201 particles.

Due to the initial velocity field, the square rotates and due to the centrifugal force it distorts; its corners stretch eventually forming long arms, which, ideally, should follow the dashed lines (see fig. 4.2). The described test case is ideal to determine the influence of the tensile instability. The fluid patch will develop a negative pressure near the center of rotation. If no treatment for the tensile instability is applied then the simulation will collapse, forming particle clusters and large voids (see fig. 4.2). On the other hand, the tensile instability treatment proposed by Monaghan is able to prevent the particle clustering and the simulation proceeds to later stages, where eventually the formed arms are greatly distorted while their width is comparable with the particle size (fig. 4.3). The kernel renormalization (fig. 4.4) is also able to prevent the unphysical fragmentation of the fluid patch due to tensile instability but, due to deformation, voids are formed between the arms (fig. 4.5 - similar behavior has been reported by Ren et al. [6]). Moreover, field variables distribution is better reproduced when using the kernel renormalization procedure, due to the increased interpolation accuracy. Also the arms tend to follow better the expected solution, represented with dashed lines.

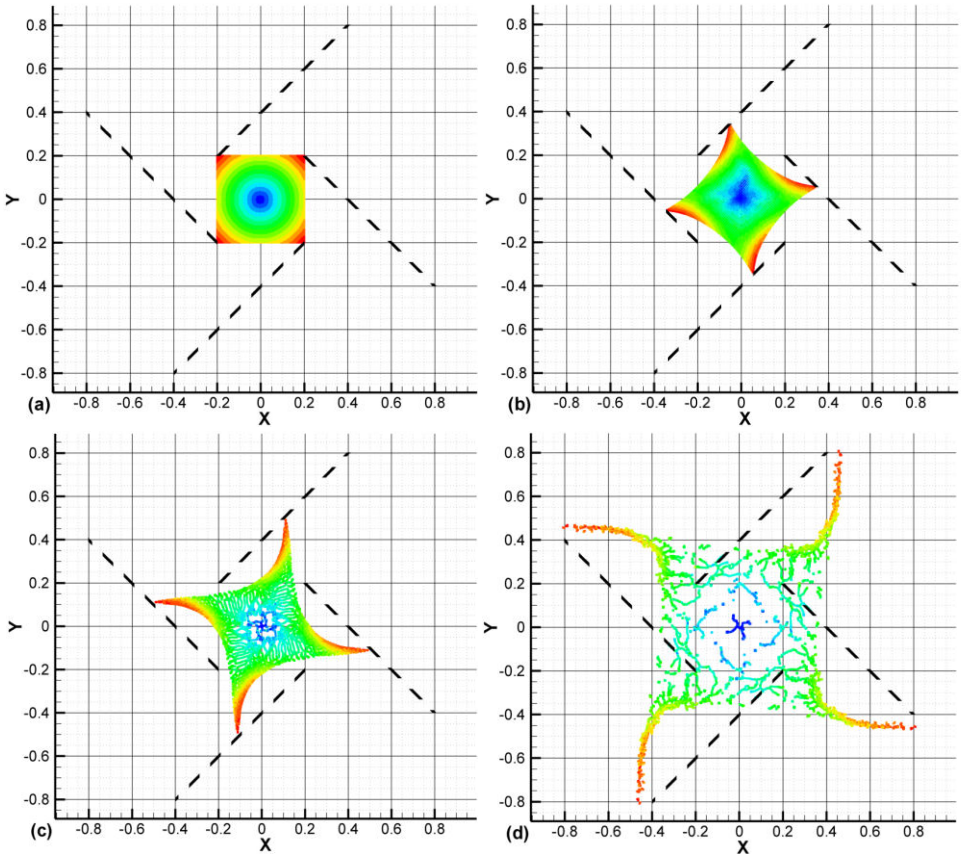


Fig. 4.2. Rotation of a square patch.
Solution without any tensile instability treatment.
Particle clustering and fragmentation is visible.

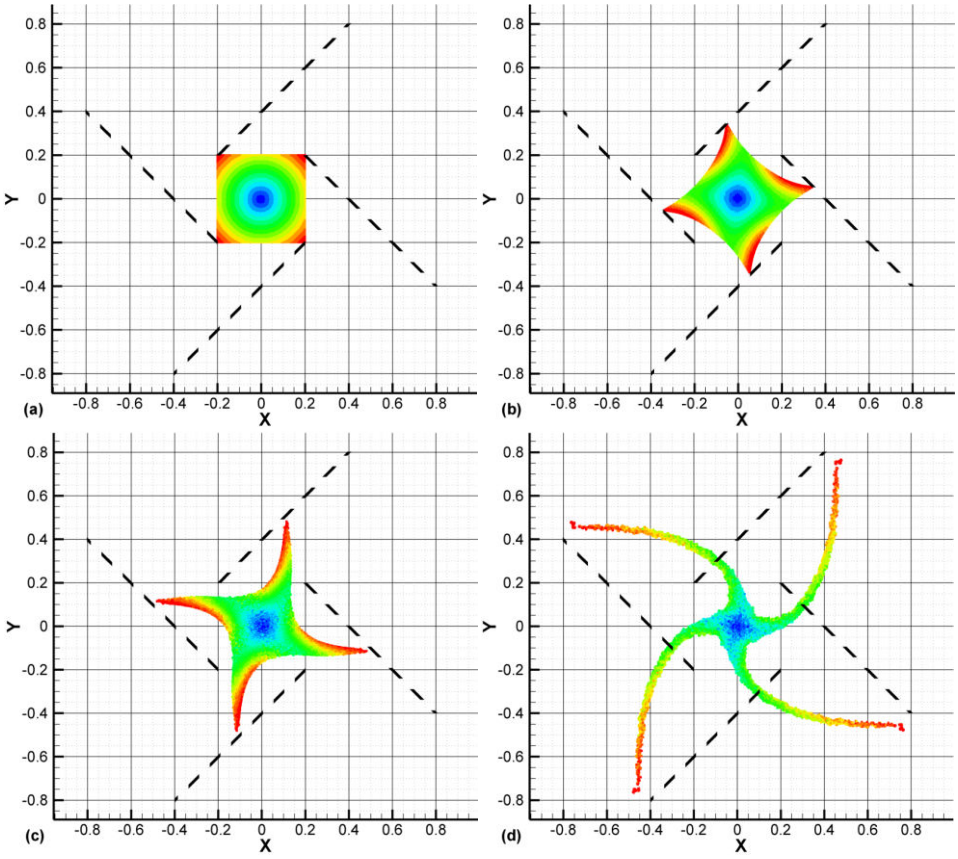


Fig. 4.3. Rotation of a square patch. Solution using Monaghan's tensile instability treatment

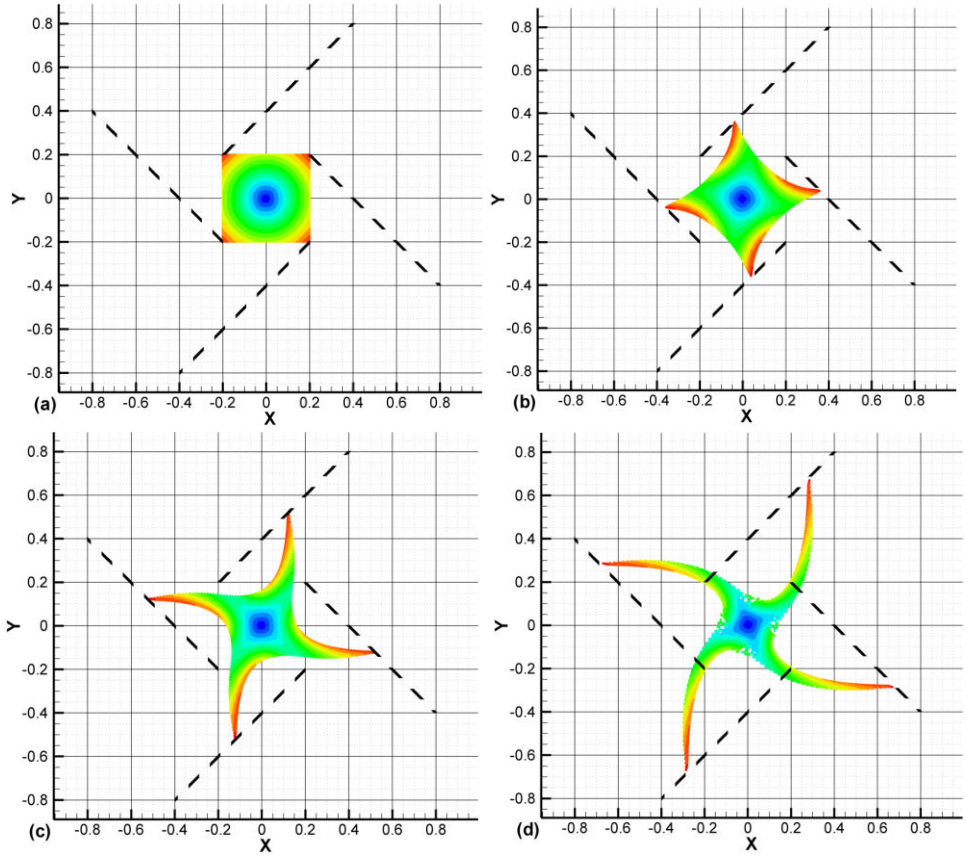


Fig. 4.4. Rotation of a square patch. Solution using kernel renormalization

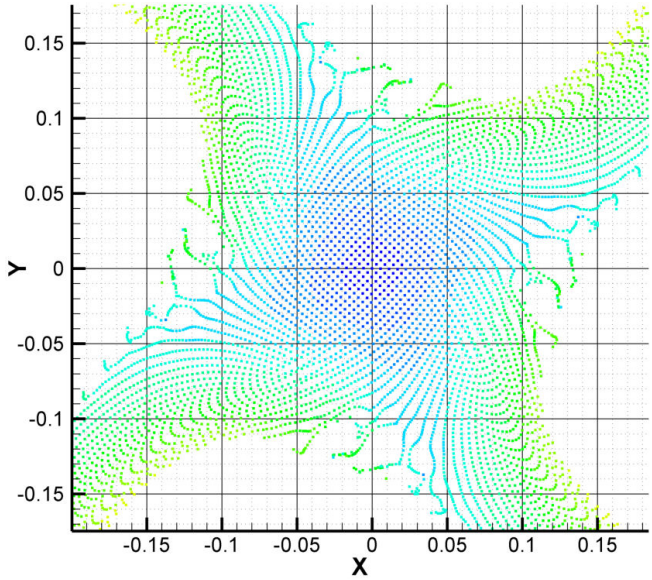


Fig. 4.5. Rotation of a square patch.
Voids formed between the arms, due to square patch deformation

Deformation of a circular patch of fluid - Conservation of volume (weakly compressible SPH)

In this test a 2D circular fluid patch is subject to compression, under a velocity field which tends to deform it into an ellipse. Since the fluid is assumed to be practically incompressible, by properly adjusting the speed of sound, the formed ellipse should occupy the same area as the initial circular patch. Moreover Monaghan [7], using the momentum equation, proved that the evolution of the ellipse semi-minor and semi-major axes is given by solving the following system of differential equations:

$$\frac{dA}{dt} = \frac{A^2\left(a^4 - \omega^4\right)}{a^4 + \omega^4} \tag{4.2}$$

$$\frac{da}{dt} = -aA \tag{4.3}$$

where:

- a is the semi-major axis and b the semi-minor axis
- ω is the initial value of a b

The condition for the fluid patch to remain elliptical with time varying axes a and b is that:

$$u = \frac{x}{a}\frac{da}{dt} \text{ and } v = \frac{y}{b}\frac{db}{dt}$$

Thus, from the equation giving the x-velocity component, it is possible to determine the initial value of A. The differential equations 4.2 and 4.3 can be solved easily using an explicit time marching method, giving a reference solution which will be used for comparing the SPH results. In the simulated case, the radius of the circular patch is 1m and the velocity field imposed at the initial time is the following:

$$\left.\begin{aligned} u(t=0) &= -100x \\ v(t=0) &= 100y \end{aligned}\right\} \tag{4.4}$$

The particle discretization used is dx=0.025m and the total number of particles involved are ~5000. The initial value of A is -100, since at the initial time step both a and b are equal to 1m. The ellipse axis evolution is practically identical to the theoretical solution from eq. 4.2 and eq. 4.3 (fig. 4.8 and fig. 4.9). Indicative results of the evolution of the deforming liquid drop are shown in fig. 4.6 and fig. 4.7, using 0^{th} and 1^{st} order density filters respectively. Results of the two different density filters are practically indistinguishable.

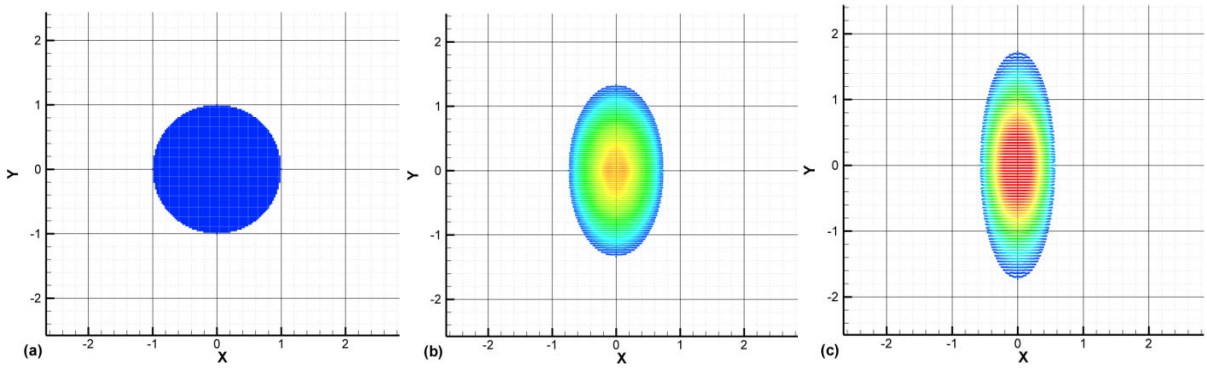


Fig. 4.6. Deformation of a circular fluid drop into ellipse.
Particle coloring according to pressure. 0^{th} order density filter.

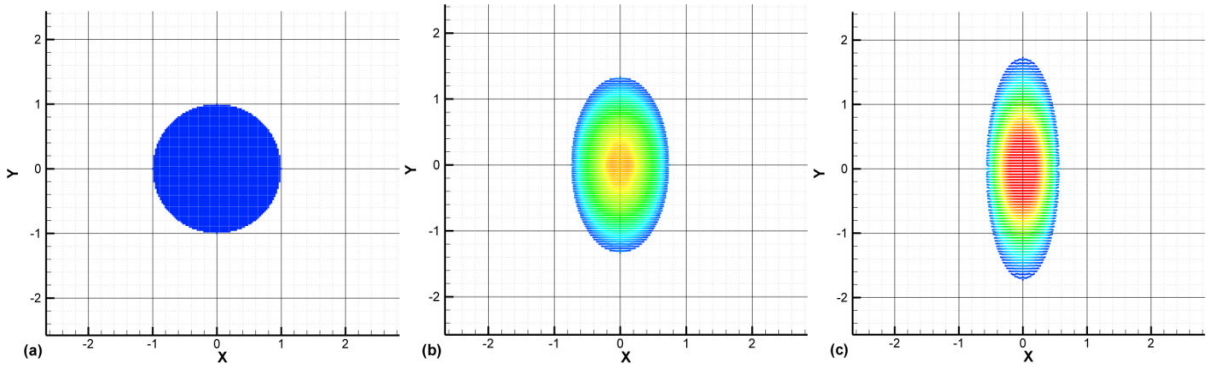


Fig. 4.7. Deformation of a circular fluid drop into ellipse.
Particle coloring according to pressure. 1^{th} order MLS density filter.

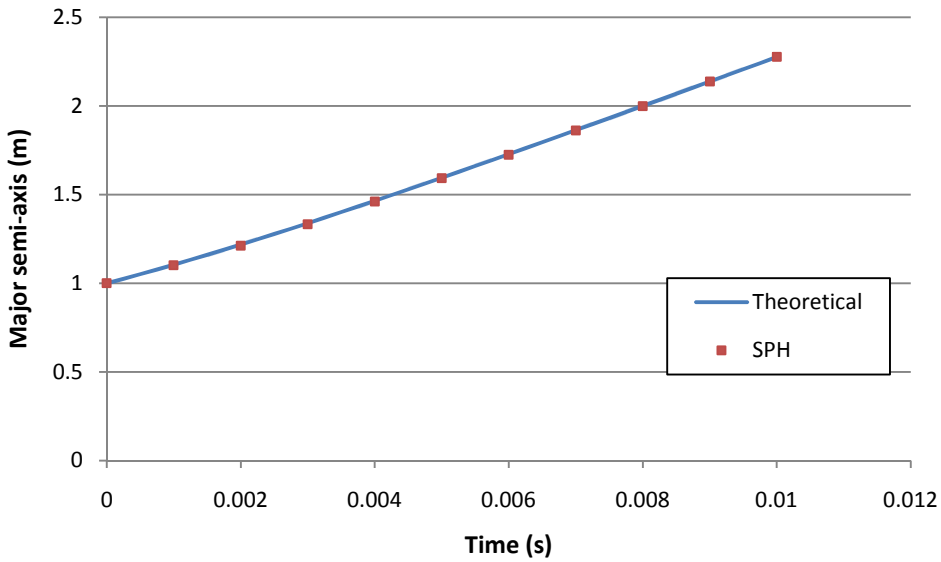


Fig. 4.8. Deformation of a circular fluid drop into ellipse. Evolution of the major semi-axis (a)

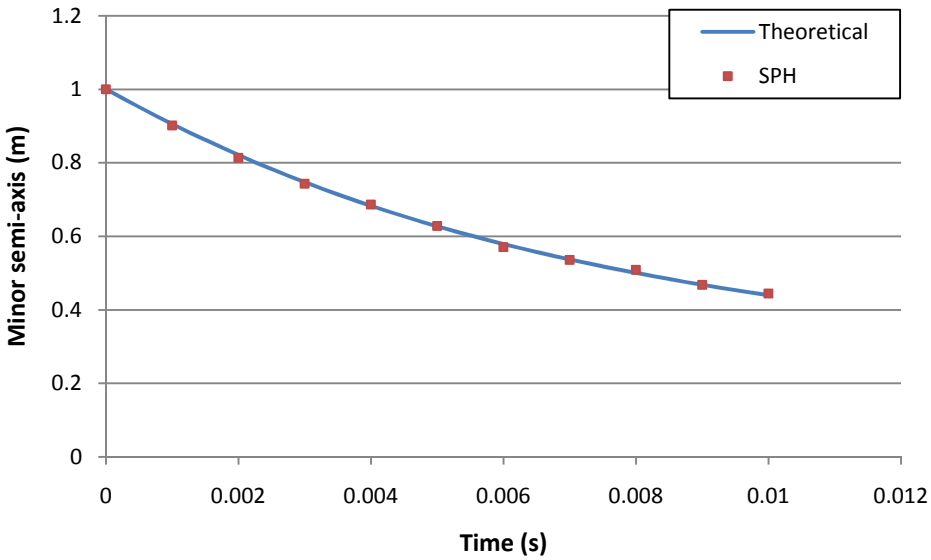


Fig. 4.9. Deformation of a circular fluid drop into ellipse. Evolution of the minor semi-axis (*b*)

4.2. Modeling viscous flows with SPH

In this part, several tests covering viscous flows will be simulated using the SPH method. The tests will examine at first some basic viscous flows in the laminar regime, such as:

- Couette flow
- Poiseuille flow
- Shear cavity flow
- 3D pipe flow
- Backward facing step flow

Then, viscous flows for higher Reynolds number are simulated. Implementation of traditional turbulence models in the SPH method is also discussed.

Couette flow

The Couette flow is a 2D flow, involving fluid flowing between two infinite parallel plates, placed at a distance $y=0$ and $y=l$. One plate remains still, while the other moves with a prescribed velocity u_0. The developing flow is moving only on the x-direction and velocity is a function of time t and y position. Assuming a constant dynamic viscosity μ, the momentum equation at the x-direction simplifies to:

$$\rho \frac{Du}{Dt} = \mu \frac{\partial^2 u}{\partial y^2} \tag{4.5}$$

For this specific flow the Lagrangian time derivative is equal to the Eulerian time derivative, since the convection term is zero ($u\frac{\partial u}{\partial x} = 0$, $v\frac{\partial u}{\partial y} = 0$). The boundary conditions are:

$$\left.\begin{aligned} u(t,0) &= 0 \\ u(t,l) &= u_0 \end{aligned}\right\} \tag{4.6}$$

and the initial condition is:

$$u(0,y) = 0 \tag{4.7}$$

It is possible to derive an exact solution expressed in series form:

$$u(t,y) = \frac{u_0}{l}y + \sum_{n=1}^{\infty}\frac{2u_0}{n\pi}(-1)^n \sin\left(\frac{n\pi}{l}y\right)Exp\left(-v\frac{n^2\pi^2}{l^2}t\right) \tag{4.8}$$

where v is the kinematic viscosity, equal to $\frac{\mu}{\rho}$.

For the simulation, using the SPH method, flow is assumed to be periodic, thus the fluid domain is extended in order to cover the support domain of the particles at the edges of the actual fluid domain (fig. 4.10). Since particles move in respect to time, when they move beyond a distance, on the $+x$ axis, then they are transferred back at the entrance between the plates at x=0. For the viscosity effects both the stress strain relations [1] and the Morris viscosity term [9] were used (see also chapter 3). However here it has to highlighted that the plates are represented by several layers of particles in order to cover the support domain of the fluid particles at $y=0$ and $y=l$. All particles at the top plate are assumed to move with the same speed u_0. If the stress strain relations are used, the stress field has to be extrapolated at the boundary particles using the local stress slope, in order to obtain accurate approximations of the viscous effects near the boundaries. Such extrapolation is not required when using the simpler Morris viscous treatment. Moreover such an extrapolation is not simple in complicated geometries, thus rendering the Morris viscosity term attractive for modeling viscous effects, in general complicated geometries.

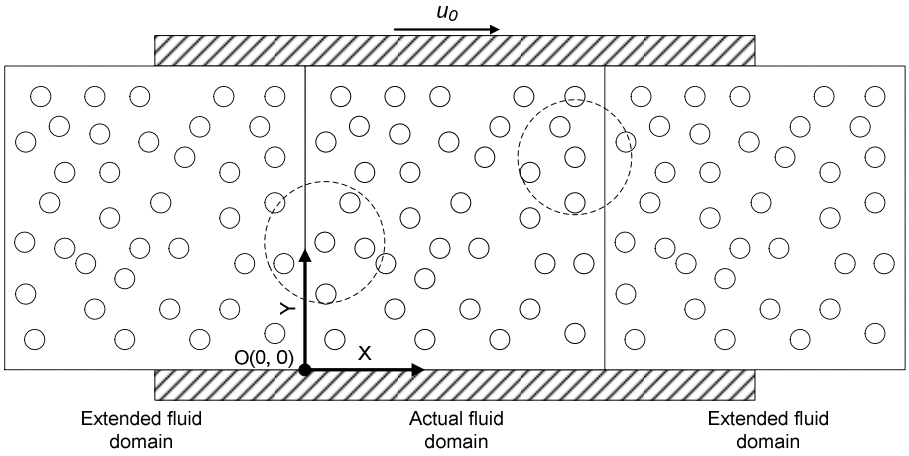


Fig. 4.10. Couette and Poiseuille flow set-up. The support domain of two particles at the edge of the actual fluid domain is also visible.

The SPH simulation was performed for:

- Distance between plates, l=1mm
- Plate velocity $u_0=1.25\cdot10^{-5}$ m/s
- Kinematic viscosity $v=1.01\cdot10^{-6}$ m^2/s
- Particle spacing is $dx=0.01l=1\cdot10^{-5}$ m (7083 total particles involved)

The Reynolds number is 0.0125. In fig. 4.11 an indicative velocity field is shown, after reaching steady state. Also in fig. 4.12 and fig. 4.13 the transient velocity profile calculated by the SPH method with stress strain relations and the Morris viscosity term is compared with the exact solution. The

stress strain relation gives better results for the velocity profile (fig. 4.12), but the error of the Morris simplified term, in comparison to the exact solution, is practically negligible (fig. 4.13). Thus, from now on in all cases involving viscosity effects, the Morris viscosity term will be included.

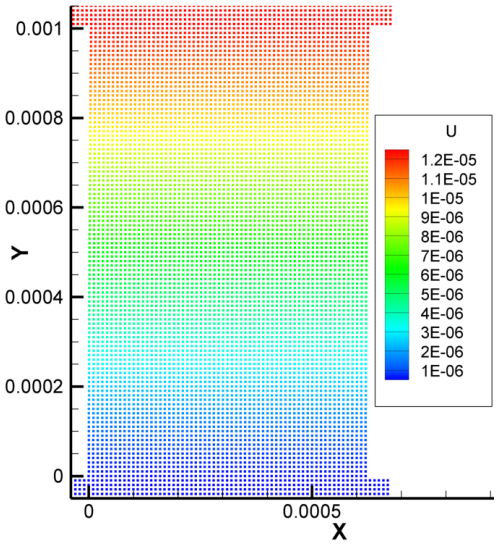


Fig. 4.11. Couette flow. *u*-velocity profile after reaching steady state.

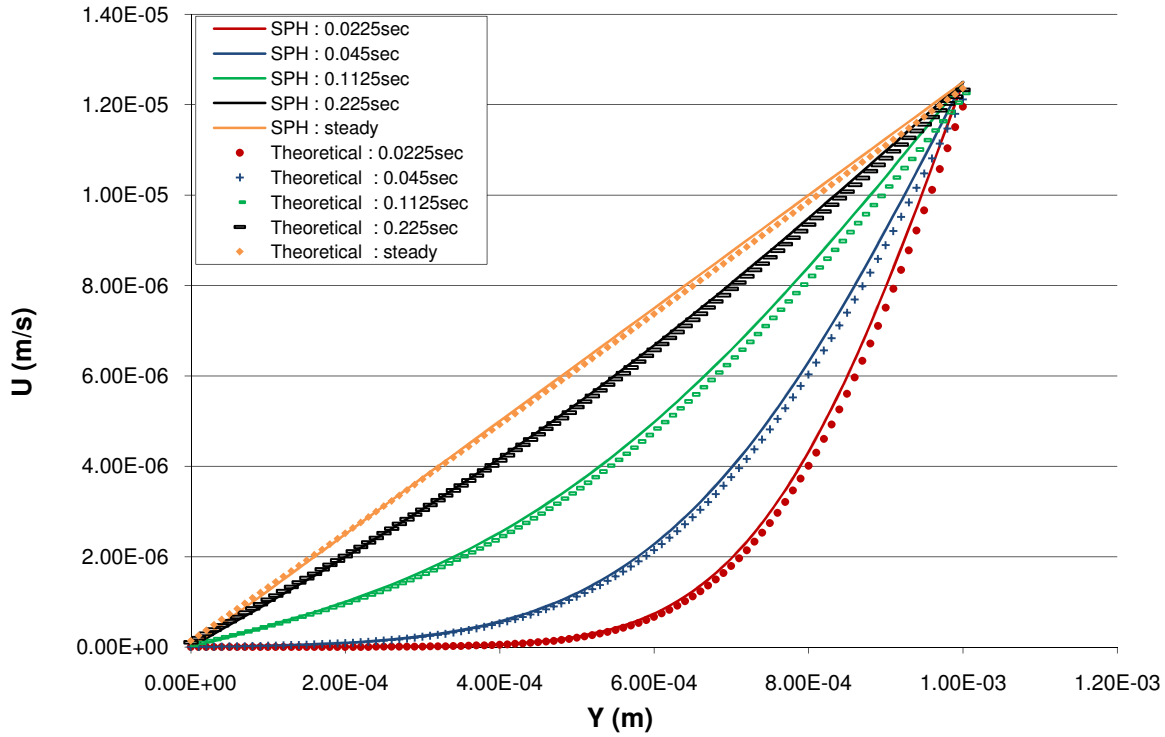


Fig. 4.12. Couette flow. Velocity distribution in respect to *y*-axis. Morris viscosity term.

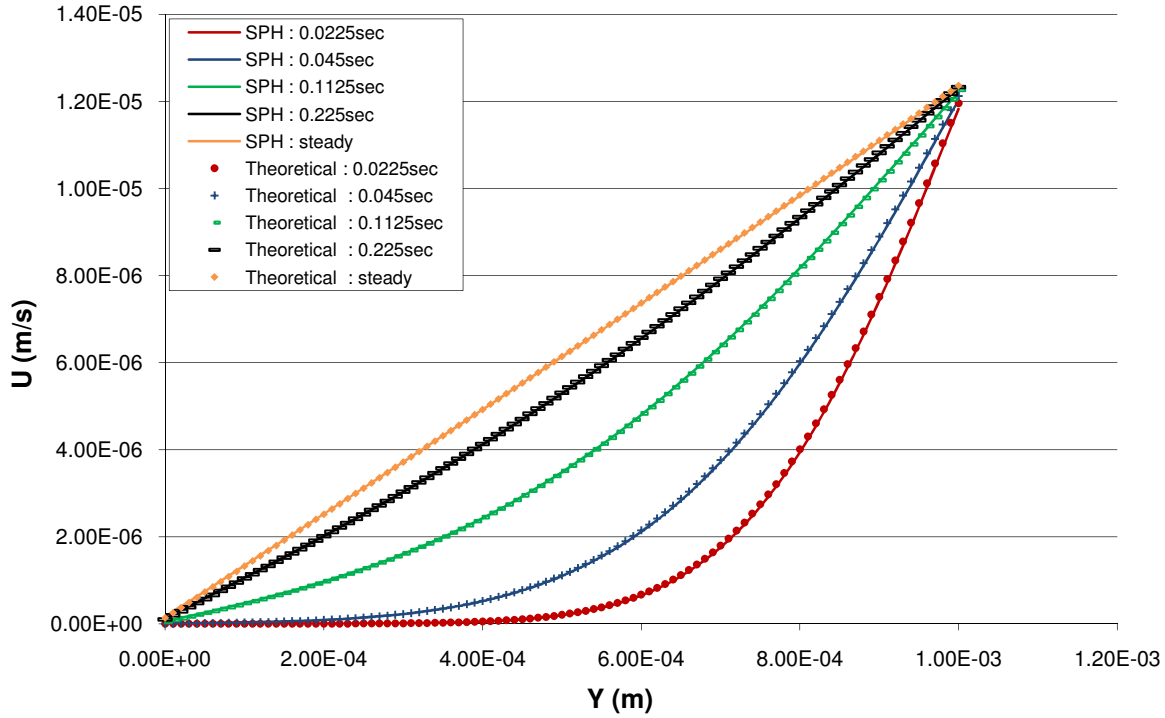


Fig. 4.13. Couette flow. Velocity distribution in respect to *y*-axis. Stress strain relation.

Poiseuille flow

The Poiseuille flow is a 2D flow, involving fluid flowing between two infinite stationary parallel plates, placed at a distance $y=0$ and $y=l$. The flow is driven due to a pressure difference, but in the present simulation a forcing term will be used instead. As in the Couette flow, the developing flow is moving only on the *x*-direction velocity is a function of time t and y position. The *x*-momentum equation reduces to:

$$\frac{Du}{Dt}=\frac{\mu}{\rho}\frac{\partial^2 u}{\partial y^2}+f \tag{4.9}$$

Again the Lagrangian time derivative is equal to the Eulerian time derivative. The boundary conditions are:

$$\left.\begin{aligned}u(t,0)=0\\u(t,l)=0\end{aligned}\right\} \tag{4.10}$$

and the initial condition is:

$$u(0,y)=0 \tag{4.11}$$

The exact solution is:

$$u(t,y)=\frac{f}{2v}y(y-l)+\sum_{n=0}^{\infty}\frac{4f\cdot l^2}{v\pi^3(2n+1)^3}\sin\left(\frac{\pi y}{l}(2n+1)\right)Exp\left(-\frac{(2n+1)^2\pi^2 v}{l^2}t\right) \tag{4.12}$$

where v is the kinematic viscosity, equal to $\frac{\mu}{\rho}$ and f is the forcing term.

The SPH simulation was performed for:

- Distance between plates, l=1mm
- Forcing term $f=2\cdot10^{-4}$ m/s^2
- Kinematic viscosity $v=1.01\cdot10^{-6}$ m^2/s
- Particle spacing is $dx=0.01l=1\cdot10^{-5}$ m (7083 total particles involved)

As with the Couette flow, periodic conditions were used to simulate the infinite plates. In fig. 4.14, an indicative instance is shown upon reaching steady state (Re=0.016).

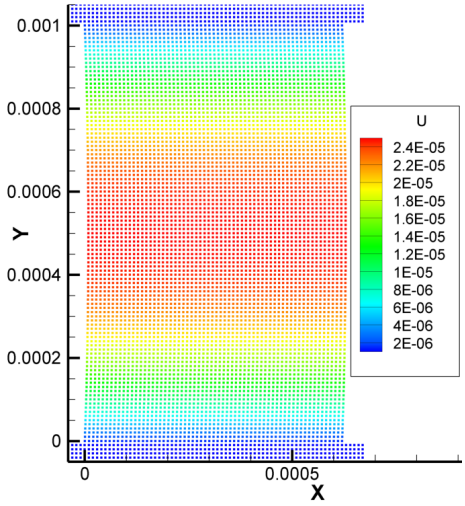


Fig. 4.14. Poiseuille flow. U-velocity profile after reaching steady state.

In fig. 4.15 indicative transient velocity profiles are shown. For all time instances the calculated profiles with the SPH method and the exact solution profiles are practically identical.

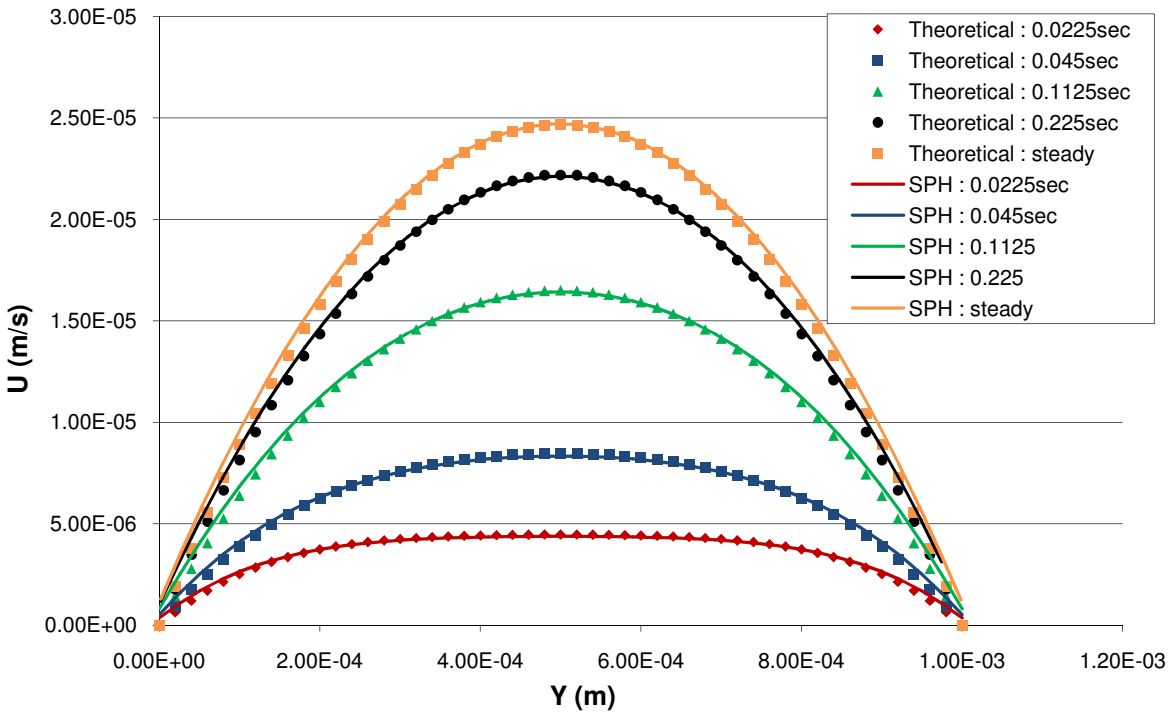


Fig. 4.15. Poiseuille flow. Velocity distribution in respect to y-axis. Morris viscosity term.

Shear cavity flow

The shear cavity flow, also known as lid-driven cavity flow, is the developing flow patterns in a closed square cavity, generated by moving the top side of the square at a constant velocity u_0, while the rest sides remain still. The flow will reach a steady state and forms a circular pattern.
In the specific simulation the parameters used are the following:

- square cavity side l=1mm
- kinematic viscosity v=10^{-6}m/s^2
- fluid density ρ_0=1000kg/m^3
- u_0 =1mm/s

leading to a Reynolds number of 1. Two different particle resolutions were used; a coarse where a total of 3600 particles where involved and a fine with 22500 particles involved. In fig. 4.16 there is an indicative instance of the developing flow, simulated with the SPH method. Also fig. 4.17 and fig. 4.18 show a comparison of the u and v-velocity distribution along the centerlines, compared with the reference solution via the Finite Difference Method [1]. Generally there are differences, which tend to become smaller as the resolution increases. In any case, the developing flow displays the same features as those found in the reference solution.

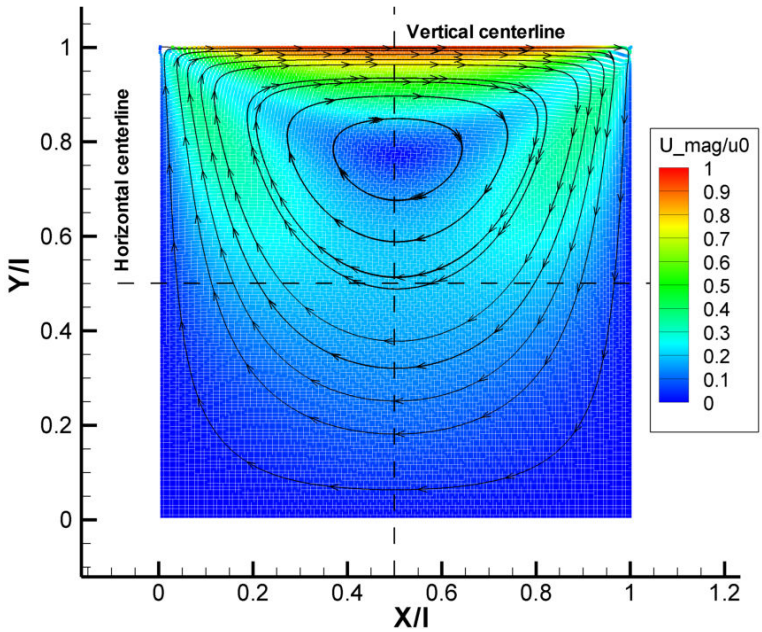


Fig. 4.16. Velocity magnitude distribution for the shear cavity problem.

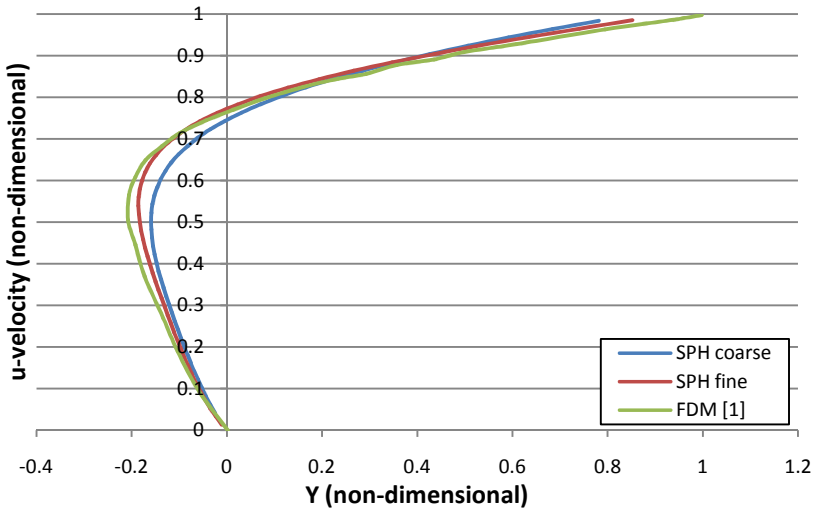


Fig. 4.17. Shear cavity. *u*-velocity distribution in respect to *y*-axis (at the vertical centerline).

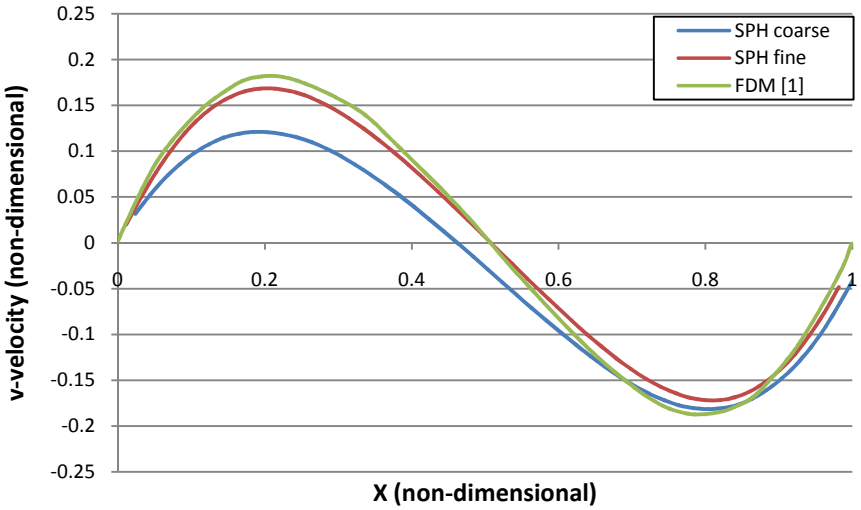


Fig. 4.18. Shear cavity. *v*-velocity distribution in respect to *x*-axis (at the horizontal centerline).

The SPH simulation of the shear cavity exhibits the characteristic of particle alignment, which was mentioned at the chapter discussing the particle approximation errors (chapter 2). Indeed, as can be seen by zooming at the top right corner of the square cavity (see fig. 4.19), particles get clumped over time, forming lines. The result of this is the increase in particle approximation errors and the great underestimation of the velocity in the specific area, in respect to the reference solution calculated with Fluent.

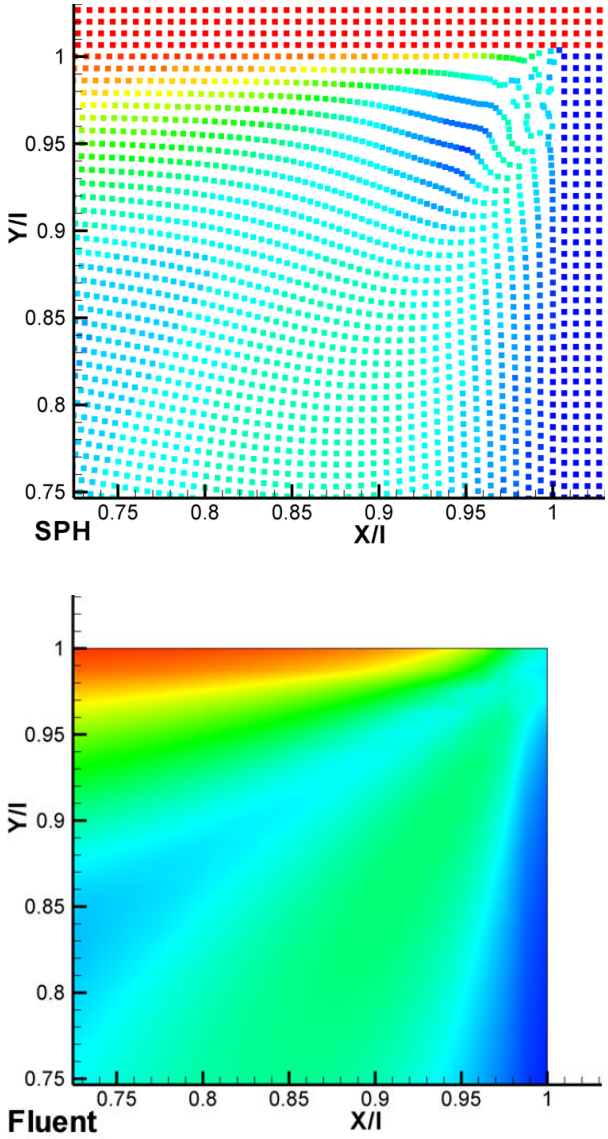


Fig. 4.19. Particle clumping near the top right corner of the shear cavity.
Note the underestimation of velocity, comparing with the Fluent solution.

Laminar flow inside a 3D pipe

This flow type resembles the 2D Poiseuille flow, but now fluid flows inside a 3D pipe instead of two parallel infinite plates. The flow is assumed periodic, applying appropriate boundary conditions, as already discussed, and a forcing term is added in the momentum equation for driving the flow. Moreover only one fourth of the pipe is solved assuming symmetry boundary conditions at the *xy* and *yz* planes. It is possible to obtain an analytical solution of the flow, solving the Navier-Stokes equations in cylindrical coordinates, finally giving [10]:

$$u(r) = 2u_m\left(1-\frac{r^2}{R^2}\right) \tag{4.13}$$

For the simulation the particle sizing was 0.01mm and the simulated pipe diameter was 0.50mm, leading to 50 particles on the diameter. The simulated fluid was water having density of 1000kg/m^3 and a dynamic viscosity of 10^{-3}Pa.s. From the forcing term added in the momentum equation at the axial direction it is possible to determine the maximum velocity of the developing flow. It is known that:

$$u_{\max} = \frac{p_2 - p_1}{4L\mu} R^2 \tag{4.14}$$

The pressure difference can be written as *F/A*, where *F* is the total force acting on the fluid and *A* is the surface of the cross section of the pipe. The maximum velocity which develops is in agreement with the analytical solution within a 3% error giving a parabolic profile (fig. 4.20). Also in fig. 4.21 the velocity distribution in the radial direction is shown in comparison with the analytical solution (eq. 4.13), showing good agreement.

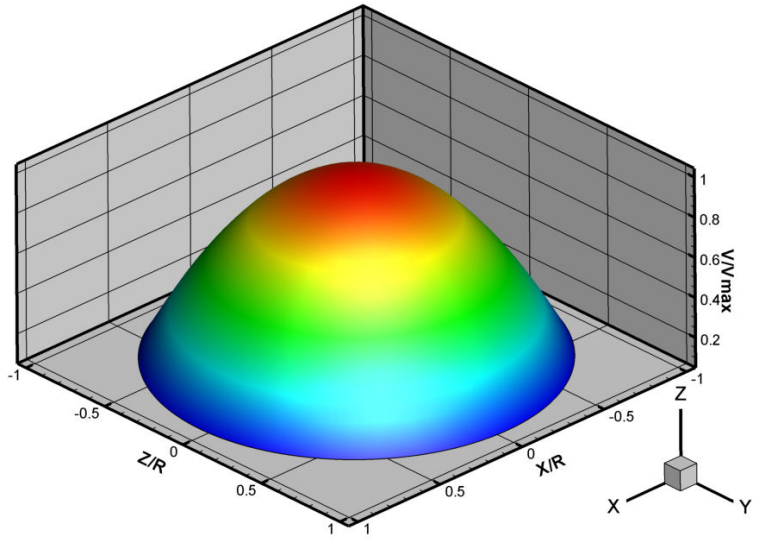


Fig. 4.20. Flow inside a 3D pipe.
Velocity distribution at the cross section of the pipe

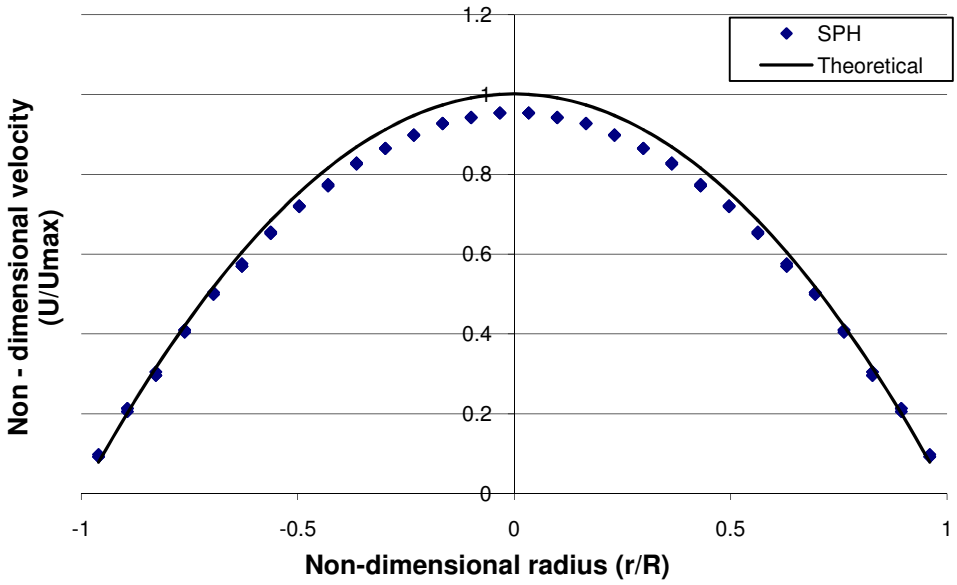


Fig. 4.21. Flow inside a 3D pipe.
Comparison of the velocity distribution calculated with SPH with the analytical solution.

Laminar backward facing step flow

The laminar backward facing step is a viscous test involving a sudden expansion, leading to detachment, recirculation and re-attachment. The simulated geometry and conditions are based on Armaly's experiment [11] for a Reynolds number of 389. A simple sketch depicting the geometry is shown in fig. 4.22. The characteristics of the simulation are:

- u=0.5323m/s
- h=5.2mm
- s=4.9mm (h/H=0.515)
- ρ=1.23kg/m^3
- μ=$1.71 \cdot 10^{-5}$ *Pa.s*
- L_1=0.01m and L_2=0.09m

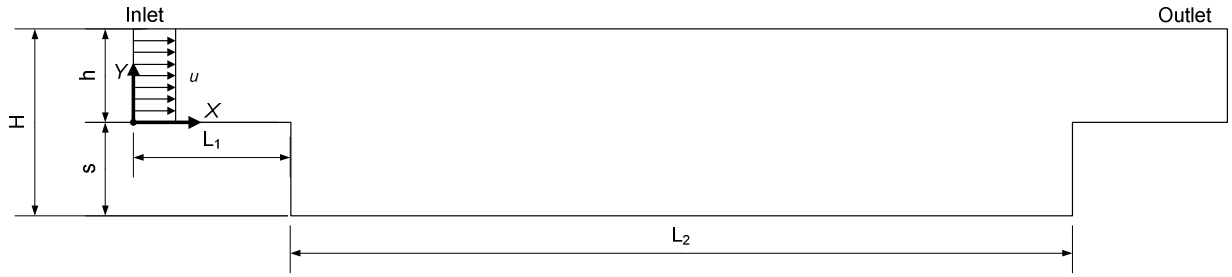


Fig. 4.22. Sketch of the backward facing step geometry

The geometry used for the SPH simulation is different from the one used in Armaly's experiments, since a forward step is assumed before the outlet. If this blockage near the exit does not occur, then particles would create voids after the backward facing step. Moreover an increased value of the numerical speed of sound is required, to avoid gap formation [12]. Particularly in the described simulations, the speed of sound used was c_0=55m/s, which is ~100 times the maximum particle velocity.

In order to enforce inlet boundary conditions, a particle buffer was used at the inlet (fig. 4.23). This particle buffer consists of particles moving at the same speed as the prescribed inlet velocity, but its density/pressure is obtained from the first line of real particles. The buffer moves, imposing the velocity inlet boundary condition. At specific time steps particles enter the computational domain, thus the buffer is returned to its original position.

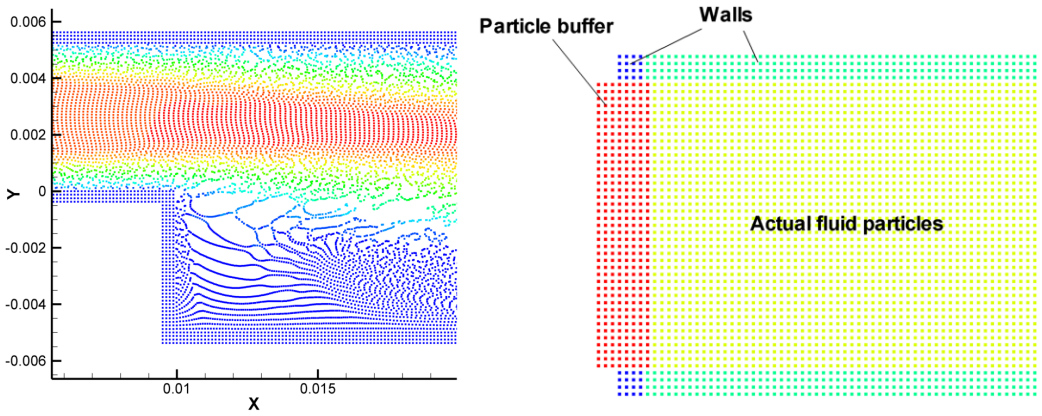


Fig. 4.23. Backward step flow. Left: Voids formed after the backstep. Right: Buffer particles at the inlet of the back step

Since there is no analytical solution for the developing flow field, the SPH method solution is compared with the solution obtained by Fluent software. For the SPH method two resolutions were used, with particle sizing *dx*=0.2mm and *dx*=0.125mm. In the following figure, the recirculation bubble is shown, from the SPH and the Fluent solutions. In both cases reattachment is calculated at X~0.04m (fig. 4.24).

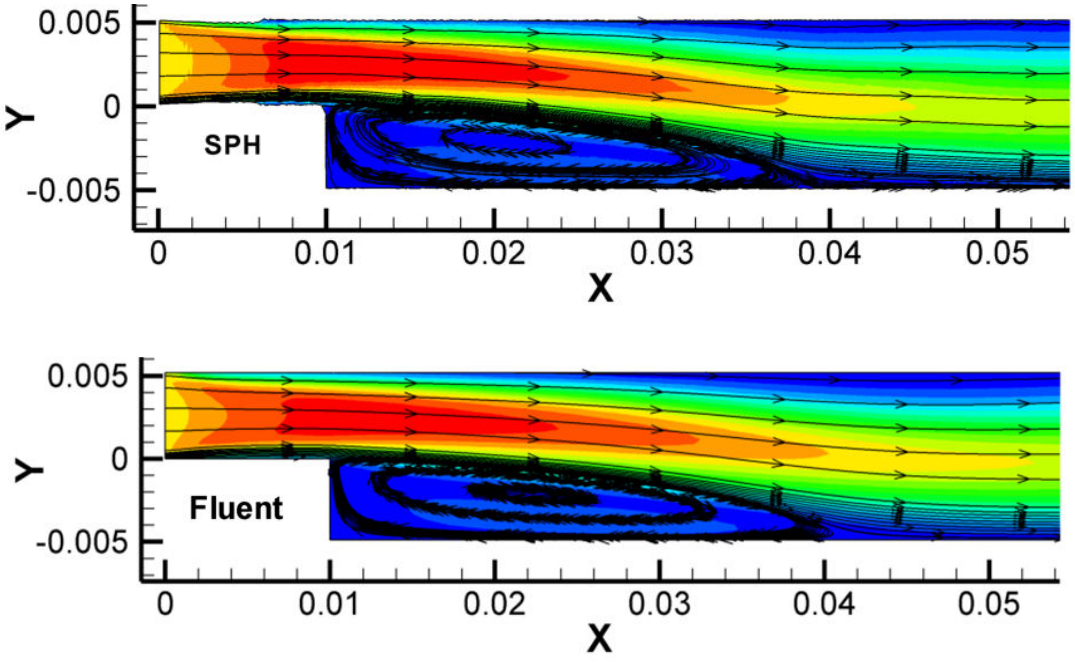


Fig. 4.24. Recirculation bubble after the backward facing step.
Comparison of the solution with SPH and Fluent.

In fig. 4.25 several indicative velocity profiles are shown, for slices on the *Y*-axis at different locations.

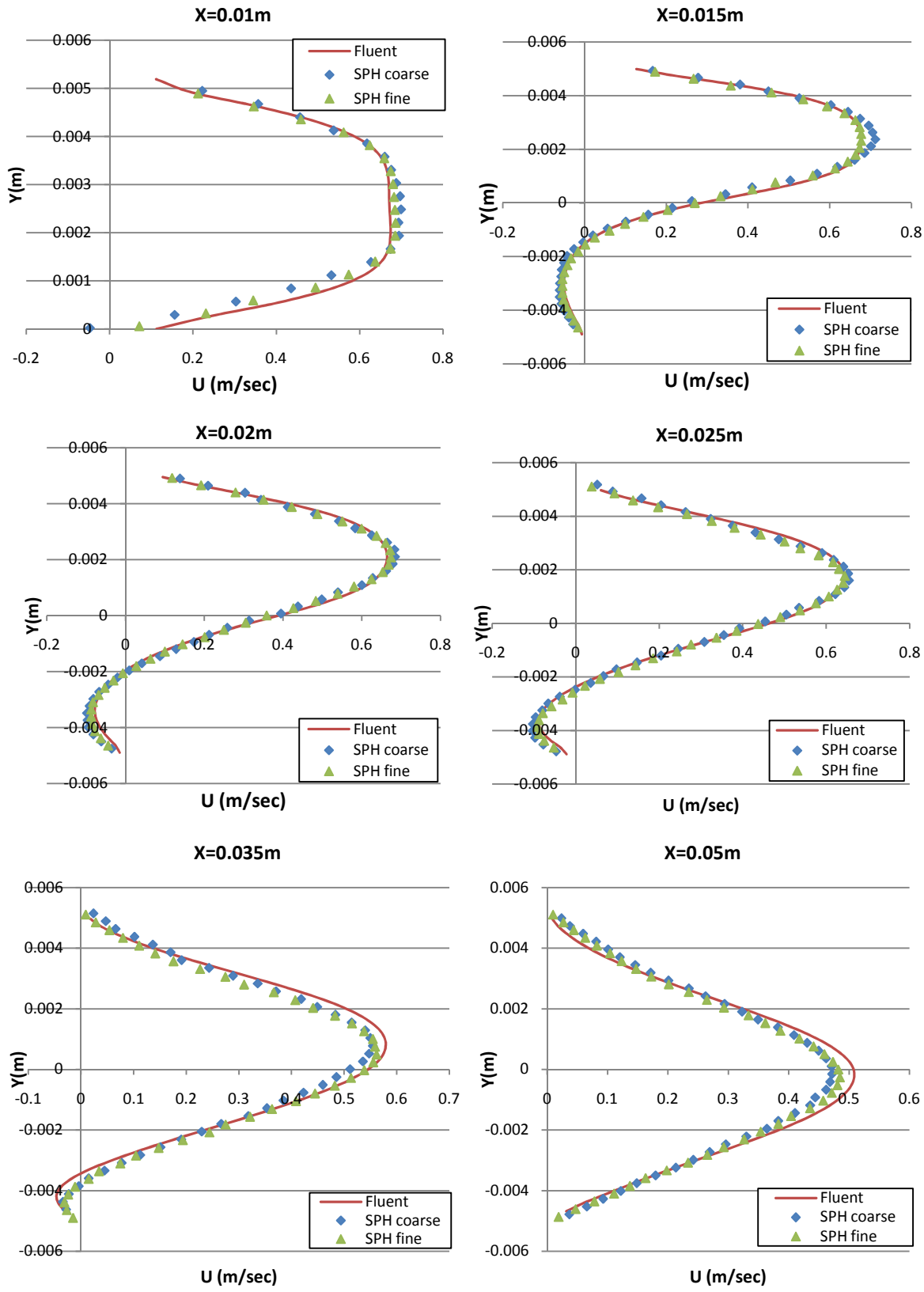


Fig. 4.25. Backward facing step flow. Velocity distribution at different locations.

The presented results show that the SPH method is able to capture properly the velocity field and the recirculation bubble formed behind the back step.

4.2.1. Particle redistribution

The fact that the SPH method loses accuracy, due to the particle disorder was already discussed at a previous chapter. A way to overcome this difficulty, while also controlling the particle distribution, is by periodically *redistributing* particles, also known as *particle remeshing* in SPH literature. The redistribution technique was initially devised by Chaniotis et al. [13, 14, 15], for controlling particle disorder and interpolation errors. Other researchers have also worked on this technique, such as Galagali [16] and Moulinec et al. [17].

The main idea of particle redistribution is to periodically move particles at a regular lattice improving their spatial distribution. Values of the field variables at the new position are found by interpolating field variables from the previous time step. In fig. 4.26 the redistribution procedure is shown.

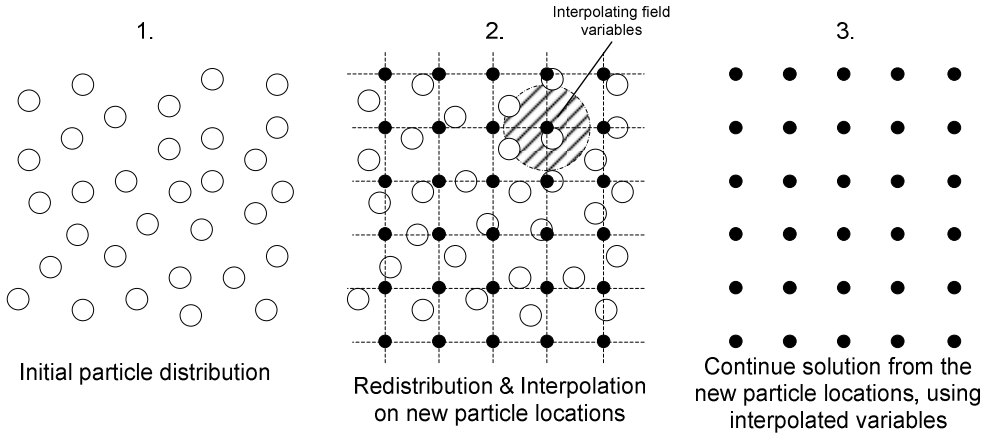


Fig.4.26. Particle redistribution procedure.

The main advantages of using particle redistribution are:

- Controlled particle distribution, thus voids cannot be formed
- Wall penetration is impossible
- Increased solution accuracy, due to the reduction of interpolation errors, since interpolations are performed on a uniformly distributed particle set. Increased accuracy comes with a relatively small computational cost, since the redistribution procedure is periodically performed every *N* steps (*N* greatly depends on the particle distortion – for the back step case, a sufficient value of *N* was 1500-3000 time steps).

However, the main disadvantage of the redistribution procedure is the increased numerical diffusion which is added during the interpolation of field variables from the old particle distribution on the new uniform particle lattice. The interpolation schemes used during the redistribution procedure are of high importance, especially in cases of high accuracy simulations, where high order conservative interpolation schemes are used [13, 14, 15, 16]. Another disadvantage is the fact that it is not easily applicable in cases with free surface flows. In such cases, reconstruction of free surface is required, which is not simple with the SPH method. For this reason, particle redistribution is primarily used in constrained flows, such as the flow in the back step geometry, where the extents of the flow field are known and the new particle locations predetermined.

Using the technique of particle redistribution, it is possible to solve the flow field, using the exact experimental conditions of Armaly's experiment [10, 11], without void formation and with high

accuracy in the solution. Moreover a parabolic inlet profile was used, where the inlet *u*-velocity component is given by:

$$u(y) = 6\bar{u}\frac{y}{h}\left(1-\frac{y}{h}\right) \tag{4.15}$$

The average velocity $\bar{u}$ is equal to 0.5323m/s, the sound speed value is set to 10m/s, (there is no need to use a very large value, since particle redistribution will prevent void formation) and density and dynamic viscosity are set to 1.23kg/m^3 and $1.71 \cdot 10^{-5}$Pa.s. The characteristic step dimensions are the same as before: h=5.2mm and s=4.9mm. Again two particle resolutions were used, one coarse and one finer, with particle sizes dx=0.2mm and dx=0.125mm (involving 25811 and 61616 particles respectively). The velocity profiles are compared with the experimental values from Armaly [11] (fig. 4.28 and 4.29).

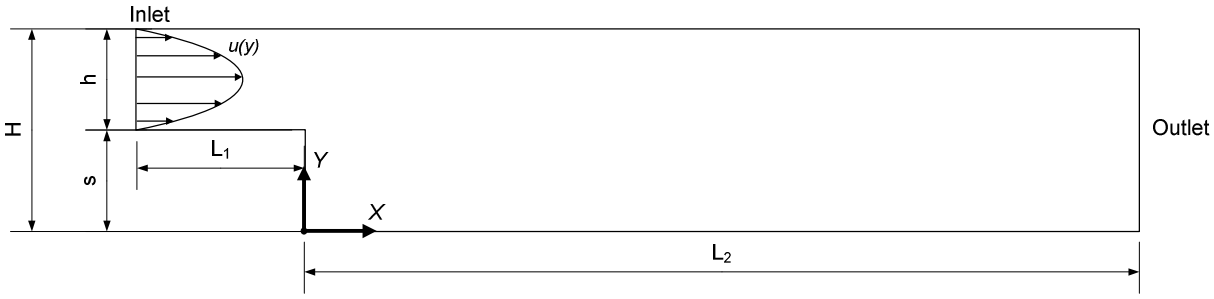


Fig. 4.27. Backward facing step – Armaly's experiment geometry.

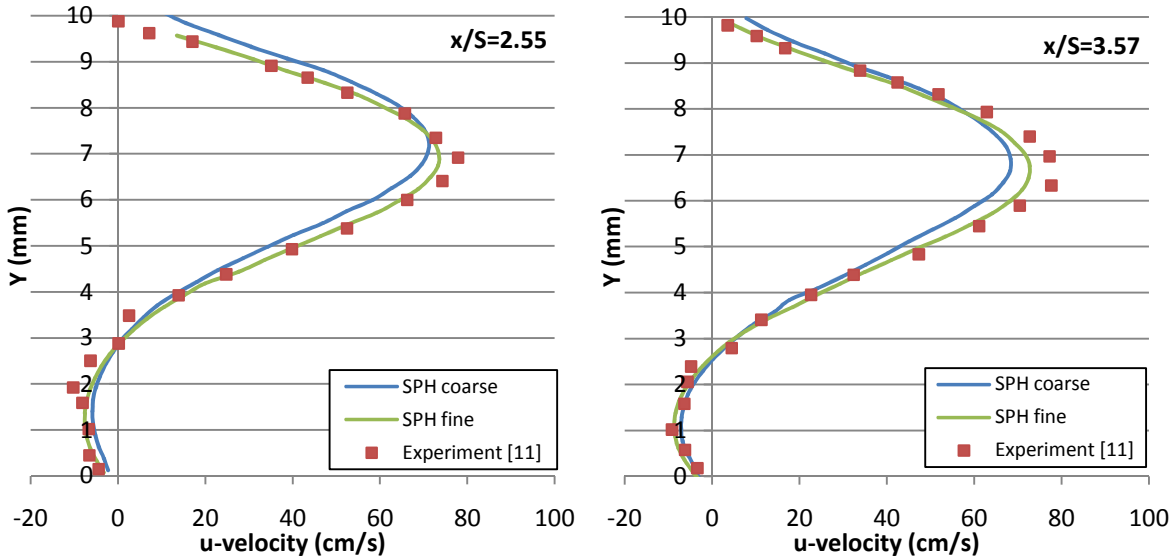


Fig. 4.28. Backward facing step flow with particle redistribution. Velocity distribution at different locations.

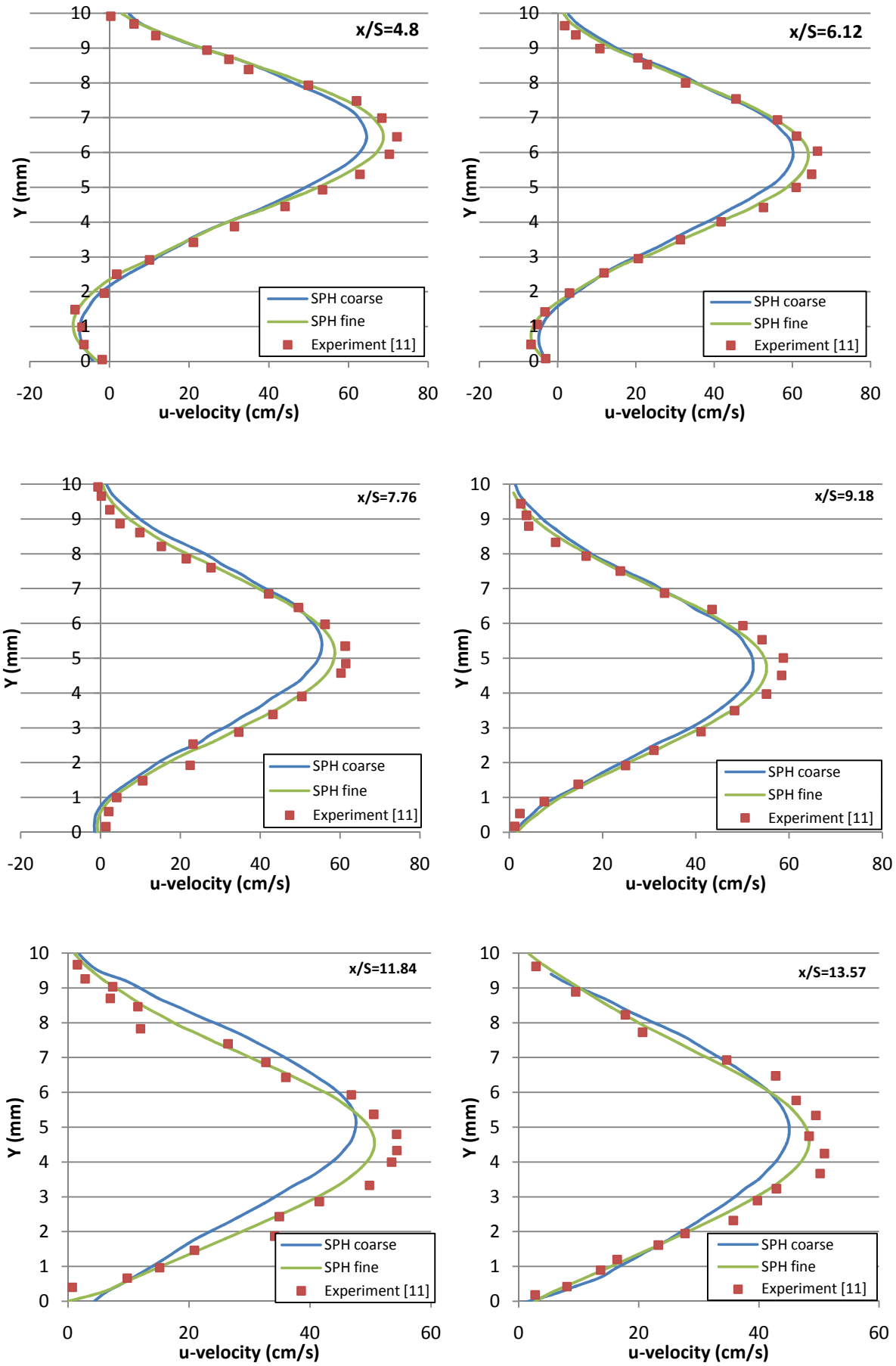


Fig. 4.29. Backward facing step flow with particle redistribution. Velocity distribution at different locations.

The SPH method estimates reattachment at ~0.04m, close to the calculated value using the Finite Difference Method in [10].

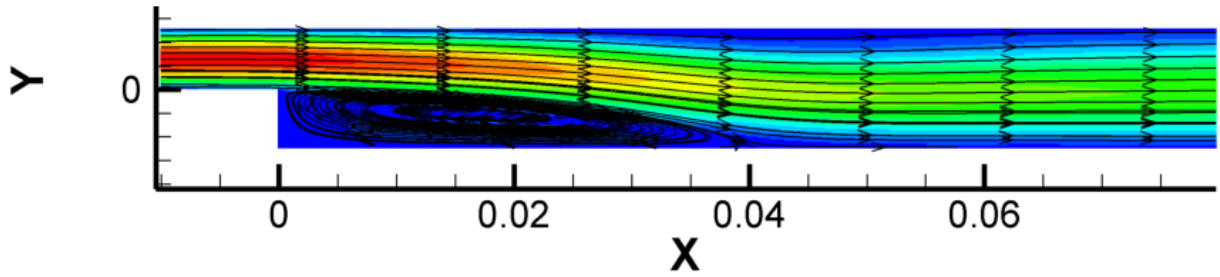


Fig. 4.30. Recirculation bubble. Reattachment at X~0.04

4.2.2. Turbulence modeling in SPH

Generally about turbulence

The main characteristic of turbulent flow is irregularity; turbulent flows are unsteady, irregular, random and chaotic [18]. The appearance of turbulence is directly related with the Reynolds number expressing the ratio of inertial to viscous forces. Below a Reynolds number, flow is smooth and adjacent layers of fluid slide past each other in an orderly fashion [19]. However, as Reynolds number increases, the flow behavior becomes unsteady and chaotic. In this regime of the Reynolds number, turbulent flow occurs.

The main idea for studying turbulent flows, is to decompose any flow variable, such as velocity $u(t)$, to a steady mean value $\overline{u}$ and to a fluctuating component $u'(t)$. Even in flows where the mean flow characteristic vary in one or two dimensions, the turbulent fluctuations always have a three dimensional spatial character. Furthermore by visualizing a turbulent flow, it is obvious that rotational structures are formed, called eddies, which have a wide range of length scales. Eddies contribute to the flow dissipation, thus mass, momentum and energy transfer is prevalent in turbulent flows. Moreover the existence of eddies greatly affects energy losses, since an energy cascade hands down energy from larger eddies to smaller eddies and eventually is dissipated due to viscosity and transformed into thermal internal energy.

In order to properly solve a turbulent flow, it is necessary to capture all underlying scales of fluid motion, i.e. resolve eddies of all sizes involved. This type of simulation is called *Direct Numerical Simulation* (*DNS*) and requires extreme computational capacity even for simple flows of relatively small Reynolds numbers, thus it is not feasible for practical applications. For practical cases, another approach is used; the Navier-Stokes equations are averaged (*Reynolds Averaged Navier Stokes - RANS*). By using time averaging, it is possible to rewrite the Navier Stokes equations, into a form involving the averaged field variables and the fluctuating components; the latter are the elements of the *Reynolds stress tensor*. The elements of the Reynolds stress tensor are the unknowns, for which a correlation has to be established – this is called the *closure problem*. In the simpler turbulence models, Reynolds stresses are directly linked to the mean flow characteristics and an *eddy viscosity* via the *Boussinesq assumption*. The eddy viscosity is calculated through an algebraic equation (*algebraic turbulence models*), or using additional transport equations (*one equation*, *two equation* or *Reynolds stress models*).

There have been several attempts of incorporating turbulence models in the SPH method. Violeau [20, 21, 22] has worked on the integration of traditional, widely accepted, turbulence models, such as the *k-ε*, *k-ω*, non-linear *k-ε* and the Reynolds stress model. Implementation of such models was successful, but generally there is difficulty enforcing boundary conditions [21] and there are discrepancies in the results from experiments [21]. Moreover practical application of 3D turbulent flows is rather cumbersome due to the increased computational cost [21 and 22].

Another proposed approach used by many SPH practitioners, is to use a *Sub-Particle Scale* (*SPS*) turbulence model [23, 24, 25, 26]. The SPS turbulence model is based on the *Large Eddy Simulation* (*LES*), where the basic idea is to simulate the larger eddies and model the smaller ones, which are below the cell/particle size. However, in order to obtain reliable results with this model, a 3D simulation with fine particle discretization is required [24].

On the other hand, efforts have been made for developing turbulence models specifically designed for the SPH method. Indeed, Monaghan, one of the inventors of the SPH method, has developed turbulence models, suited for the SPH method, such as the *XSPH-extension* [27] and the *Lagrangian Navier Stokes alpha* (*LANS-alpha*) [28] models. However, both these models have not been extensively used in the field of fluid dynamics.

Standard turbulence models in SPH (k-ε and k-ω models)

For simulating higher Reynolds numbers in the turbulent regime, an appropriate turbulence model must be included in the SPH model. In the present work, two well-known turbulence models were implemented in the SPH algorithm and tested in a benchmark case: the *k-ε* and *k-ω* turbulence models. Both models are based on the Reynolds Averaged Navier Stokes (RANS) approach, where the Navier-Stokes equations are averaged through time. They introduce two new transport equations in order to model the evolution of the eddy viscosity; one equation for the turbulent kinetic energy *k* and one equation for the turbulence dissipation *ε* or the specific dissipation rate (or turbulent frequency) *ω* – thus both belong to the two-equation closure models. Generally it is known that the *k-ε* model performs poorly in cases of separation, whereas the *k-ω* model performs better. For more information regarding turbulence models and the turbulence closure problem the reader is referred to the works of Wilcox [29] and Pope [30].

The implementation of turbulence models in the SPH method is based on the work of Violeau et al. [20, 21, 22]. First the continuity and momentum equations are written (and solved) in averaged form:

Continuity:

$$\frac{D\rho_i}{Dt} = \sum_{j=1}^{N} m_j \overline{\mathbf{u}}_{ij} \cdot \nabla_i W_{ij} \tag{4.16}$$

Averaged momentum equation:

$$\frac{D\overline{\mathbf{u}}_i}{Dt} = -\sum_j m_j \left[\left(\frac{\overline{p}_i}{\rho_i^{\,2}} + \frac{\overline{p}_j}{\rho_j^{\,2}} \right) \nabla W_{ij} - \mathbf{\Pi}_{ij} \right] + \mathbf{f}_{body} \tag{4.17}$$

Morris viscosity term:

$$\mathbf{\Pi}_{ij} = \frac{(\mu+\mu_T)_i + (\mu+\mu_T)_j}{\rho_i \rho_j} \frac{\bar{\mathbf{u}}_{ij}}{\left\|\mathbf{r}_{ij}\right\|^2} \mathbf{r}_{ij} \cdot \nabla W_{ij} \tag{4.18}$$

In the Morris viscosity term the eddy viscosity μ_T is included to account for turbulence effects. The eddy viscosity is linked to turbulent kinetic energy k and the turbulence dissipation ε or the specific dissipation rate ω, through the following equation:

$$\mu_T = \rho \nu_T = \rho C_\mu \frac{k^2}{\varepsilon} = \rho \frac{k}{\omega} \tag{4.19}$$

The equation which links ε and ω is:

$$\varepsilon = \beta^* \omega k \tag{4.20}$$

However the value of β^* is equal to the value of C_μ, for the standard k-ω model.
The exact transport equations for k, ε, ω are:

$$\frac{dk}{dt} = P - \varepsilon + \nabla\left[\left(\nu + \frac{\nu_T}{\sigma_k}\right)\nabla k\right] \tag{4.21}$$

$$\frac{d\varepsilon}{dt} = \frac{\varepsilon}{k}\left(C_{\varepsilon 1} P - C_{\varepsilon 2}\varepsilon\right) + \nabla\left[\left(\nu + \frac{\nu_T}{\sigma_\varepsilon}\right)\nabla \varepsilon\right] \tag{4.22}$$

$$\frac{d\omega}{dt} = \frac{\omega}{k}\left(C_{\omega 1} P - C_{\omega 2} k\omega\right) + \nabla\left[\left(\nu + \frac{\nu_T}{\sigma_\omega}\right)\nabla \omega\right] \tag{4.23}$$

The previous equations may be rewritten in SPH form as follows:

$$\frac{dk}{dt} = P - \varepsilon + \sum_j m_j \frac{\mu_{k,i} + \mu_{k,j}}{\rho_i \rho_j} \frac{k_i - k_j}{\left\|\mathbf{r}_{ij}\right\|^2} \mathbf{r}_{ij} \cdot \nabla W_{ij} \tag{4.24}$$

$$\frac{d\varepsilon}{dt} = \frac{\varepsilon}{k}\left(C_{\varepsilon 1} P - C_{\varepsilon 2}\varepsilon\right) + \sum_j m_j \frac{\mu_{\varepsilon,i} + \mu_{\varepsilon,,j}}{\rho_i \rho_j} \frac{\varepsilon_i - \varepsilon_j}{\left\|\mathbf{r}_{ij}\right\|^2} \mathbf{r}_{ij} \cdot \nabla W_{ij} \tag{4.25}$$

$$\frac{d\omega}{dt} = \frac{\omega}{k}\left(C_{\omega 1} P - C_{\omega 2} k\omega\right) + \sum_j m_j \frac{\mu_{\omega i} + \mu_{\omega,,j}}{\rho_i \rho_j} \frac{\omega_i - \omega_j}{\left\|\mathbf{r}_{ij}\right\|^2} \mathbf{r}_{ij} \cdot \nabla W_{ij} \tag{4.26}$$

In the previous equations:

- P is the turbulent kinetic energy production term, which can be calculated from:

$$P = C_\mu \frac{k^2}{\varepsilon} S^2 = \frac{k}{\omega} S^2 \tag{4.27}$$

where S is the scalar rate of strain, defined by:

$$S = \sqrt{2\mathbf{S} : \mathbf{S}} \tag{4.28}$$

$\mathbf{S}$ is the rate of strain tensor, consisting of 3x3 elements in 3D and 2x2 in 2D. The element S_{ij} of the $\mathbf{S}$ tensor is given by:

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u^i}{\partial x^j} + \frac{\partial u^j}{\partial x^i} \right) \tag{4.29}$$

The calculation of all velocity derivatives, through summation approximations, in order to calculate the respective S_{ij} terms, is rather inefficient. Thus, following Violeau [21], a direct estimation of the scalar rate of strain is obtained through the following equation:

$$S_i^2 = -\frac{1}{2}\sum_j m_j \frac{\rho_i + \rho_j}{\rho_i \rho_j} \frac{\left\| \overline{\mathbf{u}}_i - \overline{\mathbf{u}}_j \right\|^2}{\left\| \mathbf{r}_{ij} \right\|^2} \mathbf{r}_{ij} \cdot \nabla W_{ij} \tag{4.30}$$

- $\mu_{k,i} = \mu_i + \mu_{T,i} / \sigma_k$
- $\mu_{\varepsilon,i} = \mu_i + \mu_{T,i} / \sigma_\varepsilon$
- $\mu_{\omega,i} = \mu_i + \mu_{T,i} / \sigma_\omega$
- The parameters of the turbulence models are:

For the k-ε model:

- C_μ=0.09
- $C_{\varepsilon 1}$=1.44
- $C_{\varepsilon 2}$=1.92
- σ_k=1
- σ_ε=1.3

For the k-ω model:

- β^*=0.09
- $C_{\omega 1}$=5/9
- $C_{\omega 2}$=3/40
- σ_k=2
- σ_ω=2

Apart from the transport equations for k, ε and ω wall functions are needed to prescribe turbulent quantities and velocity at the wall. The great difference of the implementation of a turbulence model in the SPH method that other grid based method, such as the Finite Volumes, is that the wall functions is not sufficient to be employed only on the first layer of fluid particles from the wall, since the kernel function interpolation domain includes the wall particles. Thus, virtual values of velocity, turbulent kinetic energy, turbulent dissipation and specific dissipation rate must be attributed on wall particles in order to enforce a complete kernel support for particles near walls. Following Violeau [20, 21], where the wall functions which are enforced on wall particles are:

- For the parallel to the wall velocity, this is done by assuming the log-law profile:

$$\frac{\overline{u}}{u_*} = \frac{1}{\kappa} \ln\left(\frac{u_* \delta}{\nu_i} \right) + C \tag{4.31}$$

where ν_i is the kinematic viscosity, u_* is the friction velocity, κ is von Karman's constant, equal to 0.41 and C=5.1. δ is a user defined parameter representing the distance from the wall.

Its value is set to a value less than the particle size. The friction velocity is calculated through the following relation:

$$u_* = \sqrt{\frac{\tau_w}{\rho}} \tag{4.32}$$

where τ_w is the wall shear stress, equal to the derivative of the parallel velocity to the wall $u_{//}$, evaluated at the normal, to the wall, direction $\mathbf{n}$:

$$\tau_w = \mu\left(\frac{\partial u_{//}}{\partial n}\right) \tag{4.33}$$

- For the turbulent kinetic energy, it is assumed that:

$$k_i = \frac{u_{*,i}^2}{\sqrt{C_\mu}} \tag{4.34}$$

- The turbulent dissipation rate, according to [18], is calculated from:

$$\varepsilon_i = \frac{u_{*,i}^3}{\kappa\delta} \tag{4.35}$$

- The specific dissipation rate is calculated from:

$$\omega_i = \left(\beta^*\right)\frac{u_{*,i}}{\kappa\delta} \tag{4.36}$$

In order to calculate the described wall functions, the wall shear stress is needed. First of all each wall particle has its own wall normal assigned, considering the boundary geometry (fig. 4.31b). Then each wall particle is linked to a *'sampling particle'*, which is the mirror of the respective wall particle across the boundary, at the normal to the boundary direction. Sampling particles do not affect the flow. They are only used to sample the fluid characteristics at their location. From the velocity sampled and the wall normal, one may calculate the parallel to the wall velocity component $u_{//}$. This velocity component is, then, used to obtain the wall shear stress.

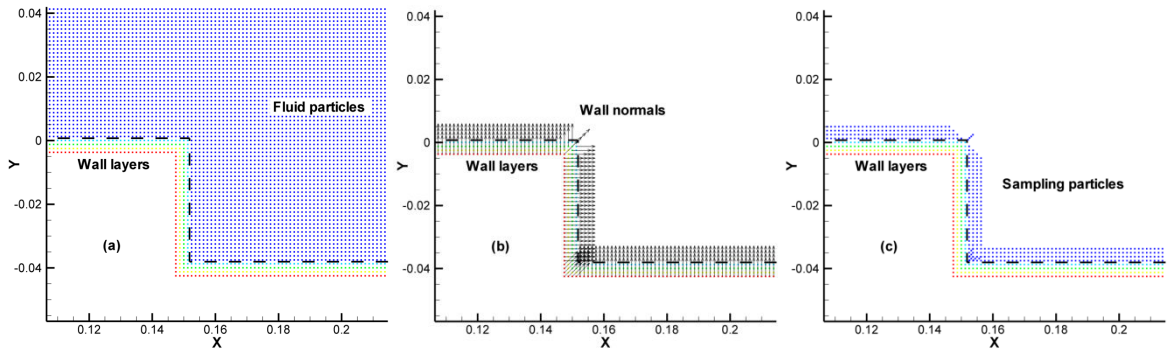


Fig. 4.31. Procedure for calculating wall functions.

Simulation of a turbulent backstep for Re=69610

As a test case for the described turbulence models, a turbulent backward facing step flow simulation was performed, attempting to replicate J.J. Kim's experiment [10]. The geometry of the test case is shown in fig. 4.32.

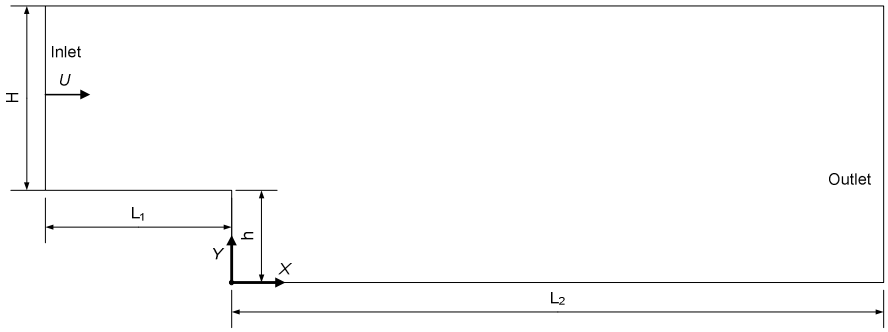


Fig. 4.32. Turbulent backstep geometry

The simulation was performed for a Reynolds number of 69610 (defined as Uh/v) and the properties of the simulated fluid were:

- ρ=1.88553kg/m^3
- μ=1.83698 $\cdot 10^{-5}$ *Pa.s*

Particle redistribution is also periodically used in both simulations with the two turbulence models. Apart from the void formation at higher velocities, simulating high Reynolds number with the SPH method leads to another side-effect. High Reynolds numbers greatly affect particle distributions inducing errors in the interpolation, which leads to unrealistic velocity profiles. In fig. 4.33, the streamlines of the flow field are shown for a high Reynolds number backward facing step flow. It seems that the particle approximation errors lead to an entirely different velocity distribution, forming additional vortices, after the step and at the top of the channel. On the other hand, when redistribution is used, only one large vortex is formed behind the back step.

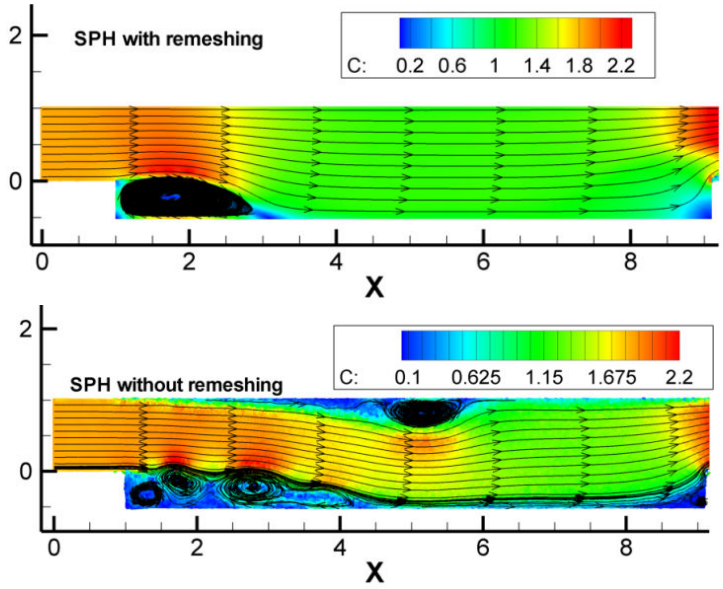


Fig. 4.33. High Re backward facing step simulation. General flow field and streamlines with/without remeshing.

In fig. 4.34 the effects of the redistribution are shown for a high Reynolds number flow (Re=132000) for different time steps. A general observation is that with redistribution the velocity field is smoother. This is valid for all field variables, since during redistribution, interpolations are

performed to determine field variables at the new particle locations. These interpolations inevitably add numerical dissipation, which stabilizes the solution and smoothes fields. Also voids are formed without particle redistribution, which, at later time steps, are filled again with particles. Another notable observation is that at high Reynolds numbers, the high velocity gradients near the walls affect the local particle distribution, resulting to overestimation of the boundary layer thickness. Thus, in order to simulate high Reynolds flows, particle redistribution is necessary.

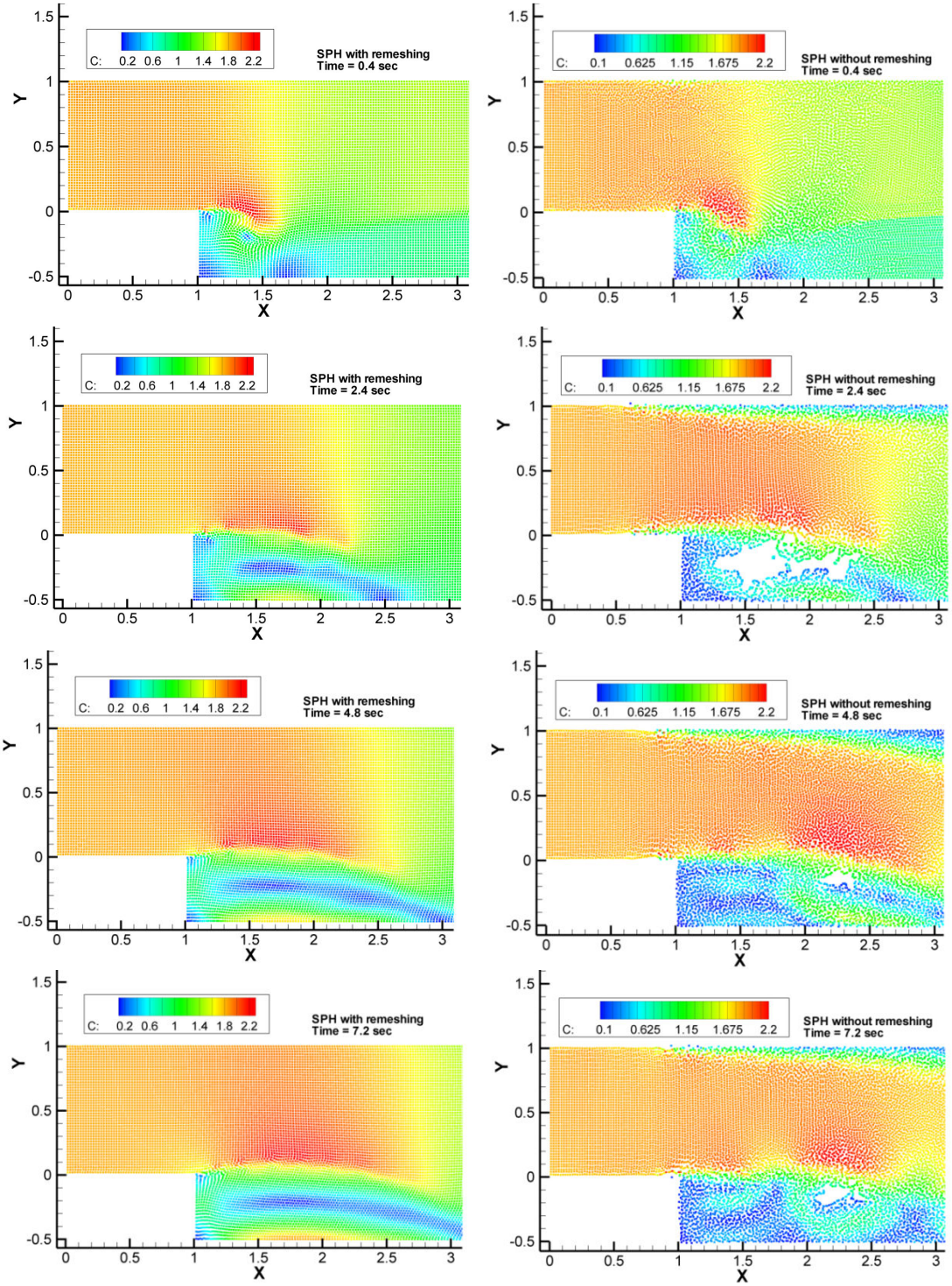


Fig. 4.34. Backward facing step. Left: standard SPH. Right: SPH with particle redistribution. Particle coloring by velocity magnitude.

In the fig. 4.35 the recirculation zone calculated with SPH *k-ε* and *k-ω* turbulence models. According to the experiment, reattachment should occur at $x/h=7.00\pm0.5$. The SPH *k-ε* model predicts reattachment at $x/h=5.2$ (Fluent *k-ε* at $x/h=5.8$). The underestimation of the reattachment point is known for the *k-ε* turbulence model. On the other hand, the SPH *k-ω* model predicts reattachment close to the experimental value, at $x/h=7$ (Fluent *k-ω* at $x/h=7.3$).

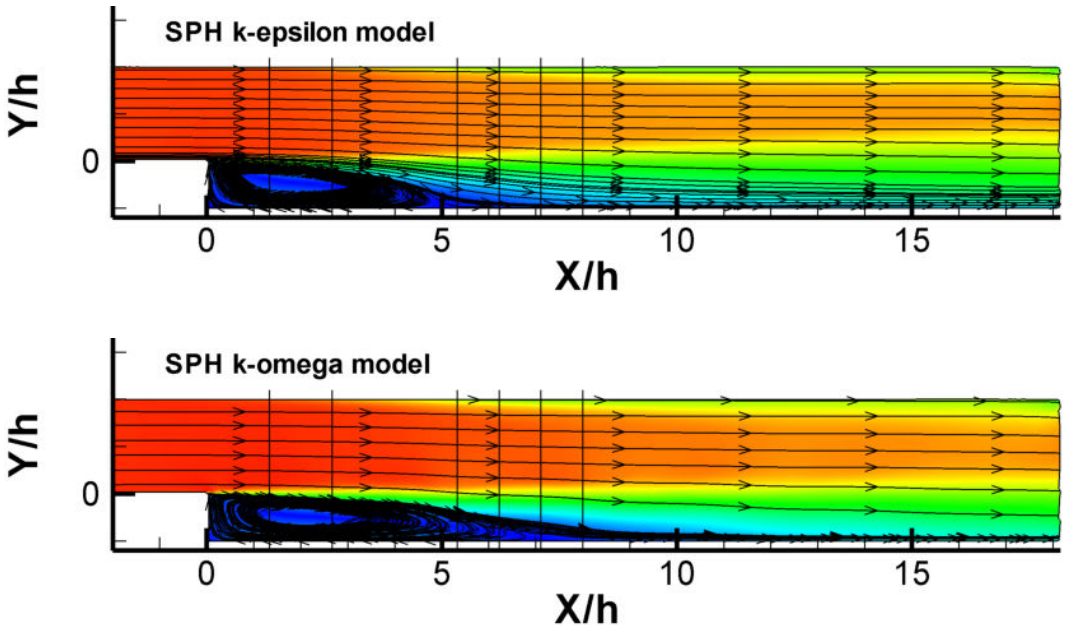


Fig. 4.35. Recirculation bubble for the turbulent backstep case.

In the figures 4.36 and 4.37, the velocity distribution with the SPH is shown for the backstep, in comparison with the experimental values from Kim [10] and a finite volume solution. For both turbulence models, there are differences between the numerical results and the experimental values. However, the SPH and Fluent solution are close to each other; practically identical near the backstep, with differences becoming more pronounced at $x/h>7.11$. In either case, the agreement between the numerical methods is satisfactory.

k-ε turbulence model

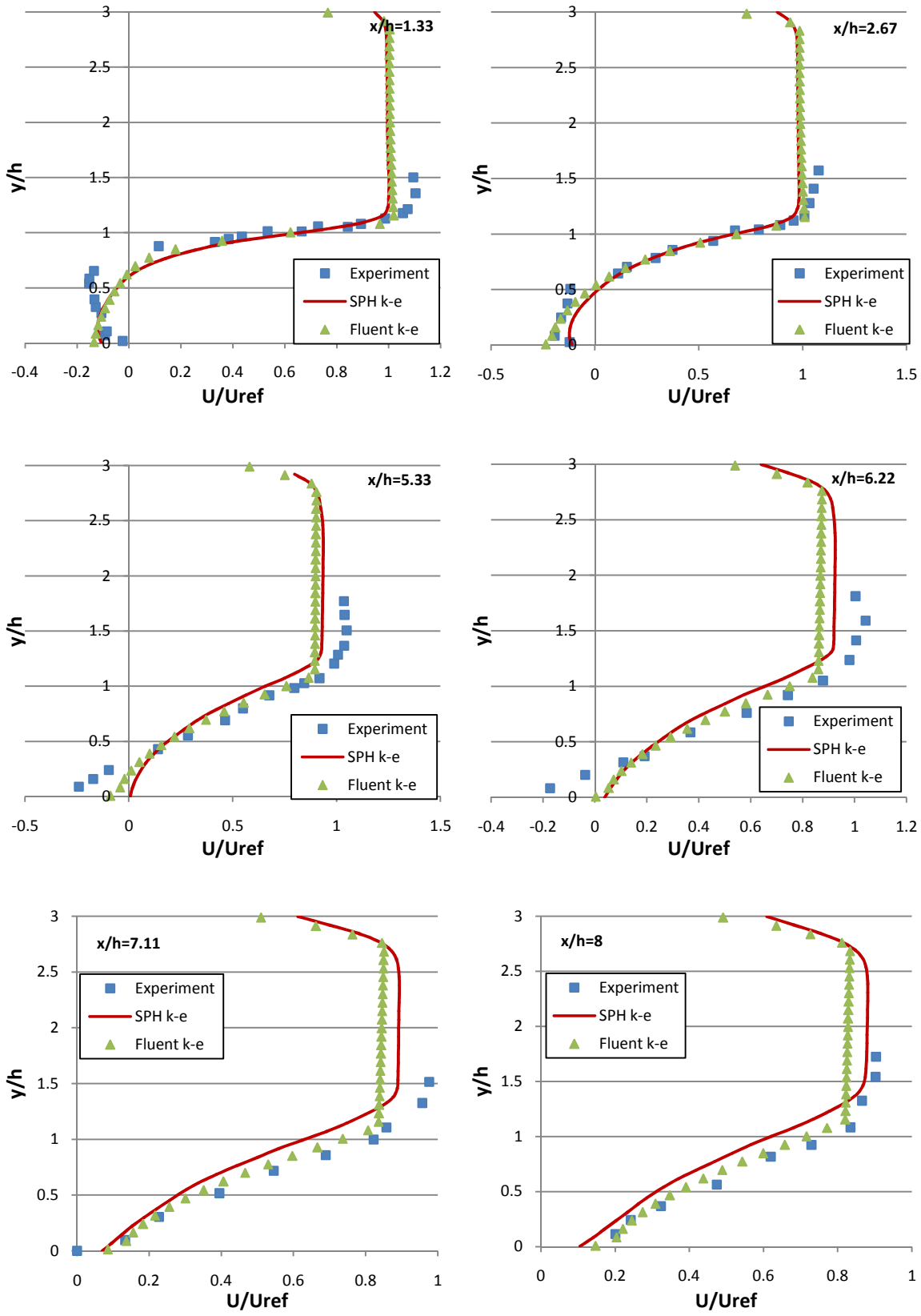


Fig. 4.36. Velocity profiles for the *k-ε* turbulence model using SPH and Fluent. Comparison with experimental data [10].

k-ω turbulence model

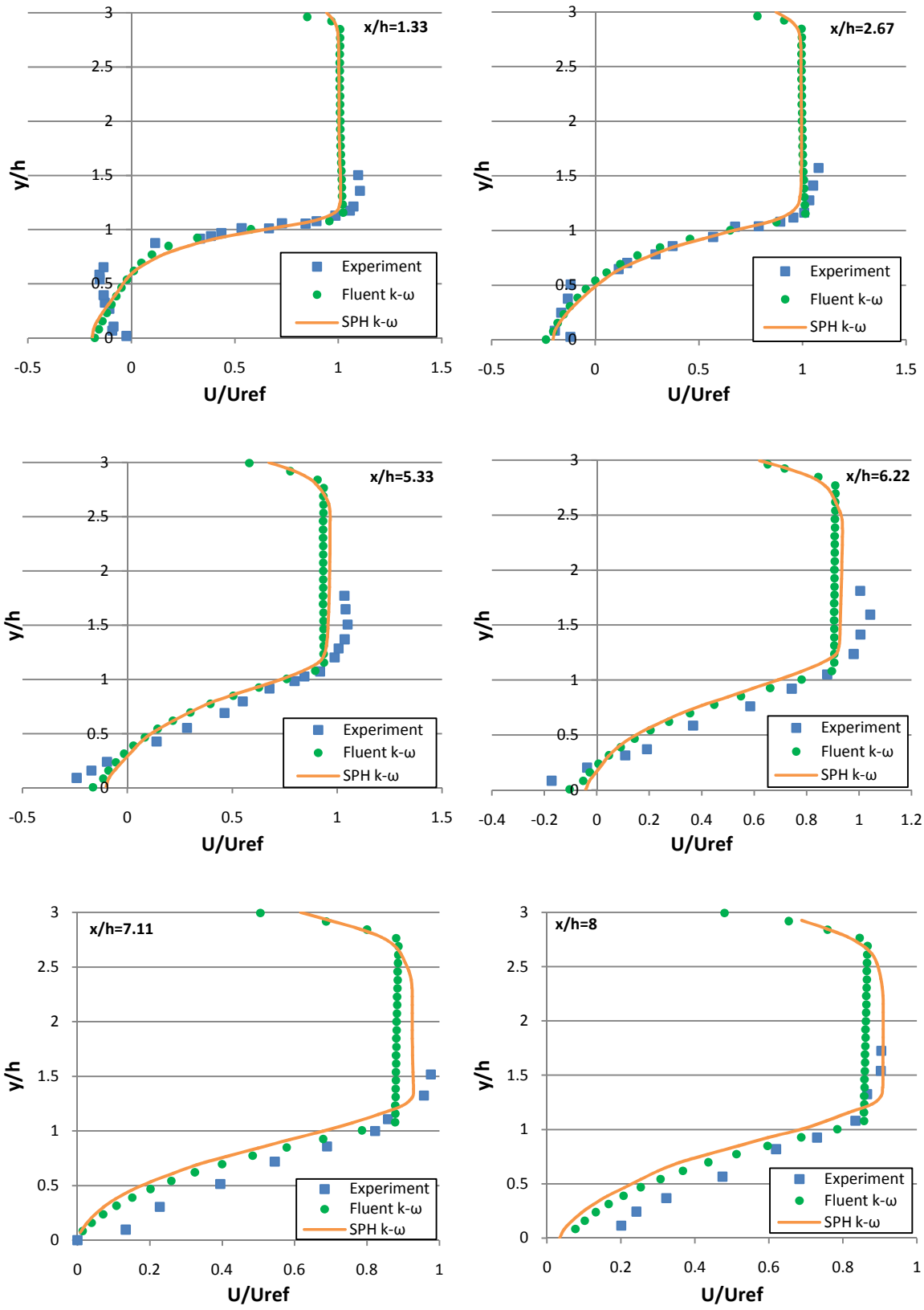


Fig. 4.37. Velocity profiles for the *k-ω* turbulence model using SPH and Fluent. Comparison with experimental data [10].

4.3. Free surface flows

The great advantage of the SPH method is the simulation of free-surface flows without the need of tracking the interface, or employing any special boundary conditions at the free-surface. Free surface simulations in SPH are generally performed by simulating only the primary phase i.e. the denser fluid involved in the simulation. This simplification is possible in multi-phase flows when the density ratio between the different phases is large enough to assume that the heavier phase is governing the flow evolution. Moreover, a two phase simulation would come at a considerably higher computational cost. For testing the free-surface capabilities of the developed SPH algorithm, the jet impingement on a flat plate will be used as a benchmark, concerning the pressure distribution and the free-surface level.

Jet impingement on a flat plate

In this benchmark case, a high velocity water jet impinges a flat plate under different impingement angles. First a particle dependence study was conducted in order to determine the optimal number of particles required to describe properly the underlying phenomena. Then, after determining the adequate resolution, four different impingement angles φ were tested, for 90^o, 60^o, 45^o and 30^o. A simple sketch illustrating the test conditions is shown in fig. 4.38. Finally the results obtained, concerning pressure distribution on the flat plate and the free-surface profile of the formed water sheet, are compared with experimental data [31, 32, 33]. The pressure distribution on the plate is determined by using sampling particles on the plate surface. Sampling is performed using an equation similar to the SPH weighted average summation formula (eq. 3.22). Since there is a steady state solution, time averaging is also performed after the transient part of the impingement, in order to smooth out pressure oscillations.

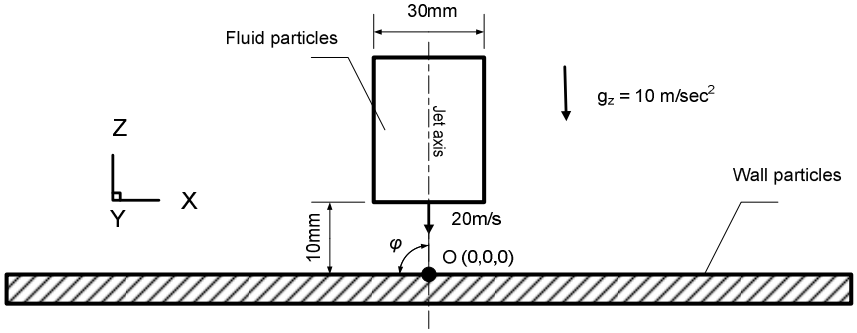


Fig. 4.38. Sketch of the jet impingement

The velocity of the water jet was 20m/s and the diameter 30mm. The artificial speed of sound was set to $10V_{jet}$= 200m/s and density was assumed to be 1000kg/m^3. Viscosity is omitted for the reasons already mentioned in section 4.2.2. For the particle dependence study three different particle resolutions were used: 3mm, 2mm and 1mm. Particles are placed on the jet in a circular manner, using cylindrical coordinates, forming successive rings on the XY plane, beginning from inside and moving to the outer surface of the jet. After placing enough rings to cover the jet radius, a layer of particles is formed. Placing particles in a regular distribution in the jet would lead to particle aligning after the jet impingement, eventually moving in lines (see fig. 4.39 left) and resulting to an unphysical solution, both in terms of free surface and pressure distribution. In order to avoid particle alignment, each ring of particles is rotated in respect to the previous ring. The same is performed for each layer of particles. The distribution of particles at the jet looks as in fig. 4.39 (right).

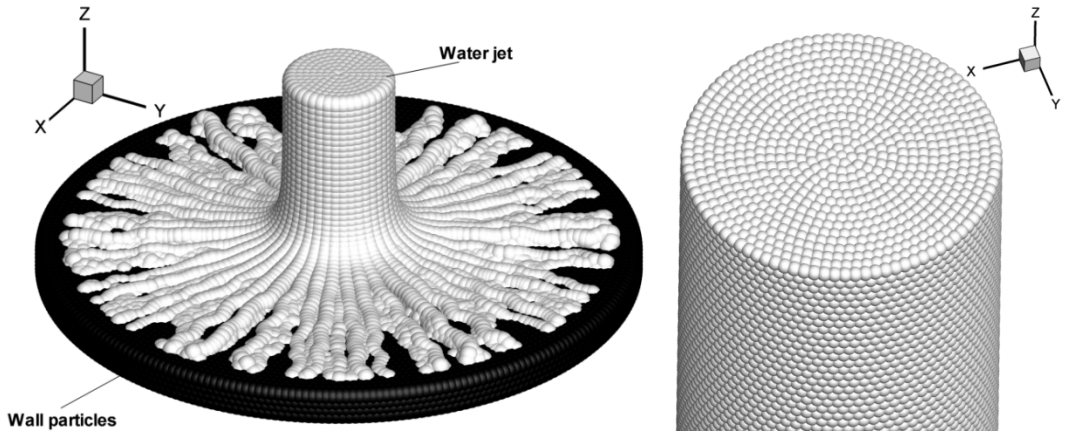


Fig. 4.39. Left: Particle alignment issue at the jet impingement. Right: Particle distribution at the jet to avoid particle alignment

The mass of each particle is calculated from the mass of the ring in which each particle belongs, divided by the number of particles in the specific ring. For example particle *i* located at radius *R*, belong to a ring spanning from $R_{out}=R+dx/2$ to $R_{in}=R-dx/2$ in the radial direction on the *xy* plane (*dx* is the particle spacing). Thus, the surface *S* of that ring is $S=\pi\left(R_{out}{}^2-R_{in}{}^2\right)$ and the volume is $V=dx\cdot S$. Eventually the mass of the ring is $m=V\cdot\rho_{water}$ and the mass of the *i* particle is $m_i=m/N$, where *N* is the number of particles in the specific ring. The number of particles on each ring is an integer number, thus:

$$N=\mathrm{int}\left(\frac{2\pi R}{dx}\right) \tag{4.37}$$

In 90^o degrees impingement it is beneficial to impose symmetry on *yz* and *xz* planes in order to speed up the simulation. In the rest cases symmetry is imposed on the *xz* plane only. Here it has to be highlighted that in the 90^o degrees impingement the solution is axis-symmetric, thus there are infinite symmetry planes and ideally it would be even more efficient to just solve a small slice of the whole computational domain. Practically, in the SPH method, if such a small slice is used, the solution would be subject to particle alignment. Indeed, reducing the computational domain size, reduces the available directions fluid particles may move, eventually leading to particle alignment and to unphysical solutions, especially near the symmetry planes [34]. It was found [34] that using two symmetry planes on *yz* and *xz* planes is a good compromise between accuracy and solution time.

Particle size dependence study

As it was mentioned earlier, a particle dependence study was conducted to determine the optimal particle size / particle number to simulate the jet impingent. The particle sizes used were 3mm, 2mm and 1mm leading to 10, 15 and 30 particles on the diameter of the jet. By comparing the pressure distribution on the plate and the free surface it is found that the particle size of 1mm is able to give accurate results, while particle size of 2mm slightly overestimates pressure at the stagnation point. On the other hand the particle resolution of 3mm leads to totally unreliable results. For the pressure distribution the pressure coefficient is used:

$$C_p = \frac{2p}{\rho V_{jet}^{\ 2}} \tag{4.38}$$

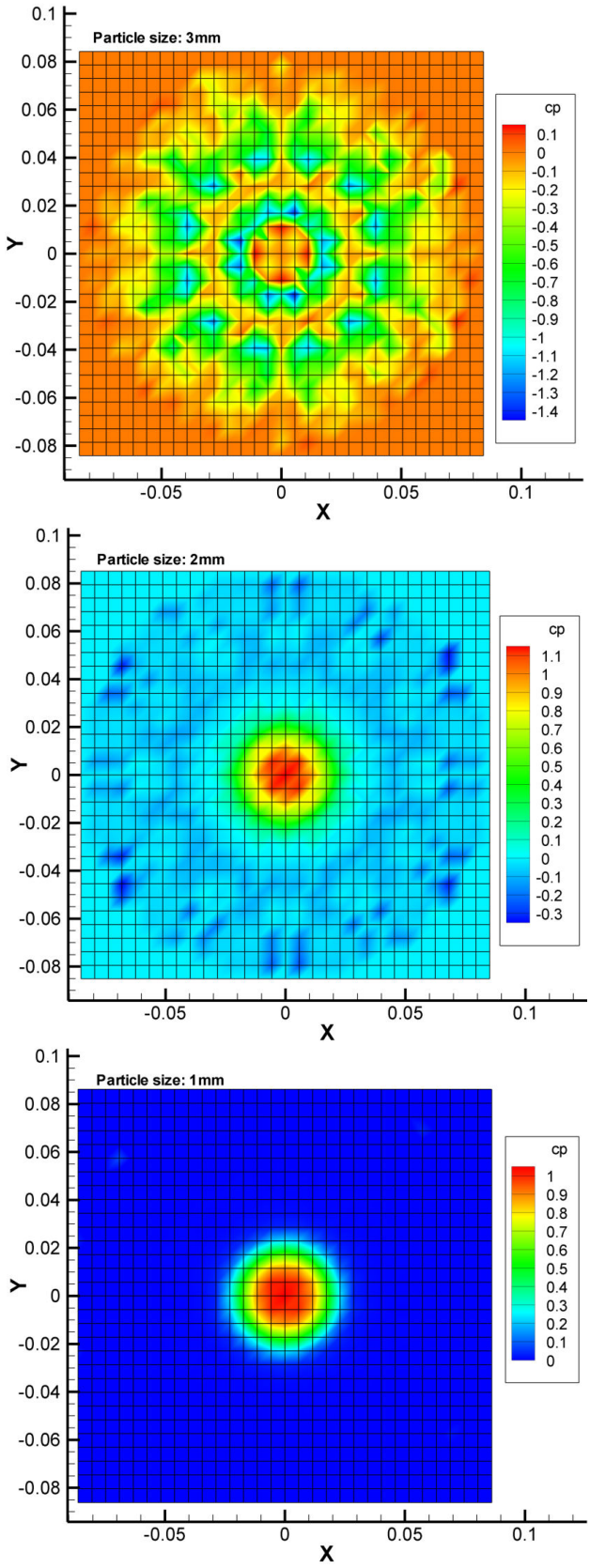


Fig. 4.40. Comparison of the pressure coefficient on the plate for different particle resolutions.

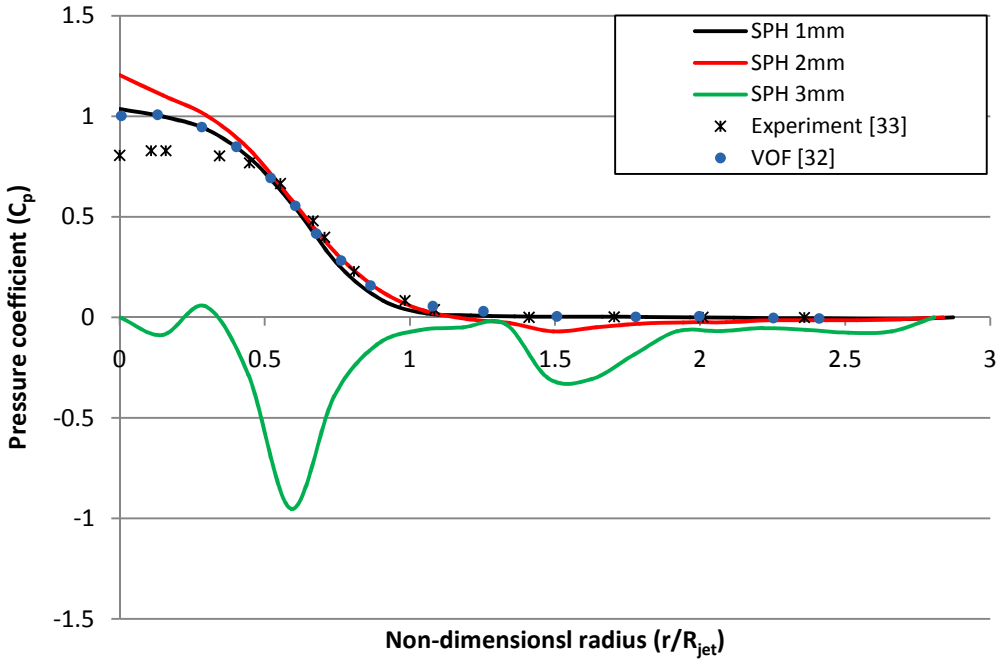


Fig. 4.41. Comparison of the pressure coefficient on the plate for different particle resolutions.

Experimental results, shown in fig. 4.41, under predict the pressure coefficient, C_p, to a value less than unity at the stagnation point, due to the influence of the spear valve adjusting the flow rate at the nozzle forming the water jet [33]. On the other hand, the numerical simulation using Finite Volumes and the Volume of Fluid method (VOF) [32], gives the same pressure coefficient distribution on the plate as the one calculated with SPH.

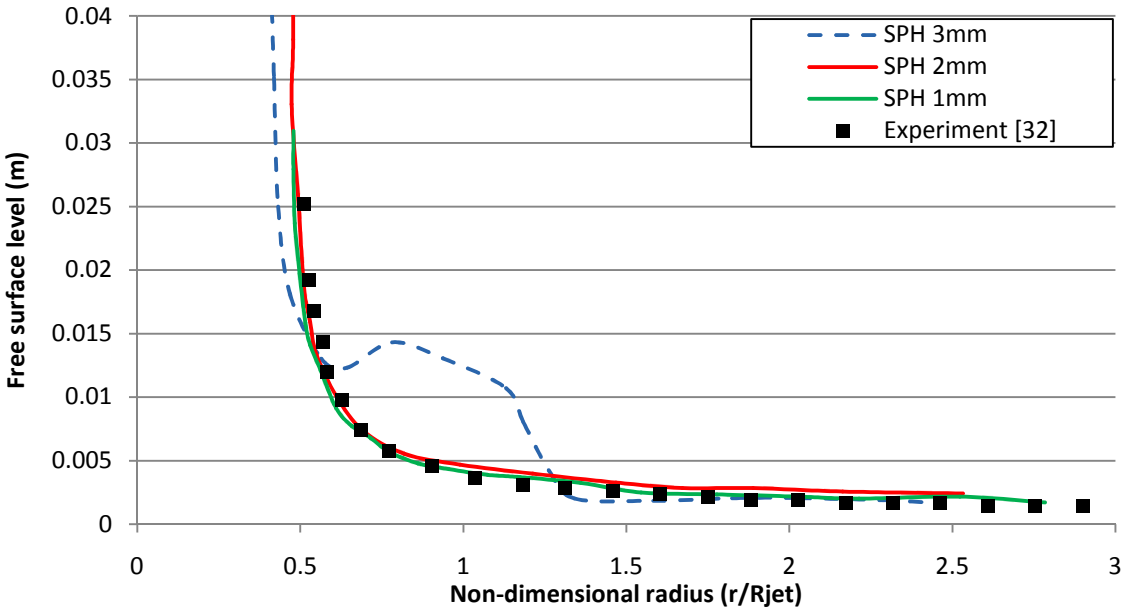


Fig. 4.42. Comparison of the free-surface level of the formed water jet for different particle resolutions.

The particle size giving the best results was the 1mm. This particle discretization will be used to simulate the rest impingement angles.

The time required to simulate 0.01sec of physical time using the SPH algorithm increases, as the particle resolution becomes finer, since more particles are involved in the simulation and need to be solved. As a matter of fact, the SPH algorithm with particle size of:

- 1mm requires ~1day and 13hours without using symmetry planes. If a symmetry plane is used then the required time is approximately reduced by a factor of 2 (~18hours). Similarly if two symmetry planes are used, the computational time is reduced by a factor of 4 (~8.8hours).
- 2mm requires approximately ~2hours, using two symmetry planes
- 3mm requires approximately ~15min, using again two symmetry planes.

All simulations were performed using OpenMP parallelization, on 2xQuad Core Xeon E5405 2.0 GHz computer, utilizing all eight available processors.

Other impingement cases

Using the particle resolution of 1mm the rest impingement cases where simulated. In fig. 4.43 a general view of the flow field is shown, using as a cut-plane the symmetry plane on the *xz* plane. The particle distribution shows the formed water sheet and its color the velocity magnitude. As it is expected by Bernoulli theorem, the velocity of the water sheet should be approximately, since there is the acceleration of gravity, equal to the velocity of the water jet.

The largest part of the flow moves tangential to the slope at the direction of the projected velocity of the jet on the slopped surface. A small part of the flow moves to the opposite direction; this part lessens as the impingement angle φ reduces. However, velocity is underestimated at that region, since particles are scarce, reducing the accuracy of the SPH interpolations.

Results regarding the free surface and the pressure distribution are shown in fig. 4.44-4.46, determined on the XZ symmetry plane; for the 30^o and 60^o degrees impingement a comparison is made with experimental values from Kvicinsky's results [32]. For the 45^o jet impingement results are compared with the results of the Fluent software. The spatial discretization used by Fluent is the same as the one selected for the SPH method, i.e. 1mm. An indicative view of the computational mesh used is shown in fig. 4.48.

From the comparison of the results it can be deduced that the SPH method is capable of describing the free-surface accurately, in respect to the experimental values and the results from the CFD program. On the other hand, there are deviations in the pressure estimations:

- At the 60^o jet impingement there is a small deviation near the stagnation point
- Generally in all cases, a negative pressure is predicted by the SPH method after the stagnation point. This issue is directly linked to the particle distribution in the area. As it was mentioned earlier, particle interpolations are no longer valid in the specific area, since there are not enough neighbors to fill the kernel support radius. Ideally this would be solved by using a much finer particle resolution, but this would result to unacceptable calculation times, even using parallel processing.

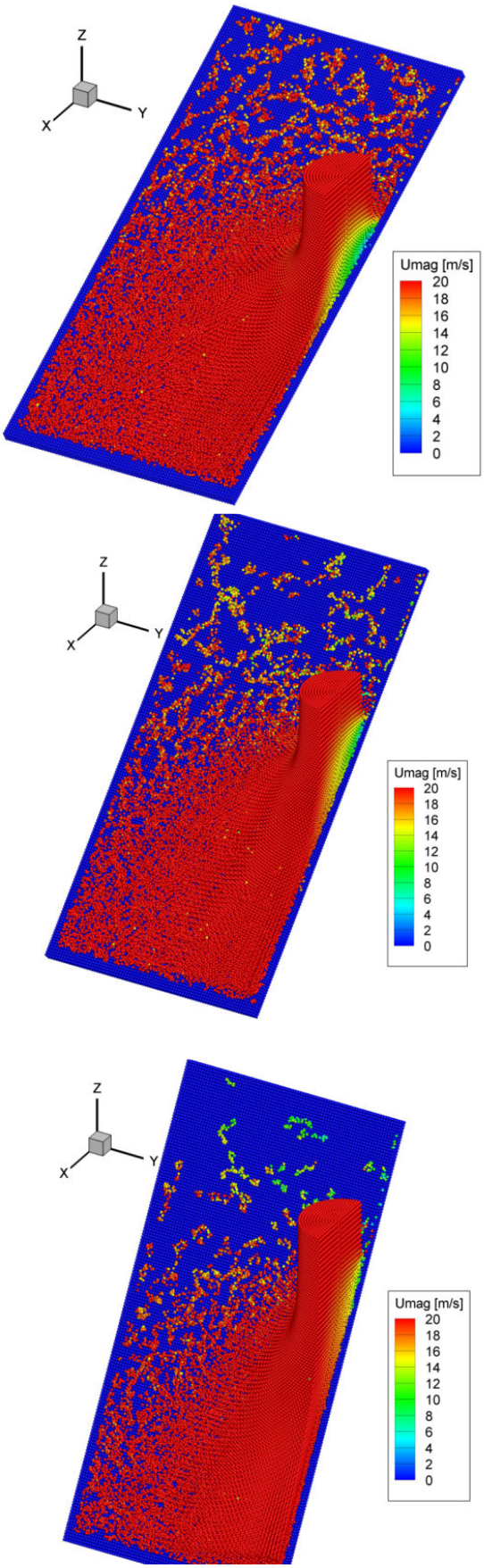


Fig. 4.43. Jet impingement on flat plate. Impingement angle, from top to bottom: 60°, 45°, 30°

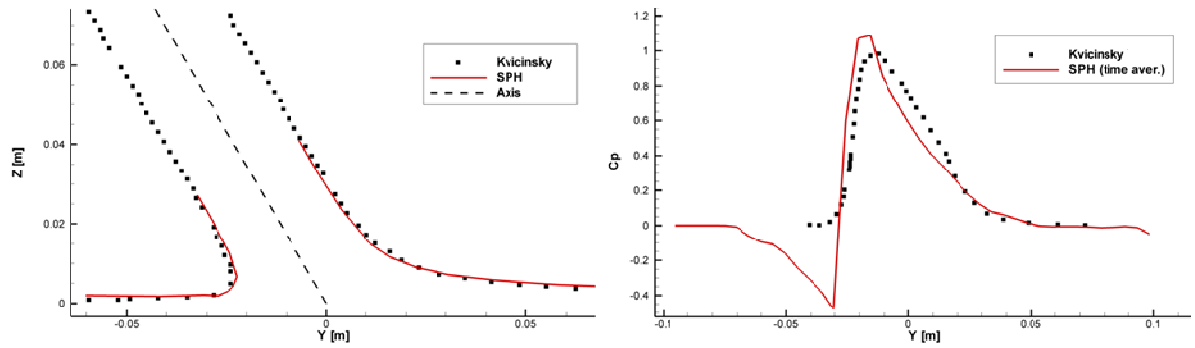


Fig. 4.44. 60^{o} jet impingement. Left: Comparison of the free surface. Right: Comparison of pressure distribution.

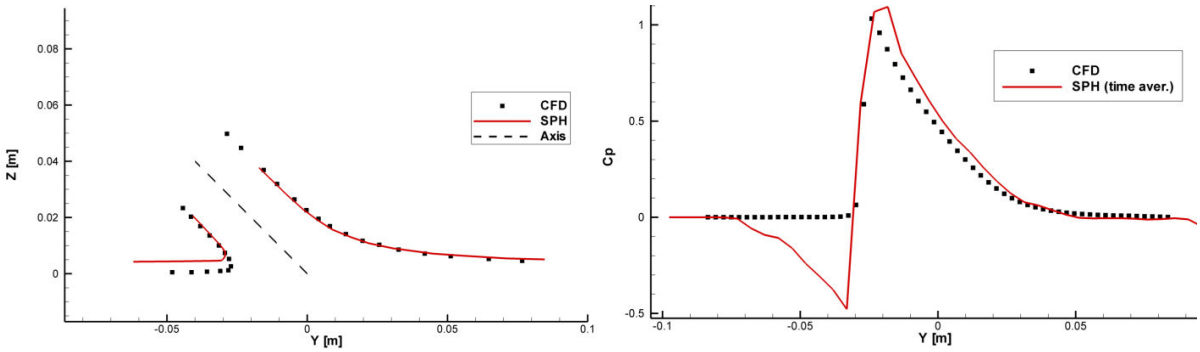


Fig. 4.45. 45^{o} jet impingement. Left: Comparison of the free surface. Right: Comparison of pressure distribution.

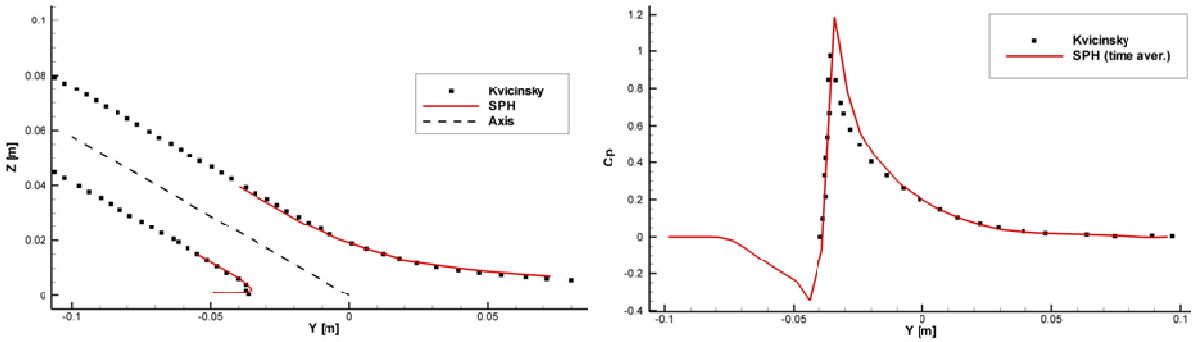


Fig. 4.46. 30^{o} jet impingement. Left: Comparison of the free surface. Right: Comparison of pressure distribution.

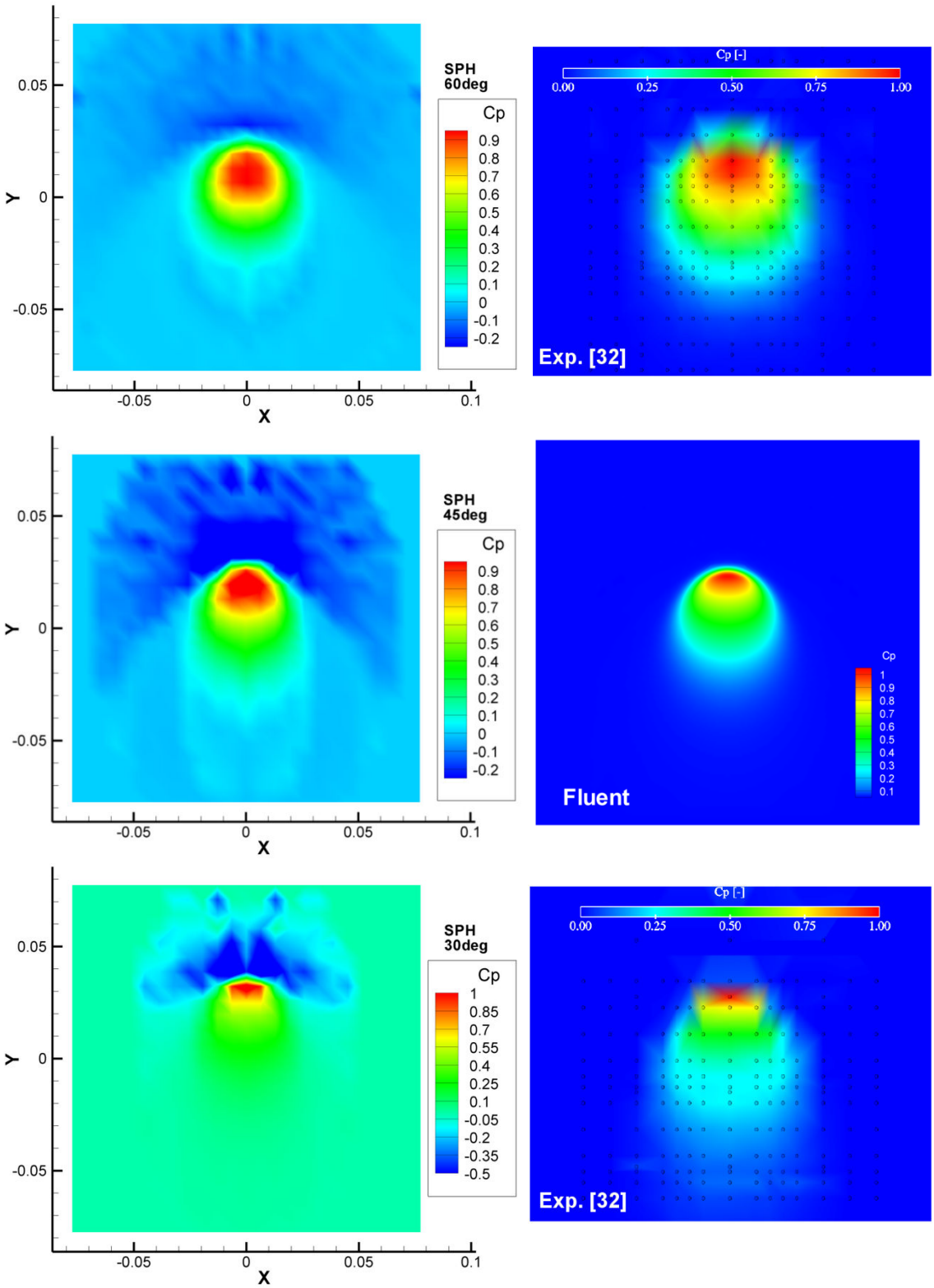


Fig. 4.47. Pressure coefficient distribution on the flat plate impingement under different angles. From top to bottom: 60°, 45°, 30° degrees impingement.

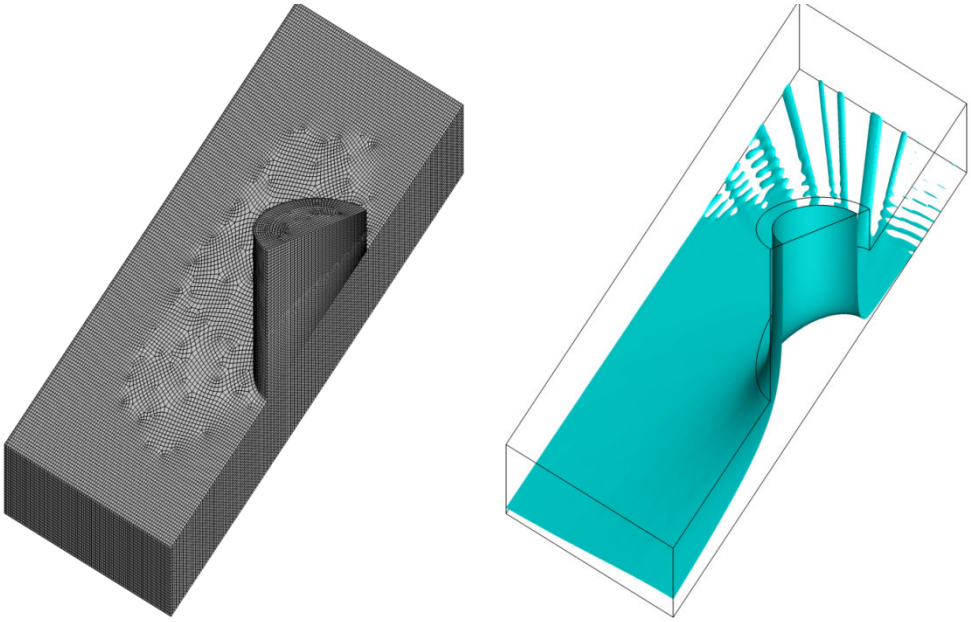


Fig. 4.48. Left: Fluent computational mesh. Right: Calculated free surface.

4.4. Concluding remarks

In the present chapter, the standard SPH algorithm is validated at several academic test cases, such as the shock tube, the rotation of a square patch of fluid and the deformation of a circular patch of fluid. The developed SPH algorithm is able to properly replicate the theoretical results.

Moreover, the application of the SPH method to viscous flows is examined. The SPH method is able to simulate accurately viscous flows but, at the examined cases, uneven particle distributions may deteriorate the solution quality. This was particularly shown in the shear cavity case and in the backward facing step case. At high Reynolds number it is necessary to use particle redistribution, in order to control the particle arrangement. In any case, it is possible to integrate traditional turbulence models, such as the k-ε and k-ω turbulence models, and reproduce turbulent flow characteristics. However, the fact that particle redistribution would be required renders direct application of turbulence at free surface simulations difficult.

The water jet impingement case shows that the SPH method is capable in simulating free surface flows, since it is able to properly reproduce the free surface. On the other hand, the inherent weakness of the SPH method in pressure estimation leads to inaccurate pressure distributions. For that reason, for all the rest cases presented with the standard SPH, the force estimation on walls is performed using the boundary forces reaction (eq. 3.88 and 3.89).

References

[1] G.R. Liu, M. B. Liu, “Smoothed particle hydrodynamics”, World scientific publishing, 2003, ISBN 981-238-456-1

[2] R. J. Leveque, “Finite volume methods for hyperbolic problems”, Cambridge University Press, ISBN 0-511-04219-1.

[3] E.F. Toro, “Riemann Solvers and Numerical methods for fluid dynamics: a practical introduction”, 3rd edition, Springer, 2009, ISBN 978-3-540-25202-3, e-ISBN 978-3-540-49834-6, DOI: 10.1007/978-3-540-49834-6.

[4] A. Khayyer, H. Gotoh, "Enhancement of the stability and accuracy of moving semi implicit method", Journal of Computational Physics, vol. 230, Issue 8, 20, p. 3093–3118, April 2011

[5] D. Le Touze, A. Colagrossi, "Free-Surface Prototype Problems Suitable to Investigate Particle Methods", 20th International Workshop on Water Waves and Floating Bodies, Spitsbergen, Norway 2005.

[6] J. Ren, J. Ouyang, B. Yang, T. Jiang, H. Mai, "Simulation of container filling process with two inlets by improved smoothed particle hydrodynamics (SPH) method", International Journal of Computational Fluid Dynamics, vol. 25, p. 365-386.

[7] J. J. Monaghan, "Simulating free surface flows with SPH", Journal of Computational Physics, vol. 110, p.399-406, 1994.

[8] R. Vacondio, P. Mignosa, "Incompressible finite pointset method for free surface flow", Proceedings of the 4th Spheric Workshop, Nantes, 27-29 May 2009, France.

[9] J. P. Morris, P. J. Fox, Y. Zhu, "Modeling low Reynolds number incompressible flows using SPH", Journal of Computational Physics, vol. 136, p. 214-226, 1997.

[10] S. Tsaggaris, "Fluid Mechanics", Symeon Press, 2005, ISBN: 960-7888-55-3.

[11] B.F. Armaly, F. Durst, J. C. F. Pereira, B. Schonung, "Experimental and theoretical investigation of backward facing step flow", Journal of Fluid Mechanics, vol. 127, p. 473-496, 1983.

[12] R.Issa, E.S. Lee, D. Violeau, D.R. Laurence, "Incompressible separated flow simulations with the smoothed particle hydrodynamics gridless method", International Journal for Numerical Methods in Fluids, vol. 47, p. 1101-1106, 2005. DOI: 10.1002/fld.864

[13] A.K. Chaniotis, D. Poulikakos, P. Koumoutsakos, "Remeshed Smoothed Particle Hydrodynamics for the simulation of viscous and heat conduction flows", Journal of Computational Physics, vol. 182, p.67-90, 2002.

[14] A.K. Chaniotis, C.E. Frouzakis, J.C. Lee, A.G. Tomboulides, D. Poulikakos, K. Boulouchos, "Remeshed smoothed particle hydrodynamics for the simulation of laminar chemically reactive flows", Journal of Computational Physics, vol. 191, p. 1-17, 2003.

[15] P. Chatelain, G.-H. Cottet, P. Koumoutsakos, "Particle mesh hydrodynamics for astrophysics simulations", International Journal of Modern Physics, vol. 18, No. 4, p. 610-618, 2007.

[16] N. Galagali, "Algorithms for particle remeshing applied to smoothed particle hydrodynamics", Master Thesis, Massachusetts institute of Technology, September 2009.

[17] G. Ramos-Becerra, C. Moulinec, D.R. Emerson, X.J. Gu, "Inlet-outlet boundary conditions and truly incompressible SPH", Proceedings of the 4th Spheric workshop, Nantes, France 2009.

[18] L. Davidson, "An introduction to turbulence models", Department of Thermo and Fluid dynamics, Chalmers University of Technology, Goteborg, Sweden, November 2003.

[19] H. Versteeg, W. Malalasekera, "An Introduction to Computational Fluid Dynamics: The Finite Volume Method", Pearson, Prentice Hall, 2nd edition, 2007, ISBN-10: 0131274988, ISBN-13: 978-0131274983.

[20] D. Violeau, C. Buvat, K. Abed-Meraim, E. de Nanteuil, "Numerical modeling of boom and oil spill with SPH", Coastal Engineering, vol. 54, p. 895-913, 2007.

[21] D. Violeau, R. Issa, "Numerical modeling of complex turbulent free surface flows with the SPH method: an overview", International Journal for Numerical Methods in Fluids, vol. 53, p. 277-304, 6 July 2006.

[22] D. Violeau, R. Issa, S. Benhamadouche, J. Chorda, M. M. Maubourguet, "Modelling a fish passage with SPH and Eulerian codes: the influence of the turbulence closure", 3rd Spheric workshop, Lausanne, Switzerland.

[23] S. Shao, C. Ji, "SPH computation of plunging waves using a 2D sub-particle scale (SPS) turbulence model" International journal of Numerical Methods in Fluids, vol. 51, p.913-936, 2006.

[24] R.A. Dalrymple and B.D. Rogers, "Numerical modeling of water waves with the SPH method", Coastal Engineering, vol. 53, p.141-147, 2006.

[25] T.S. Ting, M. Prakash, P.W. Cleary, M.C. Thompson, "Simulation of high Reynolds number flow over a backward facing step using SPH", Anziam Journal, j.47 (EMAC 2005), p.C292-C309, 2006.

[26] S. Shao, "Incompressible SPH simulation of wave breaking and overtopping with turbulence modelling", International Journal for Numerical Methods in Fluids, vol. 50, p. 597-621, 2006.

[27] J.J Monaghan, "An SPH turbulence model", 4th Spheric Workshop, Nantes France, 2009.

[28] J.J. Monaghan, "Smoothed Particle Hydrodynamics", Reports on progress in physics, vol. 68, p. 1703-1759, 2005, doi: 10.1088/0034-4885/68/8/R01

[29] D.C. Wilcox, "Turbulence modeling for CFD", 2nd edition DCW Industries Inc., July 1998 , ISBN-10: 0963605151, ISBN-13: 978-0963605153.

[30] S. B. Pope, "Turbulent flows", Cambridge University Press, October 16, 2000, ISBN-10: 0521598869, ISBN-13: 978-0521598866.

[31] S. Kvicinsky, F. Longatte, J.-L. Kueny and F. Avellan, "Free surface flows: experimental validation of volume of fluid (VOF) method in the plane wall case", Proceedings of the 3rd ASME/JSME Joint fluids engineering conference, July 18-23, San Fransisco, California.

[32] S. Kvicinsky, "Method d'analyse des ecoulements 3D a surface libre: application aux turbines Pelton", These N°2526, Ecole Polytechnique Federale de Lausanne, 2002.

[33] J.C. Marongiu, F. Leboeuf, E. Parkinson, "Numerical simulation of the flow in a Pelton turbine using the meshless method smoothed particle hydrodynamics: a new simple boundary treatment", Proceedings of the Institution of Mechanical Engineers, vol. 221, Part A: Journal of Power and Energy, 2007.

[34] P. Koukouvinis, J. Anagnostopoulos, "Flow modelling in the injector of a Pelton turbine", 4th Spheric Workshop, Nantes, France, 2009.

Chapter 5

Application of the SPH method in impulse hydraulic turbines

In this chapter several realistic applications of the SPH method in impulse hydraulic turbines are discussed. These applications have to do with the:

- simulation of the flow inside a Petlon turbine nozzle
- simulation of the behavior of an impulse turbine deflector
- simulation of the flow in impulse turbines (Turgo turbine and Pelton turbine)

5.1. Flow inside a Pelton turbine nozzle

The first real application of the SPH method is on the flow modeling of the free surface flow formed at the *nozzle spear valve* (called also as *injector*) of a Pelton impulse turbine. Impulse turbine operation requires a high velocity water jet to impinge on a runner with specific geometry. The high velocity water jet is formed at the turbine *nozzle* located at the turbine casing. In fig. 5.1 (left), there is a photograph of the model Pelton turbine installed in the Laboratory of Hydraulic Turbomachines (NTUA), where the turbine nozzles are visible, and in fig. 5.1 (right) a zoom at the nozzle, where the *needle* (also called as *spear*) is also visible.

Fig. 5.1. Left: Laboratory Pelton turbine model. Right: Turbine nozzle. The needle and water jet are also visible.

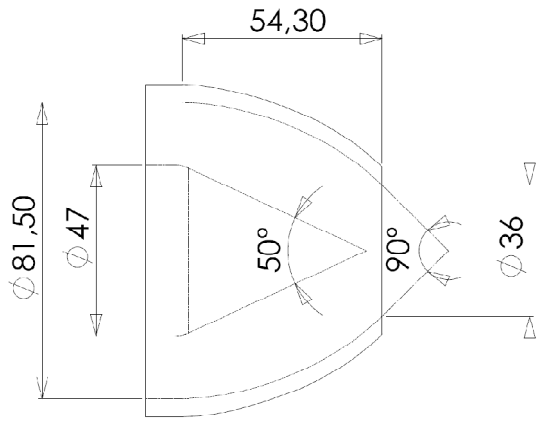


Fig. 5.2. Drawing of the Pelton nozzle with spear valve (dimensions in mm).

The purpose of the nozzle is to transform the hydraulic head into high velocity jet required for the turbine operation. The needle moves at the axial direction of the nozzle, regulating the flow rate (*needle stroke*).

The SPH algorithm was adapted in order to simulate the flow through the nozzle for different needle openings. Eventually the characteristic nozzle curve is obtained, linking the flow rate through the nozzle to the needle stroke. Then the characteristic curve is compared to the experimental data from the Pelton installation.

Case set-up

The geometry of the nozzle and needle is axis-symmetric and the basic dimensions are shown in fig. 5.2. For modeling, the geometry was simplified as shown in fig. 5.3. The geometry is represented as a planar 2d-sketch on the *yz* plane, which consists of line and curve segments. C_1, C_2 curve segments are assumed to be polynomial curves of 3rd degree in order to satisfy the conditions required (slope at *A*, *B* endpoints and coordinates of endpoints).

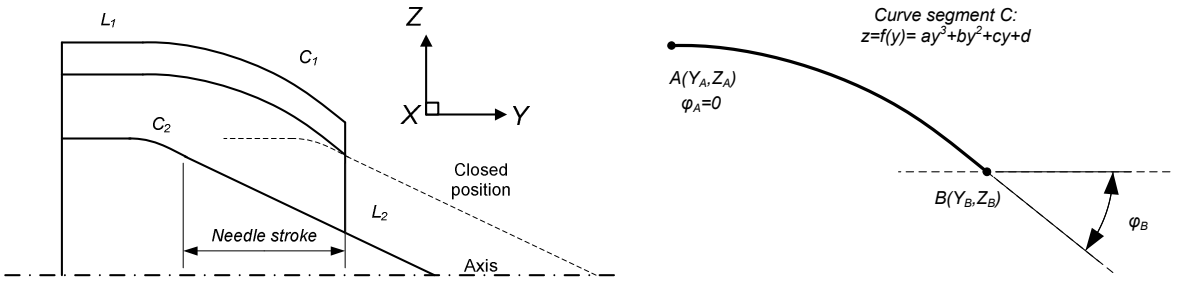


Fig. 5.3. Left: 2d-sketch, showing basic lines/curves. Right: Curve segment representation

After creating the 2d outline, then the simulated geometry is formed as a surface of revolution around the *y*-axis. For solving the flow, plane symmetry is assumed on *yz* and *xy* planes.

At the nozzle inlet, the total pressure is assumed to be constant. This resembles the actual situation of a hydro turbine operation, since the total height is practically constant, considering short term operation. In order to properly impose the boundary condition, several layers of buffer particles

are introduced at the nozzle inlet, as shown in fig 5.4. The buffer particles are placed for the same reason as in the backward facing step case (see chapter 4); in order to fill the support domain of fluid particles at the inlet, while imposing the desired pressure value. Since the total pressure is fixed, static pressure is deduced from the definition of total pressure, i.e. total pressure minus the fluid dynamic pressure:

$$p_{stat} = p_{tot} - \frac{1}{2}\rho u^2 \tag{5.1}$$

The only unknown needed to impose the static pressure, is the fluid velocity. This is obtained by averaging the fluid velocity of the first layers of fluid particles; to be more precise, for all particles with distance less or equal than the kernel support radius ($2.5h$) from the buffer region. The average velocity is also imposed on the buffer particles in order to enforce zero gradient boundary condition for velocity at the inlet. Moreover, average velocity is used to determine the volume and, consequently, mass flow rate.

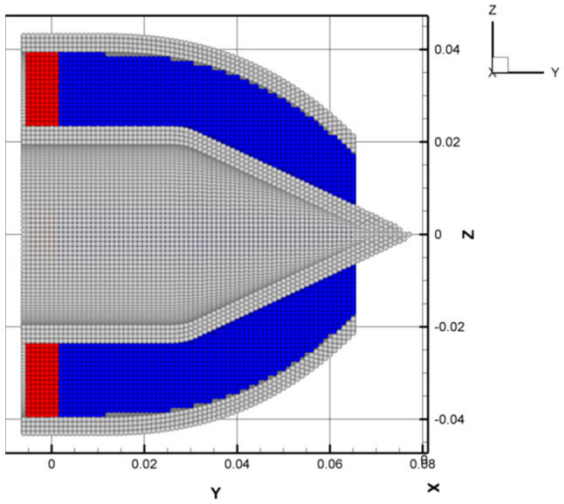


Fig. 5.4. Section view of the nozzle. Grey particles are wall particles, blue particles represent the fluid and red particles represent the particle buffer, whose pressure and velocity are imposed.

The simulations were performed for a total gauge pressure at inlet equal to 10bar for various needle strokes ranging from 43mm to 8mm. Nozzle openings less than 8mm were not simulated because particle resolution became comparable to the nozzle opening. The sound speed was set to 540m/s and fluid density was set to water density (998.2kg/m^3). Simulation time was 10msec; the flow is initially transient and after ~6msec reaches a steady state. A particle dependence study was performed to find the resolution of particles which is required to describe accurately the flow field. In all cases as initial condition it is assumed that the nozzle is filled with water.

Particle dependence analysis

In order to decide for the particle discretization, a particle-dependence analysis was performed for different particle sizes (and consequently different numbers of particles) 2.5mm, 2mm, 1.5mm and 1mm for a nozzle opening of 36mm. As can be seen from the following diagram (fig. 5.5), for particle sizes less than 1.5mm the flow rate results practically do not change. For the rest simulations the 1mm particle size was used.

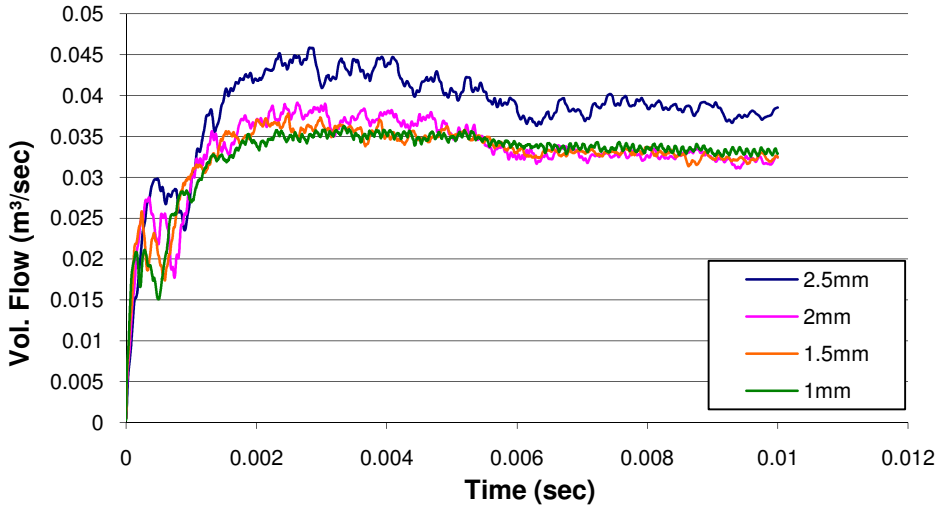


Fig. 5.5. Flow rate as a function of time for the particle dependence study.

Simulation results & comparison with experiment and Fluent solution

In this section, the results from the SPH simulation of the nozzle are compared with the experimental nozzle characteristic curve. Also the free-surface location is compared with Fluent axis-symmetric solution, since it was not possible to obtain accurate measurements of the free surface from the experimental installation.

The experimental characteristic curve was calculated by taking measurements of the flow rate through the nozzle for different needle strokes. The needle stroke is adjusted properly by turning the needle handle. Pressure is measured by a pressure transducer located before the nozzle in order to avoid including the hydraulic losses of the piping. Finally flow rate is measured by an electromagnetic flow meter. Since flow in the experimental installation is generated by a pump, it is rather difficult to ensure constant total pressure for all measurements; when the nozzle opening increases, pressure will decrease for the same pump speed. Thus, the pump speed must be increased. However, it is impossible to adjust pump speed so that total pressure matches exactly the simulation boundary condition of 10bar. For that reason, the pump speed was adjusted in order the static pressure at the measuring transducer to be ~10bar and then flow rate was properly deduced for a total pressure of 10bar. To be more specific:

$$Q' = Q\sqrt{\frac{P_{tot}'}{P_{tot}}} \tag{5.2}$$

$$P_{tot} = P_{stat} + \frac{1}{2}\rho u^2 = P_{stat} + \frac{1}{2}\rho\left(\frac{Q}{A}\right)^2 \tag{5.3}$$

- P_{stat} is the measured static pressure
- Q is the measured flow rate
- A is the surface of the cross section where pressure was measured

- P_{tot}' is the total pressure at which flow rate will be deduced (in our case 10bar)
- Q' is the deduced flow rate

In fig. 5.6 – 5.9, indicative instances are shown from the simulated flow with the SPH algorithm, showing the free-surface location and the formed water jet velocity. A comparison is also made with the free surface shape calculated by Fluent, used as reference. Generally the SPH algorithm predicts accurately the free surface shape and the water jet diameter (see also fig. 5.10). Nevertheless, it underestimates the velocity of the water jet, especially for small openings (see fig. 5.9). Following Bernoulli theorem, the velocity of the water jet is a function of the pressure difference, when viscosity is absent:

$$u_{jet} = \sqrt{\frac{2\left(P_{tot\_in} - P_{out}\right)}{\rho}} = 44.8 m/s$$

This is a side effect from the wall boundary conditions used; when particles move away from the wall then continuity equation predicts a decrease in their density, which eventually results to a pressure gradient decelerating particles moving away from walls. This is also seen at the edges of the water jet, where velocity is also underestimated.

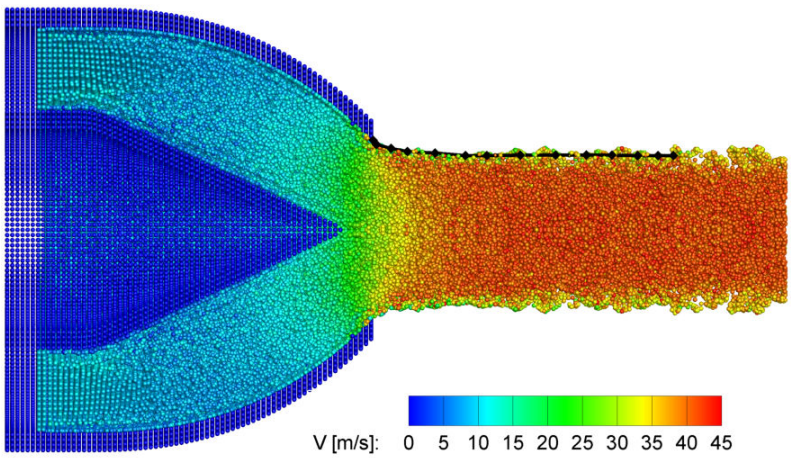


Fig. 5.6. Nozzle opening 43mm. Particles are colored according to their velocity. The black diamonds/line show the free surface calculated with Fluent.

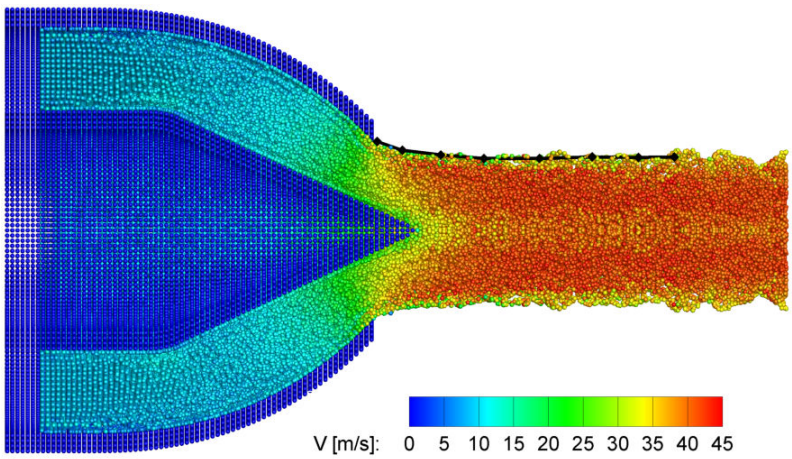


Fig. 5.7. Nozzle opening 28mm. Particles are colored according to their velocity. The black diamonds/line show the free surface calculated with Fluent.

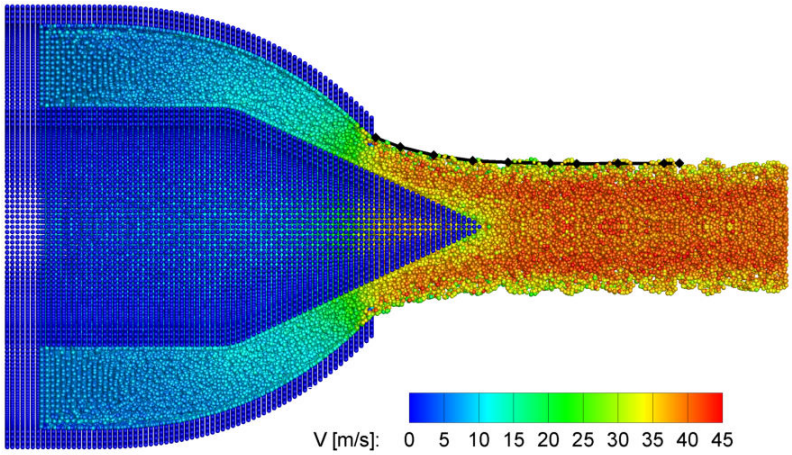


Fig. 5.8. Nozzle opening 16mm. Particles are colored according to their velocity. The black diamonds/line show the free surface calculated with Fluent.

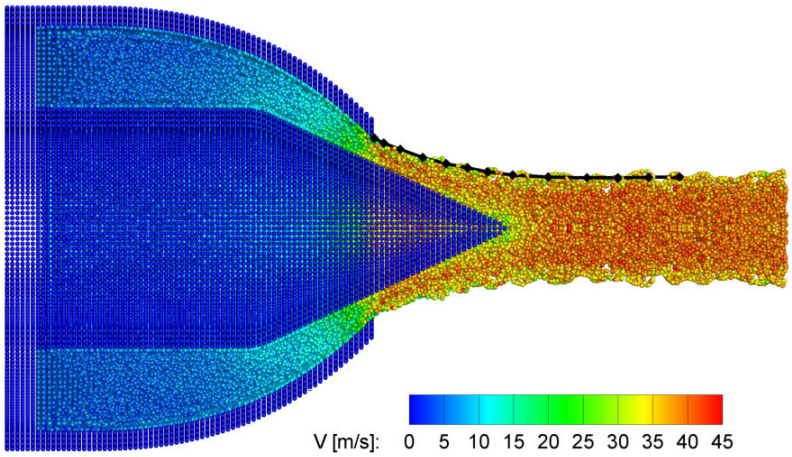


Fig. 5.9. Nozzle opening 10mm. Particles are colored according to their velocity. The black diamonds/line show the free surface calculated with Fluent.

In any case the predicted diameter of the water jet is in accordance with the jet diameter calculated by Fluent for various nozzle openings, as shown in fig 5.10.

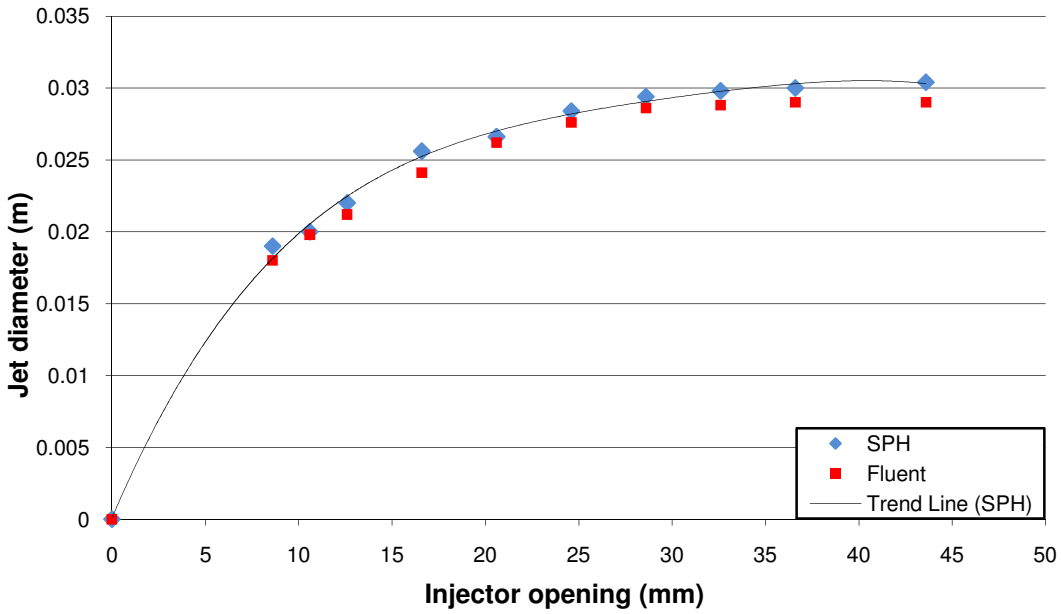


Fig. 5.10. Diameter of the jet formed after the nozzle. Comparison of the SPH and Fluent results.

The volumetric flow through the injector is calculated as $\dot{Q}_{vol} = A\,U$, where A is the surface of the section between the nozzle and the needle and U is the average axial velocity. Due to the unsteady nature of SPH, there are oscillations at the value of average velocity and, as a result, at the volumetric flow. These oscillations become greater as the injector opening becomes smaller (see fig. 5.11). The average volumetric flow rate is, hence, calculated after a certain transient period where the amplitude of the oscillation becomes constant and the average volume flow rate stabilizes.

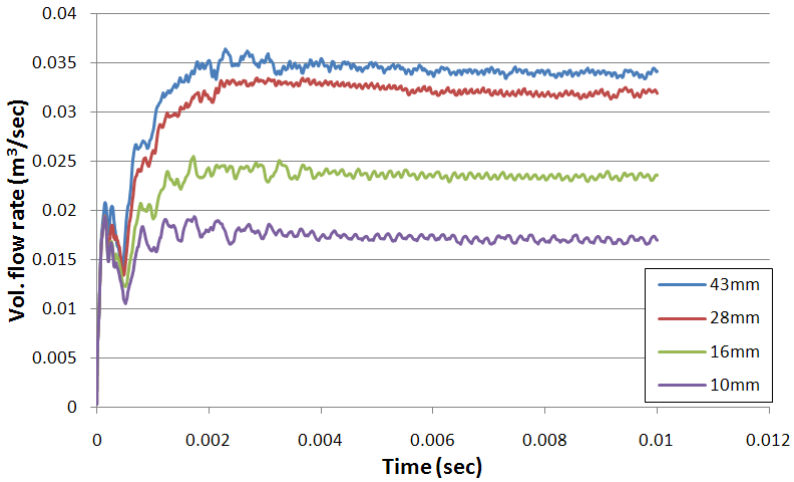


Fig. 5.11. Transient flow rate through the nozzle for different nozzle openings

By calculating the average flow rate after the transient phase and plotting the flow rate as function of the nozzle opening, the characteristic curve of the nozzle is obtained. As shown in fig. 5.12, agreement with both experimental results and the Fluent program is good.

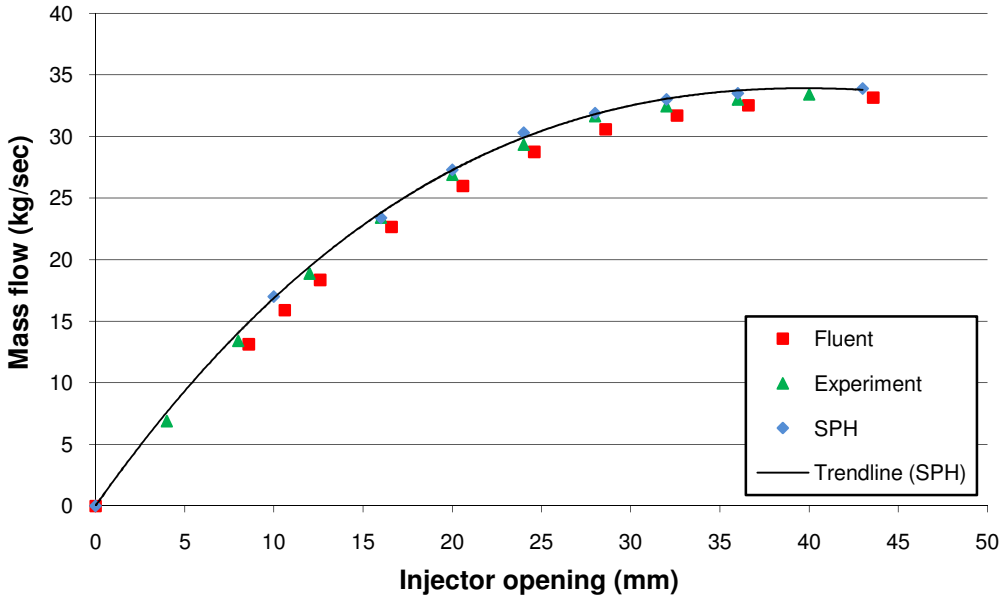
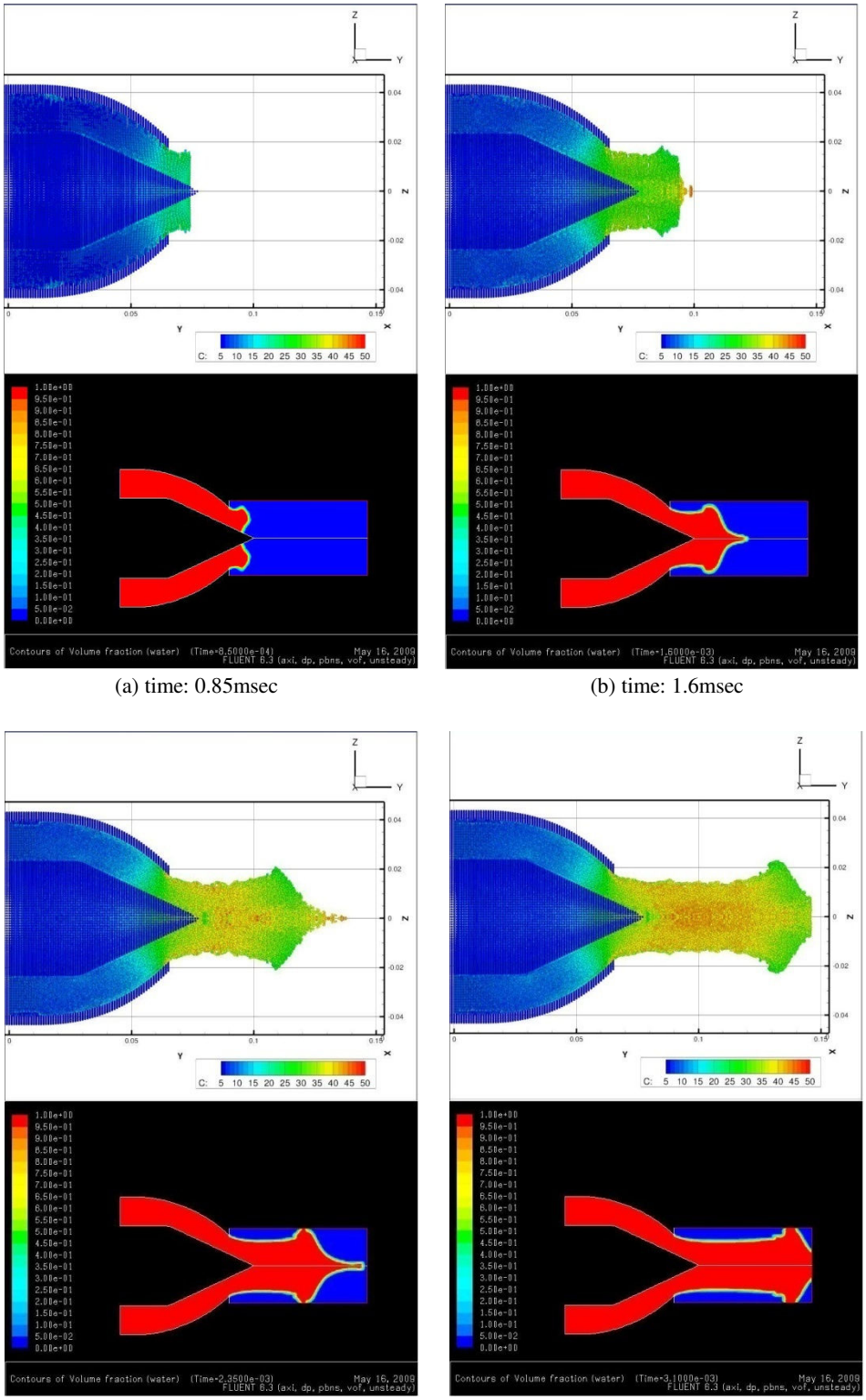


Fig. 5.12. Characteristic curve of the nozzle. Comparison of the results from SPH and Fluent simulations and experimental data.

Another interesting prediction from the SPH algorithm is the evolution of the free surface of the jet in respect to time. As it was already mentioned the nozzle was assumed to be filled with water at t=0. During the transient part of the simulation, a mushroom shaped structure is formed (fig. 5.13b, c, d). After ~4ms, the solution reaches a steady state and a cylindrical jet is formed. The same behavior is predicted by the Fluent program, at respective instances.

It has to be highlighted here that the Fluent solver used for the nozzle simulations was an axis-symmetric solver, thus the computational mesh used was 2d. This means that the Fluent simulation was performed considerably faster than the SPH simulations with any resolution used in the present section. However the comparison is not equal, since Fluent solved a 2d mesh, whereas SPH solved a 3D domain. Indeed, the situation is entirely different when Fluent is used to perform a 3D simulation and the execution time is comparable to that of the SPH algorithm.

Transient behavior



(a) time: 0.85msec (b) time: 1.6msec

(c) time: 2.35msec (d) time: 3.1msec

Fig. 5.13. Indicative instances during the transient phase calculated by SPH and Fluent.

5.2. Jet deflector

The water jet deflector is used in impulse hydro turbines in order to quickly reject load from the turbine, used in emergency shutdowns [1]. Emergency shutdowns must be done when the power line, connecting the generator to the power grid, is disconnected for some reason. The result is that the turbine operates without a resisting load from the generator, accelerating the runner and eventually leading to turbine and generator overspeed, which might damage the installation.

Load regulation is done through the nozzle needle stroke. However the hydraulic systems adjusting the nozzle needle are not able to quickly cut the water supply to the turbine runner, and, even if it was possible, such an operation would not be applicable, due to the hydraulic hammer effect [2]. Thus, a deflector is used which is inserted between the runner and nozzle, deflecting the water jet away from the turbine runner (fig. 5.14).

Fig. 5.14. Impulse turbine jet deflector (the nozzle is also visible).

Deflectors may be divided into two types, as shown in fig. 5.15. In both cases the deflector plate rotates and enters the path of the jet reducing (fig 5.15a) or totally diverting (fig 5.15b) the water flow reaching turbine runner.

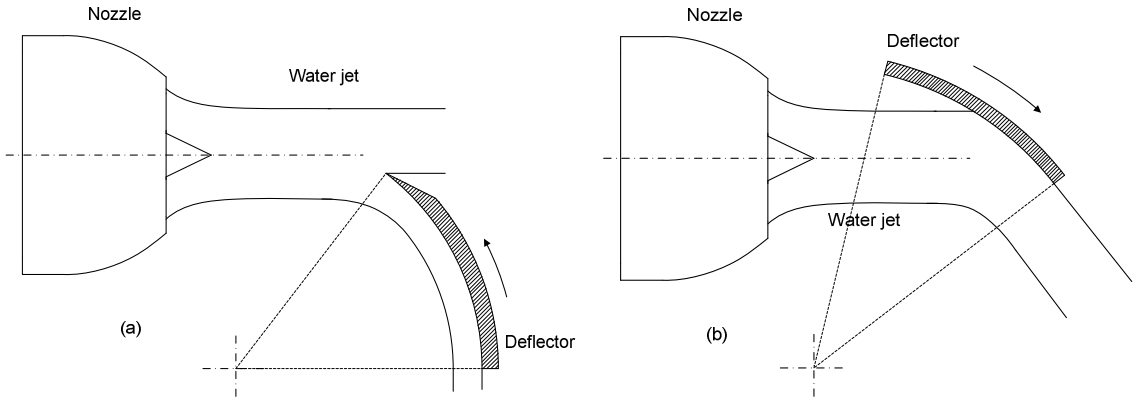


Fig. 5.15. Deflector types.

The aim of the present investigation is to determine whether a deflector, such as the one shown in fig. 5.15a, may also be used to effectively adjust the turbine load in normal operation, without regulating the spear valve. The investigation is performed primarily qualitatively, judging the jet coherence after the interaction with the deflector from indicative simulation time-steps.

Case set up

The deflector was assumed to be a section of a cylindrical arc. The simulated fluid was water (ρ=1000kg/m^3), the velocity of the jet being deflected was 32m/s and the jet radius 30mm. The numerical speed of sound was set to ~10V$_{jet}$=320m/s.

At first a simplified simulation was performed in 2D, where the deflector was positioned approximately halfway cutting the water jet. Particle size was set to 1mm, thus simulations involved ~21000 particles. Indicative results during the transient phase, showing the propagation of the water, are presented in fig. 5.16, where both the SPH and Fluent solutions are shown.

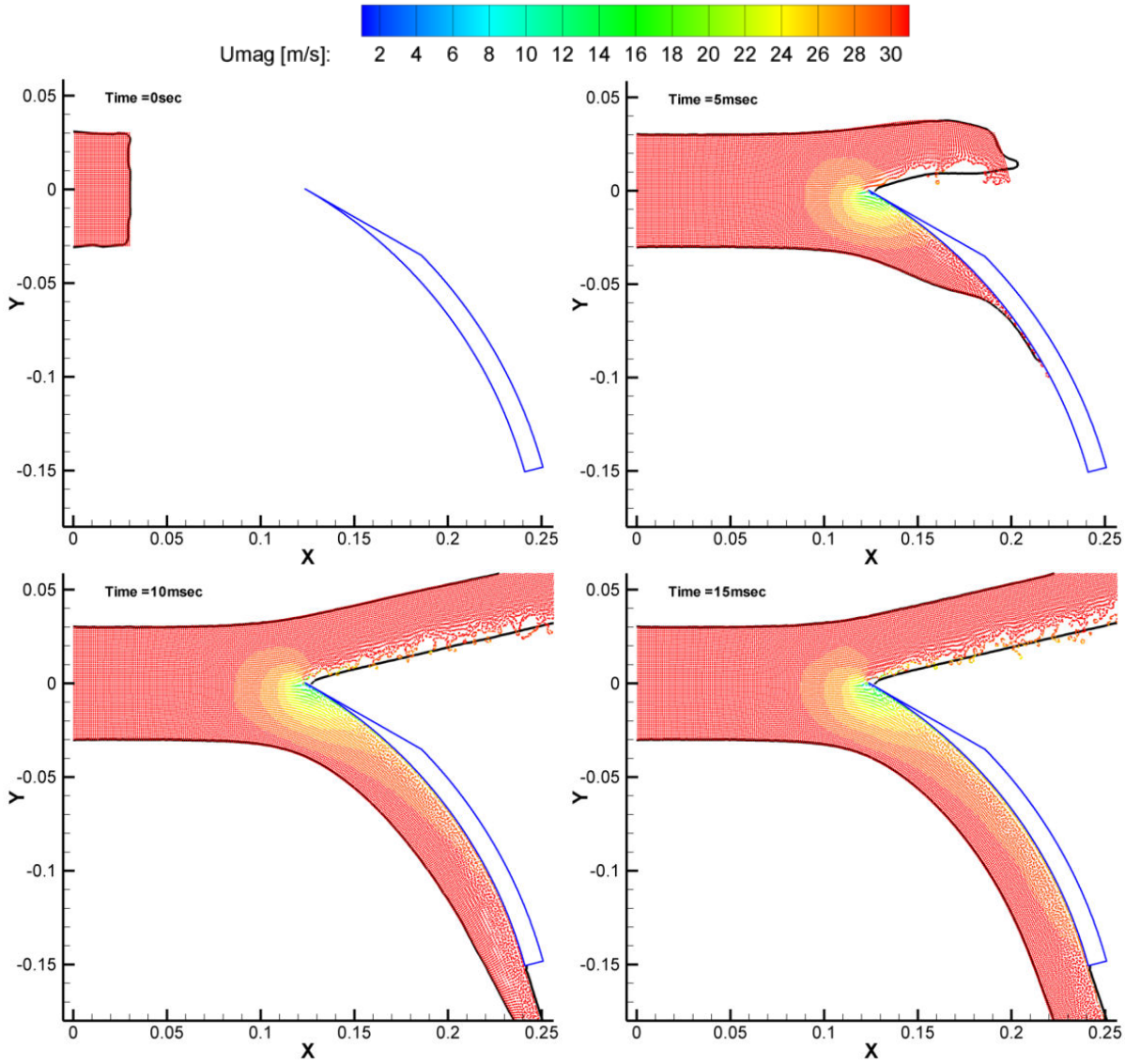


Fig. 5.16. Indicative snapshots during the transient phase of the jet-deflector interaction. Particles are colored according to their velocity magnitude. Free surface calculated with Fluent is denoted with a solid black line.

The forces on the deflector may be estimated using the conservation of momentum theorem (see also fig. 5.17):

$$\mathbf{F} = \dot{m}_{in}\mathbf{u}_{in} - \dot{m}_1\mathbf{u}_1 - \dot{m}_2\mathbf{u}_2 \tag{5.4}$$

$$\dot{m}_1 \approx \dot{m}_2 \approx \dot{m}_{in}/2 \tag{5.5}$$

After calculations using eq. 5.4 and 5.5:

$$\begin{cases} F_x = 22000\,N \\ F_y = 22400\,N \end{cases}$$

SPH and Fluent solution predict a similar deflector force:

$$SPH: \begin{cases} F_x = 20901\,N \\ F_y = 19546\,N \end{cases}$$

$$Fluent: \begin{cases} F_x = 20089\,N \\ F_y = 18980\,N \end{cases}$$

Forces are expressed per depth unit. Here it has to be highlighted that in the SPH method, forces are averaged after the initial transient stage of the flow, due to oscillations.

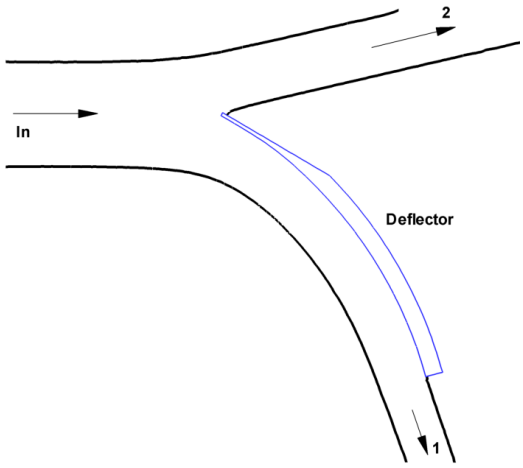


Fig. 5.17. Application of the conservation of momentum theorem for the jet deflector.

After the initial test, where the deflector was assumed to be stationary, a further test is performed to determine the influence of the deflector motion. The conditions of this test are the same as before, with the only difference that the deflector starts its motion while being out of the jet's path and then gradually deflects the jet. The angular velocity of the deflector is 5π rad/s, which is rather large for an actual impulse turbine deflector. A real deflector would require ~1sec to complete a circular arc less than 2π which will move it to the deflecting position. However, even in that case, the simulation will be able to show the transient features of the interaction. As shown in the snapshots during the transient phase (fig. 5.18), the remaining water jet is considerably affected by the presence of the deflector; its direction changes and the free-surface distorts considerably.

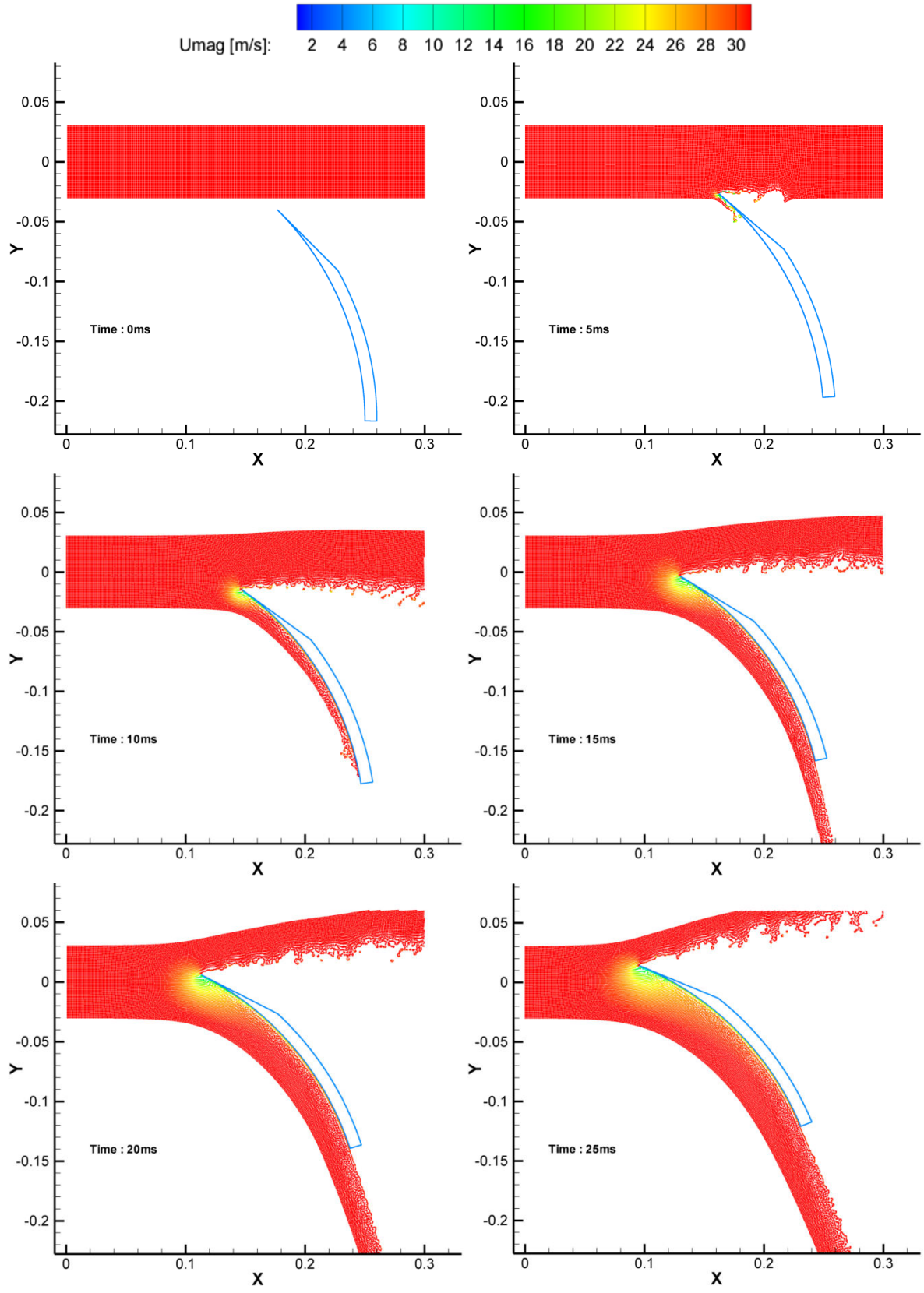


Fig. 5.18. Transient behavior of the moving deflector (2D).

In fig. 5.19 the forces acting on the deflector, due to the interaction with water, are shown. Both forces on the *x* and *y* axis exhibit oscillations, which, as it was mentioned earlier, are a side-effect of the repulsive boundary conditions used to represent walls. However, despite the oscillations, forces follow a distinct trend.

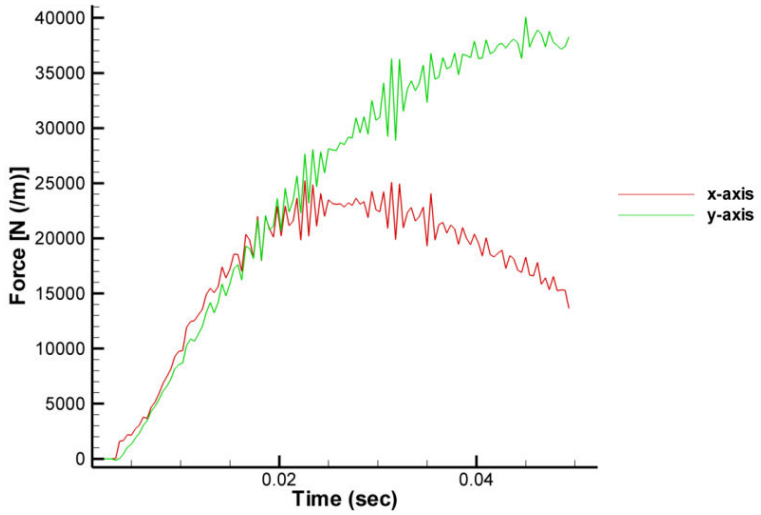


Fig. 5.19. Force history on the deflector.

As it was shown, in both the stationary and the moving deflector 2D cases, the water jet is considerably affected by the deflector presence; after the interaction, the remaining water jet is significantly distorted and its direction is altered. However this might be attributed to the assumption of two dimensional flow. Thus, in order to get a more realistic simulation it is necessary to test the influence of the deflector using a full 3D model. The same case was simulated for a cylindrical jet in 3D, for the same conditions as the 2D moving deflector case, with the only exception that the particle size used was 2mm (~210000 particles involved), since the computational cost was prohibitive for smaller particle sizes, even when running the SPH algorithm in parallel. Also for simplicity the deflector was assumed to consist of only one layer of boundary particles.

In fig. 5.20 indicative snapshots are shown during the interaction of the jet and deflector. As shown in the results, the remaining water jet is still affected by the deflector, but to a less extent than the predicted from the 2D simulation; both the jet direction and the jet shape change (see also fig. 5.21). Even in that case, the results indicate that the jet is considerably distorted. The jet shape exhibits significant scattering, which is rather likely to affect turbine efficiency, especially if the influence of the interaction with the surrounding air is accounted. The conclusion from the presented simulations is that a deflector of cylindrical shape is not well suited for adjusting the load on an impulse turbine, since the jet characteristics greatly deteriorate after the interaction with the deflector, even when the deflector cuts a small part of the jet.

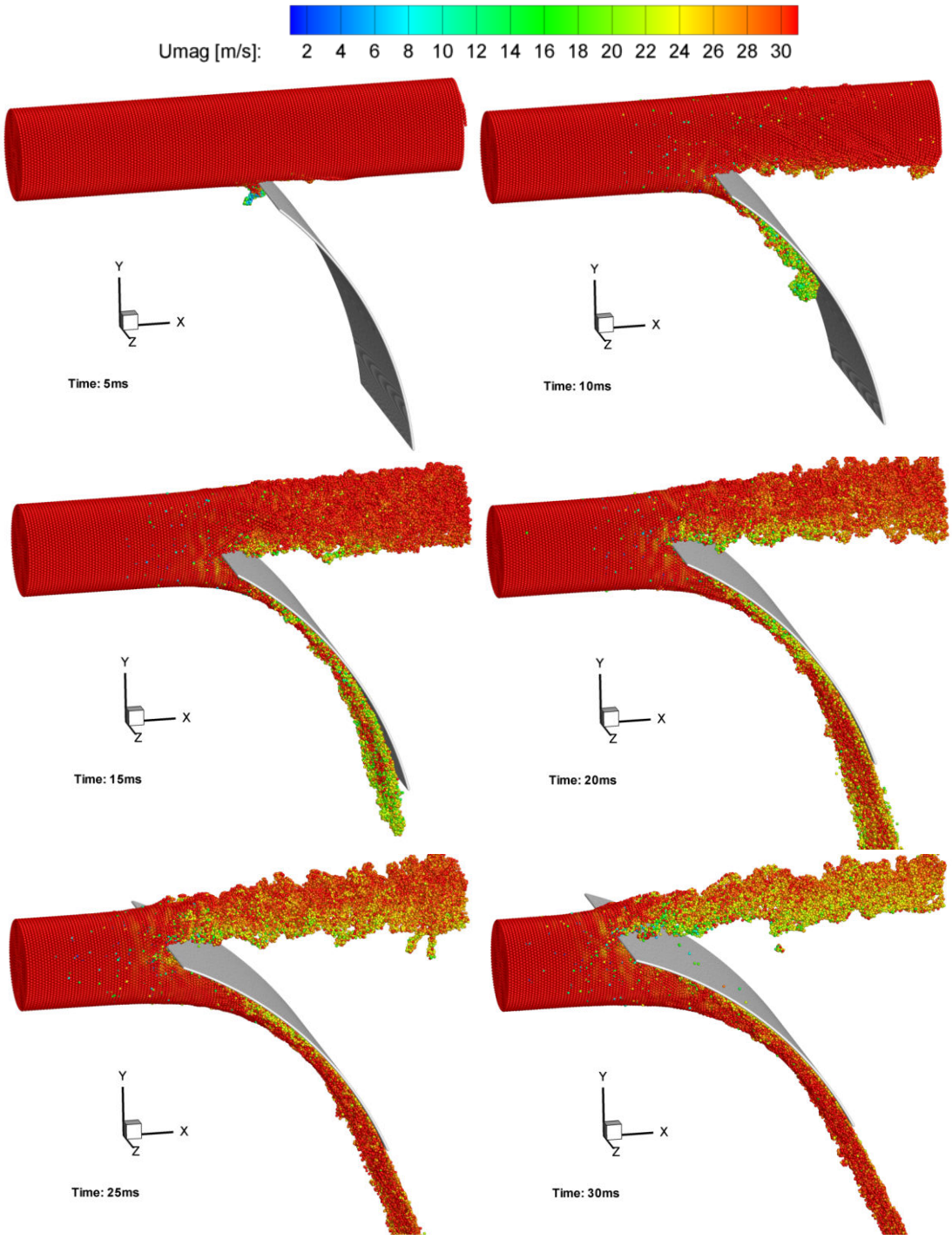


Fig. 5.20. Transient behavior of the moving deflector (3D).

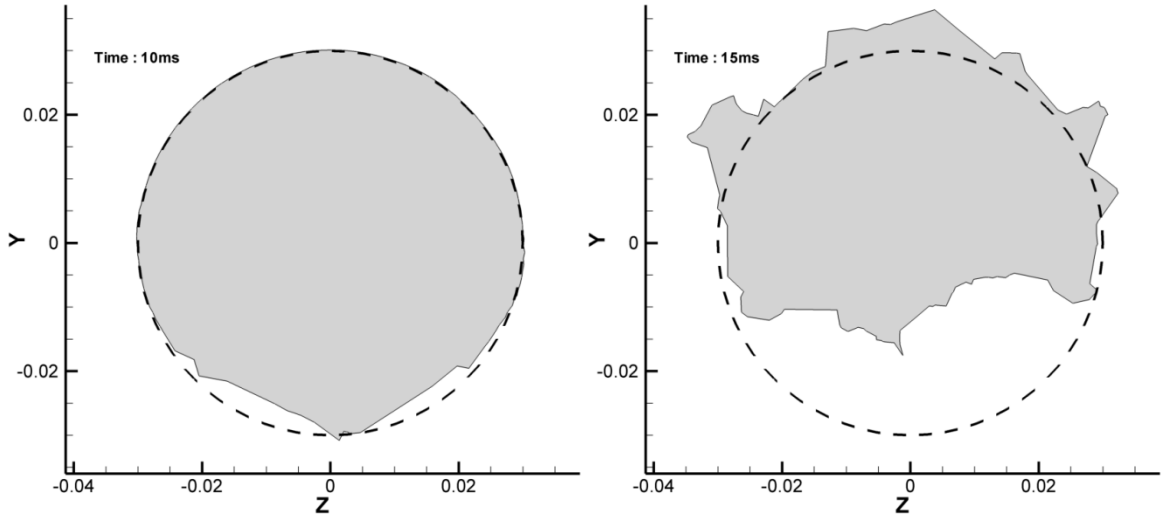


Fig. 5.21. Jet shape, after the interaction with the deflector, at two indicative snapshots. The initial jet shape is marked with a dashed line.

5.3. Impulse turbine simulations

Impulse water turbines operation is based on the interaction of a high velocity water jet with the rotating turbine runner. The turbine runner changes the direction of the flow and, as a result, torque is exerted on the runner due to the change of water jet's angular momentum. The static pressure does not change before and after the turbine runner. This enables the operation of the turbine in atmospheric environment, without the need of a sealed casing. The water jet is formed at the turbine nozzle, sometimes referred to as injector, where the water potential energy is transformed into kinetic energy. An impulse turbine may have more than one nozzles. They are located at the turbine casing, around the turbine, directing the jet tangentially at the runner blades (Turgo turbine), or buckets (Pelton turbine). The nozzles are equipped with a needle, which is used to adjust the water flow rate through the nozzle to match the requested power at the generator coupled to the water turbine. Apart from the needle the nozzles often have a deflector, which is used to deflect the water jet from the runner, in case of an emergency to quickly reject load from the turbine. The most well known impulse turbines are the Pelton and Turgo turbines.

The Pelton turbine (fig. 5.22) consists of a runner of specific geometry; the runner is comprised of specially shaped buckets (fig. 5.22). Inside the turbine casing there are the turbine nozzles at which a high velocity jet is formed and is directed tangentially at the turbine runner, impinging at the middle of the buckets. Due to their special shape, the water flow is divided into two parts which flow inside the bucket and eventually exit the bucket at the opposite direction relative to the water jet. Since the runner changes the direction of the water jet, force and consequently torque develops at the runner and energy is transferred from the water to the turbine generator. The Pelton turbine is well suited for applications of high water gauge with relative low flow rates and achieves high efficiencies.

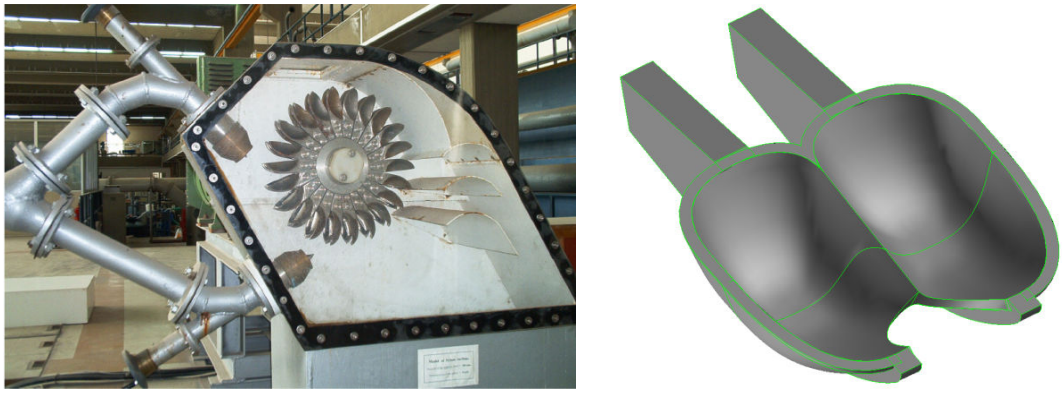


Fig. 5.22. Left: Pelton turbine. Right: Pelton turbine bucket.

Turgo turbines are designed for medium head applications. They have a flat efficiency curve and provide excellent part load efficiency, thus they can be used as an alternative of other turbine types, especially if there are large flow rate variations. The simplest Turgo turbine runner looks like a Pelton runner split in half at the plane of symmetry. The water jet enters from the one side of the runner and exits from the other. Because of that, the escaping water does not interfere with the incoming jet, or the other turbine blades, enabling the turbine to handle larger flow rates and jet diameters than a Pelton runner of the same runner diameter. As a result the Turgo turbine has higher specific speed and smaller size than a Pelton turbine of the same power. The smaller diameter allows the operation at higher angular velocities, which in turn, makes the coupling between the turbine and the generator easier, avoiding the use of a mechanical transmission system decreasing costs and increasing the mechanical reliability of the system [3].

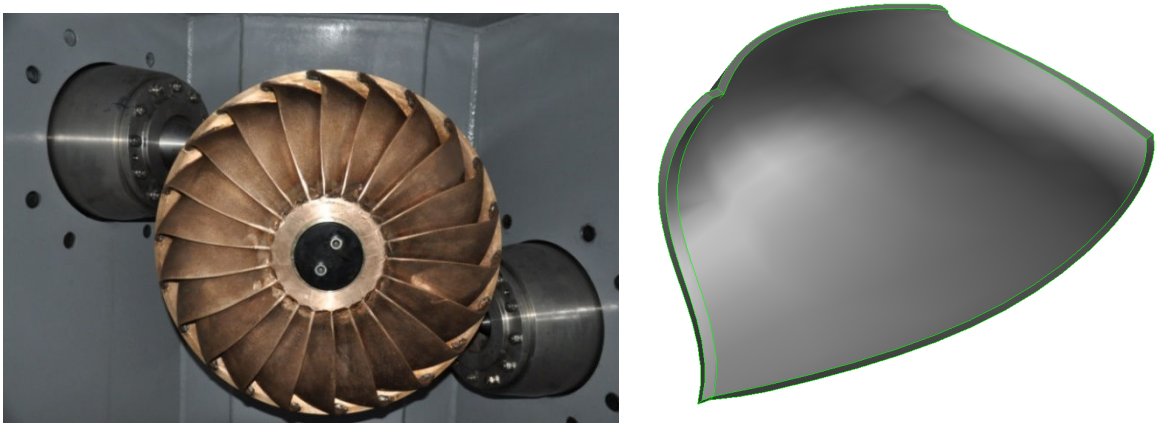


Fig. 5.23. Left: Turgo turbine. Right: Turgo turbine blade.

A common characteristic of impulse water turbines is that the developing flow of the impinging jet on the runner is unsteady, free-surface with moving boundaries, due to the runner rotation. This fact renders simulation with Eulerian methods difficult [4] since special treatments are required for capturing the underlying phenomena, such as the Volume Of Fluid (VOF) method for tracking the free-surface and sliding meshes. The aforementioned treatments increase the computational cost and requirements of the simulation considerably. Alternatively, it is possible to adopt a Lagrangian point of view, using the SPH method and exploiting its advantages. Indeed, according to Marongiu et al. [4, 5], a Lagrangian perspective simplifies the free-surface tracking and the meshless nature of the SPH method makes the handling of rotating runner easier. In the present work, the SPH method was applied for simulation of Petlon and Turgo turbines. Moreover, the SPH algorithm was used for the optimization of a Turgo turbine runner.

Geometry representation of the blade/bucket geometry

The Turgo blade geometry was generated using specialized software available to the Laboratory of Hydraulic Turbomachines department [6], which uses as an input the blade angles and the general dimensions of the blade. The average blade inlet and outlet angles may be computed from the corresponding velocity triangles. Also, additional design parameters are introduced to allow for differentiation of the inlet and outlet bucket angles along the leading and trailing edges of the bucket. Moreover, the curvature of these edges may be properly adjusted. The above make a total of 12 geometric control variables for parametric description of the bucket geometry. The mean 3-dimensional surface of the bucket is then generated using the conformal mapping methodology and interpolation techniques [6]. The above parameterization method ensures always a smooth curvature variation of the surface.

A similar approach, using NURBS [7], is used for the description of the Pelton turbine bucket [8]. According to the developed methodology a number of design control points are properly introduced for the definition of the bucket inner surface. As shown in fig. 5.24a, 6 such points are placed along the periphery of the half symmetric part of the bucket and can be moved along the x or the z direction to deform its shape. The exact location of the deepest point of the bucket may be also regulated with reference to the bucket itself and to the rotation axis of the runner. The location of the cutout lips point, as well as the radius of the cutout region can be also determined by two corresponding geometric parameters. The rest surface of the bucket is then constructed as follows: the surface is assumed to consist of a number of successive parallel slices, starting from the bucket inlet plane on the xy level and terminated at its tip. The shape of each such slice is defined by interpolation techniques between the first slice and a number of intermediate control levels that determine the rate of change of the slice width and length. Two such levels are used here (see fig. 5.24b), introducing 4 additional control parameters. Overall, the construction or the modification of the entire inner surface of the bucket is controlled by a number of 18 free geometric variables, which is a rather small number for such a complex 3dimensional shape. After parameterization, at any (x, y) point the algorithm can return the corresponding depth z of the inner surface, as well as the local slopes dz/dx and dz/dy. The bucket surface, used in the present study, is based on the geometry of an existing Pelton turbine which has a hydraulic head of 135mWG and nominal power rating of 150KW.

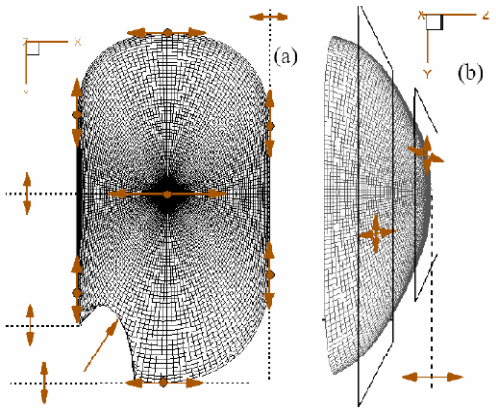


Fig. 5.24. Surface parameterization of the Pelton bucket. Location of control points.

In both cases, the geometry of the bucket/blade is eventually represented as a grid consisting of quadrilateral elements (fig. 5.25). In order to transform the 2D quadrilateral grid into a layer of particles, the blade/bucket surface was remeshed with appropriate resolution, depending on the particle sizing of each simulation.

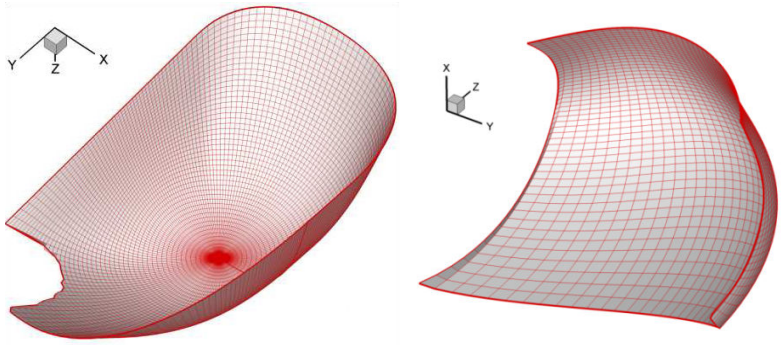


Fig. 5.25. The surface grid representing the Pelton bucket (left) and the Turgo turbine blade (right).

5.3.1. Application of the SPH method for simulation of Turgo turbine geometries

In this section the SPH algorithm is used for simulations on Turgo geometries as the one shown in fig. 5.26. In the Turgo turbine, water jet enters under a relative angle of θ~20-25^{o} in respect to the runner level of rotation. Water kinetic energy is transferred to the runner and eventually water leaves with little residual velocity mainly in the axial direction.

Initially two 2D tests are presented; the 'blade' is a section of an actual 3D Turgo turbine blade at the mean radius. One test was performed for stationary blade and the other for a moving blade performing translational motion. Following on, 3D tests have been performed for a stationary Turgo blade under various jet impingement conditions and for a rotating Turgo geometry. The rotating geometry consisted of two, or even more, blades. Eventually, the SPH method was used to evaluate the resulting Turgo geometries from the optimization with the FLS algorithm and the complete Turgo turbine runner was simulated.

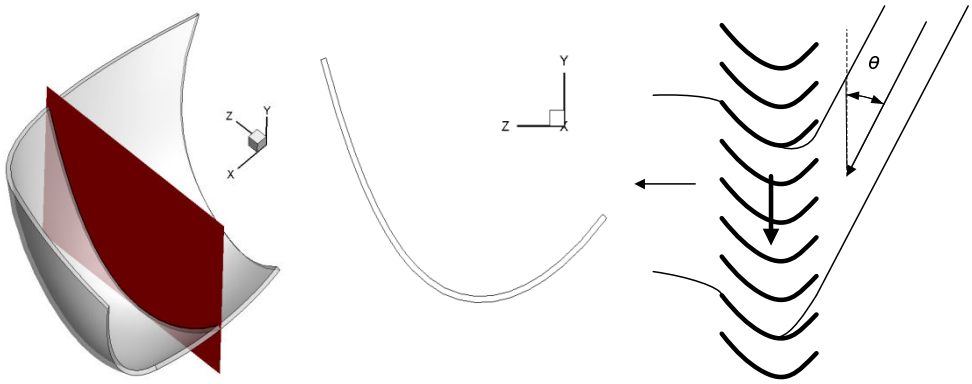


Fig. 5.26. Left: Turgo blade in 3D. Center: the 2D turgo geometry. Right: flow direction in a Turgo turbine.

2D Turgo geometry tests

The purpose of the 2D tests was primarily to explore the capability of the SPH method to simulate flow similar to the one occurring in an actual impulse turbine blade, where the wall geometry is more complicated than the simple water jet impingement on flat plate. The 2D tests correspond to the flow conditions at the mean radius of an actual Turgo turbine. The general dimensions of the 3D blade were: width 332mm, length 412mm, depth 265mm and the inner radius of the runner is 261mm and the outer radius 500mm (mean radius ~380mm). The runner was impinged by a 154mm diameter water jet with a velocity of 32m/s at 25^{o} relative to the turbine rotation plane. For the 2D case, the turbine blade was assumed to be stationary or moving with the velocity of 17.2m/s. The particle size

used was 2.5mm and the numerical speed of sound was 350m/s. The moving case was compared to the solution obtained by Fluent.

Stationary blade simulation

The simulation was performed for a total of 50msec, involving a maximum number of ~16000particles. In fig. 5.27, several snapshots of the flow are shown at consecutive time steps. The blade is positioned in such a way to approximately cut in half the impinging jet. Also the blade is simulated using a single layer of boundary particles as in the deflector case.

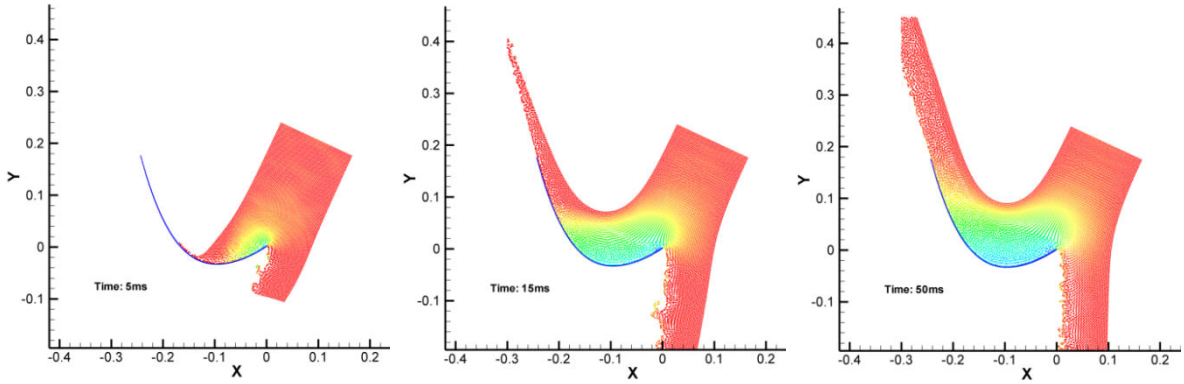


Fig. 5.27. Flow on the 2D blade at different time steps. Particle coloring according to velocity magnitude.

The force on the blade may be calculated in a similar manner as in the deflector case, using the conservation of momentum theorem. Thus it is possible to obtain forces due to the interaction of the water jet and the blade:

$$Theoretica\ l: \begin{cases} F_x = -43153\ N \\ F_y = -99703\ N \end{cases}$$

$$SPH: \begin{cases} F_x = -44165\ N \\ F_y = -103223\ N \end{cases}$$

The deviation of the calculated force by the SPH method from the analytical values is ~3%.

Moving blade simulation

This simulation was set-up in exactly the same way as the previous one, but the difference is that now the blade moves towards the $-y$ axis, with a translational velocity of 17.2m/s. The developing flow field is shown in fig. 5.28, as calculated by the SPH method and Fluent. Practically, both programs calculate the same free surface and velocity distribution; the only difference is that the free-surface calculated by SPH is susceptible to breaking, especially at the later time steps. However this difference has to be attributed to the diffusion of the Volume of Fluid method used by Fluent to track the interface, which stabilizes the free-surface. SPH calculated forces on both the x and y-axis exhibit considerable oscillations but the general trend of the force history is similar with the one calculated by Fluent. The oscillations in the force calculation do not affect greatly the total work on the blade; the calculated work by the SPH method is 27339J and by Fluent is 28300J (deviation ~3%).

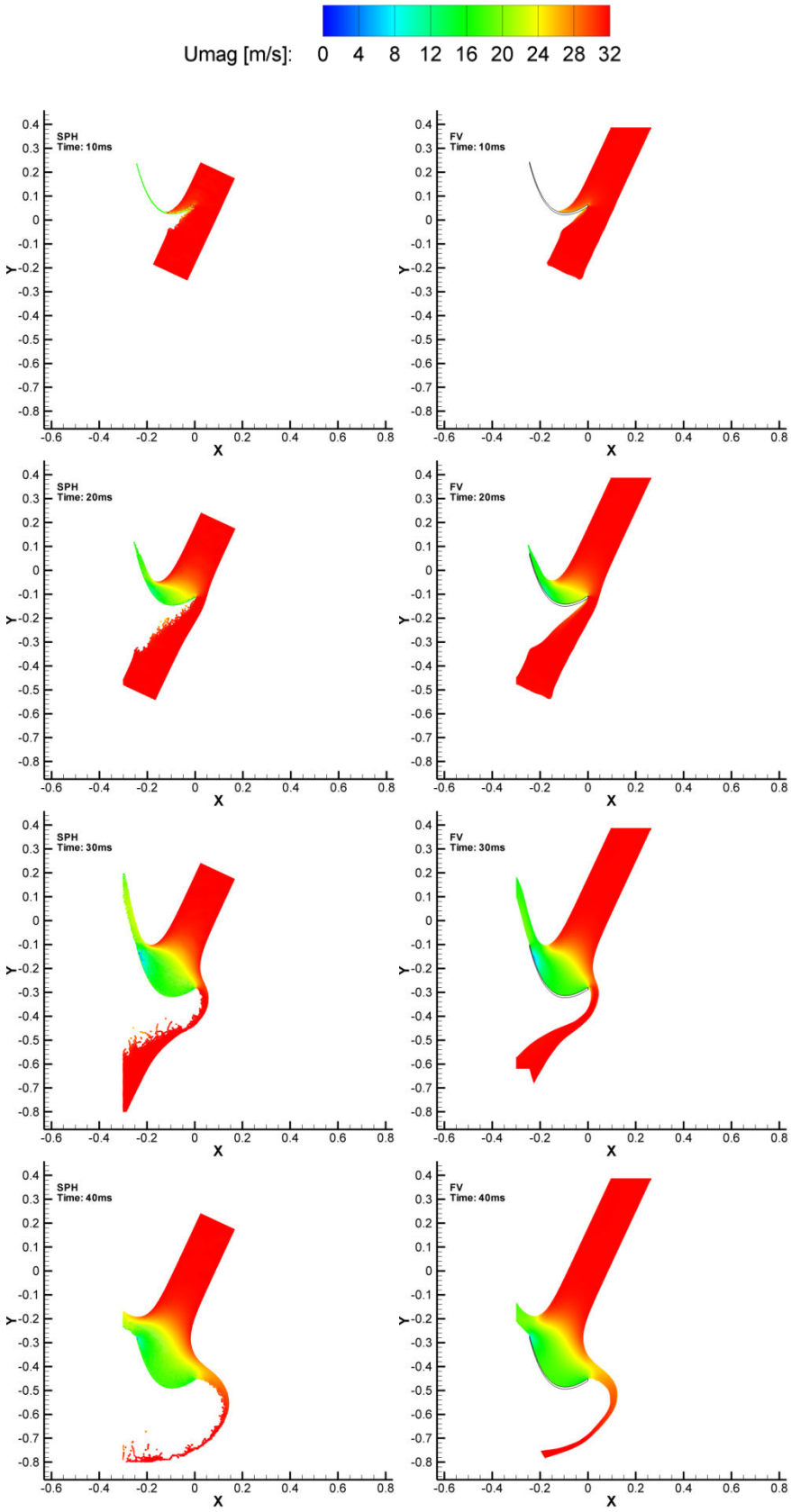


Fig. 5.28. Time instances from the 2D simulation of a moving blade impinged by water jet. Left: SPH solution, right: Fluent solution.

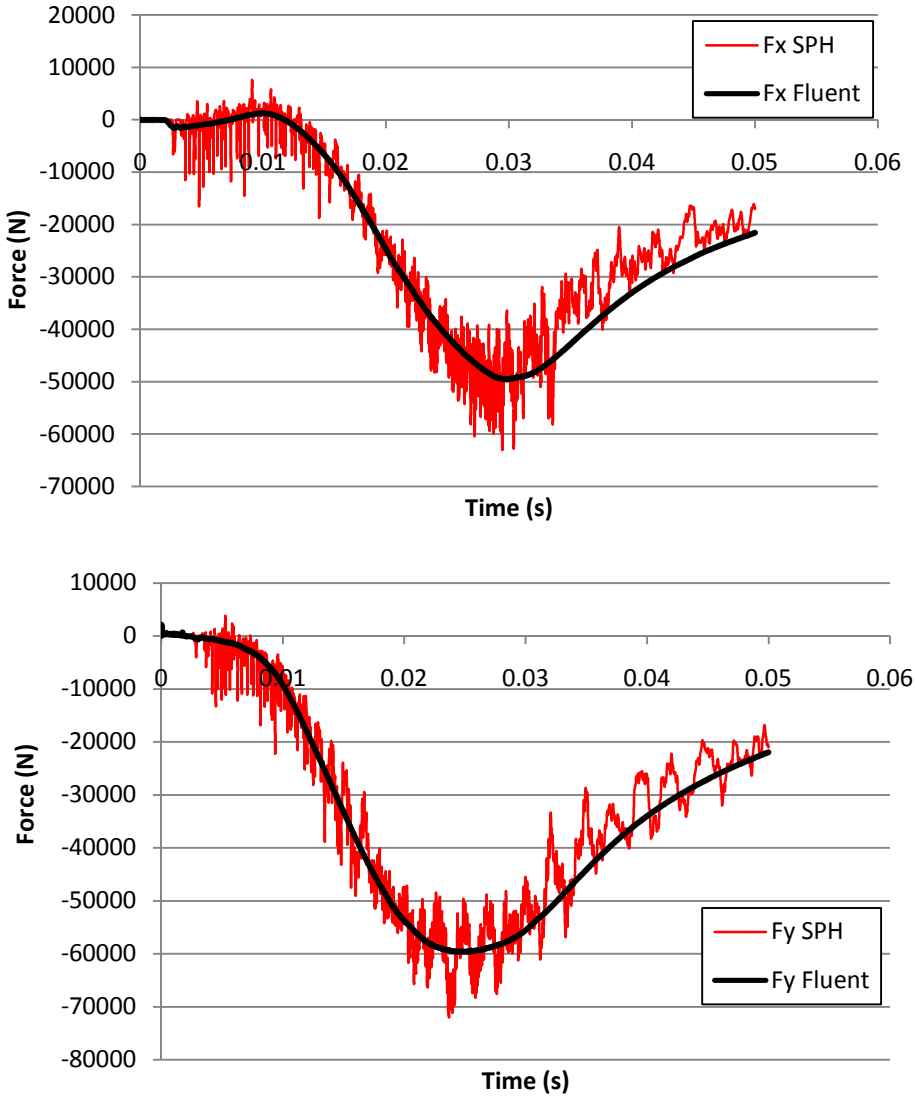


Fig. 5.29. Up: Force on the x-axis in respect to time. Down: Force on the y-axis in respect to time.

Considering the previous results of the 2D blade simulations, the conclusion is that the SPH method is able to accurately predict the flow features of the flow even in the moving blade case; the free surface is accurately resolved and the velocity field is similar. Also, even if the boundary treatment leads to a noisy time history of the forces on the blade, the general trends are reasonable, compared to the corresponding results from the Fluent solution.

Turgo 3D stationary blade

In this part the flow simulations on a 3D Turgo turbine blade is presented. The flow was simulated for different impingement angles (see fig. 5.30) relative to the *z*-axis, for a physical time of 0.1s. The water jet axis was placed on the *yz*-plane. The water jet had a diameter of 154mm and a velocity of 30m/s (the numerical speed of sound was set to 350m/s) and enters from the inlet edge. Particles in the jet are positioned in a similar manner as the one presented in the jet impingement on the flat plate, in order to avoid particle clumping.

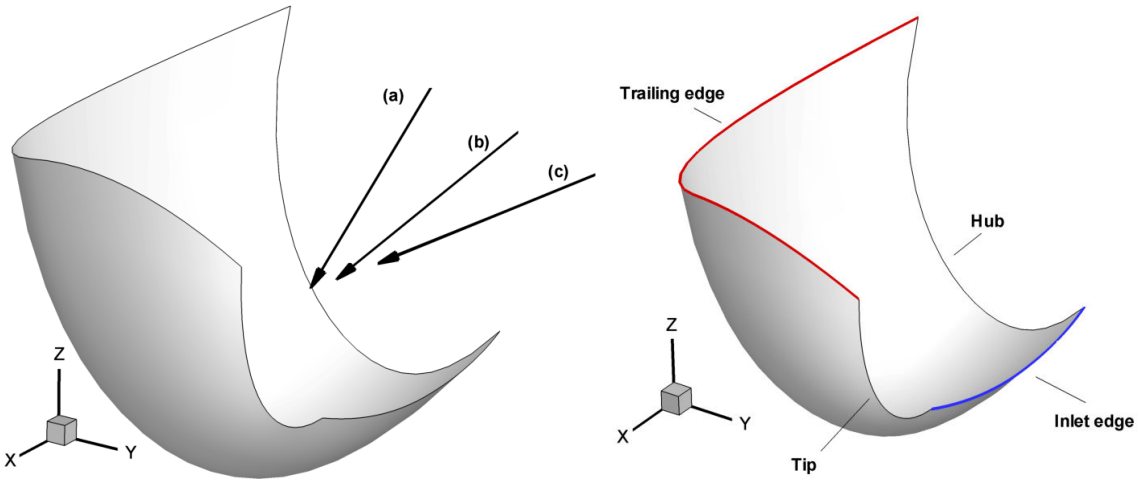


Fig. 5.30. Jet impingement on a stationary Turgo blade at impingement angles of: (a) 30^o (b) 47^o and (c) 60^o degrees from the *z*-axis.

In order to determine particle discretization of adequate accuracy which should be used for the simulations, a particle dependence analysis was carried out for various particle sizes, for impingement angle of 30° from *z*-axis. The results of the free surface, pressure profile and net forces on the blade are compared with the results from the same test case obtained by the Fluent solution. Three different resolutions have been used in this study: with 15, 21 and 31 particles on the jet diameter (with respective particle sizes 10mm, 7.5mm and 5mm). Indicative results from the simulations are presented at fig. 5.31. It is clear that by increasing the particle resolution particle distribution becomes smoother, eventually giving a more realistic flow field. From the picture of the coarse particle distribution, the estimated velocity of the water sheets, forming after the impingement, is greatly underestimated. As expected from Bernoulli theorem in the absence of friction/viscosity, the velocity of the water flowing away from the blade should be equal to the velocity of the jet, since pressure in both cases is equal to the atmospheric pressure. However, at the coarse simulation velocity magnitude of the water sheet is underestimated by ~30% (estimated velocity ~20m/s, whereas the expected velocity from the Bernoulli theorem should be 30m/s). At finer resolutions, this effect tends to minimize, however it becomes still apparent at specific areas as it will be shown later.

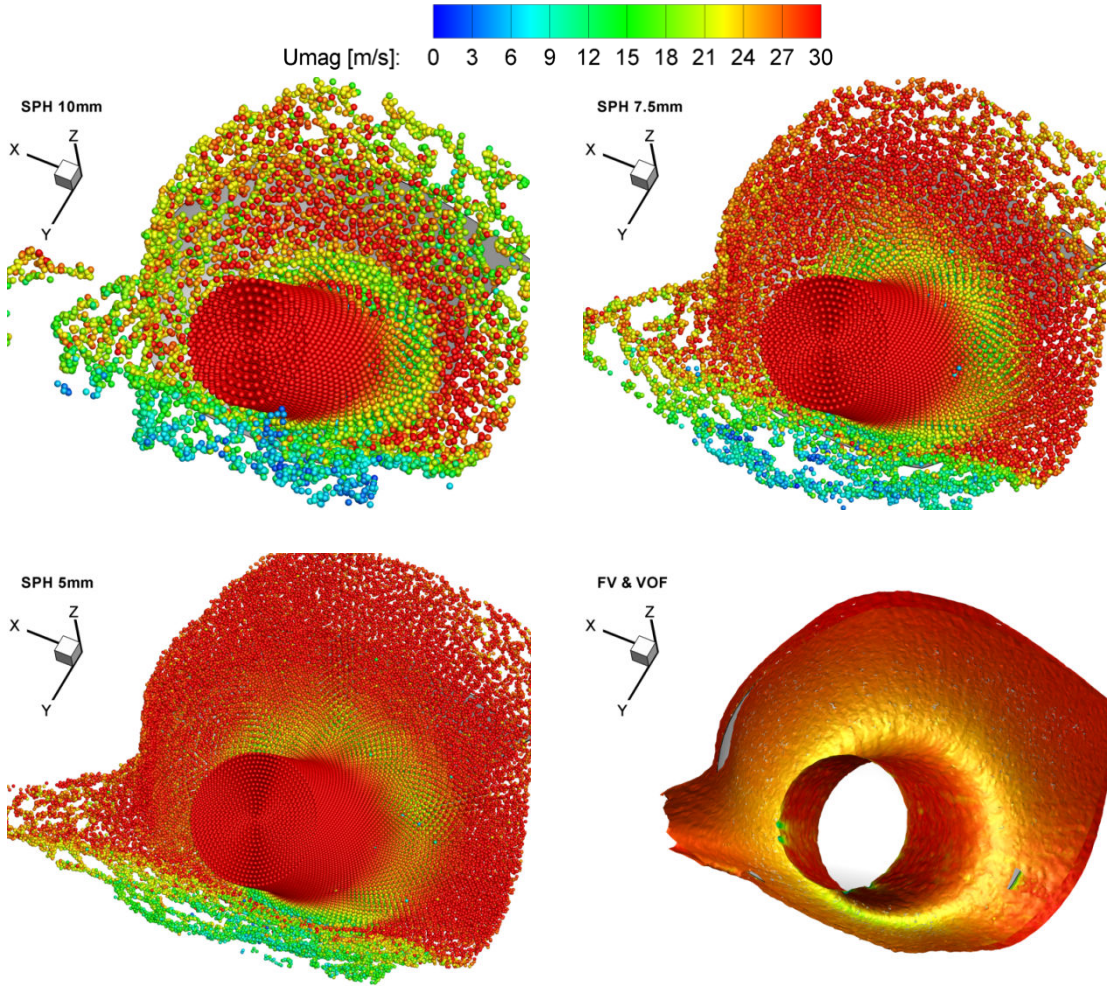


Fig. 5.31. Jet impingement on a stationary Turgo blade at angle of 30°. Particle dependence study and comparison with the Fluent solution.

In fig. 5.32, the free surface flow is compared, calculated from the SPH algorithm, for the three different resolutions employed, and Fluent. The comparison is made at a slice on the *yz*-plane for x=0. The flow calculated with SPH is represented with particles, whereas the flow calculated by Fluent is represented with a solid line. From these figures it is clear that the best representation of the free surface flow was achieved by using the fine discretization (31 particles on the jet diameter, particle size 5mm). The free surface calculated by SPH with the fine and intermediate particle discretizations (particle sizes 5 and 7.5mm respectively) is in accordance with the free surface calculated by Fluent. On the other hand, the flow calculated using a coarse particle discretization (15 particles on the jet diameter, particle size 10mm), deviates from the Fluent solution, especially at the trailing edge of the blade. Another important remark is that, with both programs, a small part of the impinging jet exits from the inlet edge; this part will still exist even when the turbine is rotating, as it will be shown later on in the rotating turbine runner simulations. It can be considered as a leakage, however it represents only a small percentage of the total flow.

In the same figure, the velocity magnitude of the water is also visible for the SPH and Fluent solution. The quality of the velocity field calculated by SPH is greatly dependent on the number of particles involved in the simulation. In the simulation using the coarse discretization (15 particles on

the jet diameter), there are large deviations mainly in the *w*-velocity component and consequently in the absolute velocity, between the velocity fields calculated by SPH and Fluent (fig. 5.32). This is attributed to the lack of sufficient neighbouring particles. With the increase in the number of particles the velocity fields calculated by both methods become similar. Apart from the absolute velocity field, the distribution of the individual velocity components calculated by SPH and Fluent was compared too, giving similar results.

Note though, that in SPH, even in the case of the fine particle discretization, particles leaving from the inlet edge of the blade, have an underestimated velocity compared to the one calculated by Fluent, because of insufficient number of particles in the area (fig. 5.33).

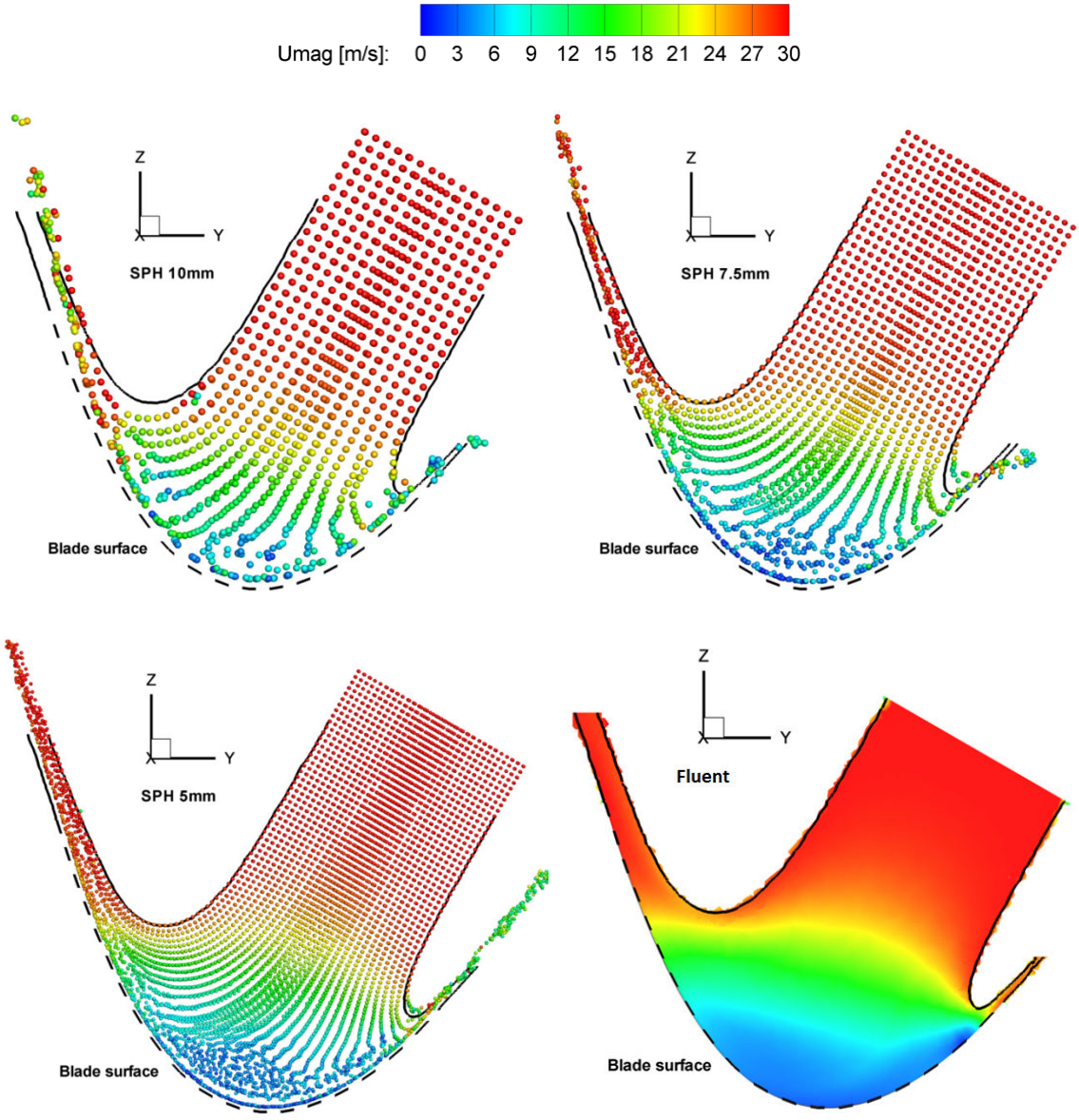


Fig. 5.32. Comparison of the free surface calculated by SPH for different resolutions. Particles represent the SPH solution, continuous line the free surface location calculated by Fluent and the dashed line represents the wall.

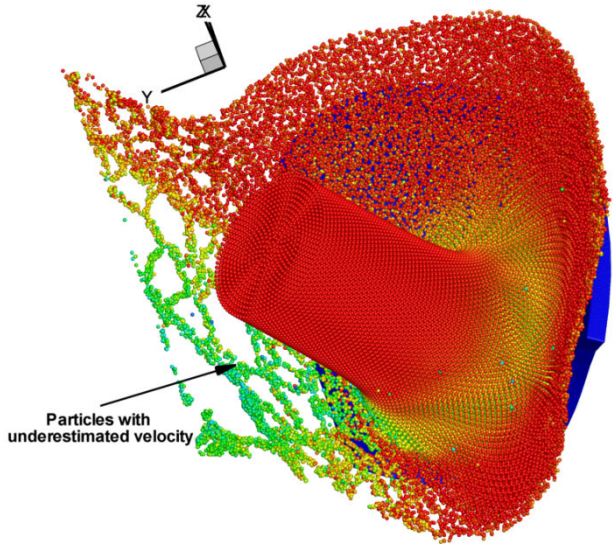


Fig. 5.33. Velocity underestimation when using the fine particle resolution at the water sheet leaving from the inlet edge (leakage). Large voids are also observable.

As regards the quality of the pressure field on the blade, it improves with the use of the finer discretization. As it appears from fig. 5.34 (see also fig. 5.35), the intermediate and coarse discretizations under predict pressure at some areas and they over predict pressure in other areas (e.g. a pressure spike at approximately 0.25m from the origin). The fine discretization is able to provide a pressure profile closer to the one calculated by Fluent, but even in this case there are some discrepancies. This can be attributed to the inherent weakness of the traditional SPH in calculating a correct pressure field [9]. Further increase in the number of particles and an even finer particle discretization would improve the pressure field. Thus, in order to avoid errors from the pressure field inaccuracies, forces on the blade will be estimated from the boundary particles reaction forces, as in the previous cases (deflector and Turgo 2d blade).

In fig. 5.35 the pressure distribution calculated by SPH, using the fine, intermediate and coarse discretizations is compared to the pressure distribution calculated by the VOF method. Pressure was evaluated along a line located on the blade surface, for $x = 0$ (fig. 5.35). Since in SPH the wall particles do not have pressure, but instead exert repulsive forces, the pressure field was estimated using pressure probes, aligned on a grid on the blade surface, on which the pressure was calculated from the fluid particles nearby, using the Shepard method [10].

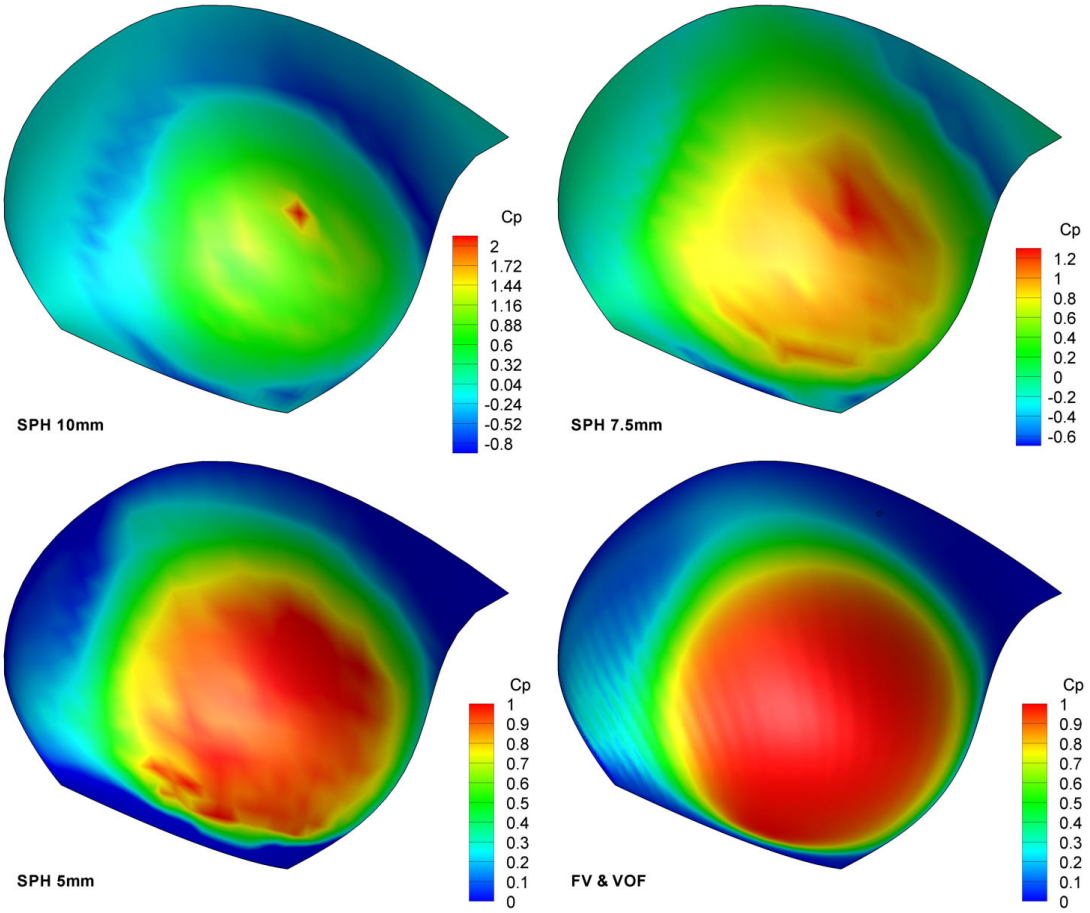


Fig. 5.34. Pressure coefficient distribution on the Turgo blade surface. Comparison of the different SPH resolutions along with the Fluent solution.

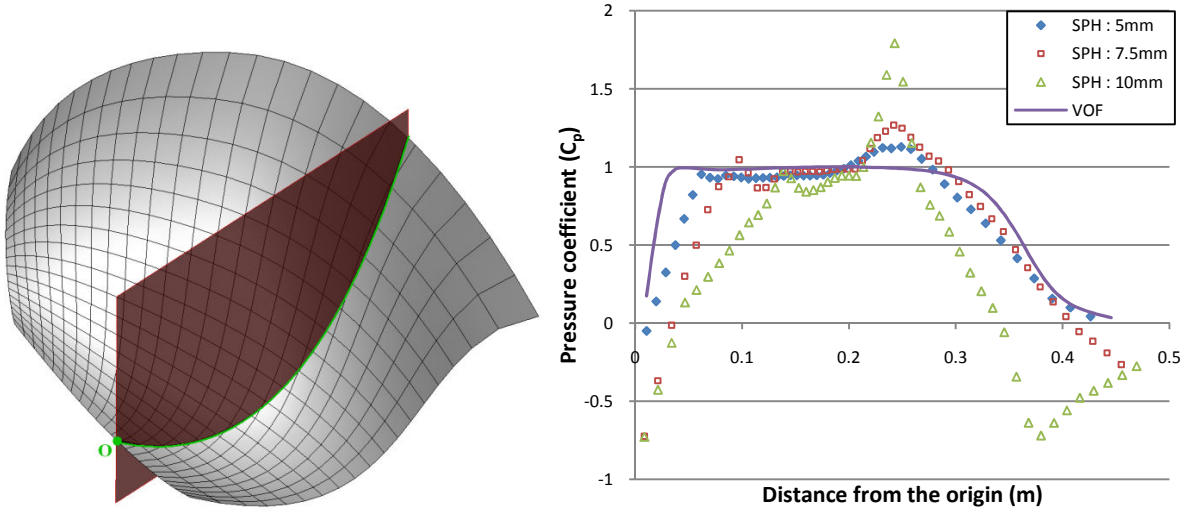


Fig. 5.35. Left: Grid of pressure probes on the blade surface. The line on which pressure is evaluated is also visible. Right: Pressure coefficient distribution on the Turgo blade surface. Comparison of the different SPH resolutions along with the Fluent solution.

During the interaction between the blade and the water jet, there is a transient period of approximately 0.02sec; during this phase the initial impact on the blade occurs, thus forces obtain

very large absolute values (see fig. 5.36, similar behavior was noticed with the other particle resolutions too). In all SPH cases, net forces exhibit oscillations; thus, time averaging is performed in all cases. The finer resolution though, is the one which gives the smoother results (fig. 5.36) after the initial transient period. The average force values for each case are presented in Table 5-I, along with results from Fluent.

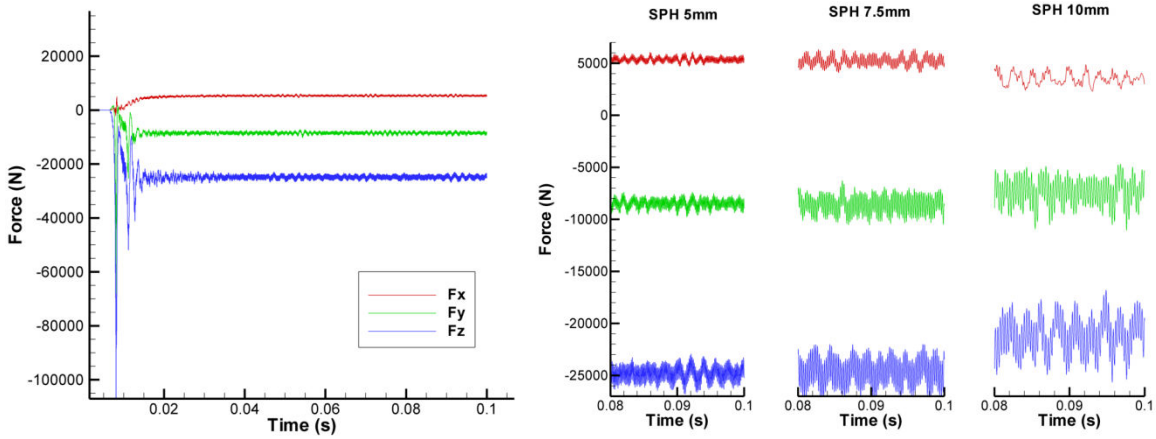


Fig. 5.36. Left: Force history on the blade. Right: Comparison of the force history after the initial transient phase, for different resolutions.

The finer SPH discretization tested (31 particles on jet diameter - particle size 5mm) was able to predict forces accurately in comparison to Fluent (max. absolute deviation 1.6%). The intermediate particle discretization (21 particles on jet diameter - particle size 7.5mm) performed with a max. absolute deviation of 4%, whereas the coarser particle discretization (15 particles in the jet diameter - particle size 10mm) the deviation becomes significant with a maximum absolute deviation of 35%.

Table 5-I. Calculated forces on the Turgo blade for 30^o jet impingement

Calculated forces	F_x [N]	F_y [N]	F_z [N]
Fluent	5425.575	-8614.56	-25216.4
SPH: fine	5338.143	-8476.81	-24802.1
SPH: intermediate	5199.245	-8577.67	-24623.3
SPH: coarse	3516.502	-7333.66	-21243.8

Considering all the above results, it is obvious that the fine particle resolution achieves the best results for the specific simulation, comparing to the other resolutions. However, the time needed to complete the simulation increases with the particle number. In Table 5-II comparative results, regarding the time needed to complete the simulation, are given for all SPH cases and Fluent. The hardware used for the SPH cases is a 2xQuad Core Xeon E5405 2.0 GHz computer. The SPH program used all 8 physical CPUs of the computer. On the other hand, the Fluent simulation was done on a i7 940 2.93GHz computer, using 4 CPUs.

Table 5-II. Time required for the execution of the programs

Resolution	Time step [sec]	CPU time	Number of particles/volumes
Fluent: 5mm	$7.10^{-7} - 5.10^{-6}$	355 hrs	594674 / 1360110 before / after mesh adaption
SPH : Fine (5mm)	4.10^{-6}	89 hrs	105679
SPH : Intermediate (7.5mm)	6.10^{-6}	14 hrs 8 min	33987
SPH : Coarse (10mm)	8.10^{-6}	5 hrs 4 min	13902

After determining the required particle resolution to properly describe the underlying flow patterns, the two other impingement cases have been also simulated. The 47° jet impingement was selected because the impingement angle was equal to the inlet angle of the blade. The free surface is correctly represented, as it is shown in fig. 5.37. The 60° jet impingement is a more interesting case, since the impingement angle is larger than the inlet angle, causing part of the leakage flow to interact with the impinging jet, leading to the formation of an unsteady bubble, which is accurately captured by the SPH method, even if the air phase is not modeled.

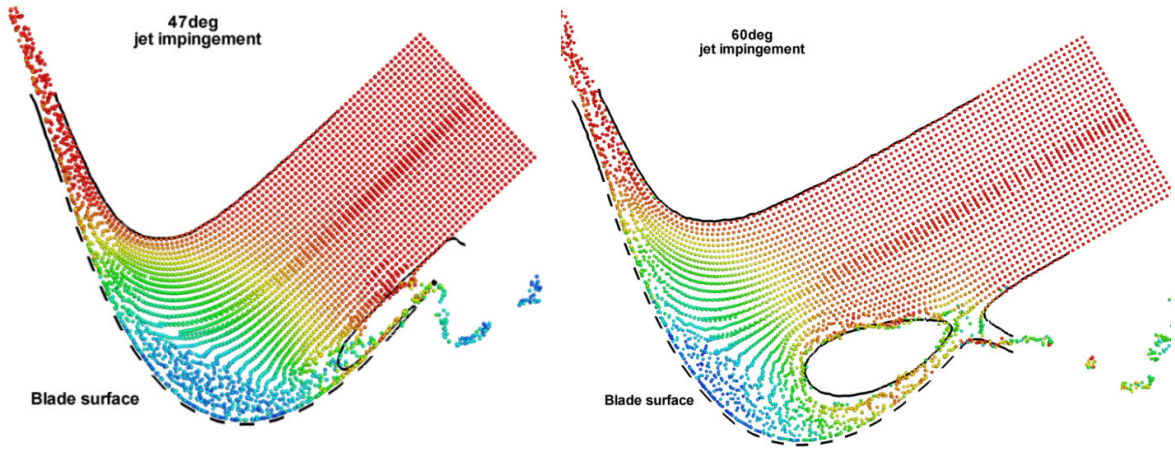


Fig. 5.37. Left: 47° jet impingement . Right: 60° jet impingement.

Table 5-III and Table 5-IV show the calculated forces on the blade for the two cases, which are in agreement; maximum deviation for the 47° jet impingement is ~2% and for the 60° jet impingement 4%.

Table 5-III. Calculated forces on the Turgo blade for 47° jet impingement

47° jet impingement	F_x [N]	F_y [N]	F_z [N]
SPH	4885.766	-12173.4	-22985.4
Fluent®	5019.615	-12424.7	-22986.1
% deviation	2.66	2.02	0.01

Table 5-IV. Calculated forces on the Turgo blade for 60° jet impingement

60° jet impingement	F_x [N]	F_y [N]	F_z [N]
SPH	3745.34	-15280.2	-16882.9
Fluent®	3927.32	-15720.2	-16695.8
% deviation	4.63	2.79	-1.12

To sum up, in the previous simulations the free-surface flow is captured accurately, even in the most complex cases, as in the 60° impingement, where part of the flow interacts with the incoming jet, forming a bubble. Also the acting forces on the turbine blade are in accordance to the results from Fluent. The above comparisons show that the standard SPH model is able to predict well the flow in the non-rotating Turgo turbine blade, comparing to other CFD tools.

Rotating Turgo turbine runner

After successfully simulating the stationary Turgo turbine blade using SPH, the SPH algorithm was extended for solving rotating geometries, such as the rotating Turgo turbine. Since the SPH method is capable of handling moving geometries, this extension practically involved only the description of the runner geometry, including hub and tip. An indicative view of the turbine runner is shown in fig. 5.38, in one jet operation. The turbine runner hub is at 260mm radius and the tip at 500mm radius from the center of rotation. The nominal diameter of the runner is at 770mm. Also the runner consists of 22 blades. The water jet had a diameter of 154mm and velocity 32m/s (discharge 596m^3/s) and entered the runner at 25° degrees from the runner rotation plane. The rotational speed of the runner was set to 44.8rad/s and rotates around the *x*-axis.

In order to reduce the computational effort, it was decided to model the flow between only two successive blades including the hub and tip walls (see also fig. 5.39), assuming periodic flow conditions on the rest blades [11, 12]. The main interest of this simulation is the torque curve expressing the developing torque on a Turgo turbine blade as a function of its angular position. Thus, in order to prove the validity of the modeling method, it is required to compare the torque curve on a blade, using the two-bladed approach described above, with the calculated torque curve by modeling more successive turbine blades (in particular seven successive blades). For the two-bladed simulation, the physical time was 42ms, while the first blade is located at φ=32^o; in this time interval the blades perform a rotation of ~107^o, which is enough to cover the complete interaction of the jet with a single blade. On the other hand, the simulation of the seven successive blades was performed for more time in order to properly describe the jet-runner interaction.

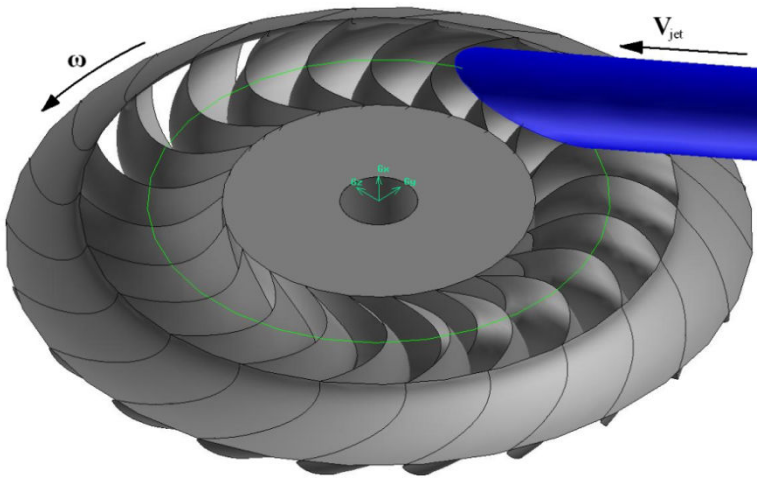


Fig. 5.38. Turgo runner view. The green line represents the nominal radius, where the jet axis is located.

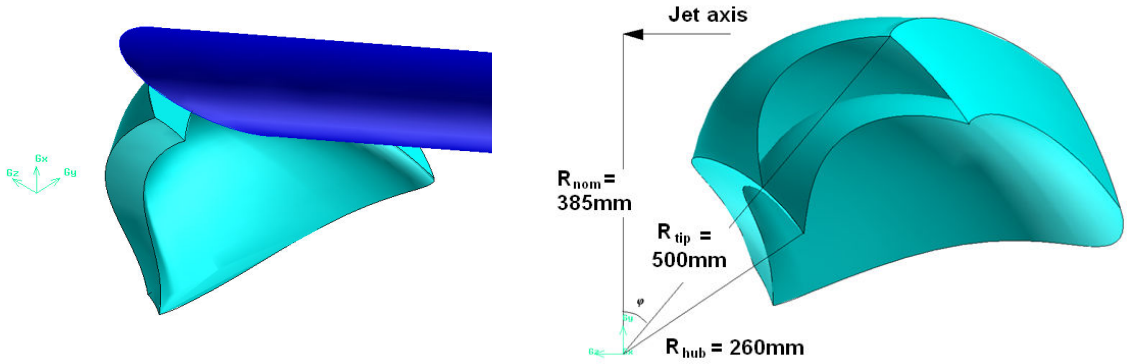


Fig. 5.39. Left: Computational domain solved. Right: characteristic dimensions of the turbine.

The general blade dimensions are the same with the one used in the stationary Turgo blade analysis, thus the same particle resolution was used, i.e. the particle size was 5mm, since it was found adequate after the particle dependence analysis. The numerical speed of sound used was set to 350m/s, larger than $10V_{jet}$ to ensure incompressibility of the simulated water.

In fig. 5.40 indicative instances of the flow are shown from the simulation involving seven successive blades. In both figures the water evacuation of the runner blades is visible and the water forms low velocity water sheets. The low velocity of the water sheets is expected, due to the energy transfer from the high velocity water jet to the turbine runner. The kinetic energy and consequently velocity of the water at the outlet should be as low as possible in order to maximize the energy conversion from hydraulic to mechanical; there is only a residual axial velocity component. Leakage is also visible from the inlet of the runner, as it was expected from the previous results of the stationary blade.

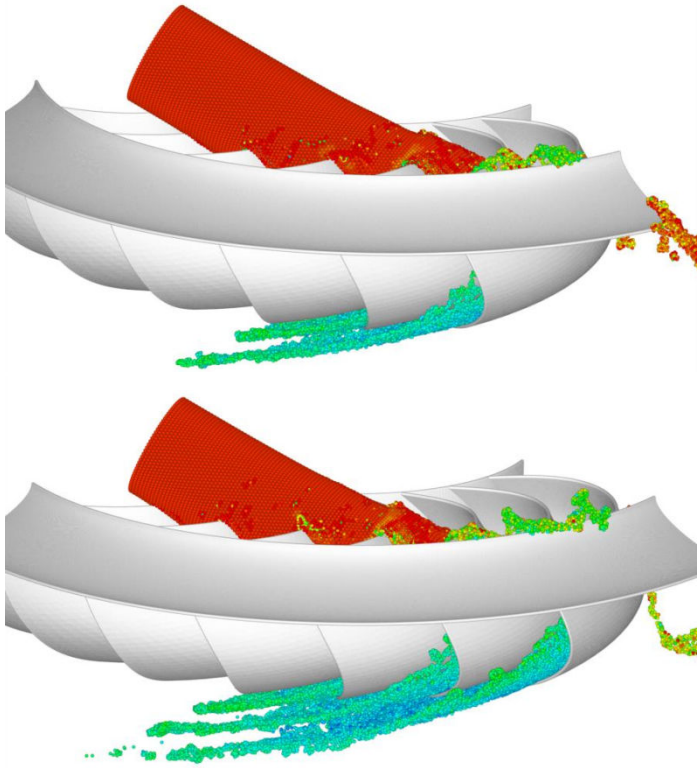


Fig. 5.40. Indicative instances from the Turgo turbine simulation, using multiple consecutive blades.

In fig. 5.41, the developing torque on the turbine blades is presented in relation to the angular location of the turbine blade. It must be highlighted here that the graph on the left expresses torque as a function of the φ angle of the first blade (as shown in fig. 5.39), whereas the right graph shows all torque curves shifted by an appropriate angle in order to emphasize their differences. Note that generated torque exhibits some oscillations similar to those in fig. 5.29. Thus, torque averaging is performed every one degree in order to obtain a smoother curve.

The torque graph exhibits a periodic behavior. However, torque graphs are not identical. Indeed, as other reserchers have also experienced, the SPH reults exhibit some dispersion due to the particle nature of the SPH method [13]. Nevertheless, by comparing the torque graphs of the consecutive blades it is obvious that differences are practicaly negligible, whereas the difference of the total work for each blade is less than 0.1%. Moreover the two-blade simulation is able to give results very close to the simulation of the seven turbine blades, the total work per blade calculated differs by less than 2%. Considering that the cost of the latter simulation is about five times higher than that of the two-blade simulation, which requires about a day to finish, the two-blade approach is adopted for all the rest simulations.

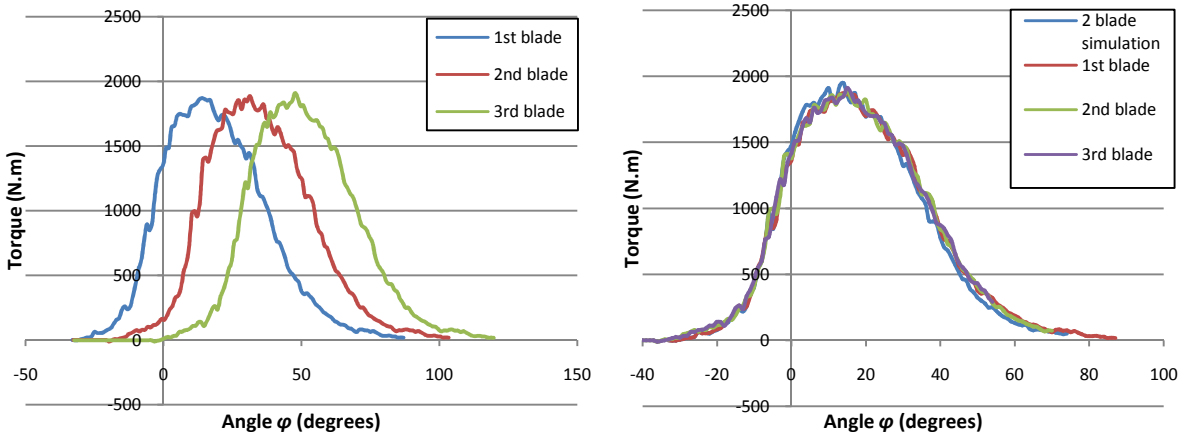


Fig. 5.41. Left: Torque on the first three consecutive blades. Right: comparison of the calculated torque from the two and seven blade simulation

At first, two different geometries were tested and compared with the results of the Fluent software in order to determine the performance and the accuracy of the SPH method. The two geometries were produced from a preliminary optimization with the FLS algorithm, for maximizing turbine efficiency. The inner shape of the geometries modeled is presented in fig. 5.42. The two geometries differ both at the inlet and the outlet edges and at the respective angles. In fig. 5.43 the complete runner of each case is shown, focusing on the outlet of the runner. A notable difference between those two geometries is the fact that geometry *B* is slightly extended at the outlet region, thus leading to a longer interaction with the water.

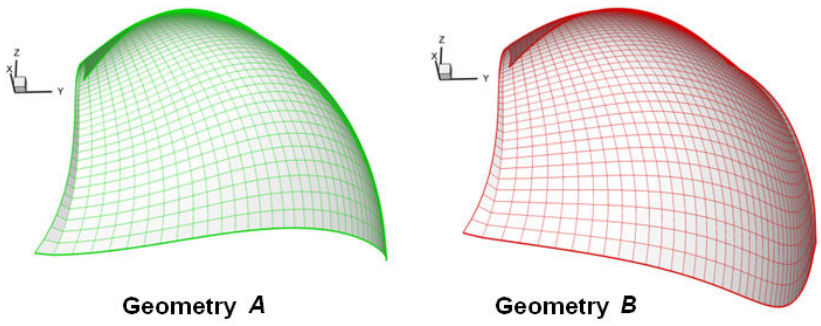


Fig. 5.42. The two different geometries used.

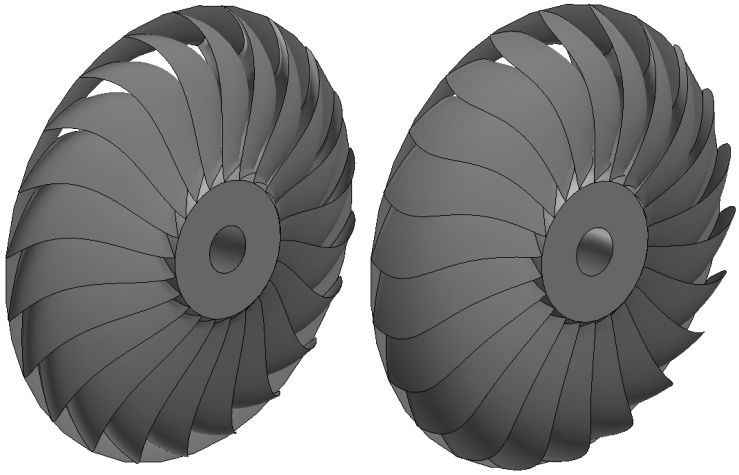


Fig. 5.43. Outlet view of the complete Turgo runner, for the two geometries: left-*A*, right-*B*.

The following figures (fig. 5.44 and fig 5.45) show the water jet and turbine runner interaction at various time steps for the two geometries during different interaction phases such as:

- at 8.4ms, during the initial impingement on the blade
- at 16.8ms, during the filling of the space between the two blades with water
- at 25.2ms, the evacuation begins, while the water jet is cut by the following blade
- at 33.6ms, the evacuation phase continues.

For clarity of the displayed images, the second blade is not shown.

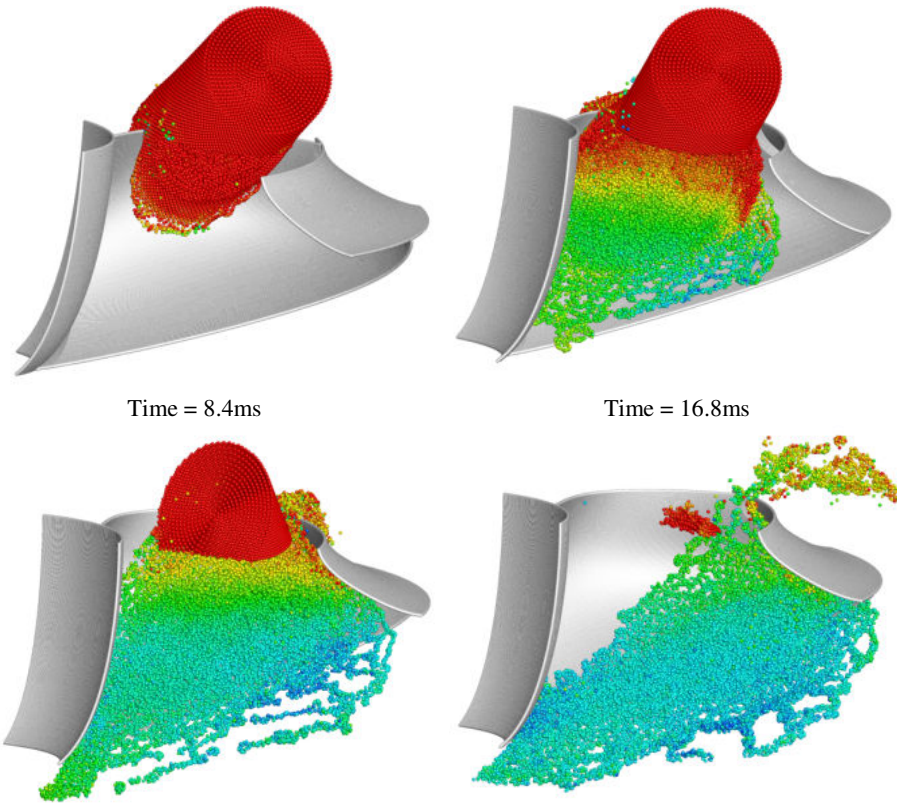


Fig. 5.44. Instances of the flow pattern for the Turgo geometry *A*. Particle coloring according to velocity magnitude.

Flow patterns are similar for geometry *B*; however water distribution is smoother through the whole blade surface.

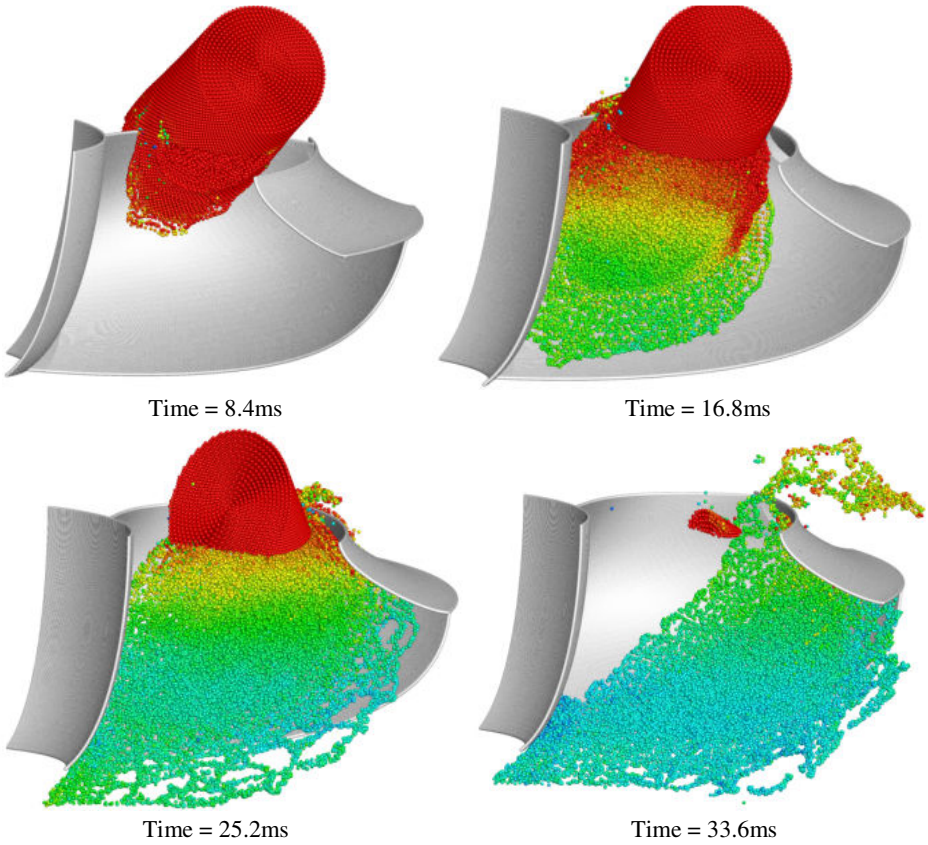


Fig. 5.45. Instances of the flow pattern for the Turgo geometry *B*. Particle coloring according to velocity magnitude.

For the Fluent solution, the sliding mesh approach was used. Thus two separate domains were defined one rotating and one stationary (see fig. 5.46). Between these two domains an interface was defined, in order to interpolate variables from the one mesh to the other. The volume size used was 5mm, similar to the SPH particle size. Larger mesh sizes, with a volume size of 7.5mm, have been used too, giving small difference in the torque curve, ensuring the grid independence of the results.

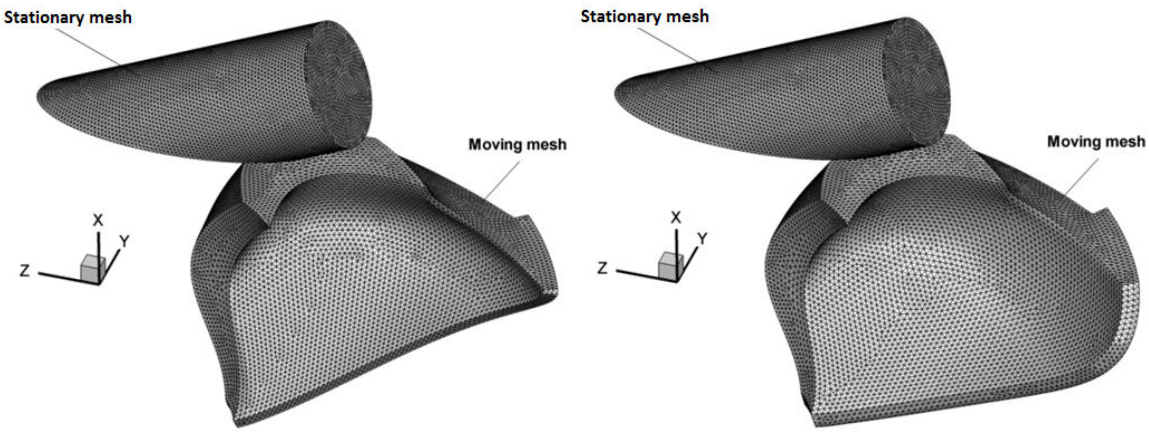


Fig. 5.46. Indicative meshes used by Fluent. Left: geometry *A*. Right: geometry *B*.

In the following figure (fig. 5.47), indicative results of the Fluent simulations are shown for both geometries. The flow field looks similar with the respective results of the SPH method in fig. 5.44 and fig. 5.45. A notable difference is that the water flow remains attached to the second blade creating a negative pressure region, which contributes to the total turbine torque. On the other hand the SPH method is unable to predict this flow feature. The main reason is the boundary conditions selected. Indeed, even if the boundary condition of repulsive boundary particles does not allow fluid particles to penetrate the wall, it does not truly enforce a zero pressure gradient on the wall.

Another observable flow feature is the splashing which occurs at the hub of the runner (fig 5.47). The SPH method predicts a similar effect but to a lesser extent, due to the resolution used. A finer particle resolution would be possible to capture this effect more accurately, but since this feature does not contribute on the developing torque, further increasing resolution was deemed unnecessary.

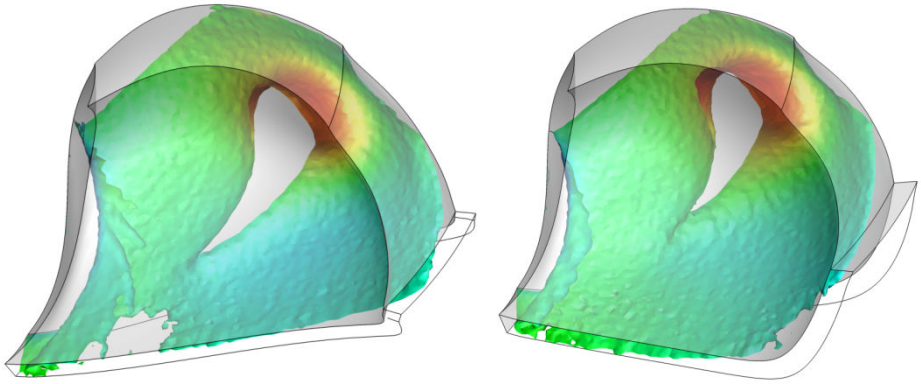


Fig. 5.47. Indicative snapshot from the Fluent solution. Time= 25.2ms. In both pictures the free surface is shown colored according to the velocity magnitude. Left: geometry *A*. Right: geometry *B*.

As in the previous cases tested with SPH, torque exhibits oscillations and averaging (in time or in respect to angular position) is required in order to smooth the torque curve. Nevertheless, the SPH method predicts the torque exerting on the impinged blade in agreement with the Fluent solution. Both methods predict that geometry *B* has better performance by ~4%. Torque on geometry *A* develops earlier due to the shape of the inlet edge; the inlet edge in this case is more curved and cuts the water jet earlier.

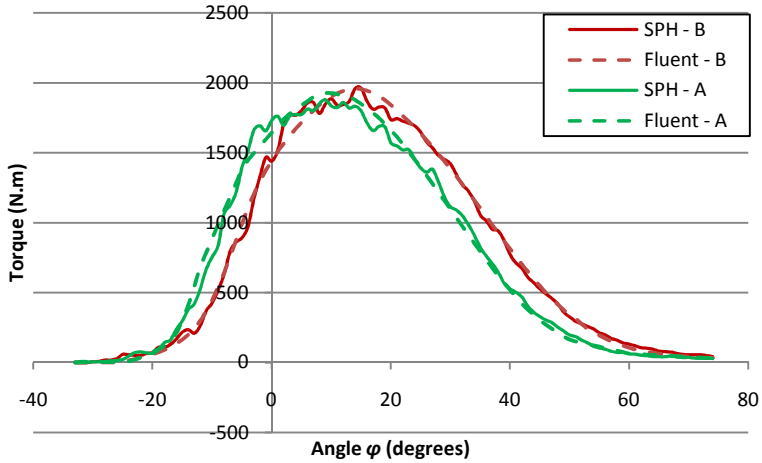


Fig. 5.48. Torque graph for the two geometries

In figures 5.49 and 5.50 the area of the blade, covered by water is shown at two different time instances for both geometries. Results by both programs are very close. The sharpness of the SPH

results is attributed to the coarseness of the interpolation grid used to obtain the respective data, note though that this mesh was not involved in the flow field calculations.

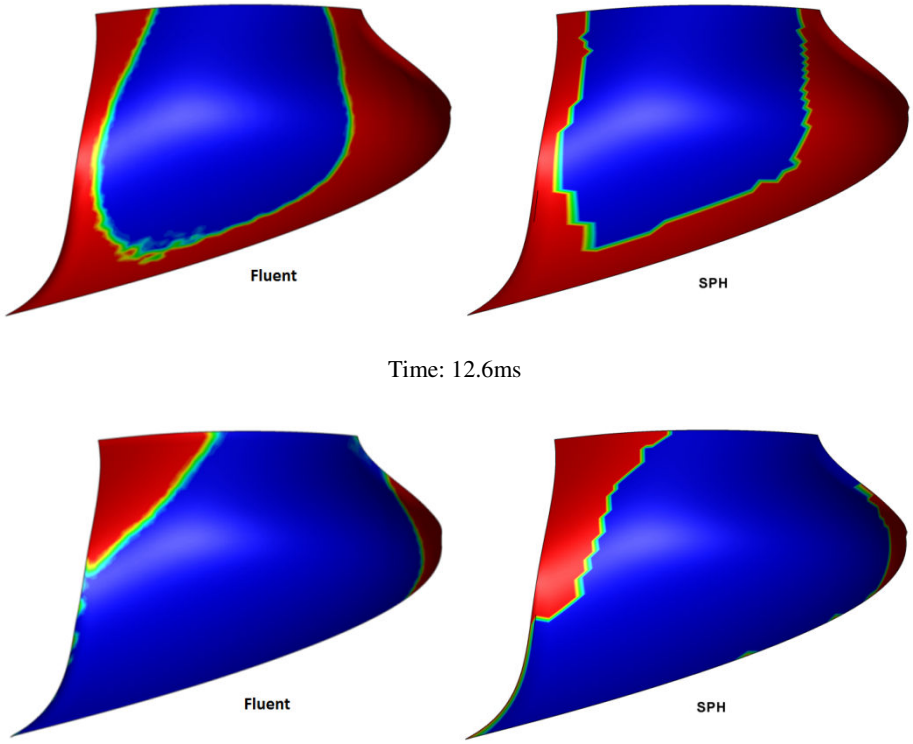


Fig. 5.49. Free surface evolution on the blade of geometry *A*. Blue represents water and red air for the Fluent solution, or void for the SPH solution.

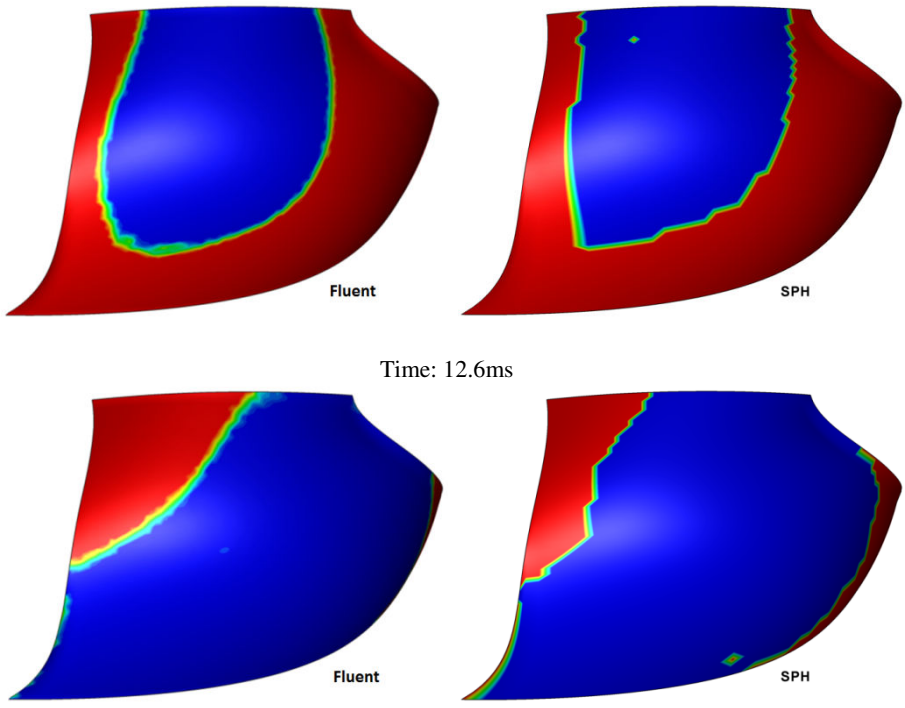


Fig. 5.50. Free surface evolution on the blade of geometry *B*. Blue represents water and red air for the Fluent solution, or void for the SPH solution.

Finally in fig. 5.51 the average torque distribution on the turbine blade is shown for the two geometries tested. In both cases maximum torque develops in curved areas where the direction of the flow changes. Geometry *B* has a smoother torque distribution at a larger area due to its design.

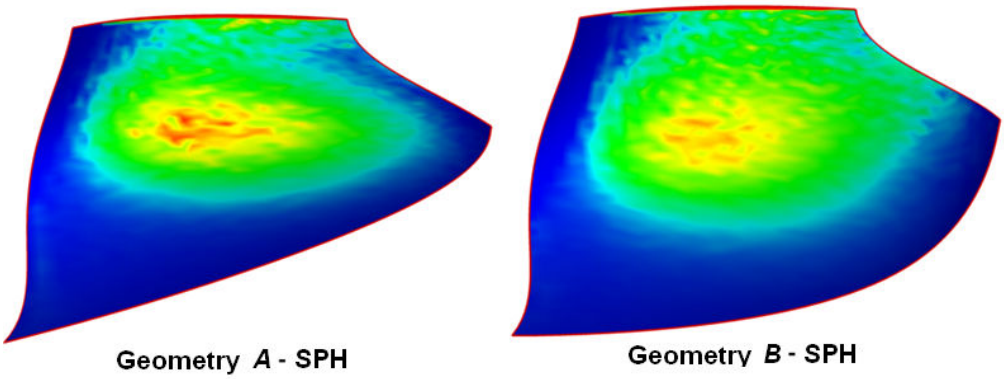


Fig. 5.51. Average torque distribution through the water jet – bucket interaction. Red corresponds to higher torque values, blue to lower.

Considering the time needed for the execution of the two programs, the SPH method is much faster than Fluent, due to the 'embarrassingly parallel' nature of the SPH algorithm. The SPH algorithm only needed 10hrs on a 2xQuad – Core Xeon 2.Ghz computer (80 CPUhrs), for each simulation. On the other hand Fluent needed ~10days using 4 parallel processes on a i7 2.97Ghz computer (960 CPUhrs).

Parametric tests – inlet angle dependence

From the previous results it has been shown that the geometry *B* performs better than geometry *A*. For that reason geometry *B* was used to perform a further design test regarding the water jet inlet angle. Simulation conditions will be the same as before, apart from the jet inlet angle. The inlet angles which were tested are 20°, 30°, 35° and 40° (fig. 5.52).

In figure 5.53 a comparison between the torque graphs for the different water jet impingement angles is made; from this figure it is shown that, by increasing the water jet impact angle, the developing torque curve becomes narrower and exhibits a higher peak value. This is further illustrated in figure 5.54, where the average torque distribution is shown; again for larger impact angle the peak value is higher and the peak torque value is moved towards the center of the blade. By calculating the integral of the torque curve it was found that the maximum efficiency is achieved for the intermediate inlet angle of 30°, as shown in figure 5.53 in the respective graph of normalized efficiency (efficiency is normalized by the maximum value, i.e. the efficiency of the turbine for 30° inlet angle).

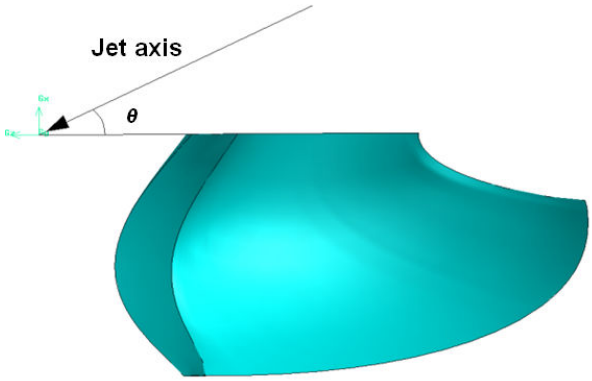


Fig. 5.52. Jet angle impingement schematic

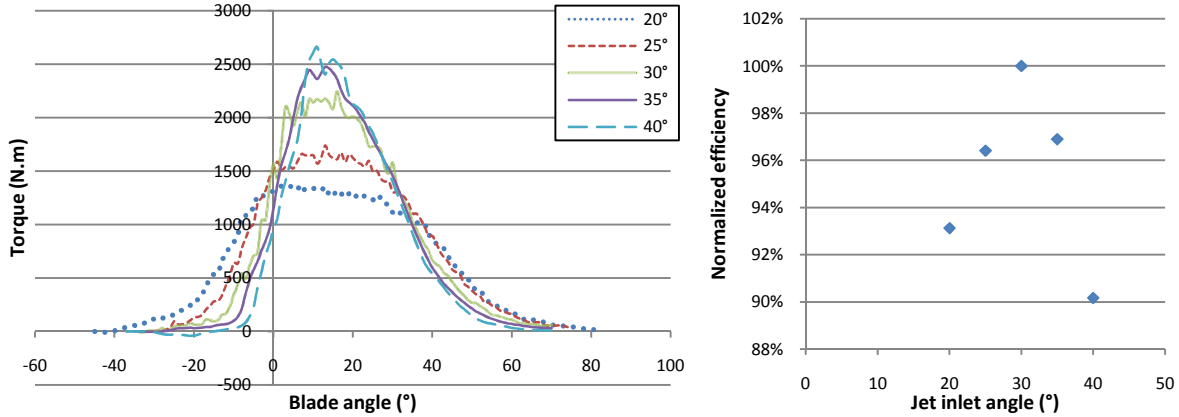


Fig. 5.53. Dependence of torque on jet inlet angle. Left: comparative torque graph. Right: normalized efficiency graph.

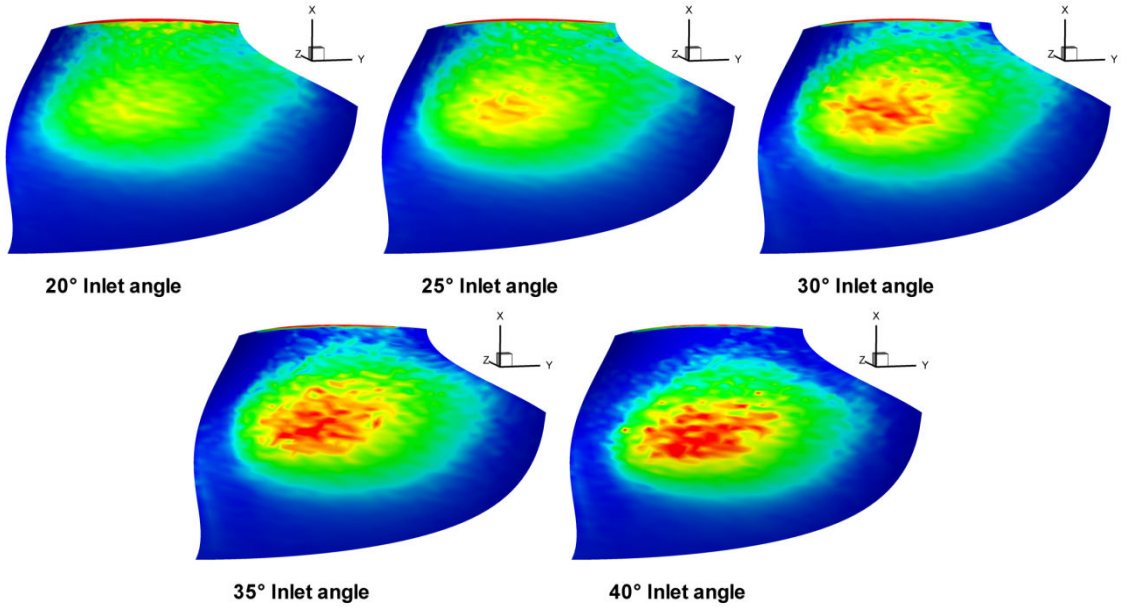


Fig. 5.54. Comparative torque distribution on the turbine blade. Red corresponds to higher torque values, blue to lower.

Parametric tests – water jet location

Another parametric test performed for the Turgo turbine is the selection of the water jet location. The water jet was shifted on the *yz* plane by a distance $D=-R_{jet}$, 0, $+R_{jet}$, while being parallel to its original axis (see fig. 5.55, positive direction is towards the $+z$ axis).

The comparison between the developing torque is made in fig. 5.56; by moving the jet towards $-z$, torque appears earlier, since the jet-blade interaction occurs earlier too. Also the torque graph becomes narrower, but with a higher peak value. Torque tends to be distributed towards the outer radius of the blade (see fig. 5.57). The opposite happens when the jet is moved towards the $+z$-axis. However the maximum efficiency is found when the jet is at its reference position, i.e. when D=0.

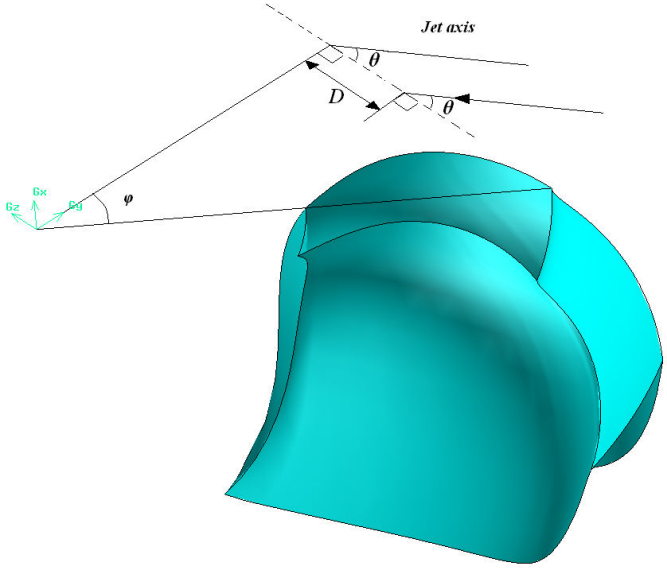


Fig. 5.55. Schematic showing the jet position.

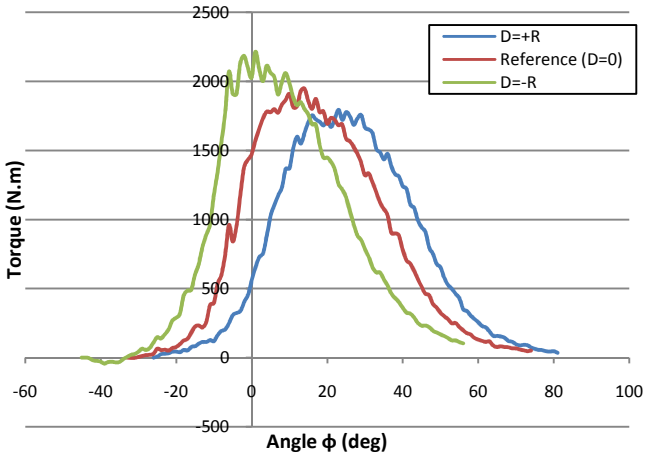


Fig. 5.56. Comparative torque graph.

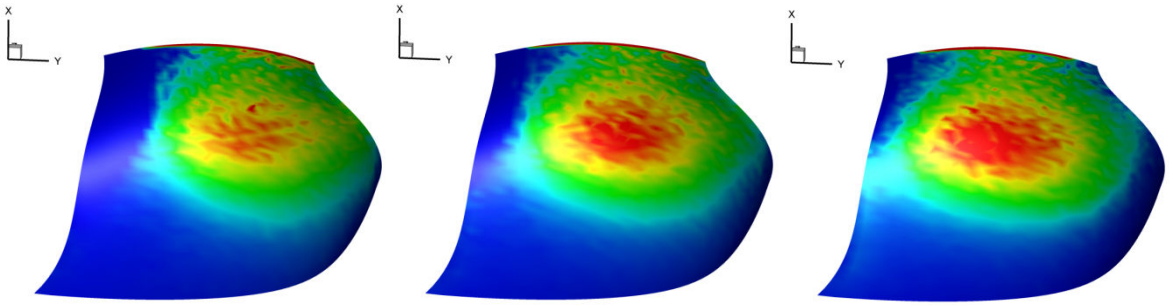


Fig. 5.57. Comparative torque distribution on the turbine blade. Red corresponds to higher torque values, blue to lower. From left to right D: -R, 0, +R

Model Turgo turbine runner optimization

The SPH algorithm was also used in the design optimization process of a small Turgo turbine for the Hydroaction project of the 7th framework program [14]. Considering all the previous results it was decided to proceed with the optimization of a Turgo turbine using a blade geometry similar to the geometry *B*. Blade dimensions are scaled down by a factor of 2.5 in respect to geometry *B*, leading to the following turbine dimensions:

- R_{hub} = 104mm
- R_{tip} = 204mm
- R_{nom} = 142.5mm (nominal diameter, where the water jet impacts)

The turbine operating point is for a nominal flow rate of 90lt/and a gauge height of 48mWG, on one jet operation (two jet operation is possible and will be investigated later on), leading to water jet velocity of $V_{jet} = \sqrt{2gH} \approx 30 m/s$, impinging the turbine runner under 25° in relation to the runner's level of rotation. The turbine runner also rotates at 1000rpm (104.72rad/s).

For the optimization procedure, *evolutionary algorithms* (*EA*) were used. In particular, the EASY software [15], developed by the Laboratory of Thermal Turbomachines of NTUA, was used. Evolutionary algorithms mimic mechanisms of biological evolution in order to find the optimal solution. Briefly described here, the main idea behind evolutionary algorithms is to code the problem parameters in a form of *genes* [16]. An initial population of *candidate solutions* is created, from random genes. The candidate solutions are evaluated; then the best solutions are used as *parents* for the next generation of candidate solutions, which are called *offsprings*. Offspring's genes are formed by gene *crossover* of the parents involved. *Mutation* is also possible to occur randomly, altering parts of the genes in order to create new possible solutions. Then the offspring generation is evaluated again and is used as parents for the next generation. The described procedure is iterated several times, until an optimal solution is found.

Evaluation may be performed using one or more objectives (*SOO-single objective optimization* and *MOO-multi objective optimization*). *Constraints* may also be considered in the optimization. The great advantage of the EAs is that the procedure is able to handle problems involving many parameters and many objectives. In computational fluid dynamics evaluations are performed by flow solvers, which generally may be rather computationally expensive, especially considering that during an optimization procedure the evaluator may be used several thousand times, in order to reach to an optimal solution. Thus, often EAs are combined with more elaborate tools, such as *metamodels*, *artificial neural networks*, etc. which are used to perform fast but inexact evaluations [15, 16] and eventually greatly speed up the whole procedure. For more information on evolutionary algorithms the reader is addressed to [16].

Since the SPH method is rather slow, requiring ~1day for a single evaluation, the FLS algorithm [3, 8] was used as a first step of the evaluating procedure. The FLS algorithm (being developed by F. Stamatelos as part of his doctoral dissertation) is based on a simple Lagrangian approach for simulating the jet as consecutive segments or frames of particles. The algorithm tracks the particles, calculating their trajectories and the energy exchange with the blade/bucket surface. The equations of motion of the fluid particles are solved in the rotating system of reference. The main advantage of the FLS algorithm is the very fast execution, since the algorithm is able to perform a single evaluation within several seconds on a single core modern personal computer, rendering it an attractive tool for optimization. However, since each particle is tracked individually from the rest particles, the particle interactions are not taken into account, thus the algorithm requires tuning in order to describe realistically the jet spreading on the blade surface. Additional tuning is required for the prediction of the energy losses due to viscosity, impact on blade of change in flow direction. These parameters

were tuned for operation in Pelton turbines, thus adjustments were required for properly describing the flow patterns in a Turgo turbine, especially for capturing jet spreading. Since an experimental installation was not, at that time, available, these adjustments were made using results of the SPH algorithm. After proper tuning, calculated flow patterns from the FLS and SPH algorithms look similar (fig. 5.58).

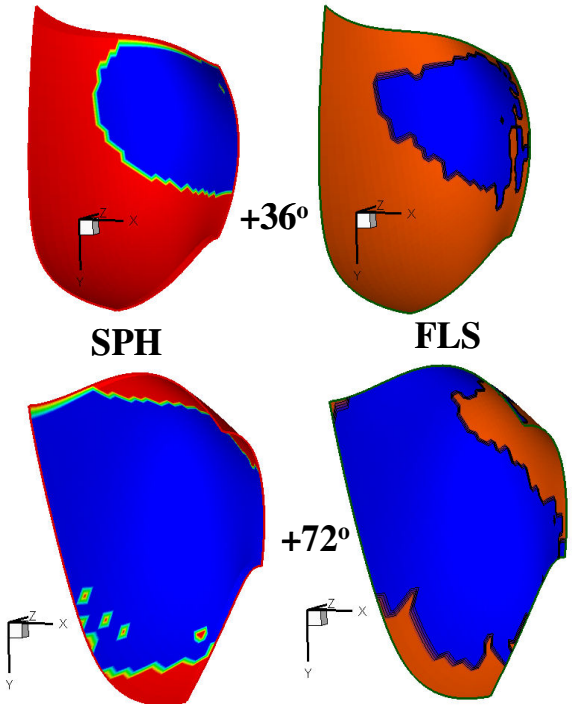


Fig. 5.58. Comparison of the flow spreading on the blade surface calculated by SPH and FLS, after adjusting FLS algorithm.

EASY software was coupled with FLS algorithm and the optimization was performed using as single objective the maximization of the turbine efficiency [6]. Optimization with the FLS algorithm resulted to several elite blade geometries, some indicative are shown in fig. 5.59. The blade geometries had differences between them, which affected the turbine efficiency by more than ~1%. The SPH algorithm was used to evaluate them in order to find the best among them. The best efficiency was found for blade geometry #3, as shown from fig. 5.60 (in fig. 5.59 and 5.60 *reference* denotes geometry *B* properly scaled). Then, geometry #3 was used to construct a full CAD model of the turbine runner, including turbine hub and tip. In order to ensure the accuracy of the geometry design, a blade was constructed directly from STL surfaces (STL - standard tessellation language, which is a format used for stereolithography CAD) using rapid prototyping. The prototype blade was then used as a model to create a mold, which was then used for the casting process (fig. 5.61). On the other hand, the hub was machined in a 5 axis CNC machine. Eventually the turbine runner was assembled and welded (fig. 5.62). More information on the turbine construction are in Appendix E.

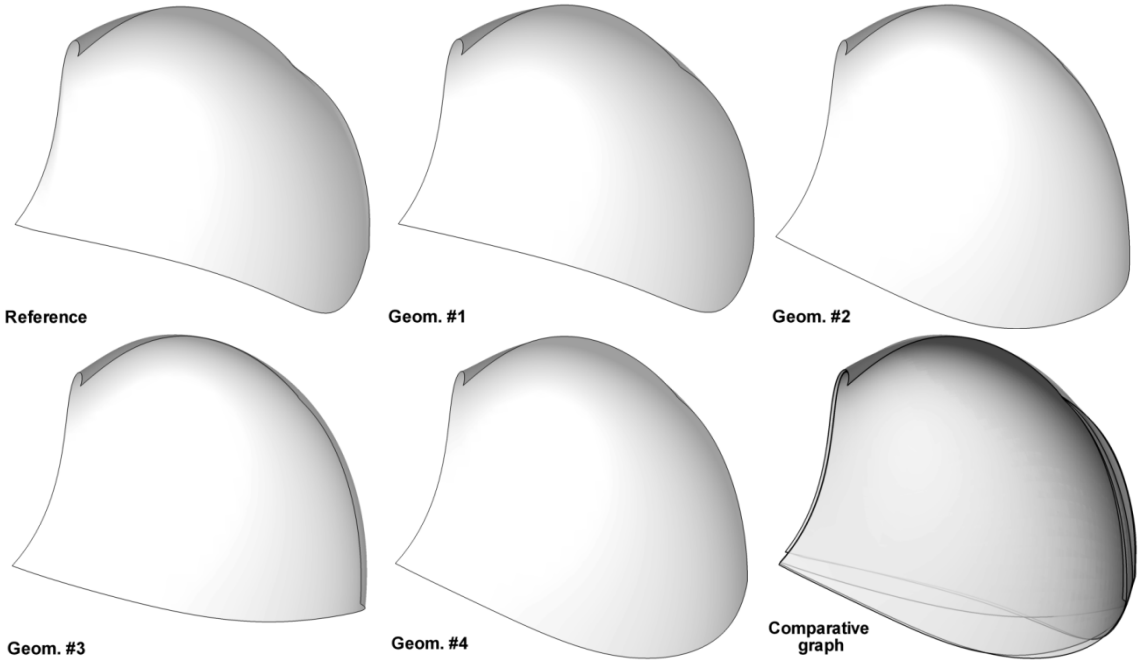


Fig. 5.59. Indicative elite geometries after the optimization procedure with the FLS algorithm.

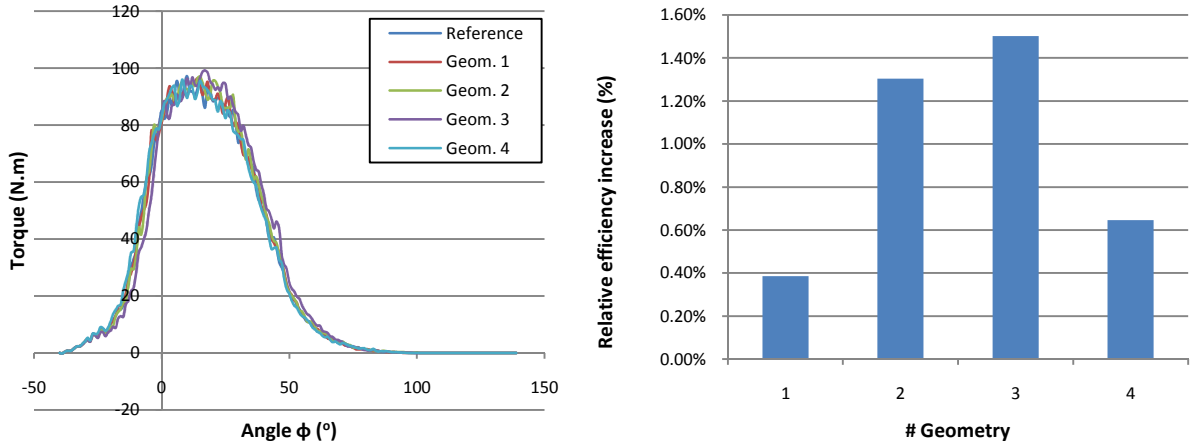


Fig. 5.60. Left: Torque curves for different geometries. Right: relative efficiency increase in respect to the reference geometry.

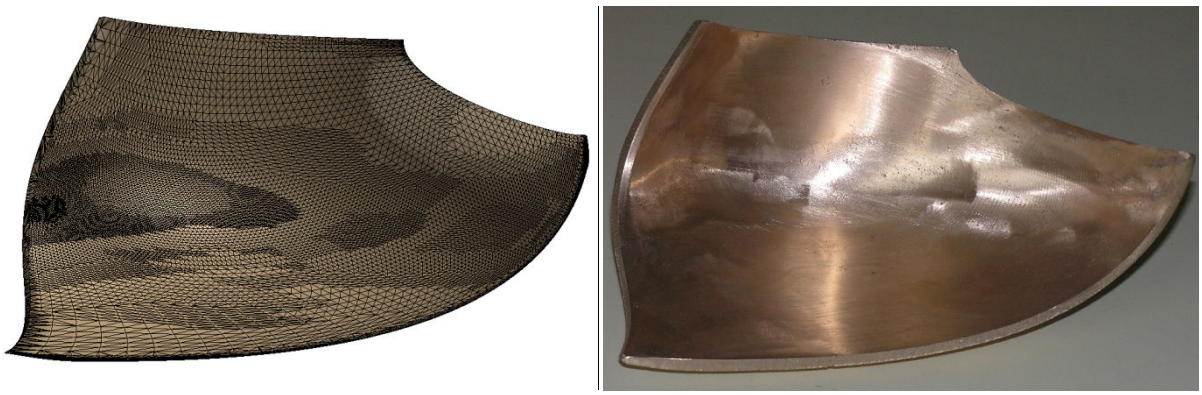


Fig. 5.61. Left: 3D CAD drawing of the turbine blade. STL triangulation is also visible. Right: Turbine blade, after casting and machining.

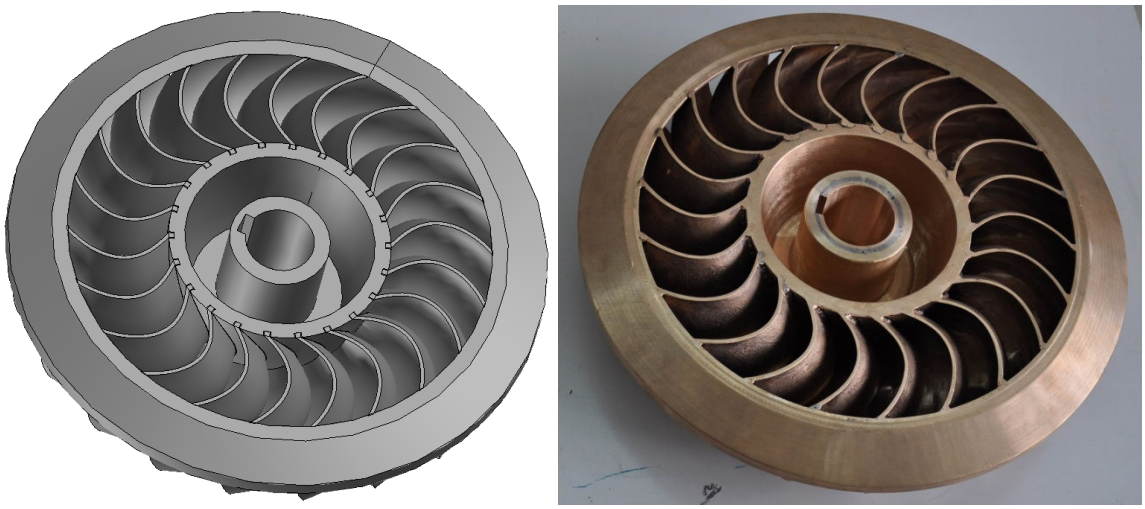


Fig. 5.62. Left: Turgo runner 3D CAD drawing. Right: Actual runner assembled.

The whole runner had a volume of ~ 4659662mm^3 and, since the runner was made of phosphorus bronze (density of 8790-8920kg/m^3), it had a total mass of ~41kg.

Apart from the torque graph, the SPH method predicted forces in x, y, z direction, on a single blade. These forces are important because they are essential for determining the bearing loads in the axial and the radial direction. In fig. 5.63 the calculated forces are shown for a single blade and for the complete runner. These forces correspond to upper jet operation. Lower jet operation would result to the same axial force, but opposite radial force due to symmetry. Forces of the complete runner have been calculated, assuming periodic flow conditions, thus periodic blade loads, by phase shifting the force curve of a single blade by an angle of 16.36^o (=360^o/22) and summing respectively forces. To be more precise, assuming that force is a function of φ angle of a specific blade, $f(\varphi)$, then the force of the complete runner may be calculated as:

$$F_{run} = \sum_{i=1}^{22} f_{blade}\left(\varphi - i\frac{360}{22}\right) \tag{5.6}$$

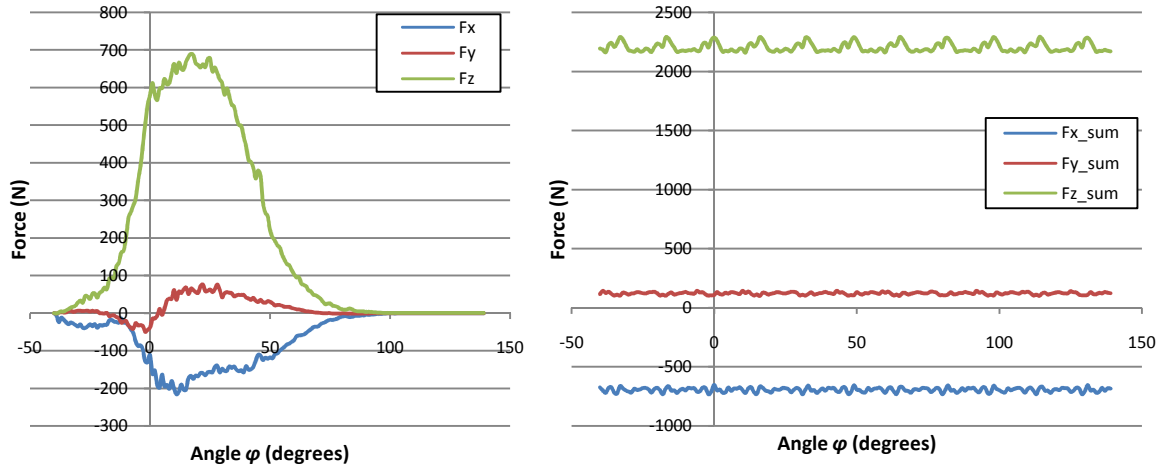


Fig. 5.63. Left: Forces on a single blade. Right: force on the complete turbine runner.

In this way it is possible to determine the axial and radial forces on the turbine runner shaft. After averaging the total runner force:

$$\begin{cases} F_x = -693.75N \\ F_y = 122.12N \\ F_z = 2205.804N \end{cases}$$

The axial force is the force on the *x*-axis, i.e. F_x. The radial force is the net force of F_y and F_z, thus $F_R = \sqrt{F_y^{\,2} + F_z^{\,2}} = 2208N$. It is possible to confirm these forces from the velocity triangles (fig. 5.64) in the turbine runner, using the momentum theorem:

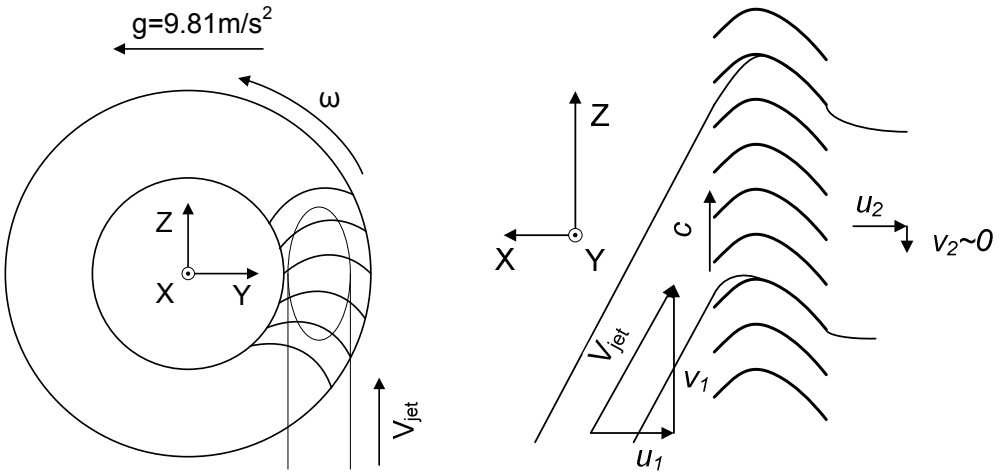


Fig. 5.64. Left: Upper nozzle operation. Right: Velocity triangles.

Inlet conditions:
$V_{jet} = \sqrt{2gH} \approx 30m/s$
u_1= -12.67m/s
v_1= 27.18m/s

Outlet conditions:
u_2~ -5m/s
v_2~0m/s

Thus:

- *x*-axis:

$$F_x = \rho Q(u_1 - u_2) = -644N \tag{5.7}$$

- *y*-axis:

$$F_y \approx 0N \tag{5.8}$$

- *z*-axis:

$$F_z = \rho Q(v_1 - v_2) = 2442N \tag{5.9}$$

The aforementioned loads occur during turbine operation with the upper nozzle. Table 5-V shows the results of turbine forces (F_x, F_y, F_z are the total runner forces due to fluid flow only).

Table 5-V. Calculated forces of the Turgo model turbine

Forces in (N)	Upper jet	Lower jet	Both jets
F_x	-693.7	-693.7	-1387.5
F_y	122.1	-122.1	0.0
F_z	2205.8	-2205.8	0.0
Weight (y-axis)	-402.2	-402.2	-402.2
F_{axial}	-693.7	-693.7	-1387.5
F_{radial}	2223.5	2267.3	402.2

Complete runner simulation

Apart from the simple simulations involving two or seven consecutive blades, one involving the complete turbine runner was performed, under one and two jet operation, including part of the turbine casing. This simulation was performed in order to investigate possible interactions of the outflowing water sheets with the turbine casing, the turbine runner or the impinging water jet. These simulations involved several millions particles (~2.5million particles for the single jet operation, ~3million particles for twin jet operation). The time needed for the simulations was about three weeks on a computer equipped with 16CPU cores.

In fig. 5.65 two indicative views of the Turgo turbine single jet operation, are shown. During the runner rotation, successive water sheets are formed, which tend to expand in an area enclosed by two lines: one line lies on the extension of the water jet and the other line is defined by the outlet angle of the turbine blade. Outflowing water does not seem to interact with the runner or the water jet, even for higher flow rates. Thus, operation at increased flow rates (off design operation) is possible, as long as the entire water jet cross section, at the runner inlet level, lies between the turbine hub and tip radii. Leakage flow is also visible from the turbine inlet.

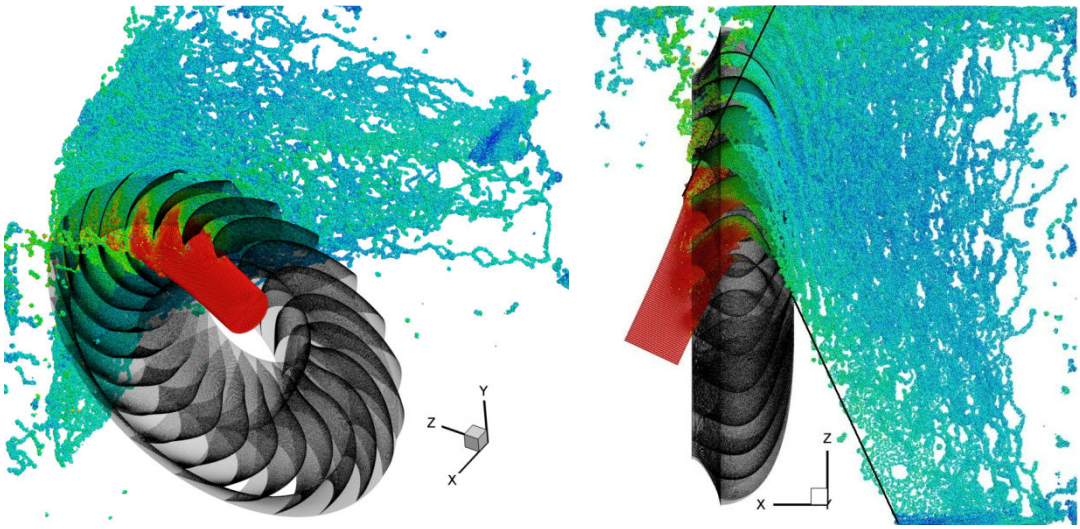


Fig. 5.65. Indicative views of upper nozzle operation, complete runner simulation (gravity is at -*y* direction).

Similar results were obtained from twin jet operation. Part of the flow is expected to follow the runner rotation (see fig. 5.66). This does not pose a problem during the upper nozzle operation, since the flow will move towards the exit of the turbine casing and will eventually leave. However, during the lower nozzle operation, water has enough kinetic energy to move upwards, contrary to gravity. Even in that case, the water eventually hits the turbine casing and spreads, without affecting the runner or jets.

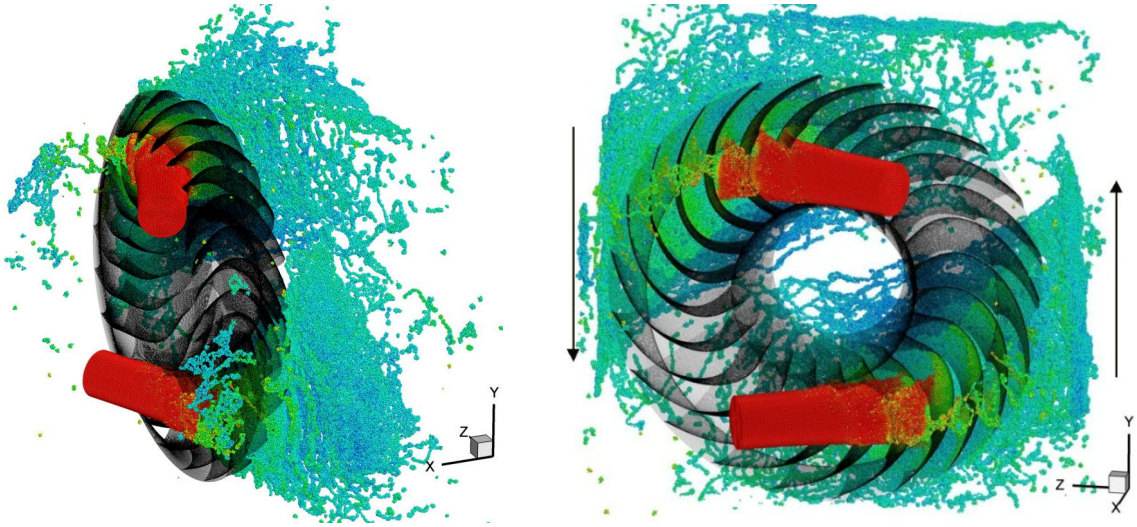


Fig. 5.66. Indicative views of both nozzle operation, complete runner simulation (gravity is at -*y* direction).

5.3.2. Application of the SPH method for simulation of Pelton turbine runners

The Pelton turbine bucket is symmetric and is formed by two half ellipsoids (fig. 5.67). A cut is made in the lip to facilitate all the water in the jet to usefully impinge on the buckets. This avoids the interference of the incoming bucket on the jet impinging on the previous bucket, during operation of the turbine [2]. The water jet impinges at the middle of the bucket, where the splitter is located. The splitter divides water flow into two equal parts. Each part moves through the respective cup shaped compartment and eventually exits at the opposite direction relative to the impingement velocity. By changing the direction of the flow, pressure develops on the bucket surface, which eventually results to the energy transfer from the fluid to the turbine runner.

In this chapter the results of the SPH algorithm used for simulations on Pelton turbine bucket are presented. As a first step, a 3D stationary Pelton bucket was impinged under various impingement angles by a water jet. The selected impingement angles correspond to various relative positions of the bucket and water jet, during the operation of a rotating Pelton turbine, as in [17]. Then, the application of the SPH method for rotating runner simulations is presented. Finally a complete Pelton runner was simulated.

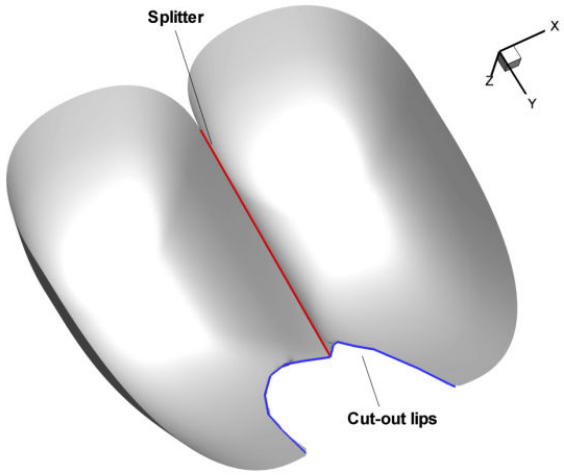


Fig. 5.67. Pelton bucket.

Stationary Pelton bucket impingement

The flow was simulated for different impingement angles (see fig. 5.68) relative to the *z*-axis, for a physical time of 0.02s. The water jet axis was placed on the *yz*-plane. The water jet had a diameter of 30mm and a velocity of 20m/s, thus the numerical speed of sound was set to 230m/s. In order to speed up the algorithm execution, symmetry boundary conditions were assumed at the symmetry plane, at the *yz*-plane. The particle size was set to 1mm since it was found adequate in similar studies in literature [19]. Results of the SPH were compared with results from Fluent.

Particles were positioned on the jet in a similar manner to the Turgo jet. The main difference here is that, in order to avoid particles moving on the splitter in an unphysical way, particles are not allowed to be on the symmetry plane. This was done by dividing any particles being on the symmetry plane and moving them by a small displacement away from the symmetry plane (see fig. 5.68).

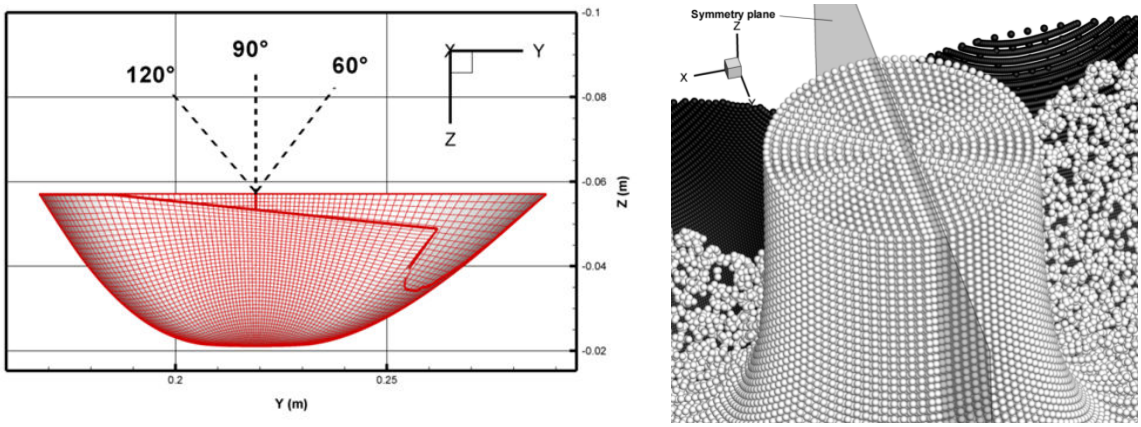


Fig. 5.68. Pelton jet impingement set up. Left: Pelton impingement angles. Right: particle positions on the jet. Note that there are no particles on the symmetry axis.

In fig. 5.69 a general view of the developing flow patterns on the Pelton bucket is shown; solution of the SPH method is compared to the Fluent solution. Generally, the forming water sheet is predicted to cover the same area on the bucket with both methods. Also, as experienced with the Turgo turbine impingement, flow velocity is under predicted with the SPH method in areas of few particles; this effect is located at the edges of the flow sheets and it is especially visible at the 90° and 60° impingements. The impingement angle plays an important role to the evolution of the flow. For the 60° impingement angle part of the water sheet exits from the back of the bucket, whereas for the 120° impingement water exits from the front. Note that the computational grid used in Fluent did not extend beyond the notch cut of the bucket (see at the end of the current section for the Fluent computational meshes, fig. 5.71) and thus the water exiting from the front is not captured. At the right part of the same figure, free surface is compared at an indicative slice at x=0. Free surface location is practically identical. Results are similar at the other slices too.

In fig. 5.70, the pressure distribution on the bucket surface is shown. Since pressure calculation through the Tait equation of state is unreliable and requires averaging, pressure was calculated indirectly through the boundary forces. Each boundary particle i was associated with a surface element of specific surface area S_i. Thus, after obtaining the reaction forces on each boundary particle, it was possible to derive the equivalent pressure distribution on the boundary through the following simple relation:

$$p_i = \frac{F_i}{S_i} \tag{5.10}$$

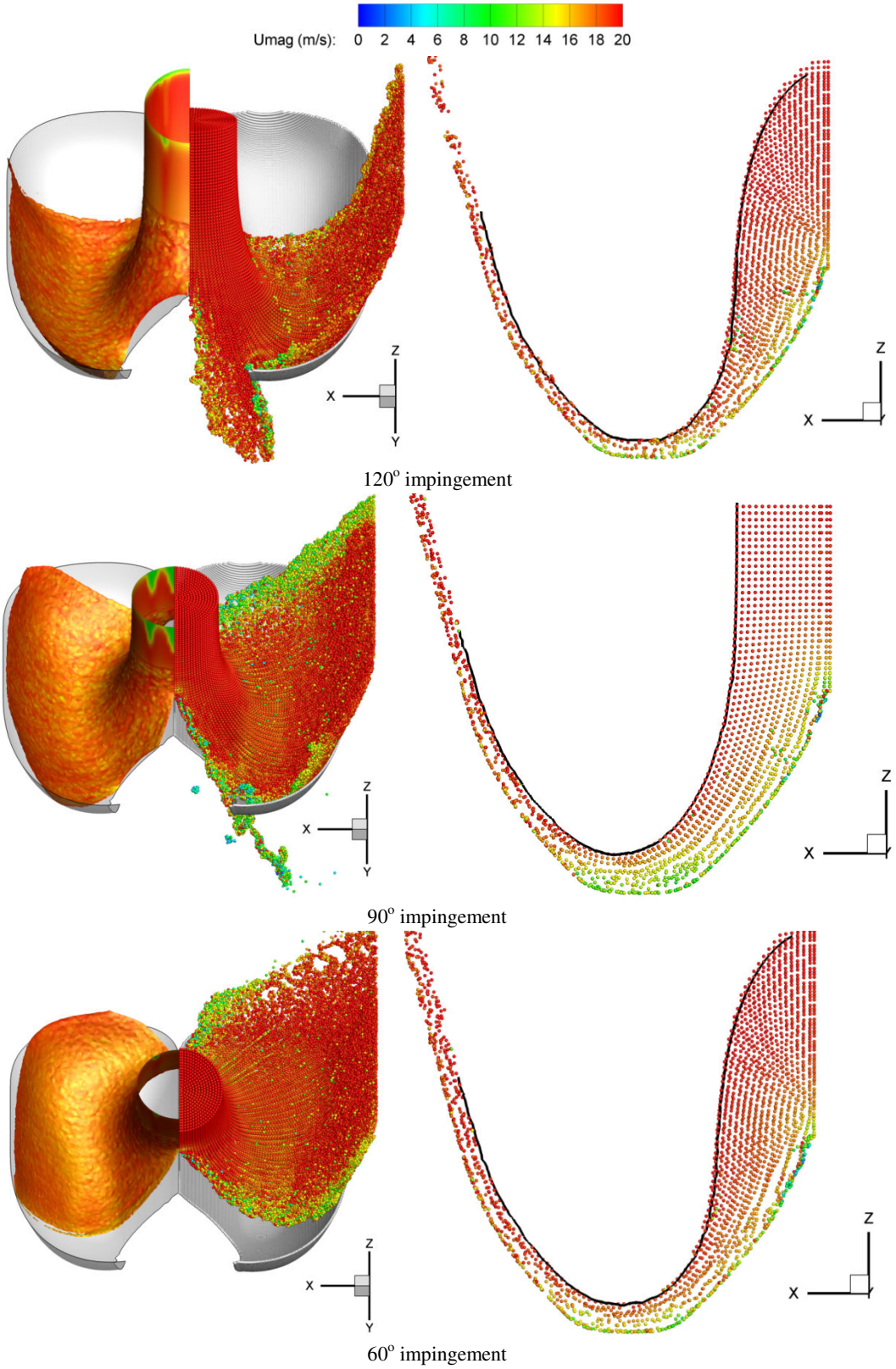
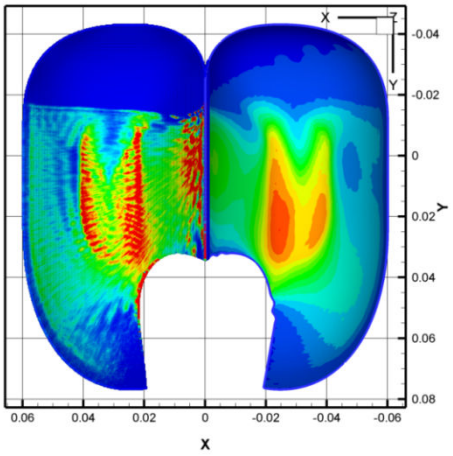
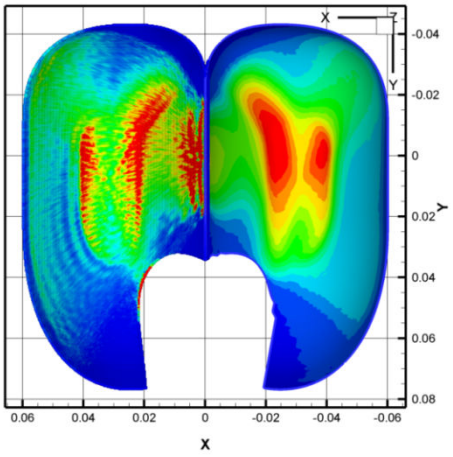


Fig. 5.69. Stationary Pelton impingement under various angles. Left: general view of the flow and comparison between the Fluent solution (isosurface) and SPH solution (particles). Right: comparison of the free surface level at slice Y=0. Particles represent the SPH solution and solid line the Fluent solution.

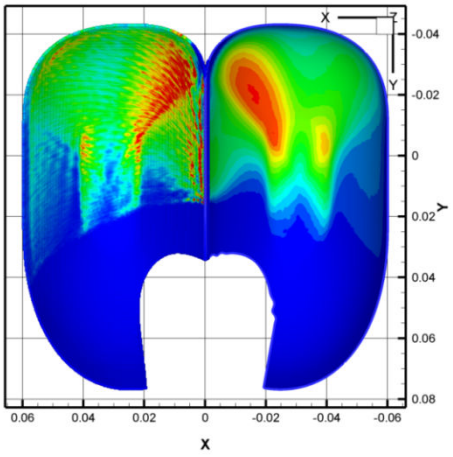
Both methods predict a similar pressure distribution. As it is expected, increased pressure occurs at regions of high curvature, since flow direction changes. However the SPH method over predicts pressure at the splitter area. This is not physical, but rather an arithmetic artifact, due to the sharp angle in the specific region. Similar behavior was experienced by other researchers too [18, 19].



120^{o} impingement



90^{o} impingement



60^{o} impingement

Fig. 5.70. Stationary Pelton impingement under various angles. Pressure distribution is shown, as calculated from the SPH solution (left part of the image) and the Fluent solution (right part of the image)

Finally, in table 5-VI the calculated forces on the bucket are shown. The SPH method underestimated the calculated forces. The cause of the underestimation is due to the influence of the density filter, which smoothes the density field, introducing numerical diffusion. The same effect occurs at the simulation of the stationary Turgo turbine blade, but it is significantly less pronounced, since the density filter is applied less times, throughout the simulation. Increasing the particle resolution does not seem to improve the quality of the results.

Table 5-VI. Calculated forces for the case of the stationary Pelton bucket

Impingement angle	**Method**	F_x [N]	F_y [N]	F_z [N]
60°	Fluent	-38.2	-102.9	-253.9
	SPH sym.	-35.1	-99.4	-236.8
90°	Fluent	-21.9	-32.1	-278.8
	SPH sym.	-17.7	-30.5	-260.6
120°	Fluent	-21.43	22.3	-219.5
	SPH sym.	-15.5	23.3	-207.5

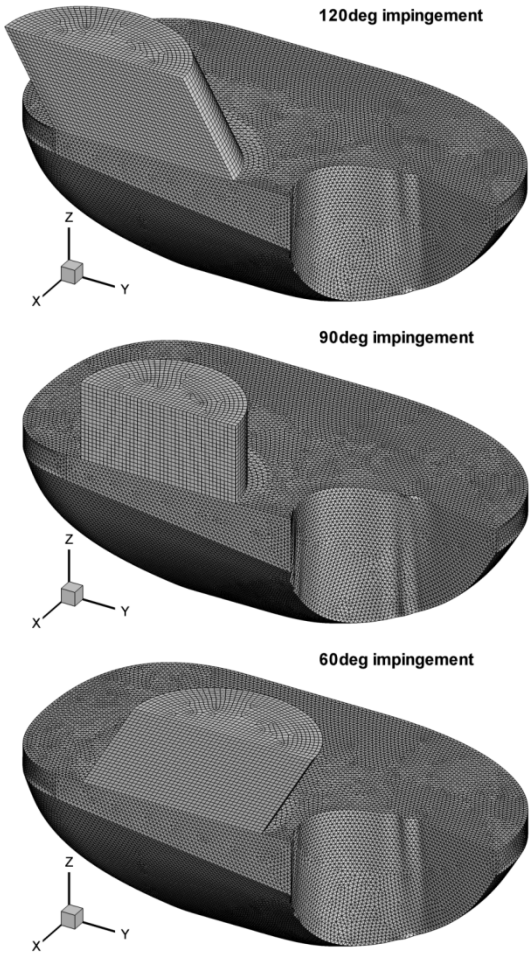


Fig. 5.71. Fluent computational meshes

Rotating Pelton turbine runner

The next step is the simulation of a rotating Pelton geometry. The bucket geometry is the same with the one used in the stationary bucket impingement. The hub radius of the runner is 177mm, the tip radius 293mm and the nominal radius 224mm. The turbine buckets are positioned with an inclination of 12° towards the direction of rotation of the runner (fig. 5.72). The runner was initially simulated for an angular velocity of 85.1rad/s, while the water jet impinging the turbine runner had a diameter of 36mm and velocity of 42m/s. The runner was simulated for a physical time of 20ms, performing a 97° arc.

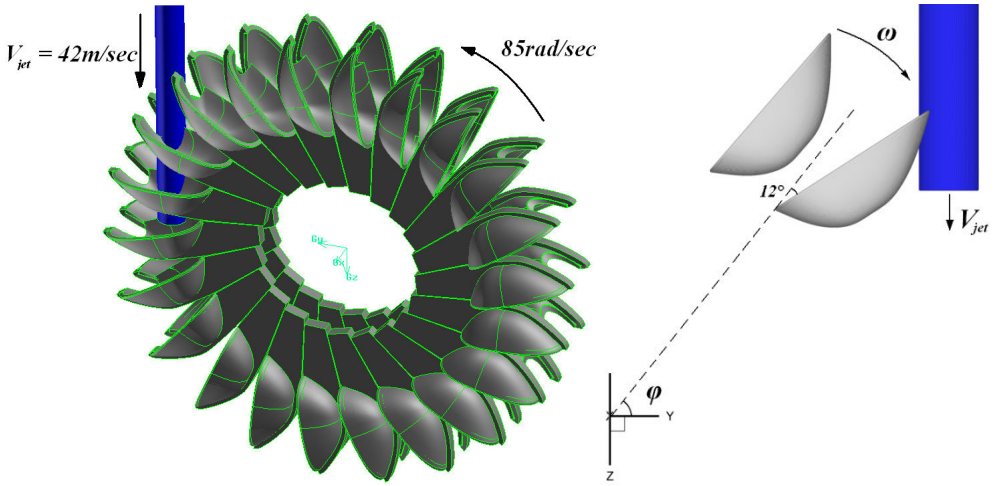


Fig. 5.72. Left: Rotating Pelton runner simulation. Right: Simulation set-up.

The particle size used was 1mm and the numerical speed of sound 500m/s. For the following simulation several simplifications were made in order to minimise the computational cost of the calculations and the computational resources needed:

- Symmetry boundary conditions are applied at the plane of symmetry of the Pelton runner. This is a reasonable simplification commonly done in the literature [20, 21] for the simulation of a Pelton runner, which essentially cuts down the computational cost by a factor of two (approximately), without sacrificing the accuracy of the solution.
- Only the inner surface of the bucket is considered for the simulation, meaning that only the inner surface of the bucket is used for creating the boundary particles layer. This layer has a uniform thickness of $2dx$ due to the boundary conditions specified above. Using the whole bucket surface to generate boundary particles for the whole bucket is possible, but would greatly increase the number of boundary particles used and, hence, the computational cost.
- Only the space between two successive buckets is simulated, assuming periodic flow for the rest of the buckets. Obviously this is an approximation for the performance of the Pelton turbine, since each bucket interacts with the water sheets leaving from the previous bucket, apart from the incoming water jet. Indeed, at least three consecutive buckets are needed to be simulated, while examining the flow patterns at the intermediate bucket, in order to determine the influence of the water sheet interaction with the buckets. In order to assess the influence of such interactions, a simulation involving three consecutive buckets was performed (fig. 5.73), which proved that, for the specific bucket geometry, water sheet interaction with the buckets is not very important and affects the turbine efficiency by a small fraction only

(~0.25%, fig. 5.73). Thus the influence of this interaction is omitted from the next calculations.

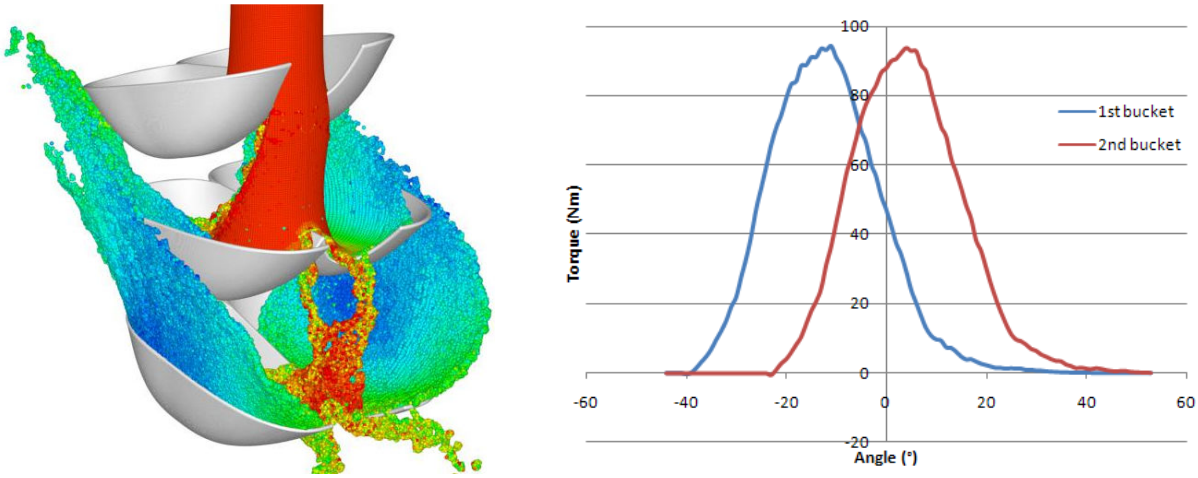
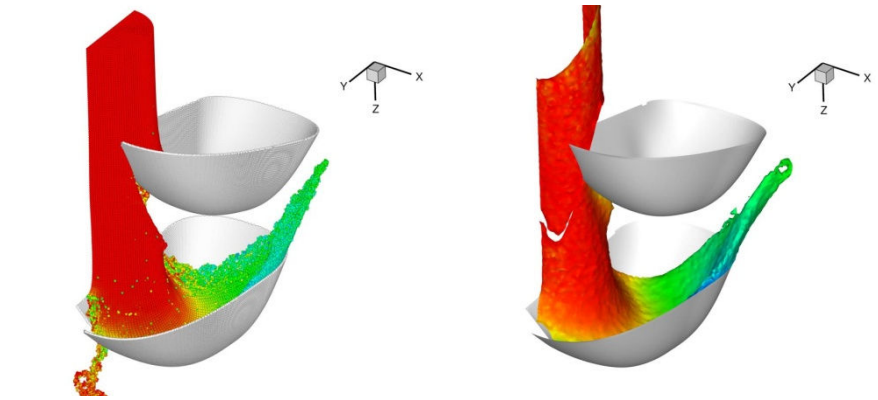


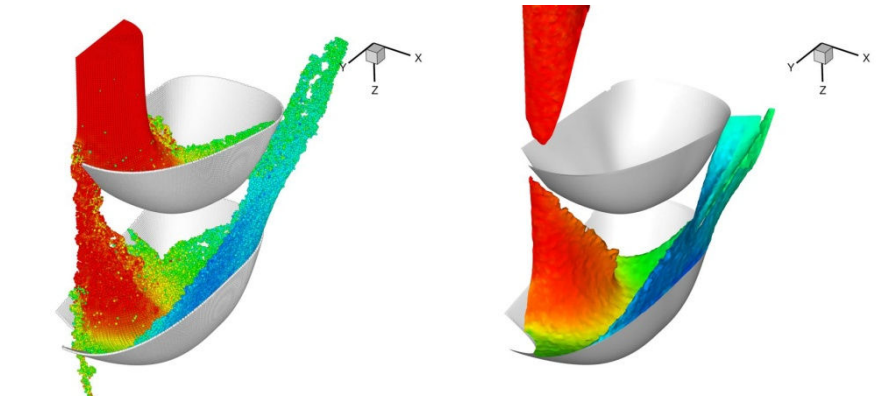
Fig. 5.73. Left: Bucket - water sheet interaction, simulated using three consecutive buckets with SPH. Right: comparison of the generated torque on the first and second bucket.

Fig. 5.74 shows indicative views of the numerical results from both methods, the SPH and Fluent. Both methods predict a similar flow field, there are some differences though, due to the set up of the simulations. The Fluent mesh did not extend beyond the second bucket, thus flow is not captured in that area. Also the computational mesh did not extend beyond the cut-out lip (see fig. 5.78). On the other hand the SPH method is able to capture these flow features, due to its inherent adaptivity, since particles move following the flow patterns. At the beginning of the interaction water leaves from the root of the bucket, towards the center of the runner (fig. 5.74b). As the runner rotates, water leaves from the side edges of the bucket. Finally at the end of the interaction, the remaining water on the bucket leaves from the front. As can be seen in the pictures, the escaping, from the rotating bucket, water sheets have little residual velocity, indicating an effective energy exchange with the runner. Moreover, the larger velocity component of the escaping water is in the x-direction, which is perpendicular to the runner's rotation plane. This residual velocity is due to the bucket design in order to avoid interference of the outflow with the next bucket. The SPH method is able to predict fine flow features, such as the interaction of the outflowing water sheet with the next bucket (see fig. 5.74b, c).

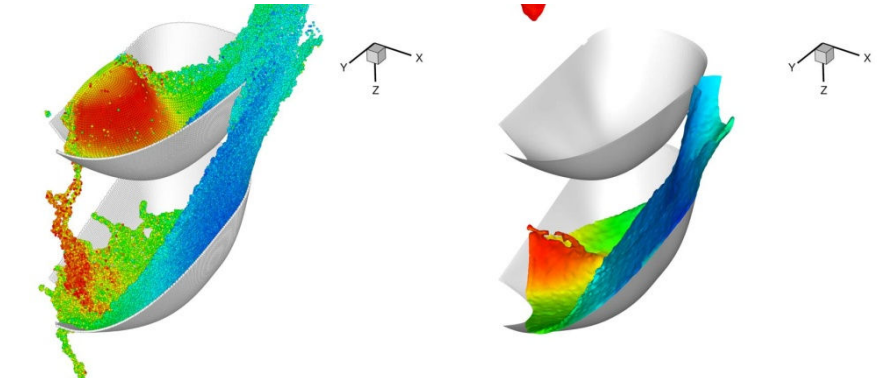
Fig. 5.75 shows the propagation of the water sheet on the bucket surface. During the beginning of the water jet – bucket interaction the water jet hits the cut-out lips of the bucket (fig. 5.75a). As the interaction continues, a larger part of the jet impinges the bucket and the free surface flow is directed towards the root of the bucket (5.75b). In fig. 5.75c the whole water jet hits the splitter, while water starts exiting from the rear sides of the bucket. The exiting water sheets from this area also interact slightly with the next bucket. In fig. 5.75d the water jet is interacting with the next bucket, thus a part of the impinging jet is cut. Water sheets start flowing out of the sides of the bucket and almost the whole bucket is covered with water. At the final stages of interaction (fig. 5.75e and 5.75f), the water jet is cut completely and the water exits from the cut-out region.



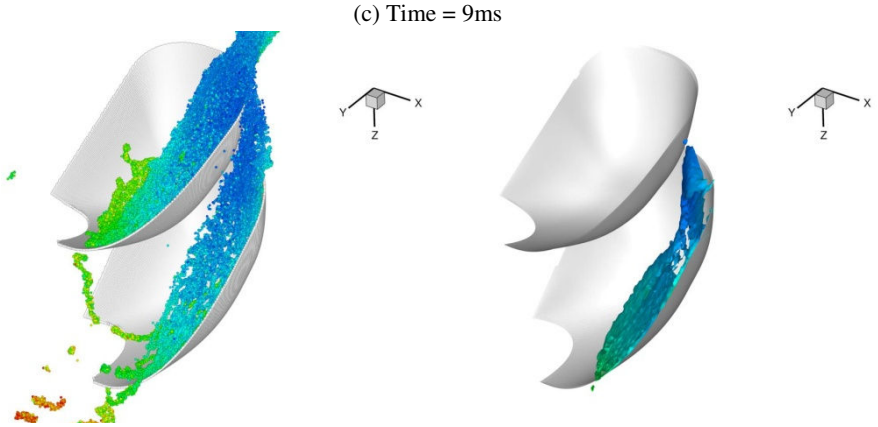
(a) Time = 6ms



(b) Time = 8ms



(c) Time = 9ms



(d) Time = 14ms

Fig. 5.74. Pelton runner simulation. Left:SPH method. Right: Fluent. Colouring according to velocity magnitude.

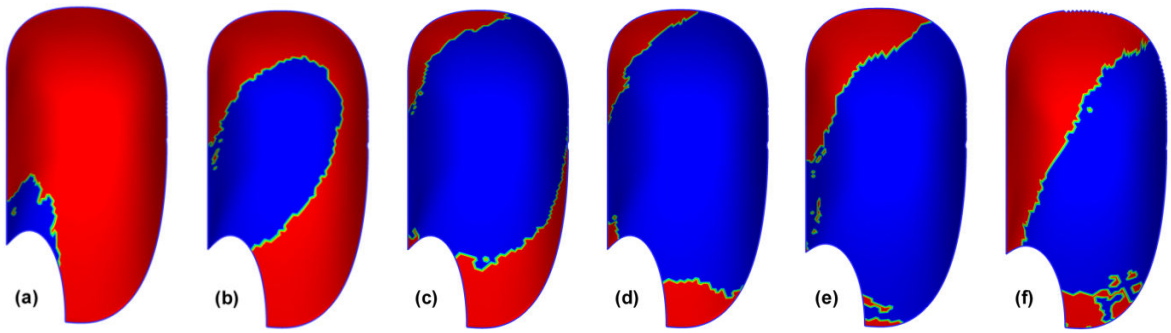


Fig. 5.75. Evolution of the flow on the Pelton bucket, interval 2ms. The position of the water on the bucket is marked with blue contour levels, whereas the rest of the bucket is shown using red contour levels.

Fig. 5.76 shows a plot of the developing torque and axial force on the half-bucket in respect to its angular position. As with the Turgo turbine, forces and consequently torque, exhibit oscillations and, thus, force/torque averaging is performed in order to obtain a smooth curve. Note that, due to symmetry, there is no axial force on the complete runner. The SPH method is able to predict accurately the onset of the developing torque and force, captures the peak value but slightly underestimates forces during the bucket emptying. Increased particle resolution (particle size 0.75mm) does not improve the torque curve. However it must be highlighted that the SPH algorithm required ~20hrs, whereas Fluent ~5days on the systems already mentioned (SPH on 2xQuad core Xeon 2Ghz, Fluent on i7 940 2.93GHz computer, using 4 CPUs: 160 and 480CPUhrs respectively).

Also in fig. 5.77 the distribution of the developing torque on the bucket surface is shown, throughout the entire jet-bucket interaction period. As it is expected, force and consequently torque, develops in areas where there is change in the direction of the flow.

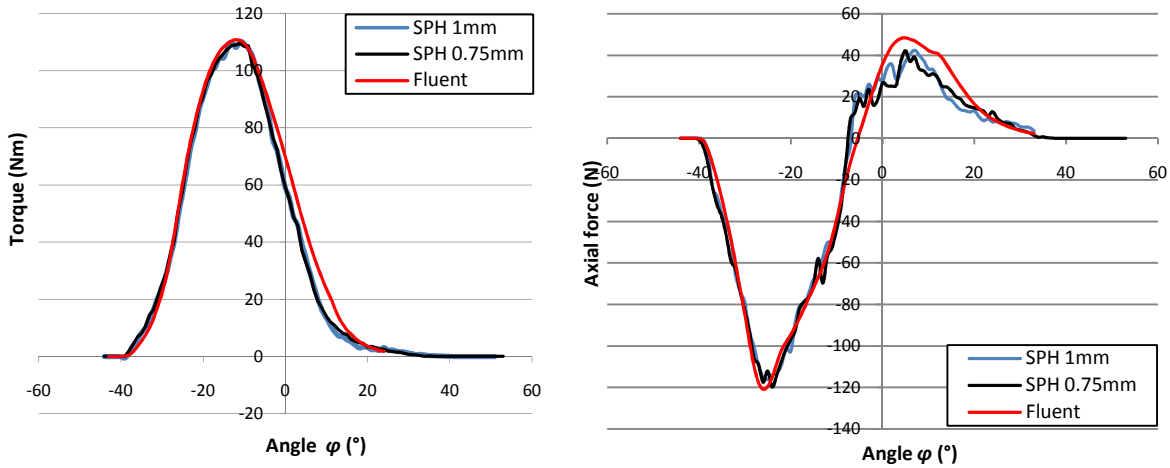


Fig. 5.76. Left: Torque generated on the half-bucket. Right: Axial force generated on the half bucket.

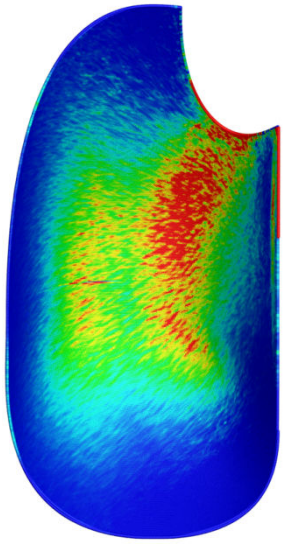


Fig. 5.77. Torque distribution on the bucket surface.

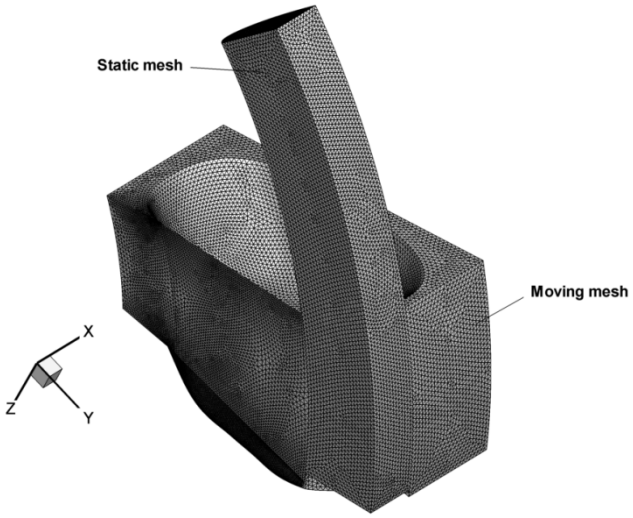


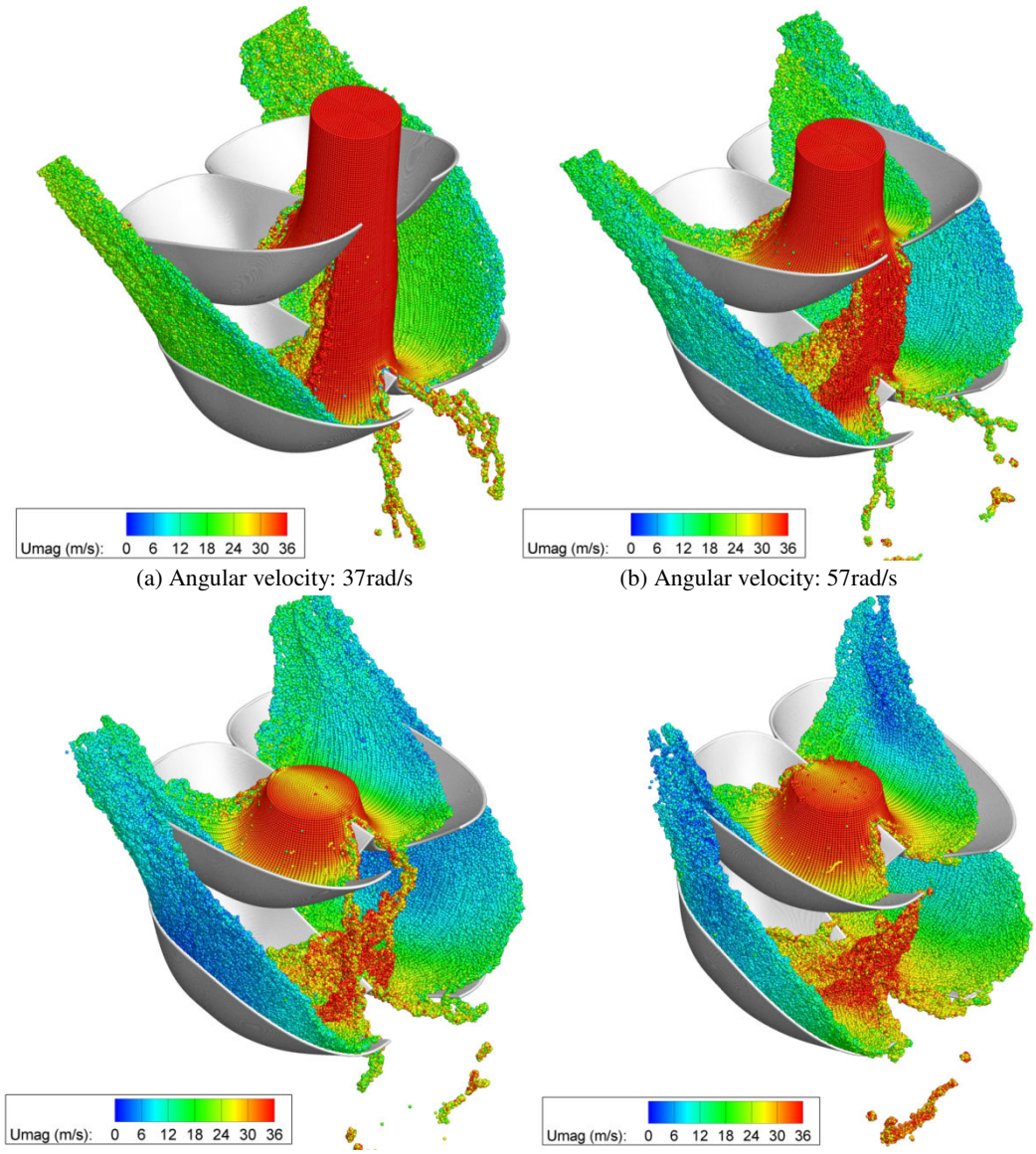
Fig. 5.78. Computational mesh used by Fluent. Note the refinement in areas with sharp angles, such as the splitter.

Varying angular velocity

Several simulations were performed for various nozzle openings and turbine water gauge height. The water jet velocity was estimated from the relation $V_{jet} = C_{los}\sqrt{2gH}$ (uniformly set along the jet cross section) and the jet diameter $D_{jet} = \sqrt{\frac{4Q}{\pi V_{jet}}}$. The factor C_{los} represents the losses at the Pelton turbine nozzle and was set to 0.97.

In figure 5.79 indicative instances of the flow are shown for a nozzle opening of 20mm for various angular velocities during the bucket - water jet interaction. From the results in fig. 5.79c, it is shown that the water sheets leaving the bucket at optimal rotational speed have a small residual absolute velocity, in the stationary frame, in comparison to the incoming water jet. This is expected due to the energy transfer from the water jet to the runner, leaving the out-flowing water with little

kinetic energy. On the other hand figure 5.79a, b, d shows the absolute velocity in the stationary frame of the out-flowing water sheets for several non - optimal angular velocities, two less than the optimal (fig. 5.79a, b) and one higher than optimal (fig. 5.79d). In these cases the water sheets have a considerable residual velocity, which eventually leads to the reduction of the turbine efficiency. Another notable observation is that the low velocity region of the outflowing water sheet changes location in respect to the angular velocity. This is expected, since energy transfer is maximized at the radius where the peripheral velocity is approximately half the jet velocity. Thus, the low velocity region is at a greater radius than the nominal for low angular velocities and at a smaller radius for higher angular velocities. At the nominal operating point the low velocity region is located at the nominal radius and the efficiency of the turbine is maximized. The instances of these figures are not taken at the same angular position of the bucket, because the bucket - water jet interaction ends at different positions due to the different rotational speed.



(a) Angular velocity: 37rad/s (b) Angular velocity: 57rad/s

(c) Angular velocity: 73rad/s (optimal operating point) (d) Angular velocity: 94rad/s

Fig. 5.79. Pelton turbine simulation for various angular velocities. Note that the water sheet low speed region position changes.

The torque graphs in fig. 5.80, show the effect of the rotational speed to the developing torque on the runner: for low rotational speed the torque curve is narrow with a high peak value, whereas for a high rotational speed the torque curve has longer duration, but with a lower peak value. The maximum efficiency is achieved when the peripheral velocity of the bucket is approximately half the water jet velocity, i.e. for the 73.1rad/sec. The same trend is observed for all different nozzle openings tested.

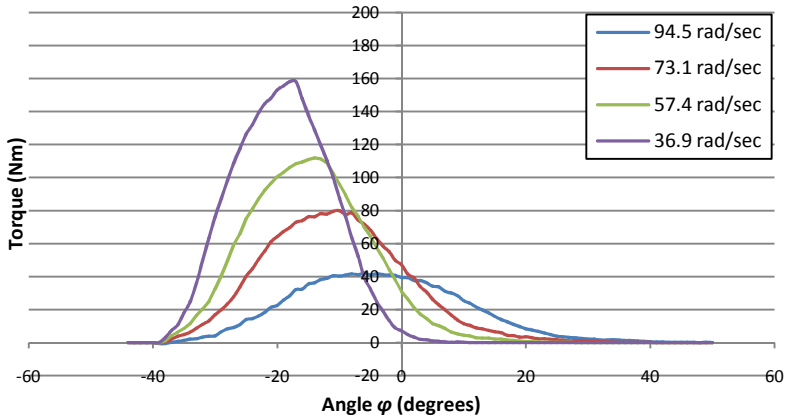


Fig. 5.80. Comparative torque graph for different angular velocities. Results are for the half-bucket.

Complete Pelton runner simulation

Finally the simulation of the complete Pelton turbine runner was performed, simulating all 22 buckets for single jet operation. Fig. 5.81 shows the water jet – runner interaction and the consecutive water sheets formed. The out flowing water sheets do not interact with the water jet, at least in the absence of the casing.

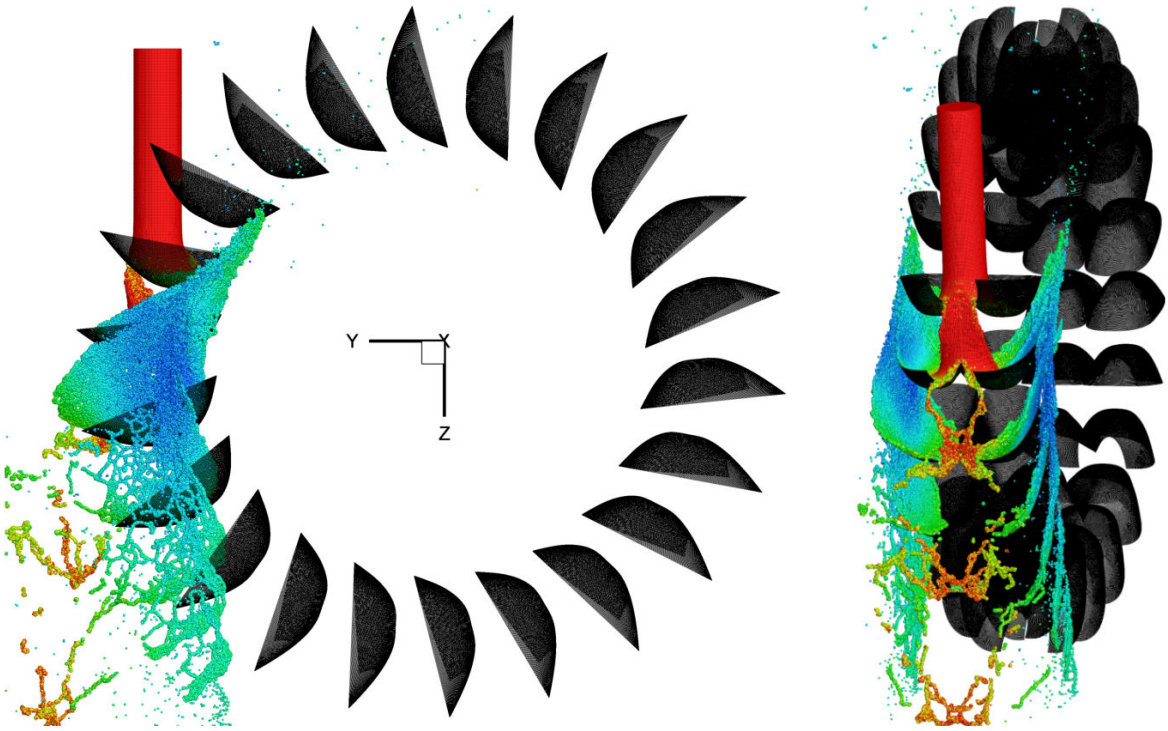


Fig. 5.81. Simulation of the complete Pelton runner.

Figure 5.82 shows the exerted torque on the runner. As it is expected, after the initial transient phase, a periodic behavior is achieved. Eventually torque resembles closely a periodic function of the form:

$$T(\varphi) = \overline{T} + A\sin(22\varphi + \varphi_0)$$

where:

- 22 is the number of turbine buckets
- $\overline{T} \approx 139\text{Nm}$
- φ is the runner rotation angle
- $\varphi_0 \approx 1.3\pi$ rad, is an arbitrary angle added to match the periodic sinus function with the torque curve.

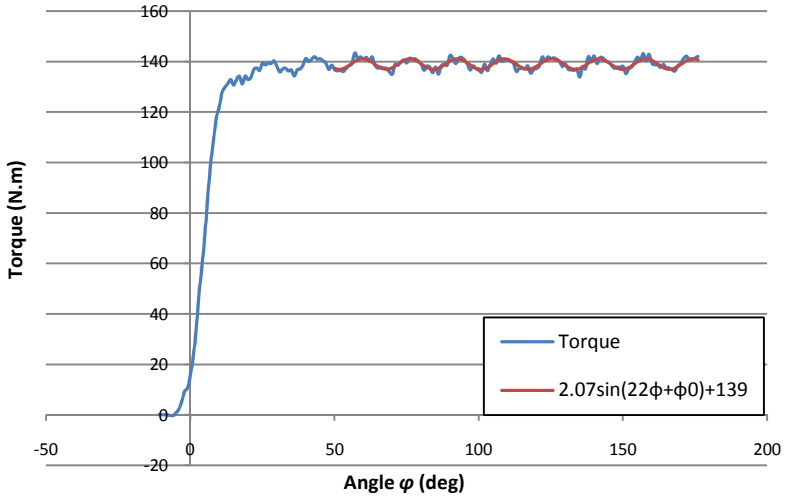


Fig. 5.82. Torque graph of the complete runner in respect to the rotation angle.

5.4. Concluding remarks

Considering all the previous results regarding the SPH simulations of free surface flows and flows in impulse turbines parts, it can be concluded that the SPH method is an attractive alternative of traditional mesh based CFD methods. It is able to capture properly the free surface effects, while it is considerably faster than the mesh based program. Subtle flow features are captured, such as the water sheet interaction with the next bucket of the Pelton runner, or the recirculation bubble in the 60° stationary Turgo jet impingement. Simulations of complete turbine runners are possible using the SPH method, whereas mesh based methods would require a very fine mesh even at long distances from the runner in order to properly resolve the thin water sheets. The adaptivity of the SPH method enables proper resolution of such water sheets, provided enough particles are used, without needing enormous computational resources.

On the other hand the major drawback of the SPH is its inaccuracy. Indeed the SPH method underestimates the efficiency of the impulse turbines simulated in comparison to experimental data by ~5%. The main reason of this discrepancy is primarily attributed to the weakness of the boundary particles to enforce a truly zero gradient of pressure on the wall. Moreover, density/pressure oscillations are responsible for the oscillatory behavior of the forces on walls, even in steady state

cases. Density filtering affects the numerical diffusion of the SPH algorithm, by smoothing the density field and is another possible cause of the forces underestimation. Increasing particle resolution has been examined, but it did not result to a significantly different result, meaning that particle independence has been achieved.

In the next chapters, new advanced techniques to overcome the density/pressure oscillations and the weakness of the boundary conditions will be presented and discussed.

References

[1] European small hydropower association (ESHA), "Guide on how to develop a small hydropower plant", 2004.

[2] C. P. Kothandaraman, R. Rudramoorthy, "Fluid Mechanics and Machinery", 2nd edition, New Age International Publishers, ISBN (13): 978-81-224-2558-1.

[3] J.S. Anagnostopoulos, D.E. Papantonis, "Flow modeling and runner design optimization in Turgo water turbines", International Journal of Applied Science, Engineering and Technology, vol. 4, no. 3, 2007, ISSN: 1307-4318.

[4] J. C. Marongiu, F. Leboeuf, E. Parkinson, "Numerical simulation of the flow in a Pelton turbine using the meshless method smoothed particle hydrodynamics: a new simple solid boundary treatment", Proceedings of IMechE vol.221, Part A: Journal of Power and Energy, DOI: 10.1243/09576509JPE465.

[5] J.C. Marongiu, F. Leboeuf, J. Caro, E. Parkinson, "Free-surface flowns using an hybrid SPH-ALE method", Journal of Hydraulic Research, vol. 48, extra issue 2010, p. 40-49, DOI: 10.3826/jhr.2010.0002.

[6] J. Anagnostopoulos, P. Koukouvinis, F. Stamatelos, D. Papantonis, "Optimal design and experimental validation of a Turgo model Hydro turbine", Proceedings of the 11th Conference on Engineering Systems Design and Analysis (ESDA 2012), Nantes, France, 2012.

[7] L. Piegl, W. Tiller, "The NURBS Book – Monographs in visual communication", Springer, 2nd edition, ISBN-10: 3540615458, ISBN-13: 978-3540615453.

[8] J. Anagnostopoulos, D. Papantonis, "Experimental and numerical studies of runner design of Pelton turbines" Hidroenergia 2006, Crieff, Scotland UK, 7-9 June 2006.

[9] M. Gómez-Gesteira, B.D. Rogers, R.A. Dalrymple, A.J. Crespo, and M. Narayanaswamy, 2009, User Guide for the SPHysics Code v2.0, http://wiki.manchester.ac.uk/sphysics.

[10] Y. Jang, J.C. Marongiu, E. Parkinson, "Analysis of SPH and mesh based simulations using point based post processing tool", Proceedings of the IIIrd Spheric Workshop, Lausanne, Switzerland 2008.

[11] P. Koukouvinis, J. Anagnostopoulos, D. Papantonis, "Development and application of the SPH method for the simulation of impulse turbines", Proceedings of Flow 2010 conference, Thessaloniki, 12-13 November 2010, Greece.

[12] P. Koukouvinis, J. Anagnostopoulos, D. Papantonis, "SPH method used for flow predictions at a Turgo impulse turbine: comparison with Fluent", International Conference on Fluid Mechanics 2011, Paris, 27-29 July 2011, France.

[13] J. C. Marongiu, E. Parkinson, S. Lais, F. Leboeuf, J. Leduc, "Application of SPH-ALE method to Pelton hydraulic turbines", 5th Spheric Workshop, Manchester, 23-25, June 2010, United Kingdom.

[14] Hydroaction website: http://www.hydroaction.org/index.php@id=45.html.

[15] The Evolutionary Algorithm System, User's manual, http://velos0.ltt.mech.ntua.gr/EASY.

[16] K. Giannakoglou, "Optimization methods for aerodynamics", 4rd edition, NTUA publishing, Athens 2006.

[17] B. Zoppe, C. Pellone, T. Maitre, P. Leroy, "Flow analysis inside a Pelton turbine bucket", ASME Journal of Turbomachinery, vol. 128, pages 500-512, 2006.

[18] J.C. Marongiu, F. Leboeuf, E. Parkinson, "Riemann solvers and efficient boundary treatments: an hybrid SPH-finite volume numerical method", 3rd Spheric Workshop, Lausanne, 4-6 June 2008, Switzerland.

[19] J.C. Marongiu, F. Leboeuf, J. Caro, E. Parkinson, "Low mach number numerical schemes for the SPH-ALE method. Application in free surface flow in Pelton turbines", 4th Spheric Workshop, Nantes, 27-29 May 2009, France.

[20] A. Santolin, G. Cavazzini, G. Ardizzon, G. Pavesi (2009) "Numerical investigation of the interaction between jet and bucket in a Pelton turbine", Proceedings of the IMechE, Vol.223 Part A : Journal of Power and Energy.

[21] R. Mack, W. Rohne, S. Riemann, W. Knapp, R. Schilling, (2006), "Using the potential of CFD for Pelton turbine development", Proceedings of the 23rd IAHR Symposium, Yokohama, October 2006.

Chapter 6

Incorporating Riemann solvers for particle interactions in SPH (SPH-R)

In the previous chapters the implementation of the standard SPH algorithm was discussed. The strong points of the SPH method have been highlighted, whereas its weaknesses have been also analyzed. One major issue of the SPH method is the quality of the pressure / density field. Various techniques have been employed to smooth the density field, such as using density filtering, upwind flux formulation or introduction of diffusive terms. An attractive alternative which is used more and more in modern SPH algorithms, is the implementation of Riemann solvers for solving the inter-particle interactions using the Godunov, or a higher order scheme. The main advantage of the implementation of Riemann solvers is that the density / pressure fields become smooth without adding arbitrary diffusion terms or correction schemes such as the density filtering.

In the present chapter implementation of Riemann solvers in conjunction with the standard SPH model will be discussed. This form of SPH was first formulated by Parshikov et al. [1] and has been used by other researchers too [2]. Parshikov reformulated the standard SPH equations, in the absence of viscous effects, in a way which enabled treating the two interacting particles as the two states of a one dimensional Riemann solver. Non-conservative formulation is employed, as with the standard SPH. In this chapter, the implementation of such an algorithm is discussed along its extension using high order schemes. From now on the SPH method using Riemann solvers for the inter-particle interactions will be referred to as *SPH-R*.

6.1. Derivation of the SPH-R equations

Parshikov noted that, beginning from the standard SPH continuity and momentum equations (eq. 3.21 and 3.39 without the viscous effects):

$$\frac{D\rho_i}{Dt} = \sum_j m_j \mathbf{u}_{ij} \cdot \nabla_i W_{ij} \tag{6.1}$$

$$\frac{D\mathbf{u}_i}{Dt} = -\sum_j m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla W_{ij} + \mathbf{f}_{body} \tag{6.2}$$

it is possible to rewrite them after projecting particle i and j velocities on the unity vector pointing from particle i to particle j (see fig. 6.1), as follows:

$$\frac{D\rho_i}{Dt} = \sum_j m_j \mathbf{u}_{ij} \cdot \nabla_i W_{ij} = \sum_j m_j \left(\mathbf{u}_i - \mathbf{u}_j \right) \cdot \frac{\mathbf{r}_{ij}}{\left\| \mathbf{r}_{ij} \right\|} \frac{dW}{dr} \tag{6.3}$$

Considering that:

$$U_i^{proj} = -\mathbf{u}_i \cdot \frac{\mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|} \tag{6.4}$$

$$U_j^{proj} = -\mathbf{u}_j \cdot \frac{\mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|} \tag{6.5}$$

The continuity equation becomes:

$$\frac{D\rho_i}{Dt} = -\sum_j m_j \left(U_i^{proj} - U_j^{proj}\right) \frac{dW}{dr} \tag{6.6}$$

Parshikov replaced U_i^{proj} , U_j^{proj} and p_i, p_j using the solution of the one dimensional Riemann problem assuming that particles *i* and *j* are the left (L) and right (R) states and a discontinuity lies between them. This method is known as the Godunov method in the Finite Volume methodology.

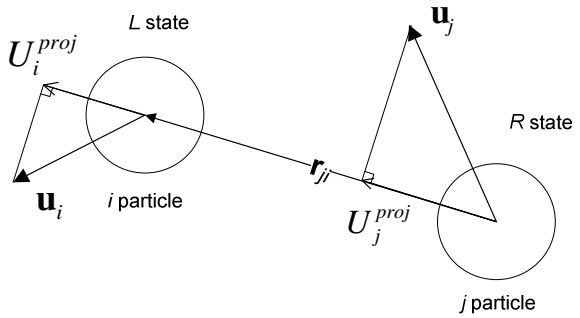


Fig. 6.1. Particle interactions between *i* and *j*.

The following substitutions are made:

$$\frac{1}{2}\left(U_i^{proj} + U_j^{proj}\right) \rightarrow U_{ij}^*$$

$$\frac{1}{2}\left(p_i + p_j\right) \rightarrow p_{ij}^*$$

and the continuity and momentum equations become:

$$\frac{D\rho_i}{Dt} = -\sum_j m_j 2\left(U_i^{proj} - U_{ij}^*\right) \frac{dW}{dr} \tag{6.7}$$

$$\frac{D\mathbf{u}_i}{Dt} = -\sum_j m_j \left(\frac{2p_{ij}^*}{\rho_i \rho_j}\right) \nabla W_{ij} + \mathbf{f}_{body} \text{, or} \tag{6.8}$$

$$\frac{D\mathbf{u}_i}{Dt} = -\sum_j m_j \left[\frac{p_i}{\rho_i^2} + \frac{\left(2p_{ij}^* - p_i\right)}{\rho_j^2}\right] \nabla W_{ij} + \mathbf{f}_{body} \tag{6.9}$$

Note that in order to solve correctly the Riemann problem, one has to identify the wave structure arising from the interaction of the two states (see Appendix A for more information). However in our cases, only nearly incompressible flows will be covered, thus the Riemann problem solution is expected to be the star region.

For the solution of the Riemann problem, it is possible to use an exact Riemann solver, but has a significant computational cost, due to the non-linear nature of the underlying equation that has to be

solved. Approximate Riemann solvers are used instead; indeed, Parshikov proposed the use of the following approximate Riemann solver based on the acoustic approximation:

$$U_{ij}^{*} = \frac{c_L \rho_L u_L + c_R \rho_R u_R + p_L - p_R}{c_L \rho_L + c_R \rho_R} \tag{6.10}$$

$$p_{ij}^{*} = \frac{c_R \rho_R p_L + c_L \rho_L p_R + c_R \rho_R c_L \rho_L (u_L - u_R)}{c_L \rho_L + c_R \rho_R} \tag{6.11}$$

Other possible approximate Riemann solvers are the PVRS:

$$U_{ij}^{*} = \frac{1}{2}(u_L - u_R) - \frac{\overline{c}}{2\overline{\rho}}(\rho_R - \rho_L) \tag{6.12}$$

$$\rho_{ij}^{*} = \frac{1}{2}(\rho_L - \rho_R) - \frac{\overline{\rho}}{2\overline{c}}(u_R - u_L) \tag{6.13}$$

or the entropic variable solver:

$$U_{ij}^{*} = \frac{1}{2}(u_L - u_R) - \frac{1}{2\overline{\rho c}}(p_R - p_L) \tag{6.14}$$

$$p_{ij}^{*} = \frac{1}{2}(p_L - p_R) - \frac{1}{2}(u_R - u_L)\overline{\rho c} \tag{6.15}$$

In all previous relations, c is the numerical speed of sound for the specific conditions. It is calculated by the following equation:

$$c_K = c_0 \sqrt{\left(\frac{\rho_K}{\rho_0}\right)^{\gamma - 1}} \tag{6.16}$$

where c_0 is the speed of sound at the reference density ρ_0. Also $u_L = U_i^{proj}$ and $u_R = U_j^{proj}$, $\overline{\rho}$ and $\overline{c}$ are the average density and speed of sound between states L and R.

After obtaining ρ^* from the PVRS solver, it is possible to obtain p^* through the equation of state. For more information regarding approximate Riemann solvers, the reader is referred to Appendix A or the more detailed books of Toro [3], Leveque [4], Artzi [5] and the work of Irvings et al. on compressible fluids [6]. All the described solvers give practically identical results, when used for simulations.

The advantage of the described modification of SPH is that it requires little changes in the algorithm, in order to incorporate the solution of the Riemann problem and modify the equations solved. Boundary conditions may be described in the same way as in the standard SPH, however Riemann solvers enable a better implementation of boundaries which will be further discussed later on. Moreover, Godunov method inserts enough numerical diffusion, to stabilize the solution and density/pressure fields. However, since the Godunov method is only 1st order accurate, numerical diffusion in practical situations is rather prohibitive to obtain any meaningful results, i.e. sharp gradients are excessively smeared. This effect occurs in the Finite Volumes method too, however it becomes more pronounced in the SPH method, since the interaction distance between particles is larger for the same particle/volume discretization (the reader is reminded that particles interact at a radius of $2.5h$, with h being generally larger than the particle discretization dx). Thus a high order extension of the method is required in order to limit numerical viscosity to acceptable levels.

6.2. MUSCL 2nd order scheme / Limiter

In the present work a MUSCL type 2nd order scheme has been especially developed for the SPH method. The key point of the MUSCL scheme is that the piecewise approximation of the left and right states is replaced by the extrapolated data on the interface between the two interacting particles. This is done by using spatial derivatives of the respective field variables (see fig. 6.2, $\mathbf{\Phi}$ represent the vector of field variables used for the solution of the Riemann problem, such as ρ, u, v, etc.).

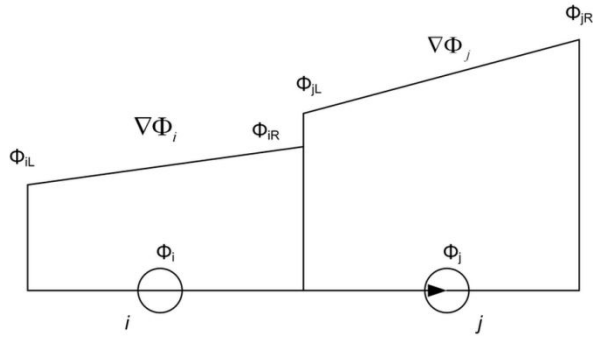


Fig. 6.2. MUSCL scheme. Instead of using field variables $\mathbf{\Phi}_i$, $\mathbf{\Phi}_j$ the extrapolated values $\mathbf{\Phi}_{iR}$, $\mathbf{\Phi}_{jL}$ are used.

A simple way of performing the extrapolations is to directly use the spatial variables, as follows:

$$\mathbf{\Phi}_{iR} = \mathbf{\Phi}_i + \tfrac{1}{2}\nabla\mathbf{\Phi}_i \cdot \mathbf{r}_{ij} \tag{6.17}$$

$$\mathbf{\Phi}_{jL} = \mathbf{\Phi}_j - \tfrac{1}{2}\nabla\mathbf{\Phi}_j \cdot \mathbf{r}_{ij} \tag{6.18}$$

Unfortunately, use of these relations, in order to calculate the extrapolated values, will result in spurious oscillations near discontinuities, since the resulting method is not Total Variation Diminishing (TVD) and, thus, new local extrema are formed; the resulting scheme is unstable, producing numerical dispersion [4], due to Godunov's theorem. In order to remedy this, a limiter function has to be used, which will limit gradients to physical values.

In this work a different procedure for the MUSCL scheme is adopted than the one used in the recent SPH literature [2, 7, 8], following instead the proven scheme used in the Finite Volume method. In the latter method, forward and backward derivatives are used for the limiting procedure. The principle is to compare these two derivatives for each particle and decide, depending on their values, whether or not the high order scheme should be applied. On the other hand, the derivative approximation in the SPH method is the equivalent of central difference and thus is not very well suited to the role of limiting. For that reason, two new derivative approximations are derived from eq. 2.17, in order to calculate the forward and backward derivatives.

Finite differences: $\left\langle \nabla\mathbf{\Phi}_i \right\rangle_F = \dfrac{\mathbf{\Phi}_{i+1} - \mathbf{\Phi}_i}{x^a_{i+1} - x^a_i} \Rightarrow$

$$\text{SPH: } \left\langle \nabla\mathbf{\Phi}_i \right\rangle_F = 2\sum_{X^a_j > X^a_i} \frac{m_j}{\rho_j}\left(\mathbf{\Phi}_j - \mathbf{\Phi}_i\right)\nabla W_{ij} \tag{6.19}$$

Finite differences: $\left\langle \nabla\mathbf{\Phi}_i \right\rangle_B = \dfrac{\mathbf{\Phi}_i - \mathbf{\Phi}_{i-1}}{x^a_i - x^a_{i-1}} \Rightarrow$

$$\text{SPH:}\left\langle \nabla \mathbf{\Phi}_i \right\rangle_B = 2 \sum_{X_j^a < X_i^a} \frac{m_j}{\rho_j} \left(\mathbf{\Phi}_j - \mathbf{\Phi}_i \right) \nabla W_{ij} \tag{6.20}$$

The reason the summations are multiplied by 2 in eq. 6.19 and 6.20 is because only half of the kernel was used for the approximation (fig. 6.3). The equations above satisfy also the relation of central derivative:

$$\left\langle \nabla \mathbf{\Phi}_i \right\rangle_C = \frac{\left\langle \nabla \mathbf{\Phi}_i \right\rangle_F + \left\langle \nabla \mathbf{\Phi}_i \right\rangle_B}{2} \tag{6.21}$$

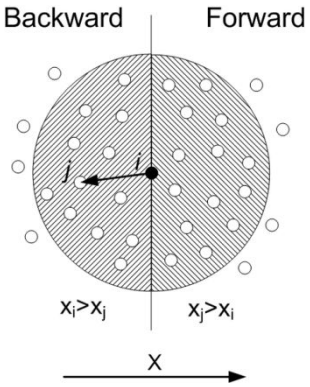


Fig. 6.3. Support domain for calculating forward and backward derivatives.

The procedure for the implementation of the MUSCL scheme is based on the procedure described by E. Toro [4] and is the following:

- First of all it is needed to obtain the forward and backward derivatives before the Riemann problem solution, using the equations 6.19 and 6.20.
- Then for each interacting set of particles *i*, *j* it is needed to calculate the slopes $\mathbf{\Delta}_i$ and $\mathbf{\Delta}_j$ using forward and backward derivatives. The slope can be calculated as :

$$\mathbf{\Delta}_{i,F} = \nabla \mathbf{\Phi}_{i,F/B} \cdot \mathbf{r}_{ij} \text{ and} \tag{6.22}$$

$$\mathbf{\Delta}_{i,B} = \nabla \mathbf{\Phi}_{i,F/B} \cdot \mathbf{r}_{ij} \tag{6.23}$$

Here it must be highlighted that the projection (eq. 6.22 and 6.23) for the slope calculation is done considering the local coordinate system. Hence the calculation of the local slope may involve both forward and backward derivatives, depending on which quadrant the $\mathbf{n}_{ij}$ direction vector (unity vector of $\mathbf{r}_{ij}$) lies. For example, in fig. 6.4 the unity vector $\mathbf{n}_{ij}$ points towards the second quadrant, thus the forward local slope, denoted with $\Delta f\big|_{F,i}$, involves the forward derivative on the *y*-direction, but the backward derivative on the *x*-direction, i.e.:

$$\Delta f\big|_{i,F} = \left.\frac{df}{dx}\right|_{i,B} \left(x_j - x_i \right) + \left.\frac{df}{dy}\right|_{i,F} \left(y_j - y_i \right)$$

On the other hand, the backward local slope $\Delta f\big|_{B,i}$, involves the backward derivative on the *y*-direction and the forward derivative on the *x*-direction, i.e.:

$$\Delta f\big|_{i,B} = \left.\frac{df}{dx}\right|_{i,F}\left(x_j - x_i\right) + \left.\frac{df}{dy}\right|_{i,B}\left(y_j - y_i\right)$$

In a similar manner it is possible to calculate the local slope for particle *j* for 2D and 3D problems.

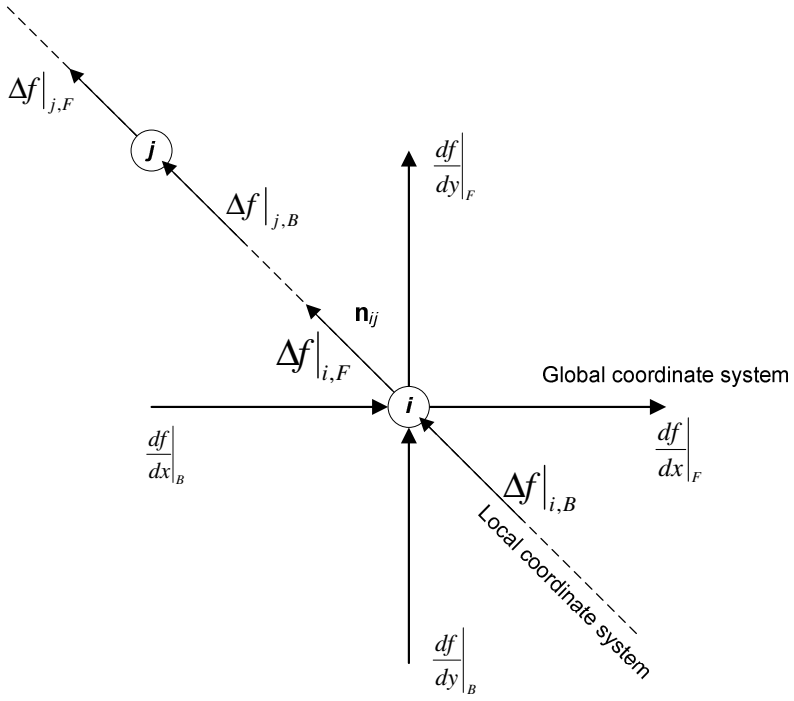


Fig. 6.4. Projection using local coordinate system.

- For each particle the limiting procedure is done, by comparing the slope calculated with the forward and the backward derivative. The following limiter function was used:

$$\overline{\Delta}_i = \begin{cases} \max[0, \min(\beta\Delta_{i,B}, \Delta_{i,F}), \min(\Delta_{i,B}, \beta\Delta_{i,F})] & \Delta_{i,F} > 0 \\ \min[0, \max(\beta\Delta_{i,B}, \Delta_{i,F}), \max(\Delta_{i,B}, \beta\Delta_{i,F})] & \Delta_{i,F} < 0 \end{cases} \tag{6.24}$$

  Setting the β parameter value equal to 2, the limiter reproduces the SuperBee flux limiter, which is shown to have a good performance [4].
- Reconstruction of data at the interface between particles:

$$\mathbf{\Phi}_{iL} = \mathbf{\Phi}_i - \tfrac{1}{2}\overline{\mathbf{\Delta}}_i \tag{6.25}$$

$$\mathbf{\Phi}_{iR} = \mathbf{\Phi}_i + \tfrac{1}{2}\overline{\mathbf{\Delta}}_i \tag{6.26}$$

$$\mathbf{\Phi}_{jL} = \mathbf{\Phi}_j - \tfrac{1}{2}\overline{\mathbf{\Delta}}_j \tag{6.27}$$

$$\mathbf{\Phi}_{jR} = \mathbf{\Phi}_j + \tfrac{1}{2}\overline{\mathbf{\Delta}}_j \tag{6.28}$$

- Solution of the Riemann problem using $\mathbf{\Phi}_{iR}$ as the left state and $\mathbf{\Phi}_{jL}$ as the right state, instead of using $\mathbf{\Phi}_i$ and $\mathbf{\Phi}_j$.

Using the above procedure it is possible to construct a high order SPH-R algorithm, which is able to produce results with little numerical dissipation and without dispersion.

6.3. Tests/Applications

In this section, several indicative tests will be presented, where the performance of the 1st and 2nd order SPH-R method will be examined.

Shock tube

The shock tube test was already described before, in the standard SPH method (section 4.1). The simulated medium is at two different states (left and right) which are separated by a discontinuity. During the interaction of the two states, a characteristic wave structure is formed. The resolution used is the same as the previous SPH simulation; particle distribution is denser in the high density region (spacing dx=0.018m) and sparser at the low density region (spacing dx=0.02m). The initial conditions are:

$$\begin{cases} \rho_L = 1100, \ u_L = 100 & x < 0 \\ \rho_R = 1000, \ u_R = 0 & x > 0 \end{cases}$$

Figure 6.5 shows the results of the simulations. An important feature of the SPH-R solver, regardless of the 1st or 2nd order scheme, is that it does not exhibit oscillations near the contact discontinuity wave, as it was experienced with the standard SPH solver. As it is expected, the solution of the 1st order SPH-R method smears sharp gradients. On the other hand the 2nd order SPH-R method gives results close to the exact Riemann problem solution.

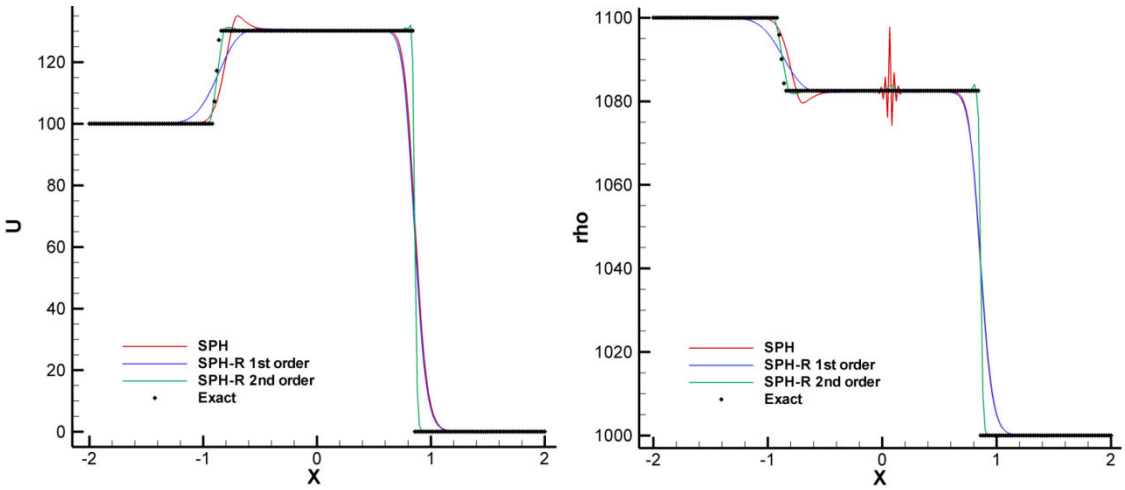


Fig. 6.5. Results of the SPH and SPH-R method for the shock tube case. Left: *u*-velocity component. Right: density. Comparison with the exact solution.

2d jet impingement

A 2d jet impingement under 60^{o} angle has been used as a benchmark to test the behavior of the SPH-R method in hydrodynamic applications. Results of the pressure coefficient and velocity magnitude distribution are shown in fig. 6.6 and fig. 6.7. As a general remark, the results of the SPH-R method, both when using the 1st and 2nd order scheme, are smoother than the standard SPH, with or without corrections, such as the density filter. This is also illustrated when considering fig. 6.8, showing the instantaneous pressure distribution on the flat plate, in conjunction with the respective results from the standard SPH simulation (see chapter 3); SPH-R pressure distribution is smoother and closer to the theoretical solution.

Regarding the performance of the two schemes, the 1st order scheme tends to underestimate greatly the velocity of the branches formed after the impingement (fig. 6.7) - fluid velocity at branches should be equal to the impinging jet velocity. This eventually results to an overestimation of the branch thickness. Also there are some numerical artifacts on the pressure field at the highly curved regions (fig. 6.6), where pressure is overestimated.

On the other hand, the 2nd order scheme calculates correctly the velocity of the branches, and consequently, their thickness, while it also reproduces a smooth, accurate pressure field.

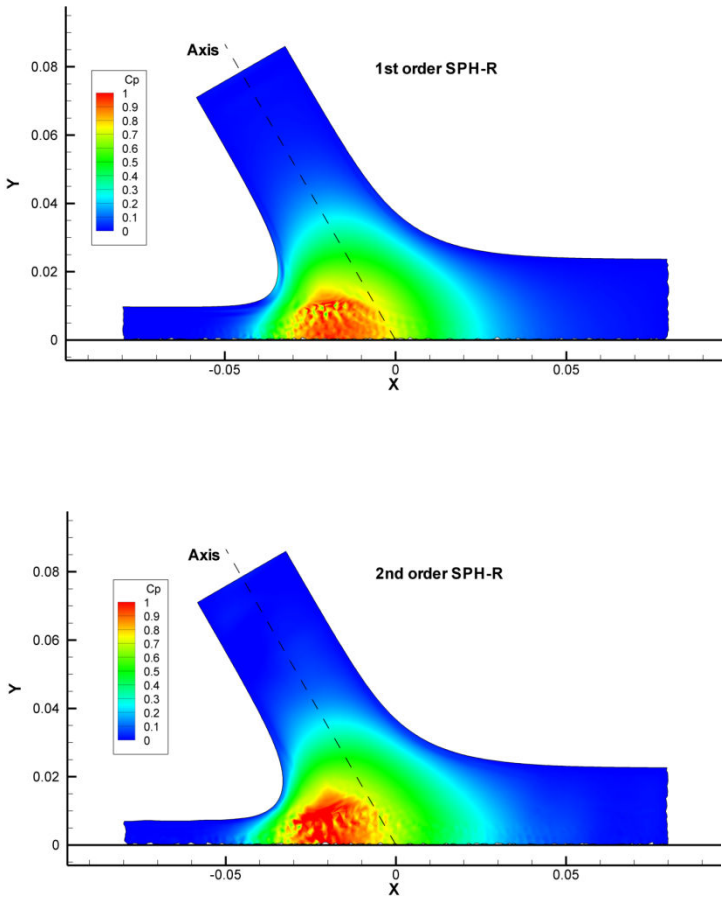


Fig. 6.6. Pressure coefficient distribution for the 1st and 2nd order schemes.

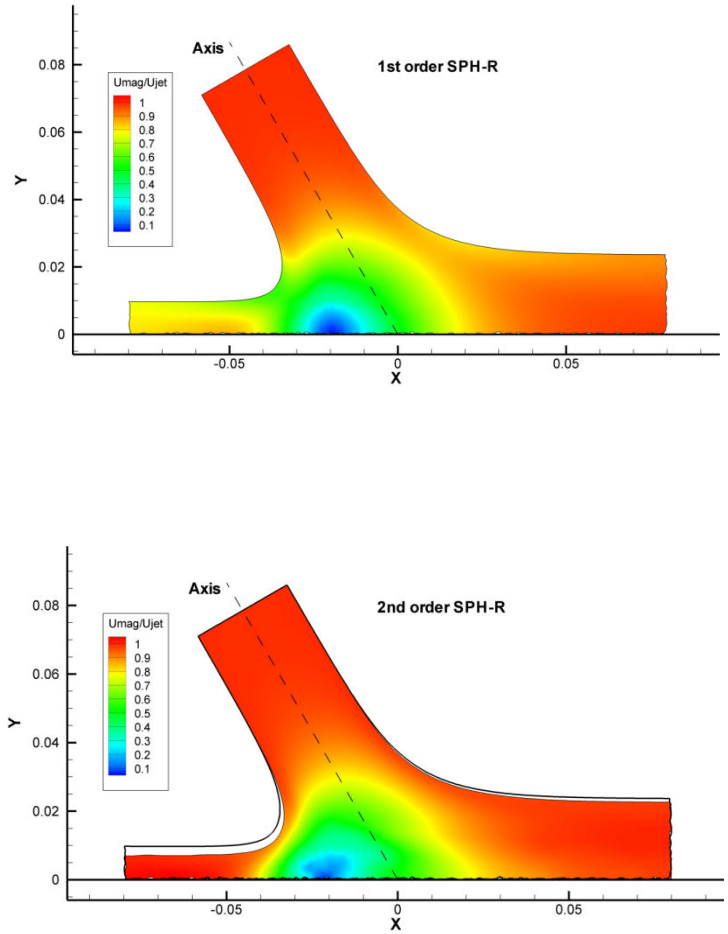


Fig. 6.7. Non-dimensional velocity for the 1st and 2nd order (water jet velocity is used as reference). The free surface of the 1st order scheme is shown in the 2nd order scheme results to highlight the influence of the numerical diffusion of the 1st order scheme.

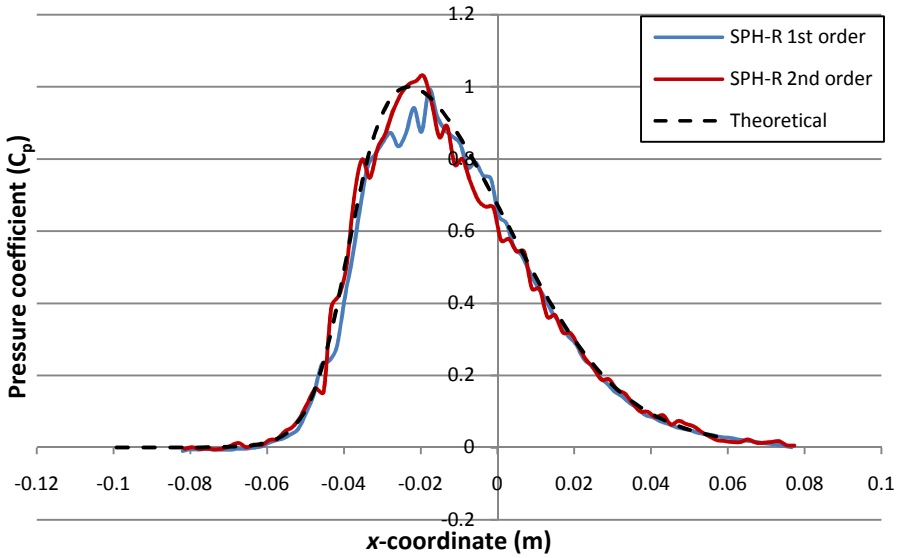


Fig. 6.8. Instantaneous pressure coefficient on the plate.

It is important to mention here that the SPH-R method is prone to particle alignment to a greater extent than the standard SPH method. In the standard SPH method, particle aligning sometimes is destroyed, due to local instabilities. On the other hand the inherent numerical viscosity, even of the 2nd order SPH-R scheme, tends to prevent such instabilities from occurring and, thus, particles remain aligned (see also fig. 3.6). Eventually, the distribution of field variables is affected by this effect, as shown in fig. 6.9. Note that away from the stagnation point particles are equidistant from each other. When particles move close to the stagnation point, they form lines and eventually wavy structures.

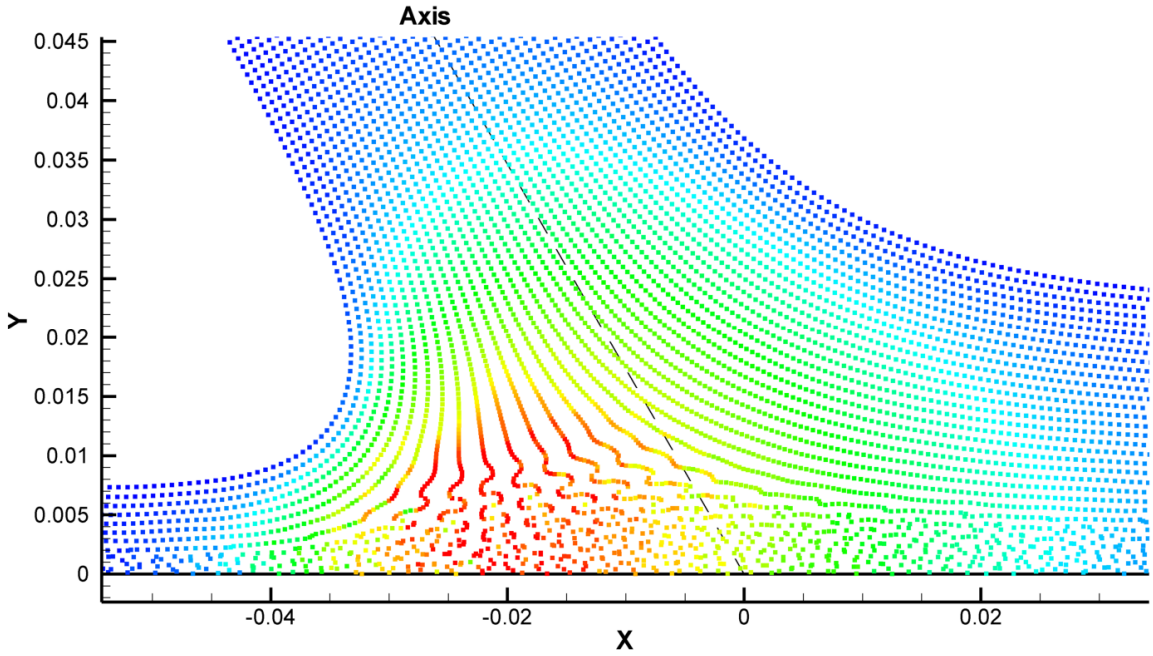


Fig. 6.9. Particle alignment near the stagnation point.

2d jet impingement on a moving blade

The 2nd order SPH-R method was used to simulate the case of the jet impingement on the 2d Turgo turbine blade, using the same parameters as the standard SPH simulation. An indicative view of the results is shown in fig. 6.10. Free-surface fragmentation has been reduced, comparing with the standard SPH, due to the numerical viscosity effects. The effect of numerical viscosity is visible in the force history over time; SPH-R results are much less oscillatory than the standard SPH method results.

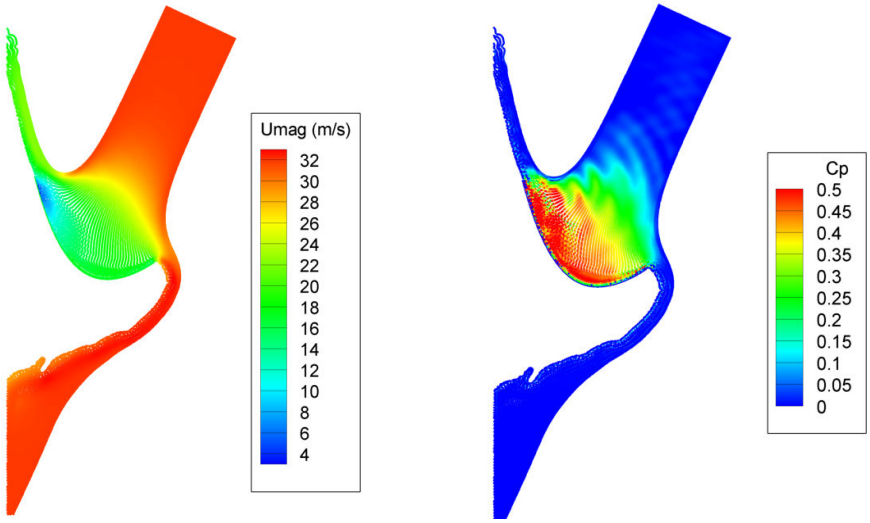


Fig. 6.10. Indicative instance of the 2d Turgo jet impingement (t=0.03s). Left: velocity magnitude. Right: Pressure distribution.

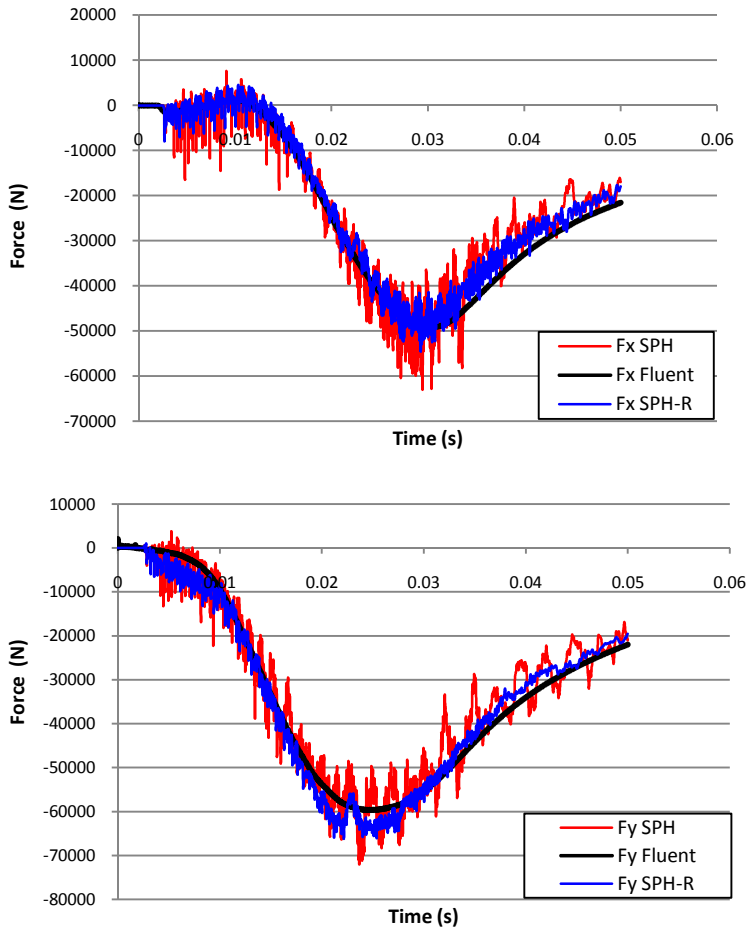


Fig. 6.11. Forces history on the moving blade. Up: x-axis. Down: y-axis.

3d jet impingement on Turgo blade and Pelton bucket

Next, the 2nd order SPH-R method was used to simulate the impingement on the stationary Turgo blade (30° impingement) and Pelton turbine bucket (90° impingement). In fig. 6.12 the general view of the flow is shown for the two impingement cases. Comparing with the relevant results from the standard SPH method, it is obvious that the velocity distributions are smoother due to the effects of numerical viscosity. In both cases the outlet velocity of the formed water sheet is less than the jet velocity, despite using the 2nd order scheme. Moreover, in the Pelton bucket the flow tends to form thread-like structures. This is the result of particle alignment in conjunction with the scheme's numerical viscosity. However, in this particular case, it does not affect negatively the simulation, since there are no stagnation points (apart from the splitter) and there is no particle clumping.

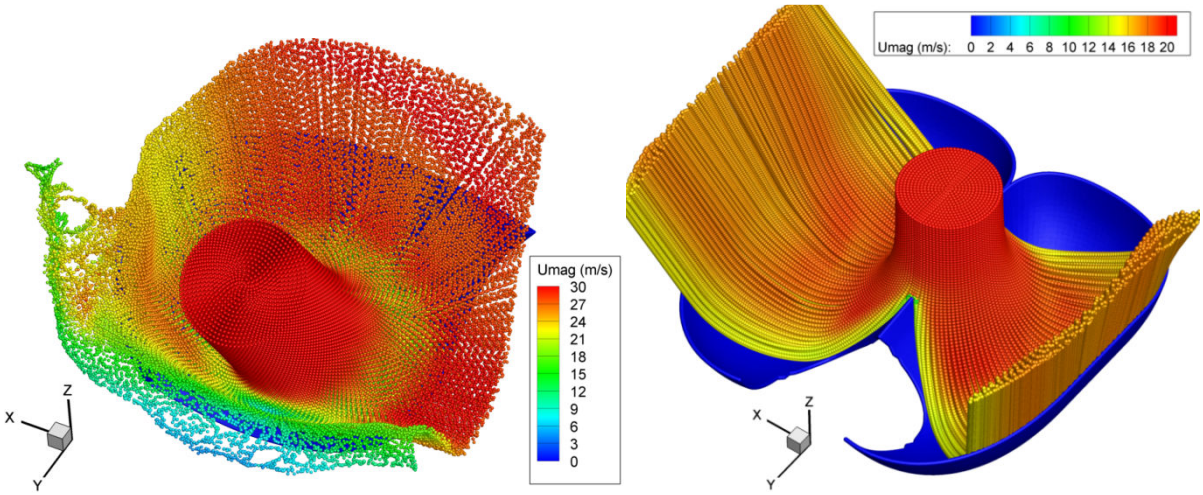


Fig. 6.12. Simulation of stationary jet impingement of a: Turgo turbine blade (left) and a Pelton turbine bucket (right).

In both cases forces are underestimated, both in respect to the Fluent and standard SPH solutions. Again this is mainly attributed to the inherent numerical viscosity of the scheme.

Table 6-I. Comparison of the calculated forces with SPH and SPH-R. Turgo jet impingement.

Calculated forces	F_x [N]	F_y [N]	F_z [N]
Fluent	5425.575	-8614.56	-25216.4
SPH	5338.143	-8476.81	-24802.1
SPH-R	4546.51	-8130.29	-23572.25

Table 6-II. Comparison of the calculated forces with SPH and SPH-R. Pelton jet impingement.

Calculated forces	F_x [N]	F_y [N]	F_z [N]
Fluent	-21.9	-32.1	-278.8
SPH	-17.7	-30.5	-260.6
SPH-R	-25.0	-29.5	-249.6

Simulation of impulse turbines using the SPH-R method

Although in the stationary impingement cases the 2nd order SPH-R method proved overly diffusive, underestimating forces, in the simulation of impulse turbines it gave similar results in respect to the SPH method (see fig. 6.13 and 6.14). First of all the torque curve is smoother, without the intense oscillations of the SPH method. Then the SPH-R method tends to predict better the torque during the evacuation of the Pelton bucket. It is reminded here, that during the evacuation phase the SPH method under predicted torque. SPH-R method behaves better in the moving geometry cases because generally flow features change less than in the stationary cases. The reason of this effect lies in the numerical scheme used; the numerical viscosity is expected to increase with the increase of the difference between the two states of the Riemann problem. Thus, in cases where flow features change rapidly, it is expected that the influence of numerical viscosity is increased.

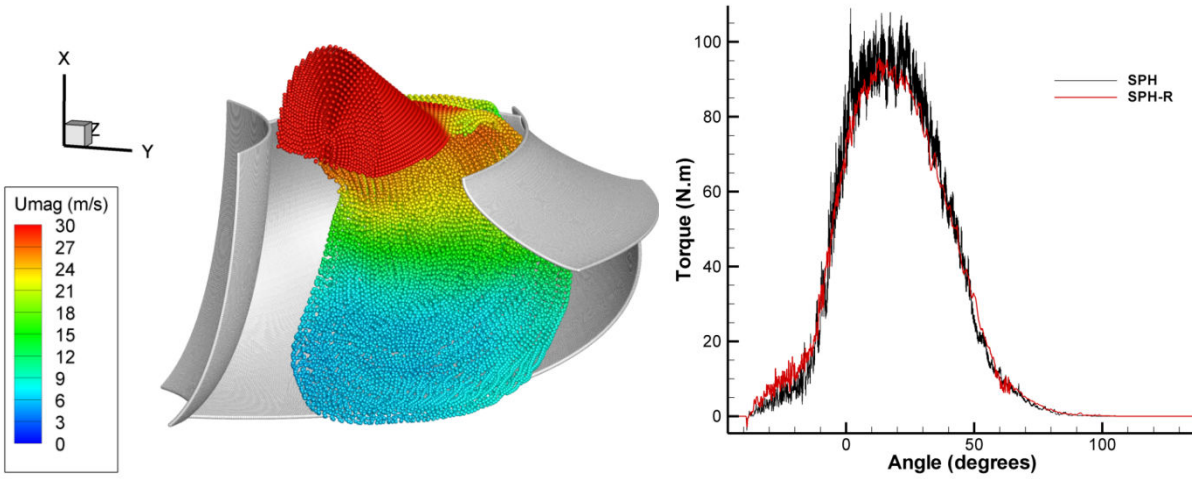


Fig. 6.13. Left: Simulation of a rotating Turgo turbine. Right: torque curve comparison (SPH and SPH-R).

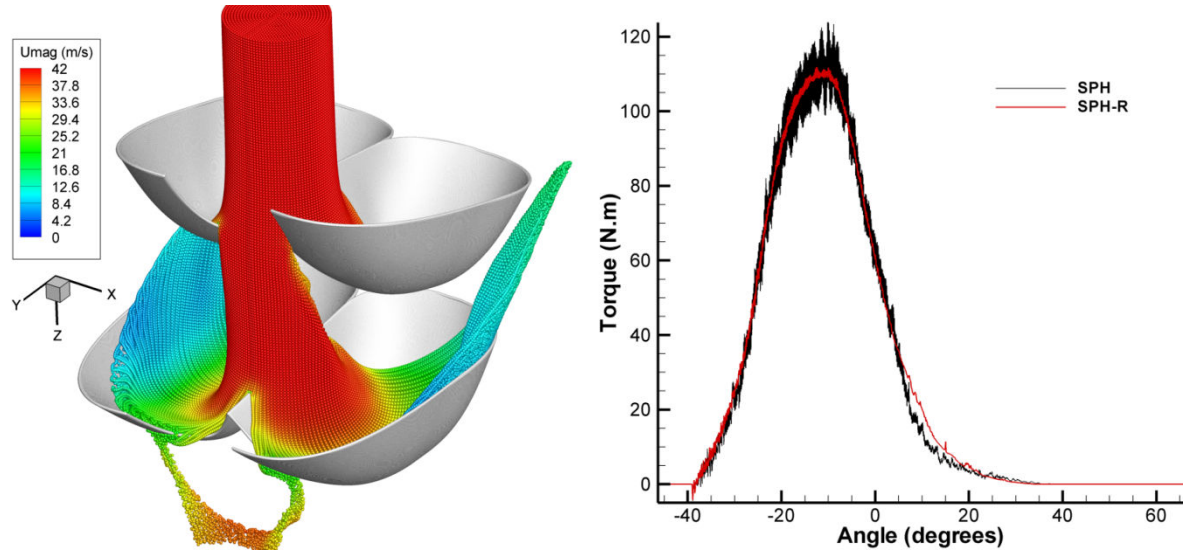


Fig. 6.14. Left: Simulation of a rotating Pelton turbine. Right: torque curve comparison (SPH and SPH-R).

6.4. Further possibilities with SPH-R

In the previous section the SPH-R method was described. While the method reduces the pressure oscillations of the SPH method, it still handles boundaries in a problematic way. An alternative is the implementation of boundaries, considering the surface integral term that has been omitted from the SPH derivative approximation (see chapter 2 eq. 2.7). However this introduces an additional problem; the field variables have to be defined on the boundary surface elements. This is done by solving the partial Riemann problem, which was based on the work of F. Dubois [9] (this treatment was initially adapted to the SPH-ALE method by Marongiu et al. [7, 10] and [11] and will be further described in chapter 7).

The partial Riemann problem is practically identical to the Riemann problem already described (see also Appendix A), with the difference that only the left state is available and the other state is missing, since it extends beyond the boundary. However the right state is possible to be deduced from the boundary condition to be enforced. For example for a wall boundary condition, the left and right states should have the same density/pressure. The velocity of the right state is assumed to be, expressed in the local coordinate system, as [4]:

$$u_R = -u_L + 2u_{wall} \tag{6.29}$$

In order to determine the interaction of a fluid particle with a boundary element, the velocity of the fluid particle is projected on the local boundary surface normal vector $\mathbf{n}_j$ and not the $\mathbf{n}_{ij}$ vector (fig. 6.15). Then the partial Riemann solver is solved, using the left state and the deduced right state. Once the solution is obtained, using an approximate Riemann solver, the solution is added to the derivative approximation, as follows:

$$\frac{D\rho_i}{Dt} = V_{term} - \sum_{j\in\partial\Omega} \rho_j S_j 2\left(U_i^{proj} - U_{ij}^*\right) W_{ij} \tag{6.30}$$

$$\frac{D\mathbf{u}_i}{Dt} = \mathbf{V}_{term} + \mathbf{f}_{body} + \sum_{j\in\partial\Omega} S_j \left(\frac{2p_{ij}^*}{\rho_i}\right) W_{ij}\mathbf{n}_j \tag{6.31}$$

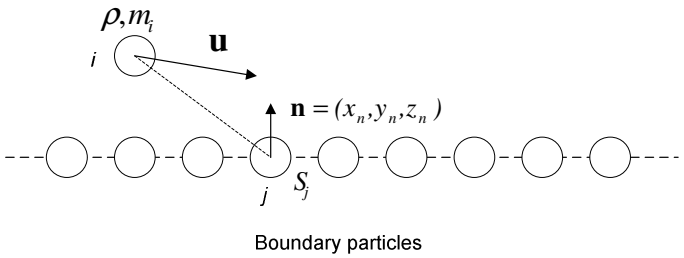


Fig. 6.15. Interaction of a fluid particle with a boundary element

In eq. 6.30 and 6.31 V_{term} is the volume term in respective equations 6.7 and 6.8. Also S_j represents the surface of the finite boundary element and $\left[p_{ij}^*, U_{ij}^*\right]$ is the solution of the partial Riemann problem between particle *i* and the boundary element *j*.

Field variable values on the boundary are obtained using the following relations for pressure/density and local velocity respectively:

$$p_j = 2\sum_{i\in\Omega}\frac{m_i}{\rho_i}p_E W_{ij} \overset{Tait}{\rightarrow} \rho_j \tag{6.32}$$

$$\text{Local velocity: } \begin{cases} u_j = u_{wall} \\ v_j = \dfrac{\sum_{i\in\Omega}\frac{m_i}{\rho_i}v_i W_{ij}}{\sum_{i\in\Omega}\frac{m_i}{\rho_i}W_{ij}} \end{cases} \tag{6.33}$$

Boundary values may be used for the derivative calculation of the MUSCL scheme. Note that velocity calculated by eq. 6.33 is the velocity of the local Riemann problem, i.e. projected on the local wall normal. Thus it must be transformed to the global coordinate system in order to be used properly.

This boundary implementation enables proper handling of any boundary condition and good accuracy. However a possible issue may arise in cases when the support domain of a fluid particle, truncated by the boundary, is concave as in fig. 6.16. In that case, the fluid particle *i* will interact with boundary particle *j*. The solution of the partial Riemann problem will result to a negative pressure, eventually leading to an unphysical attraction of the fluid particles towards the boundary. The cause of the problem is the fact that the particle interaction radius is much larger than the boundary discretization.

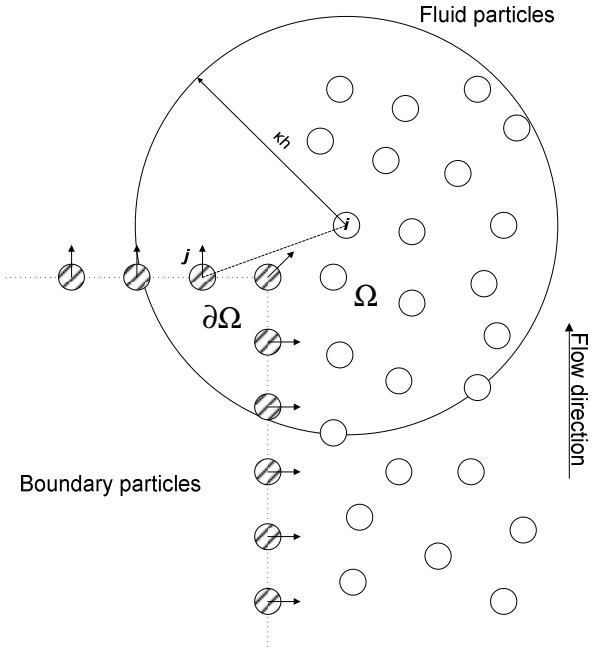


Fig. 6.16. Concave support domain.

A possible solution of that problem is to divide boundary particles into two types. In the first type, boundary particles will behave as already described and will be placed in areas away from corners such as in fig. 6.15. In the second type, the boundary particles will behave as described, but if a negative pressure (from the solution of the Riemann problem) is detected, then the interaction with the

fluid particle will be omitted. These particles will be placed on corners and while they will prevent boundary penetration, they will also prevent the unphysical attraction described.

The boundary treatment described above will be further developed and mainly used in the SPH-ALE method, which will be described in the next chapter.

6.5. Hybrid SPH/SPH-R

Another interesting approach is to use the partial Riemann problem approach for boundaries in combination with the SPH with the upwind flux correction, described in eq. 3.75. Eventually:

$$\frac{D\rho_i}{Dt} = \sum_{j\in\Omega} m_j \left(\mathbf{u}_{ij} \cdot \nabla_i W_{ij} + \mathbf{n}_{ij} \cdot \nabla_i W_{ij} \frac{c_{ij}}{\rho_j} \left(\rho_i - \rho_j \right) \right) - \sum_{j\in\partial\Omega} \rho_j S_j \left(\mathbf{u}_i - \mathbf{u}_{ij}^* \right) \cdot \mathbf{n}_j W_{ij} \tag{6.34}$$

$$\frac{D\mathbf{u}_i}{Dt} = -\sum_{j\in\Omega} m_j \left(\frac{p_i}{\rho_i^{\,2}} + \frac{p_j}{\rho_j^{\,2}} \right) \nabla W_{ij} + \sum_{j\in\partial\Omega} S_j \frac{2p_{ij}^*}{\rho_i} \mathbf{n}_j W_{ij} + \mathbf{f}_{body} \tag{6.35}$$

The first term in eq. 6.34 and eq. 6.35 is the volume term appearing in eq. 3.75 and eq. 3.39 respectively. The next term is the surface integral term used to enforce the boundary conditions. Star marked values are the solution of the partial Riemann problem from the interaction of a fluid particle with a boundary particle.

The combined approach ensures enforcement of boundary conditions, while limiting numerical diffusion, since Riemann solvers are only used for the interaction with the boundary particles. Pressure oscillations are treated by the diffusion term (upwind monotone flux) used in the density equation. No further corrections, such as XSPH or density filtering, are required.

In fig. 6.17 a comparison of the experimental, SPH and SPH/SPH-R hybrid results are shown for the dam break case; results are practically identical, indicating that the hybrid scheme is accurate. In the next figure (fig 6.18) indicative instances of the dambreak problem are shown. The SPH/SPH-R hybrid method is able to reproduce an accurate hydrostatic pressure field, something that the standard SPH method fails to do (see fig. 6.19).

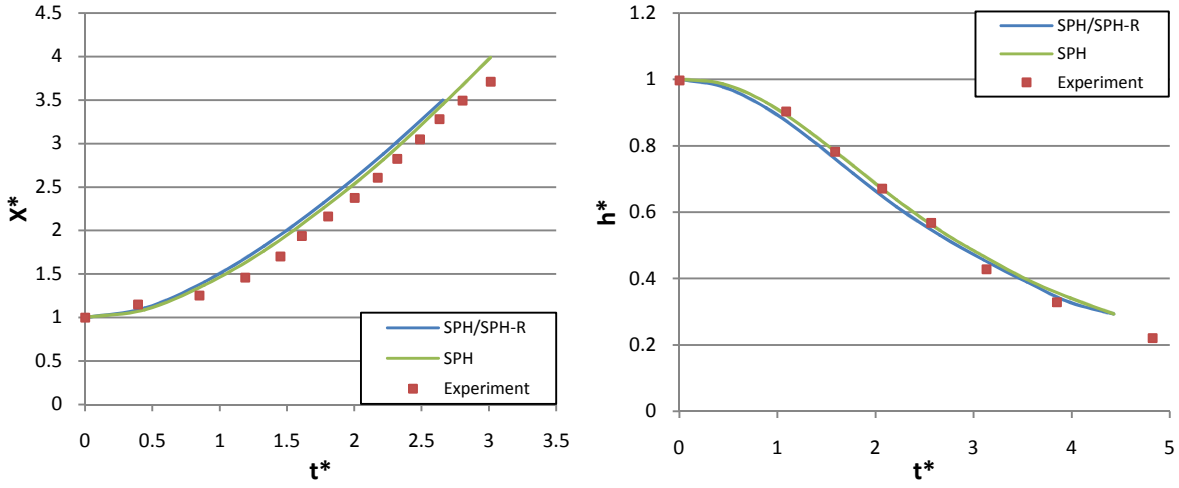


Fig. 6.17. Dam break case. Non-dimensional wave propagation (left) and column height evolution (right), in respect to non-dimensional time.

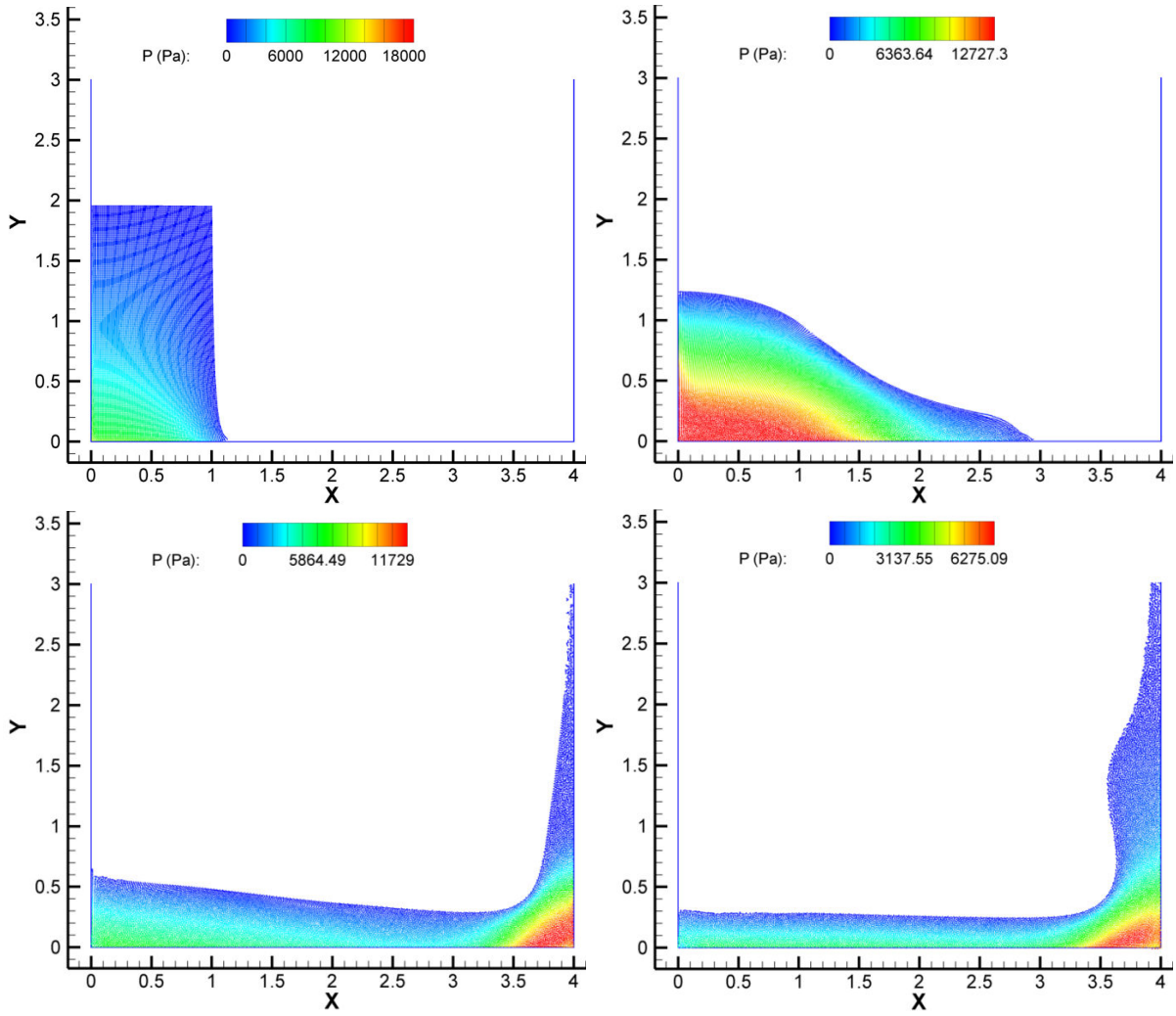


Fig. 6.18. Indicative instances from the dambreak simulation using the hybrid SPH/SPH-R.

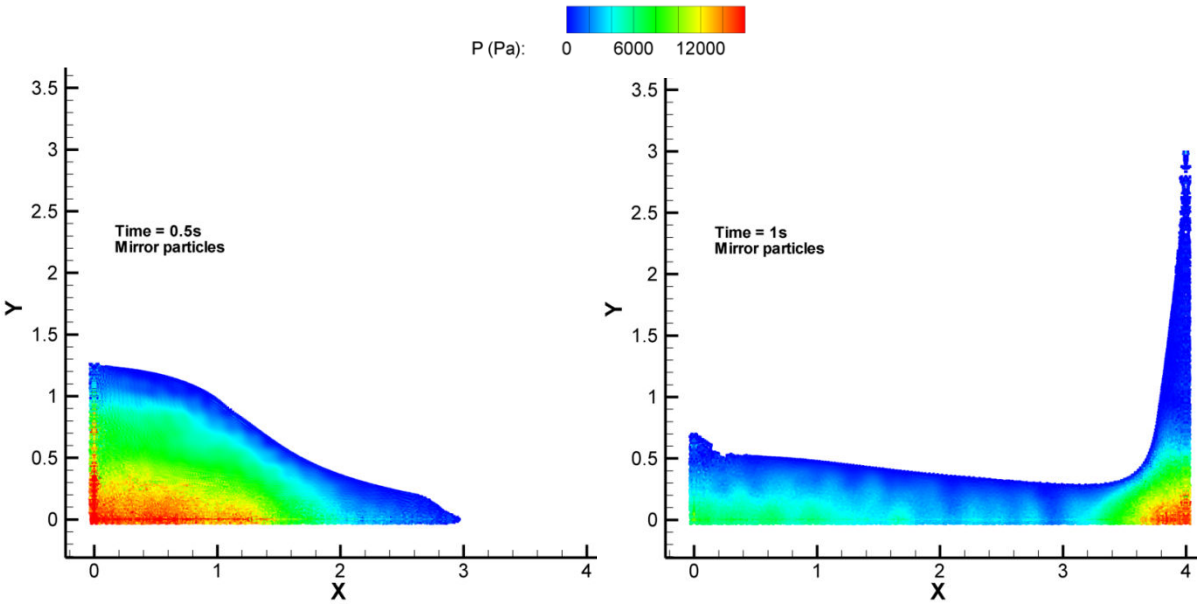


Fig. 6.19. Indicative instances from the dambreak simulation using the standard SPH. Boundaries are treated with mirror particles.

In fig. 6.20 a 2d jet impingement case is shown using the hybrid SPH/SPH-R method. The method is able to calculate correctly the free surface with a little deviation from the 2nd order SPH-R scheme. On the other hand the pressure distribution on the plate is very close to the theoretical solution.

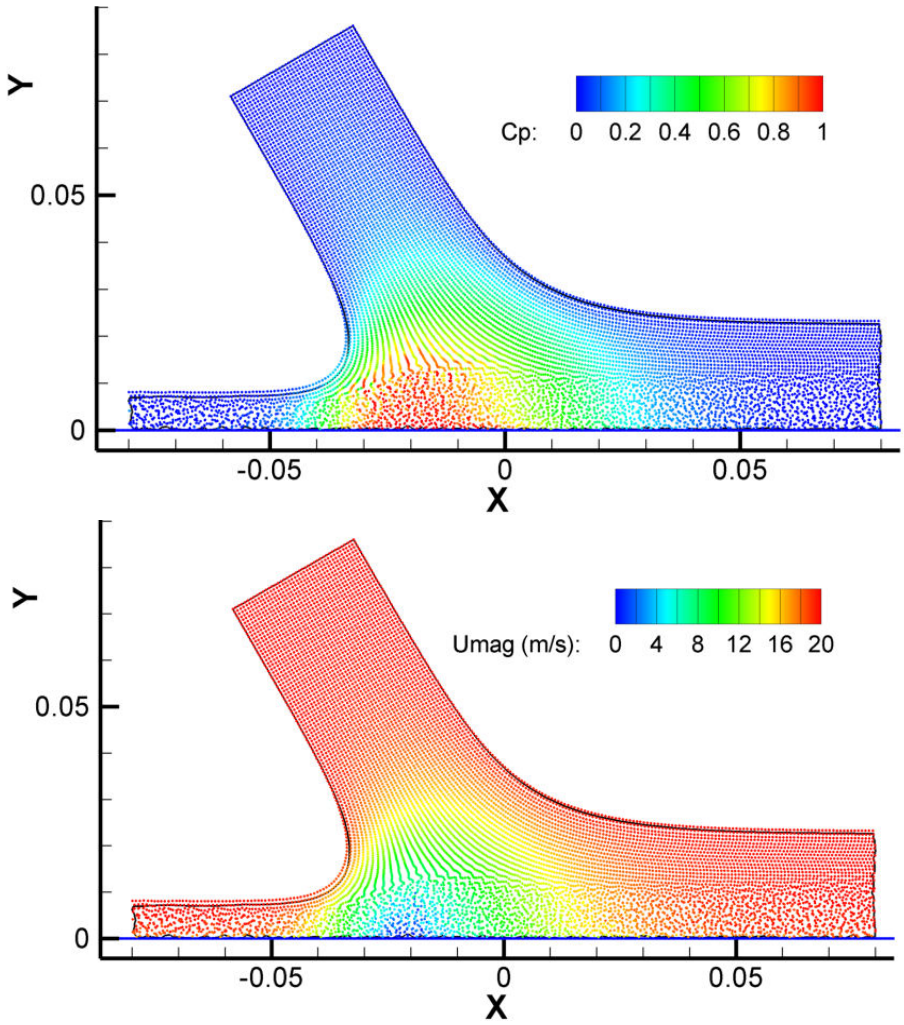


Fig. 6.20. Jet impingement on a flat plate. SPH/SPH-R results for: pressure coefficient (up) and velocity magnitude (down). The 2nd order SPH-R solution is shown as reference with a continuous line.

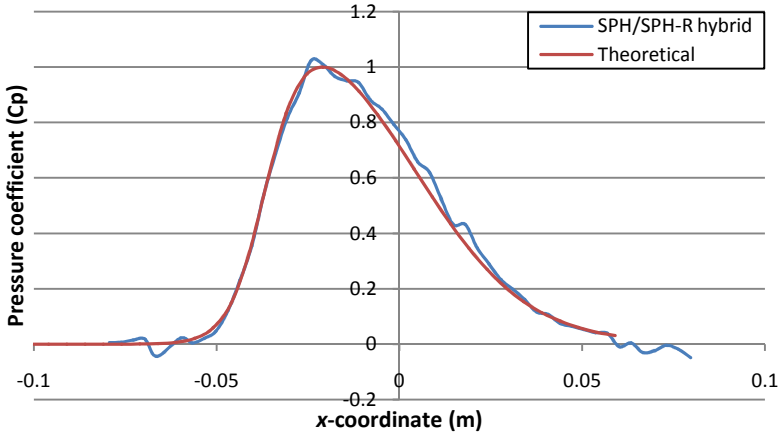


Fig. 6.21. Instantaneous pressure coefficient distribution on the flat plate. Comparison with the theoretical solution.

6.6. Concluding remarks

In this chapter, the combination of the SPH method with Riemann solvers was examined. The resulting method is able to remedy the pressure oscillations of the SPH method. Also there is the capability to describe accurately boundary conditions, using partial Riemann problems. However, even the 2nd order scheme still suffers from numerical diffusion and its effects are even more observable in 3d simulations. Moreover, the MUSCL treatment requires the calculation of the derivatives of field variables and the implementation of the limiter, which contains several "*if*" statements. Considering that these calculations have to be performed many times at each time step, this results to an increased computational cost. Indeed the 2nd order SPH-R algorithm may require around 5 $\times$ the time needed for the SPH algorithm to complete the same simulation. Eventually the SPH-R method might not be the best alternative from the SPH method, since it is slower and some inaccuracies still appear.

An interesting alternative is the hybrid SPH/SPH-R method presented in the final section of the current chapter. This method does not require the MUSCL scheme, involving limiters/derivative calculation etc. On the other hand it is able to treat boundaries efficiently, due to the implementation of partial Riemann problems at boundaries. All the previous come to a computational cost comparable to the standard SPH method.

References

[1] A. Parshikov, S. Medin, "Smoothed Particle Hydrodynamics using interparticle contact algorithms", Journal of computational physics, vol. 180, p. 358-382, 2002. DOI: 10.1006/jcph.2002.7099.

[2] M. Gomez-Gesteira, B.D. Rogers, R.A. Dalrymple, A.J.C. Crespo, M. Narayanaswamy "Users guide for the SPHysics code", http://wiki.manchester.ac.uk/sphysics/

[3] R. J. Leveque, "Finite volume methods for hyperbolic problems", Cambridge University Press, ISBN 0-511-04219-1.

[4] E.F. Toro, "Riemann Solvers and Numerical methods for fluid dynamics: a practical introduction", 3rd edition Springer, ISBN 978-3-540-25202-3, e-ISBN 978-3-540-49834-6, DOI: 10.1007/978-3-540-49834-6

[5] M. Ben-Artzi, J. Falcovitz, "Generalized Riemann Problems in computational dynamics", Cambridge Monographs on Applied and Computational Mathematics, April 14, 2003, ISBN-10: 0521772966, ISBN-13: 978-0521772969.

[6] M.J. Ivings, D.M. Causon, E.F.Toro, "On Riemann solvers for compressible liquids", International Journal for Numerical Methods in Fluids, vol. 28, p.395 – 418, 1998.

[7] J. C. Marongiu, E.Parkinson, "Riemann solvers and efficient boundary treatments: a hybrid SPH-finite volume numerical method", Proceedings of the 3rd Spheric workshop, Lausane, June 2008.

[8] Y. Blacodon, J. Bohbot, "About compressible treatment and solid boundary conditions aspects of smoothed particle hydrodynamics", Proceedings of the 4th Spheric workshop, Nantes France, May 2009.

[9] F. Dubois, "Partial Riemann problem, boundary conditions and gas dynamics", Absorbing boundaries and layers, domain decomposition methods: application to large scale computations, p. 16-77, New York 2001, Nova Science Publishers.

[10] J. C. Marongiu, F. Leboeuf, J. Caro, E. Parkinson, "Low Mach number numerical schemes for the SPH-ALE method. Application in free surface flows", Proceedings of the 4th Spheric workshop, Nantes France, May 2009.

[11] F. Magoules, "Computational Fluid Dynamics", Chapman & Hall/CRC 1st edition, ISBN-10: 1439856613, ISBN-13: 978-1439856611.

Chapter 7

SPH-ALE method

In order to overcome the issues of the standard SPH and the SPH-R method, the SPH-ALE method was developed and tested. The SPH-ALE method was initially implemented by Vila [1] and has considerable differences from the traditional SPH. First of all the conservative system of the Euler equations is solved $[\rho, \rho u, \rho v, \rho w]^T$, instead of using the primitive variable set of the standard SPH method. Also it adopts an Arbitrary Lagrangian Eulerian point of view, meaning that the computational elements move due to the influence of a transport field not necessarily equal to the velocity field, thus the particles may move following the flow field (Lagrangian description), may remain still (Eulerian description), or move in any other arbitrary way. Finally, the discretization of the Euler equations resembles the discretization of the Finite Volume Godunov method and Riemann solvers are used to calculate the fluxes between the computational elements which are now considered moving volumes, rather than particles (but they will be referred as particles for simplicity). The great advantage of the SPH-ALE method is that it enables robust handling of the boundary conditions. Moreover, the Godunov method introduces numerical diffusion, damping pressure oscillations, providing much smoother results than the standard SPH method.

7.1. Governing equations

The Euler equations, using the ALE description, may be described in the following form [1]:

$$L_{u_0}(\mathbf{\Phi}) + \sum_{l=1,d} \frac{\partial}{\partial x^l}\left(\mathbf{F}_E^l - u_0^l \mathbf{\Phi}\right) = 0 \tag{7.1}$$

In the above equation:

- Φ is the vector of conserved variables: $[\rho, \rho u, \rho v, \rho w]^T$ in 3D
- $L_{u_0}(\mathbf{\Phi}) = \frac{\partial \mathbf{\Phi}}{\partial t} + \frac{\partial (u_0 \mathbf{\Phi})}{\partial x} + \frac{\partial (v_0 \mathbf{\Phi})}{\partial y} + \frac{\partial (w_0 \mathbf{\Phi})}{\partial z}$
- u_0^l is the l-component of the transport field velocity vector $\mathbf{u}_0 = (u_0, v_0, w_0)$
- $\mathbf{F}_E^l$ is the flux vector. For a 3D calculation the flux vectors would be [2]:

$$\mathbf{F}_E^1 = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \end{bmatrix} \quad \mathbf{F}_E^2 = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \end{bmatrix} \quad \mathbf{F}_E^2 = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \end{bmatrix}$$

The discretization of eq. 7.1 using SPH would lead to one-dimensional moving Riemann problems between interacting particles [1]. Thus for two interacting particles:

$$\begin{cases} \dfrac{\partial \mathbf{\Phi}}{\partial t} + \dfrac{\partial}{\partial x}\left(\mathbf{F}_E(\mathbf{\Phi})\cdot\mathbf{n}_{ij} - \mathbf{u}_0(\mathbf{x}_{ij},t)\cdot\mathbf{n}_{ij}\mathbf{\Phi}\right) = 0 \\ \mathbf{\Phi}(x^{(n_{ij})},0) = \begin{cases} \mathbf{\Phi}_i & if \quad x^{(n_{ij})} < 0 \\ \mathbf{\Phi}_j & if \quad x^{(n_{ij})} > 0 \end{cases} \end{cases} \tag{7.2}$$

where $\mathbf{\Phi}_i$ and $\mathbf{\Phi}_j$ are the states of particles i and j respectively and $\mathbf{n}_{ij}$ is the unit vector pointing from particle i to particle j, $\mathbf{x}_{ij}$ is the midpoint between particle i and j and $x^{(nij)}$ is the x-coordinate at the local coordinate system, defined by $\mathbf{n}_{ij}$, assuming $\mathbf{x}_{ij}$ is the origin. Vila [1] proved that the solution of the moving Riemann problem in eq. 7.2 can be linked to the classical Riemann problem for the Euler equation, stated as:

$$\begin{cases} \dfrac{\partial \mathbf{\Phi}}{\partial t} + \dfrac{\partial}{\partial x}\left(\mathbf{F}_E(\mathbf{\Phi})\cdot\mathbf{n}_{ij}\right) = 0 \\ \mathbf{\Phi}(x^{(n_{ij})},0) = \begin{cases} \mathbf{\Phi}_i & if \quad x^{(n_{ij})} < 0 \\ \mathbf{\Phi}_j & if \quad x^{(n_{ij})} > 0 \end{cases} \end{cases} \tag{7.3}$$

$$\mathbf{\Phi} = \mathbf{\Phi}_E\left(\frac{x^{(n_{ij})} + X_0(t)}{t}, \mathbf{\Phi}_i, \mathbf{\Phi}_j\right)$$

$$X_0(t) = \int_0^t \mathbf{u}_0(\mathbf{x}_{ij},t)\cdot\mathbf{n}_{ij}\,d\tau$$

The resulting particle approximation for the 3D inviscid Euler equations is the following [1]:

$$\frac{d\mathbf{r}_i}{dt} = \mathbf{u}_0(\mathbf{r}_i,t)$$

$$\frac{d}{dt}(\omega_i) = \omega_i\sum_{j=1}^{N}\omega_j\left(\mathbf{u}_0(\mathbf{r}_j) - \mathbf{u}_0(\mathbf{r}_i)\right)\cdot\nabla_i W_{ij}$$

$$\frac{d}{dt}(\omega_i\rho_i) + \omega_i\sum_{j=1}^{N}\omega_j 2\rho_E(\mathbf{u}_E - \mathbf{u}_0)\cdot\nabla_i W_{ij} = 0$$

$$\frac{d}{dt}(\omega_i\rho_i u_i) + \omega_i\sum_{j=1}^{N}\omega_j 2\left[\left[\rho_E u_E(u_E - u_0) + p_E\right]\frac{\partial W}{\partial x} + \left[\rho_E u_E(v_E - v_0)\right]\frac{\partial W}{\partial y} + \left[\rho_E u_E(w_E - w_0)\right]\frac{\partial W}{\partial z}\right] = \omega_i\rho_i g_x$$

$$\frac{d}{dt}(\omega_i\rho_i v_i) + \omega_i\sum_{j=1}^{N}\omega_j 2\left[\left[\rho_E v_E(u_E - u_0)\right]\frac{\partial W}{\partial x} + \left[\rho_E v_E(v_E - v_0) + p_E\right]\frac{\partial W}{\partial y} + \left[\rho_E v_E(w_E - w_0)\right]\frac{\partial W}{\partial z}\right] = \omega_i\rho_i g_y$$

$$\frac{d}{dt}(\omega_i\rho_i w_i) + \omega_i\sum_{j=1}^{N}\omega_j 2\left[\left[\rho_E w_E(u_E - u_0)\right]\frac{\partial W}{\partial x} + \left[\rho_E w_E(v_E - v_0)\right]\frac{\partial W}{\partial y} + \left[\rho_E w_E(w_E - w_0) + p_E\right]\frac{\partial W}{\partial z}\right] = \omega_i\rho_i g_z$$

(7.4-7.9)

In all the above equations, ω represents the volume of the particle, the E index indicates the solution from the Riemann problem between particle i and particle j. Again, pressure is calculated through the Tait equation of state, reformulated as:

$$p_i = k\rho_i^{\gamma} - B \tag{7.10}$$

where $B = \dfrac{\rho_0 {c_0}^2}{\gamma}$ is the stiffness parameter of the equation of state and $k = \dfrac{B}{\rho_0^{\gamma}}$.

7.2. Implementation of the SPH-ALE model (1st order Godunov method)

In order to solve the system of equations described above, the solution of the Riemann problem between particles *i* and *j* has to be found. The Riemann problem has to be solved on the direction defined by the $\mathbf{n}_{ij}$ unit vector pointing from particle *i* to particle *j*. To do this, all velocity components have to be transformed from the global coordinate system to a local coordinate system pointing to the $\mathbf{n}_{ij}$ direction (fig. 7.1). This is accomplished using a global-to-local transformation matrix which in 3D would have the following form:

$$\begin{bmatrix} u_{loc} \\ v_{loc} \\ w_{loc} \end{bmatrix} = \begin{bmatrix} x_n & y_n & z_n \\ x_m & y_m & z_m \\ x_o & y_o & z_o \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{7.11}$$

In the above transformation *u, v, w* are the velocity components, x_n, y_n, z_n are the components of the $\mathbf{n}_{ij}$ unity vector pointing from particle *i* to particle *j* and the *loc* subscript represents local coordinate system variables. Furthermore x_m, y_m, z_m and x_o, y_o, z_o are the components of $\mathbf{m}_{ij}$ and $\mathbf{o}_{ij}$ unity vectors respectively. These vectors are normal to each other and normal to the $\mathbf{n}_{ij}$ vector. Obviously for density and pressure there is no need for any transformation, since they are scalars.

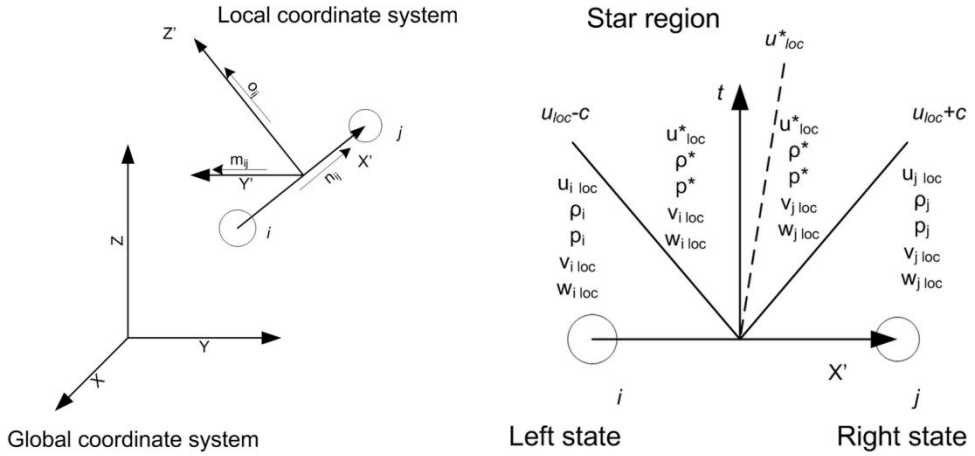


Fig. 7.1. Coordinate system transformation (left) and the solution of the Riemann problem for the Tait equation of state (right), needed for the SPH-ALE algorithm

Then, the local steady one-dimensional Riemann problem (eq. 7.12) has to be solved using either an exact or an approximate solver. An exact solver for the Euler equations, using the Tait equation of state, is presented in appendix A, following Ivings et al. [3]. However the disadvantage of the exact solver is that it involves an iterative procedure for the calculation of the Riemann problem solution, which will be computationally expensive, since it has to be evaluated for all particles interactions for all time steps. Eventually, as with the SPH-R method, an approximate Riemann solver is used. However, in the SPH-ALE method, the approximate solver of Roe is used instead, which is based on the conservative variable set. The jacobian matrix is linearized in respect to the *i, j* states of the Riemann problem. Following Toro [2], the split three-dimensional Riemann problem for the Euler equations is:

$$\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \end{bmatrix}_t + \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \end{bmatrix}_x = 0 \overset{Tait}{\Rightarrow} \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \end{bmatrix}_t + \begin{bmatrix} \rho u \\ \rho u^2 + k\rho^\gamma \\ \rho uv \\ \rho uw \end{bmatrix}_x = 0 \tag{7.12}$$

The *t* and *x* indices denote the time and spatial derivative respectively.

In the previous equation, density ρ and the *u* velocity component play the important role in the Riemann problem. The rest velocity components can be treated as passive scalars. The resulting linearized jacobian of the system of equations (eq. 7.12) is:

$$\tilde{\mathbf{A}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \tilde{c}^2 - \tilde{u}^2 & 2\tilde{u} & 0 & 0 \\ -\tilde{u}\tilde{v} & \tilde{v} & \tilde{u} & 0 \\ -\tilde{u}\tilde{w} & \tilde{w} & 0 & \tilde{u} \end{bmatrix} \tag{7.13}$$

$$\text{where} \quad \tilde{\rho} = \sqrt{\rho_L \rho_R} \qquad \tilde{c} = \sqrt{\gamma\, k\, \tilde{\rho}^{\gamma-1}}$$

$$\tilde{u} = \frac{u_L\sqrt{\rho_L} + u_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \quad \tilde{v} = \frac{v_L\sqrt{\rho_L} + v_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \quad \tilde{w} = \frac{w_L\sqrt{\rho_L} + w_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}$$

and has eigenvalues $\lambda_1 = \tilde{u} - \tilde{c}$, $\lambda_{2,3} = \tilde{u}$ and $\lambda_4 = \tilde{u} + \tilde{c}$ and respective eigenvectors

$$\mathbf{K}_1 = \begin{bmatrix} 1 \\ \tilde{u} - \tilde{c} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} \quad \mathbf{K}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \mathbf{K}_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{K}_4 = \begin{bmatrix} 1 \\ \tilde{u} + \tilde{c} \\ \tilde{v} \\ \tilde{w} \end{bmatrix}$$

Each eigenvalue represents a wave. Eigenvalues λ_1 and λ_4 represent non-linear waves (shocks or rarefactions) and define the intermediate star region (fig. 7.1), whereas eigenvalues λ_2, λ_3 represent linear degenerate contact discontinuity waves [2], playing important role for the passive scalar transport. The solution of the local Riemann problem is found at the moving interface between the particles, which will be assumed to move at the average transport velocity ($\mathbf{u}_0$) of the two particles *i*, *j*. Since the particles move with a velocity much lower than the speed of sound the main interest is in finding the solution at the star region of the Riemann problem (fig. 7.1) [4]. This can be done using the jump relations for the four waves [2]:

$$\Delta\mathbf{\Phi} = \sum_1^4 a_i \mathbf{K}_i \tag{7.14}$$

where $\Delta\mathbf{\Phi}$ is the difference between left and right states ($\mathbf{\Phi}_R$-$\mathbf{\Phi}_L$). From the above equation it is possible to calculate the wave strengths a_1, a_2, a_3, a_4 which can then be used to evaluate the field variables at the star region, which will be the solution to the Riemann problem. Starting from the left state (particle *i*), the star values for *u*-velocity component (local) and the density can be found from the following equation:

$$\mathbf{\Phi}_* - \mathbf{\Phi}_L = a_1 \mathbf{K}_1 \tag{7.15}$$

Eq. 7.15 may be used for the calculation of the Riemann problem solution for the rest velocity components, if the interface velocity is less than the velocity at the star region (u_*). In the opposite case, the influence of the respective discontinuity wave must be considered.

Before applying the solution of the Riemann problem in the SPH-ALE system of equations, the solution needs to be transformed back to the global coordinate system, by applying the inverse transformation of eq. 7.11. The local-to-global transformation is the following:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} x_n & x_m & x_o \\ y_n & y_m & y_o \\ z_n & z_m & z_o \end{bmatrix} \begin{bmatrix} u_{loc} \\ v_{loc} \\ w_{loc} \end{bmatrix} \tag{7.16}$$

After the transformation, the solution of the Riemann problem is applied at eq. 7.5-7.9 and the system of PDEs is integrated with an explicit method. The time step used is defined by taking into account the maximum wave speed travelling inside the moving volume dimensions.

$$dt = C_{CFL} \frac{dx}{d\ S_{\max}} \tag{7.17}$$

In the above relation dx is the volume size, C_{CFL} is the courant number ($0<C_{CFL}<1$), S_{max} the maximum wave speed calculated from the interactions of all particles and d is the dimension of the problem, since Toro proves [2], that the stability range in a d-dimensional problem reduces to the $1/d$ of the stability range of the one dimensional problem.

7.3. MUSCL-Hancock 2nd order scheme / Limiter

The procedure for the implementation of the MUSCL scheme is similar to the one described previously in the 2nd order SPH-R method (section 6.2). However it includes one more step called *evolution step*, resembling the MUSCL-Hancock method described by E. Toro [2]. The procedure is the following:

- Forward and backward derivatives are obtained before the Riemann problem solution, using the following equations:

$$\left\langle \nabla \mathbf{\Phi}_i \right\rangle_F = 2 \sum_{X_j^a > X_i^a} \omega_j \left(\mathbf{\Phi}_j - \mathbf{\Phi}_i \right) \nabla W_{ij} \tag{7.18}$$

$$\left\langle \nabla \mathbf{\Phi}_i \right\rangle_B = 2 \sum_{X_j^a < X_i^a} \omega_j \left(\mathbf{\Phi}_j - \mathbf{\Phi}_i \right) \nabla W_{ij} \tag{7.19}$$

- Slope, $\mathbf{\Delta}_i$ and $\mathbf{\Delta}_j$, calculation using forward and backward derivatives:

$$\mathbf{\Delta}_{i,F} = \nabla \mathbf{\Phi}_{i,F/B} \cdot \mathbf{r}_{ij} \text{ and} \tag{7.20}$$

$$\mathbf{\Delta}_{i,B} = \nabla \mathbf{\Phi}_{i,F/B} \cdot \mathbf{r}_{ij} \tag{7.21}$$

- Limiting procedure:

$$\overline{\Delta}_i = \begin{cases} \max[0, \min(\beta \Delta_{i,B}, \Delta_{i,F}), \min(\Delta_{i,B}, \beta \Delta_{i,F})] & \Delta_{i,F} > 0 \\ \min[0, \max(\beta \Delta_{i,B}, \Delta_{i,F}), \max(\Delta_{i,B}, \beta \Delta_{i,F})] & \Delta_{i,F} < 0 \end{cases} \tag{7.22}$$

The β parameter is set to 2, reproducing the SuperBee flux limiter [2].

- Reconstruction of data at the interface between particles:

$$\mathbf{\Phi}_{iL} = \mathbf{\Phi}_i - \tfrac{1}{2}\overline{\mathbf{\Delta}}_i \tag{7.23}$$

$$\mathbf{\Phi}_{iR} = \mathbf{\Phi}_i + \tfrac{1}{2}\overline{\mathbf{\Delta}}_i \tag{7.24}$$

$$\mathbf{\Phi}_{jL} = \mathbf{\Phi}_j - \tfrac{1}{2}\overline{\mathbf{\Delta}}_j \tag{7.25}$$

$$\mathbf{\Phi}_{jR} = \mathbf{\Phi}_j + \tfrac{1}{2}\overline{\mathbf{\Delta}}_j \tag{7.26}$$

- Evolution of $\mathbf{\Phi}_{iR}$ and $\mathbf{\Phi}_{jL}$ at the local one-dimensional Riemann problem:

$$\overline{\mathbf{\Phi}}_{iR} = \mathbf{\Phi}_i + \frac{1}{2}\frac{dt}{dx}\left[\mathbf{F}\left(\mathbf{\Phi}_{iL}\right) - \mathbf{F}\left(\mathbf{\Phi}_{iR}\right)\right] \tag{7.27}$$

$$\overline{\mathbf{\Phi}}_{jL} = \mathbf{\Phi}_j + \frac{1}{2}\frac{dt}{dx}\left[\mathbf{F}\left(\mathbf{\Phi}_{jL}\right) - \mathbf{F}\left(\mathbf{\Phi}_{jR}\right)\right] \tag{7.28}$$

- Solution of the Riemann problem using $\overline{\mathbf{\Phi}}_{iR}$ as the left state and $\overline{\mathbf{\Phi}}_{jL}$ as the right state, instead of using $\mathbf{\Phi}_i$ and $\mathbf{\Phi}_j$.

7.4. Boundary conditions – Partial Riemann problem

The main advantage of the SPH-ALE method is the robustness in dealing with the boundaries [1, 4]. The method is able to take into account different boundary conditions, without the need to extrapolate variable fields beyond the boundary. This is done by considering the boundary surface elements as the interface between two fluid states and using them to calculate the surface integral term of the derivative approximation [4]. Thus, in eq. 7.5-7.9 another summation term is added, in order to account the interactions of a particle i with a surface element j. Equations 7.5-7.9 become:

$$\frac{d}{dt}(\omega_i) = V_{term} + \omega_i \sum_{j\in S} S_j \left(\mathbf{u}_0(\mathbf{r}_j) - \mathbf{u}_0(\mathbf{r}_i)\right)\cdot \mathbf{n}_j W_{ij}$$

$$\frac{d}{dt}(\omega_i\rho_i) + V_{term} + \omega_i \sum_{j\in S} S_j 2\rho_E\left(\mathbf{u}_E - \mathbf{u}_0\right)\cdot \mathbf{n}_j W_{ij} = 0$$

$$\frac{d}{dt}(\omega_i\rho_i u_i) + V_{term} + \omega_i \sum_{j\in S} S_j 2\left[\left[\rho_E u_E\left(u_E - u_0\right) + p_E\right]W_{ij}x_n + \left[\rho_E u_E\left(v_E - v_0\right)\right]W_{ij}y_n + \left[\rho_E u_E\left(w_E - w_0\right)\right]W_{ij}z_n\right] = \omega_i\rho_i g_x$$

$$\frac{d}{dt}(\omega_i\rho_i v_i) + V_{term} + \omega_i \sum_{j\in S} S_j 2\left[\left[\rho_E v_E\left(u_E - u_0\right)\right]W_{ij}x_n + \left[\rho_E v_E\left(v_E - v_0\right) + p_E\right]W_{ij}y_n + \left[\rho_E v_E\left(w_E - w_0\right)\right]W_{ij}z_n\right] = \omega_i\rho_i g_y$$

$$\frac{d}{dt}(\omega_i\rho_i w_i) + V_{term} + \omega_i \sum_{j\in S} S_j 2\left[\left[\rho_E w_E\left(u_E - u_0\right)\right]W_{ij}x_n + \left[\rho_E w_E\left(v_E - v_0\right)\right]W_{ij}y_n + \left[\rho_E w_E\left(w_E - w_0\right) + p_E\right]W_{ij}z_n\right] = \omega_i\rho_i g_z$$

(7.29-7.33)

In all the previous equations V_{term} is the volume term presented in equations 7.5-7.9 respectively. It must be highlighted that in equations 7.29-7.33 S_j represents surface, not volume. Moreover $\mathbf{n}_j$ should not be confused with $\mathbf{n}_{ij}$ used during the interactions between particles, since it is the local

boundary normal pointing away from the fluid region. It is used instead of the $\mathbf{n}_{ij}$ vector to all calculations. The procedure for the interaction of a particle i with a surface element is similar to what was described previously, i.e. the coordinate system must be transformed to a local system, defined now by the $\mathbf{n}_j$ normal vector, the Riemann problem solved and the solution transformed back to the global coordinate system. However, in the Riemann problem only the left state (from particle i) is known. There is no right state, but it can be deduced from the left state and the boundary condition type [4, 5]. For example in the case of a wall boundary condition, it is known [2] that the right and left states have the same density. Furthermore for the right state local u-velocity component, the following relation can be used [2]:

$$u_R = -u_L + 2u_{wall} \tag{7.34}$$

The rest local velocity components (v_{loc}, w_{loc}) are the same for both states.

From the above, it is obvious that surface elements do not need to have any field values by themselves, since the needed data can be deduced from the fluid particles. However they have to be included in the derivative calculation for the 1st step of the MUSCL scheme. Thus it is needed to attribute field values to them too. This is done following Marongiu et al. [4], by using the relation for j-particle's pressure:

$$p_j = \sum_{i=1,N} \omega_i 2 p_E W_{ij} \tag{7.35}$$

Using the equation of state it is possible to obtain density too. For the rest of conserved variables (ρu, ρv, ρw) a similar approach can be used, based on Shepard filtering [6]:

$$(\rho u)_j = \frac{\sum_{i \in \Omega} \omega_i (\rho u)_i W_{ij}}{\sum_{i \in \Omega} \omega_i W_{ij}} \tag{7.36}$$

7.5. Low speed preconditioning

The implementation of preconditioning in SPH-ALE has been introduced by Marongiu et al. [7]. In cases where the SPH-ALE method is used to describe the fluid as nearly incompressible, the fluid velocity is small compared to the, artificially set, speed of sound. It is known [7, 8] that at low Mach numbers the large disparity between the waves traveling with speeds $u+c$, $u-c$ and the wave traveling with speed u, renders the solution of such problems problematic [8]. A way to overcome this is by multiplying the time derivative of the compressible flow equations with the inverse of the preconditioning matrix in order to reduce the speed of acoustic waves to the flow velocity. In this way it is possible to further reduce the numerical viscosity and reduce the noise from the pressure field.

The preconditioner used in this work is the preconditioner of Turkel [7, 8, 9], which in its simplest form is written as:

$$\mathbf{P} = \begin{bmatrix} \beta^2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.37}$$

The β parameter in the preconditioner matrix is set to:

$$\beta = \min\left[1, \max\left(K M_i, K M_j, \beta_{bound}\right)\right], \tag{7.38}$$

with the K parameter set at the same scale as the ratio of the numerical sound speed to the maximum velocity, c_0/u_{max} (usually ~10). In the previous relation M_i represents the Mach number in the local neighbor of particle i. Note that the value of β is bounded by a problem dependent lowest value in order to avoid instabilities at stagnation points [7, 8]. The above preconditioner is written for entropic variables $\mathbf{W}$: $[p, u, v, w]^T$. In order to use it for the conservative variables $\mathbf{W}_c$: $[\rho, \rho u, \rho v, \rho w]^T$ it must be transformed according to the relation [8]:

$$\mathbf{P}_c = \frac{\partial \mathbf{W}_c}{\partial \mathbf{W}} \mathbf{P} \frac{\partial \mathbf{W}}{\partial \mathbf{W}_c} \Rightarrow$$

$$\mathbf{P}_c = \begin{bmatrix} 1/c^2 & 0 & 0 & 0 \\ u/c^2 & \rho & 0 & 0 \\ v/c^2 & 0 & \rho & 0 \\ w/c^2 & 0 & 0 & \rho \end{bmatrix} \cdot \begin{bmatrix} \beta^2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} c^2 & 0 & 0 & 0 \\ -u/\rho & 1/\rho & 0 & 0 \\ -v/\rho & 0 & 1/\rho & 0 \\ -w/\rho & 0 & 0 & 1/\rho \end{bmatrix} = \begin{bmatrix} \beta^2 & 0 & 0 & 0 \\ \beta^2 u - u & 1 & 0 & 0 \\ \beta^2 v - v & 0 & 1 & 0 \\ \beta^2 w - w & 0 & 0 & 1 \end{bmatrix} \tag{7.39}$$

The preconditioning matrix is multiplied with the linearized Jacobian and the resulting eigenvalues and eigenvectors are used to find the intermediate star region, as described in the previous section. The preconditioned Riemann solver helps in reducing the noise in the instantaneous pressure field as it is shown in fig. 7.2, which depicts a 2d jet impingement on a flat plate with a velocity of 20m/s (simulation parameters are the same as with the previous 2d jet impingement with the standard SPH, i.e. particle size 1mm, c_0=230m/s). Also, as it is shown in fig. 7.3, the distribution of the instantaneous pressure coefficient on the flat plate is smoother for the preconditioned scheme with a maximum value of $(C_p)_{max}$=1.01, whereas the non-preconditioned scheme has a maximum value of $(C_p)_{max}$=1.09. The integral of the pressure on the boundary, results to force of F_y= -12019N for the preconditioned scheme, whereas to a force of F_y= -12268N for the non-preconditioned scheme. The normal force to the boundary for the same case can be found through the conservation of momentum and its value is F_y= -11964N (0.5% deviation for the preconditioned scheme, 2.5% deviation for the non-preconditioned scheme). Note that the SPH-ALE results, even without preconditioning, are smoother and closer to the theoretical solution, than the SPH results, no matter using density filter of upwind flux formulation. Another effect is that the preconditioned scheme helps in the reduction of the numerical viscosity.

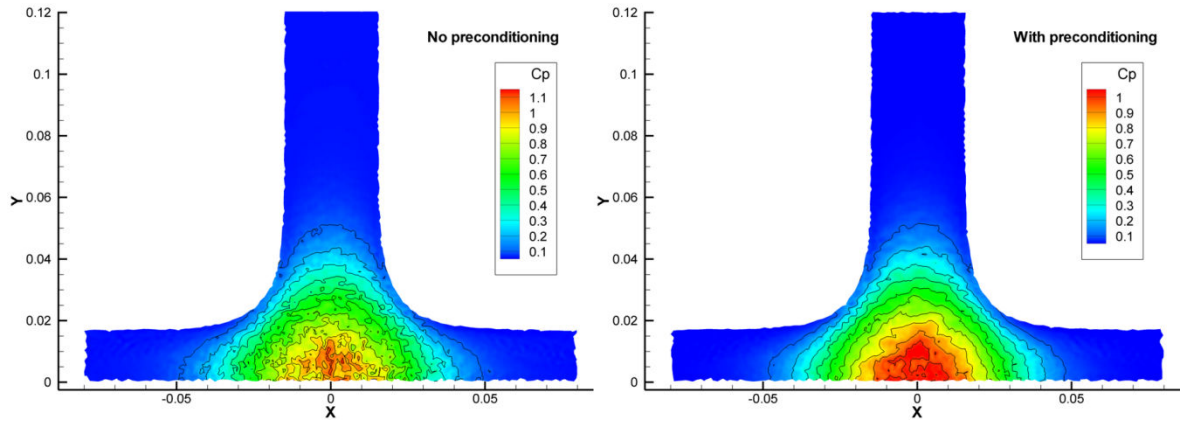


Fig. 7.2. Non-preconditioned (left) versus the preconditioned scheme (right). Pressure coefficient distribution.

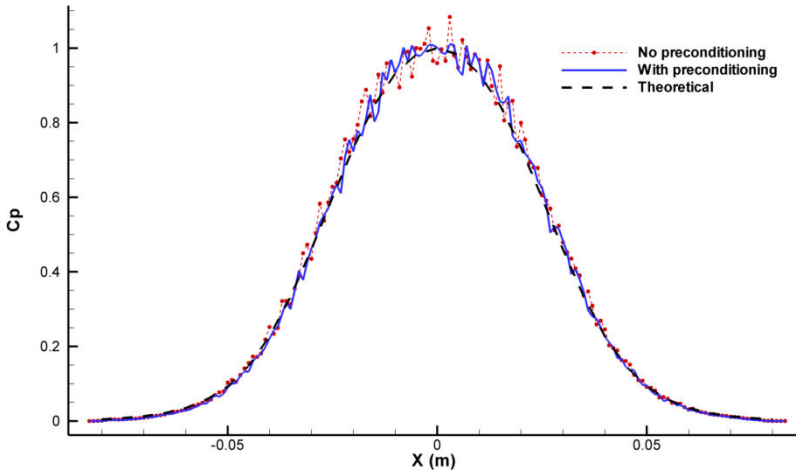


Fig. 7.3. Comparison of the instantaneous pressure coefficient profile on the plate for the two schemes

7.6. Validation of the method

Shock tube

The described 2nd order SPH-ALE method is able to perform well in shock tube tests, without overshoots and/or oscillations near discontinuities (fig. 7.4), in contrast to similar cases in SPH-ALE literature [4], for the same resolution (dx=1mm, 100 particles simulated). The comparison is made between the exact Riemann solver [3] and the 1st and 2nd order SPH-ALE method. The reference density of the fluid was set to 997.04kg/m^3, with a speed of sound c_0=1466.7m/s and γ=7.15. The initial conditions for the shock tube test were:

$$L\begin{cases}\rho = 1100 kg/m^3 \\ U = 0 m/s\end{cases}, x<0 \qquad R\begin{cases}\rho = 1000 kg/m^3 \\ U = 0 m/s\end{cases}, x>0$$

The following results were obtained using Eulerian description, though the Lagrangian description produces similar results (CFL=0.8).

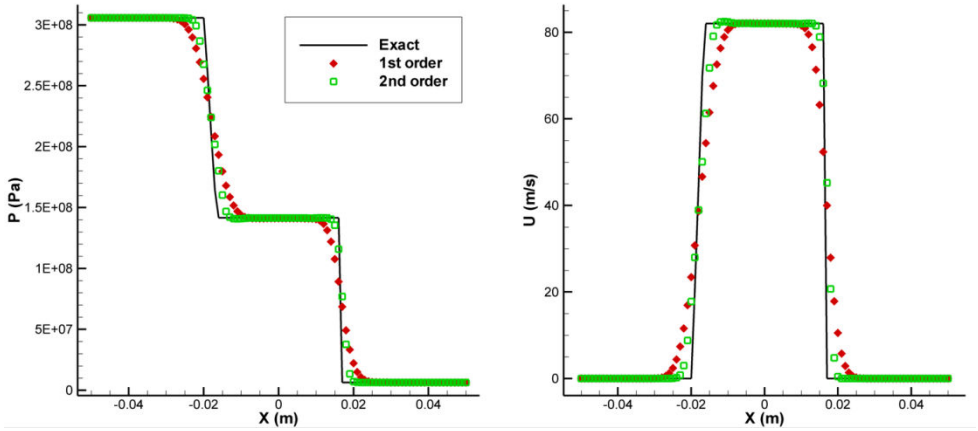


Fig. 7.4. Solution of the Shock tube problem: left, pressure and right, u-velocity component.

Green Taylor vortex

The Green Taylor vortex flow (fig. 7.5) is simulated to measure the numerical viscosity of the described scheme. The Green-Taylor flow is a 2D, periodic flow involving four counter-rotating vortices within one pattern length. For this specific problem there is an analytical solution (eq. 7.40).

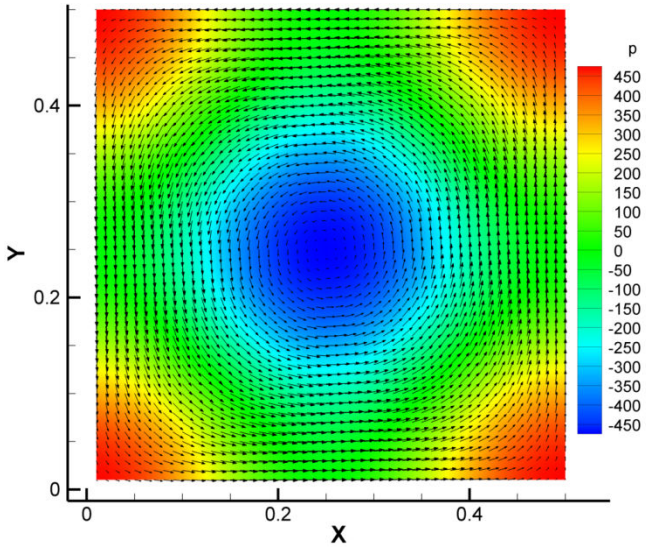


Fig. 7.5. Green Taylor vortex. 1/4th of the periodic pattern.

$$\begin{cases} u = Ue^{-\frac{8\pi^2}{\mathrm{Re}}t}\cos(2\pi y)\sin(2\pi x) \\ v = -Ue^{-\frac{8\pi^2}{\mathrm{Re}}t}\sin(2\pi y)\cos(2\pi x) \\ p = \dfrac{\rho U}{4}e^{-\frac{16\pi^2}{\mathrm{Re}}t}\left[\cos(4\pi y)+\cos(4\pi x)\right] \end{cases} \tag{7.40}$$

By including in the analytical solution the viscosity dissipation, it is possible to determine the amount of the numerical diffusion of the SPH-ALE method, since the method does not include any viscosity terms. Indeed, using a resolution 100x100 for a grid of particles with sizing dx=0.01m and an initial velocity u=1m/s, the present scheme is able to increase the Reynolds number to ~2000 (fig. 7.6), almost four times the Reynolds number in the same case in the literature [7]. The Turkel preconditioner further decreases numerical viscosity, eventually leading to a Reynolds number of ~3500.

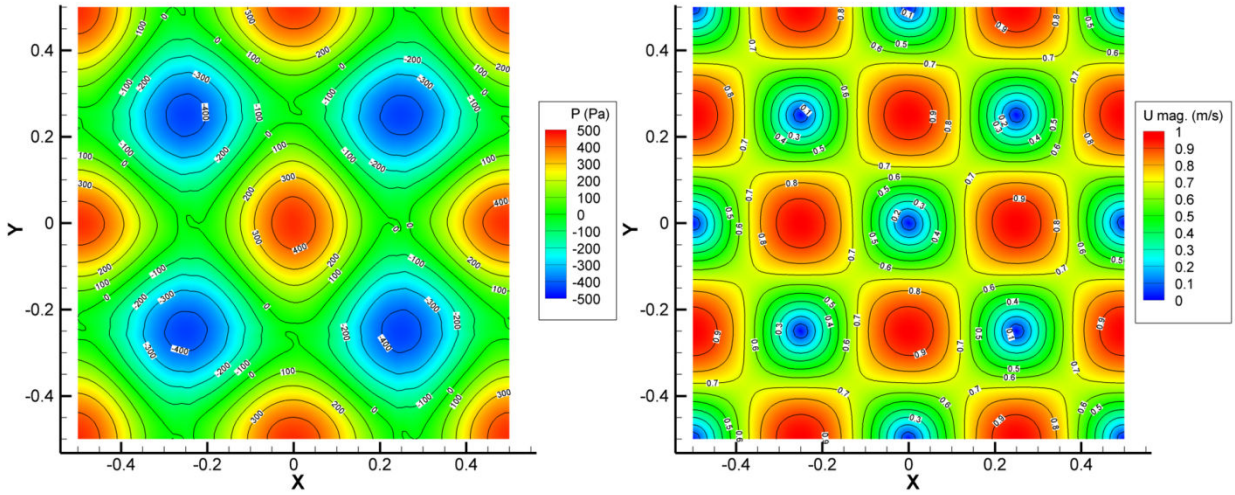


Fig. 7.6. Solution of the Green Taylor Vortex problem, pressure (left) and velocity magnitude (right) distribution. Results are after 1s of simulation, using Eulerian description.

2D explosion/implosion

Another case, which was used as a benchmark for the SPH-ALE method, was the calculation of a 2-dimensional explosion and implosion. The explosion was simulated using a 200x200 grid of particles with size 0.05m. Fluid parameters were set exactly as in the shock tube case, i.e. reference density of the fluid was set to 997.04kg/m^3, with a speed of sound c_0=1466.7m/s and γ=7.15. For the explosion case, the density inside a circle with radius of 1m, placed at the center of the grid, was set to 1100kg/m^3, whereas at the rest area to 1000kg/m^3. During the simulation a shock wave is generated, traveling outwards, whereas a rarefaction wave moves towards the center of the grid.

Indicative results of the simulation are shown in fig. 7.7 at 0.25ms. The circular wave pattern is properly reproduced. Also in fig. 7.8 the distribution of density and radial velocity are shown, calculated by the 2nd order SPH-ALE method. Also for comparison the solution of the 2nd order 1d axis-symmetric FV method, using 1000 finite volumes in the radial direction, and the 1st and 2nd order 2d FV method, using 200x200 finite volumes, are shown. The 2nd order SPH-ALE solution and the 2nd order FV solution have similar behavior, with the SPH-ALE solution being slightly more diffusive, due to the increased interaction radius. In any case the SPH-ALE solution is close to the high resolution 2nd order 1d FV solution.

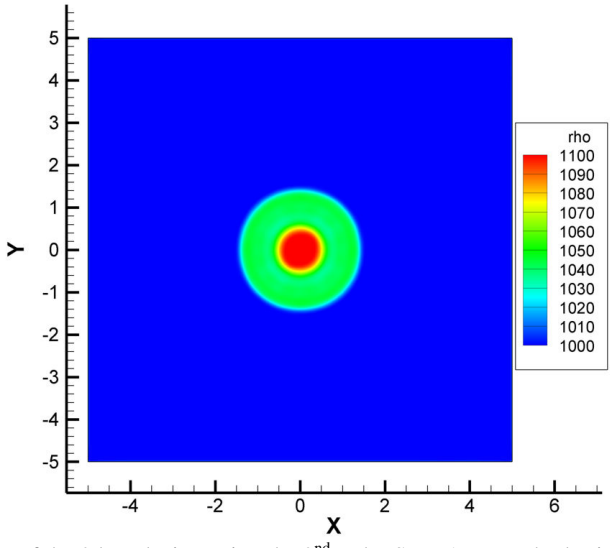


Fig. 7.7. Solution of the 2d explosion using the 2nd order SPH-ALE method. Time:0.25ms.

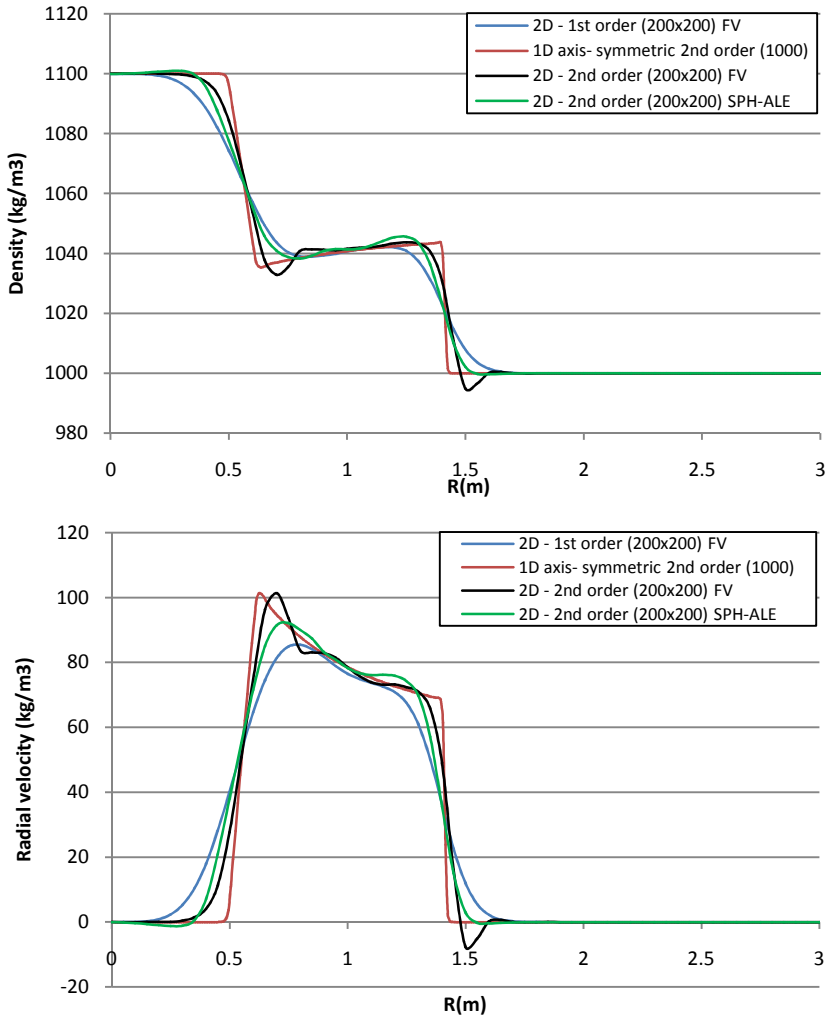


Fig. 7.8. Solution of the 2d explosion: density (up) and radial velocity (down). Comparison with the FV solution.

The implosion problem has the opposite set-up; inside a circle with radius of 1m, placed at the center of the grid, the density was set at 900kg/m^3, whereas at all the rest area density was set at 1000kg/m^3. In order to compare convergence, several resolutions are used: 200x200, 250x250 and 333x333 (equivalent particle sizes 0.05, 0.04 and 0.03m). Initially a shock wave is generated, moving towards the center of the low density region, and a rarefaction wave at the edge of the circle defining the low density region. As the simulation proceeds, the shock wave focuses at the center of the low density region, creating a high density region. A new shock wave is generated expanding outwards, following the rarefaction wave.

The SPH-ALE solution is shown in fig. 7.9 at 0.5ms, while the shock wave travels towards the center of the low density region, and at 1.25ms, after the shock wave focusing. Results are compared to the 1d axis-symmetric, 2nd order FV solution. For the latter solution, 1000 computational volumes are used, resulting to approximately 10 times higher resolution at the radial direction. From the comparison of the solution between the two methods in fig. 7.10 and fig. 7.11, it is obvious that both methods produce similar results. The SPH-ALE method calculates correctly the wave speeds, and, with proper resolution, is able to reproduce high gradient areas (fig. 7.10).

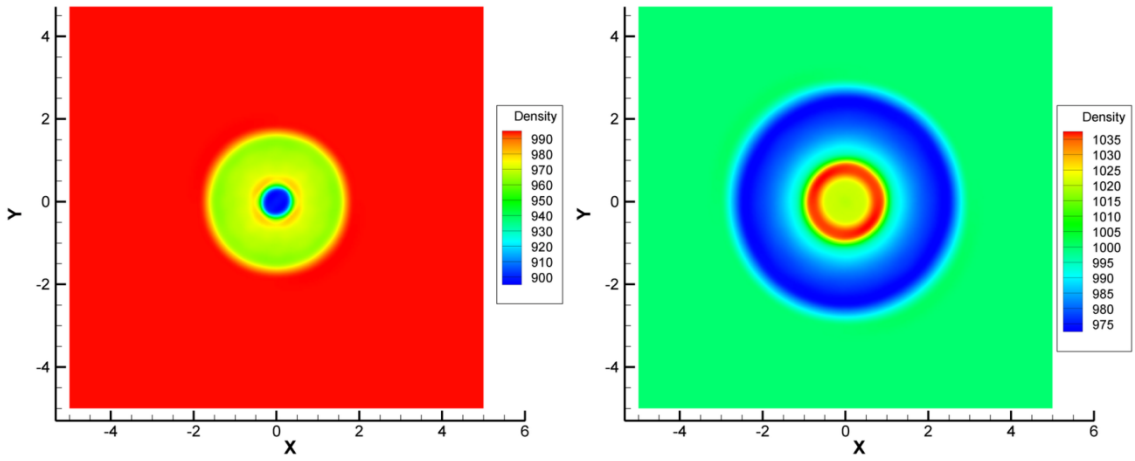


Fig. 7.9. Implosion calculated using the 2nd order SPH-ALE method. Left: Implosion at 0.5msec, shock wave moving towards the center of the low density area. Right: Implosion at 1.25msec, after shock wave focusing, shock wave is expanding.

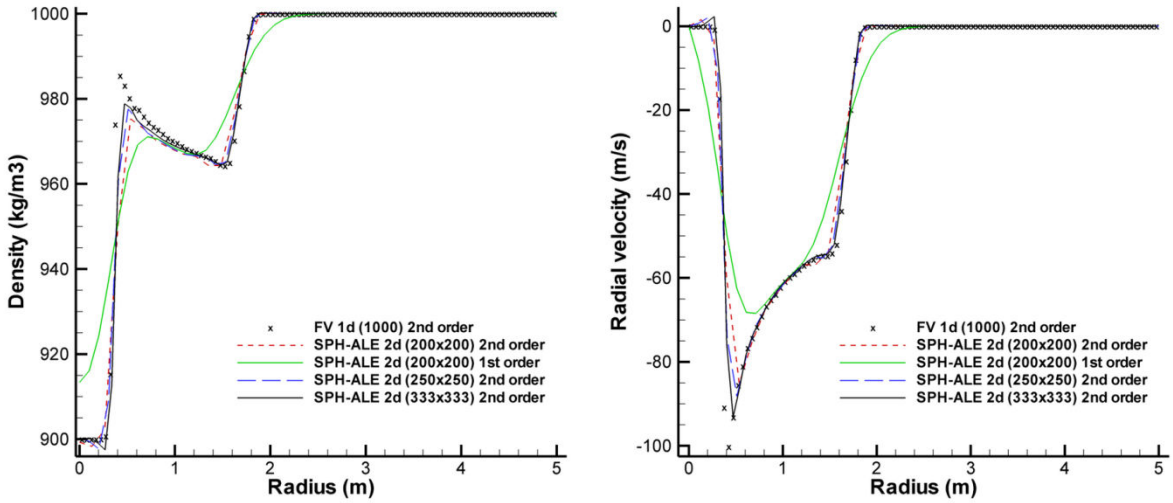


Fig. 7.10. Implosion at 0.5ms. Comparison of: left, calculated density and right, radial velocity

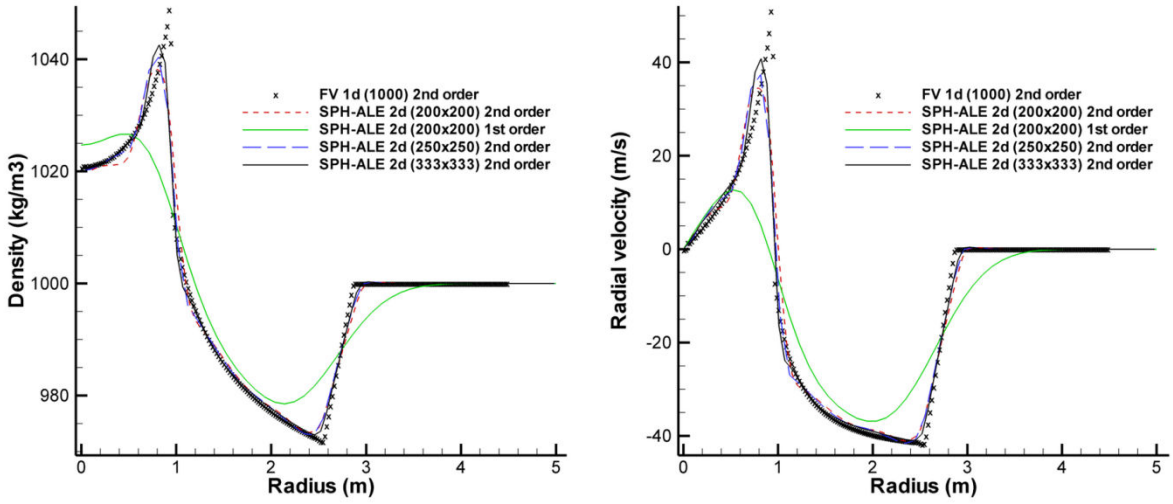


Fig. 7.11. Implosion at 1.25ms. Comparison of: left, calculated density and right, radial velocity

The simulations described above are handled easily with the SPH-ALE method due to the ALE perspective; indeed the used of the SPH or SPH-R methods would be problematic, due to induced

errors from the particle Lagrangian motion. Moreover the simulation of the Green-Taylor vortex is impossible using a pure Lagrangian approach, due to particle alignment and clumping at the stagnation points.

For both explosion-implosion problems, the axis-symmetric solution was obtained by solving the 2^{nd} order 1-dimensional FV problem using the MUSCL-Hancock scheme and considering appropriate source terms. Also the 2d FV solution was obtained using the *unsplit* MUSCL-Hancock finite volume approach [1]. A more detailed description is in appendix A (for more information, see Toro [1]).

7.7. Hydrodynamic problems

In this part several practical applications of the SPH-ALE method will be presented. These applications involve:

- simulation of wedge impacts on the water surface along with comparison with experimental data from Yettou et al. [10]
- simulation of the 3d jet impingement
- simulation of the flow at stationary impulse turbine geometries (Turgo blade and Pelton bucket)

Wedge impact on water surface

Boat hull impacts on the water surface last only several milliseconds [10], during which pressure distribution on the hull surface changes rapidly. Estimation of the forces on the hull is crucial to determine the hull motion and the loads which the hull should withstand. There are several ways of analyzing the impact, ranging from a simplified 2d analysis using potential flow, to a full 3d analysis of the complete hull. Lewis et al. [11, 12] suggested solving consecutive slices of the ship hull in 2D, simplified as wedges of appropriate geometry (while the same logic was used by other researchers too [13]). This method has the advantage of solving several simplified 2D problems using Navier Stokes equations, instead of solving a considerably time consuming 3D problem.

The SPH-ALE method is suited for such a kind of flows, since there is no need for a mesh and due to the Lagrangian description, the computational elements follow the flow features, adapting to the flow patterns naturally. Furthermore, it is able to handle moving geometries, without needing special mesh treatments. Moreover the method does not require any free surface tracking algorithm. All the above characteristics of the SPH-ALE method render it an attractive alternative to the traditional mesh based methods. In this part several simulations of wedge impacts will be presented, in conjunction with experimental data from Yettou et al. [10].

The problem is very well described in [10, 11]. A wedge is dropped from a specified height on the water surface. The primary interest is the forces and the wedge motion after the impact. Several test configurations have been examined and the basic experimental characteristics are the following:

- Wedge square top has a section of 1.2m× 1.2m in all cases
- The deadrise angle θ (see also fig. 7.12) of the wedges ranges from 15^o to 35^o degrees, with a interval angle of 5^o degrees. However, experimental data are provided only for the 15^o, 20^o and 25^o angles.
- The wedge impacts the surface after being released from a height of 1 or 1.3m.
- Adjustable wedge mass ranging from 89 to 158kg.

The wedge is equipped with pressure sensors on its side, positioned every 50mm (fig. 7.12).

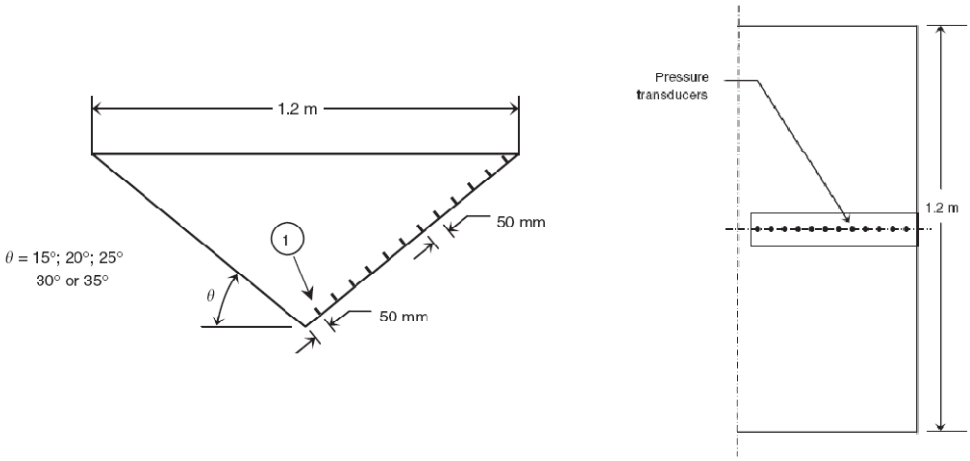


Fig. 7.12. Wedge schematic [10]

The problem was modeled using the SPH-ALE method, using symmetry boundary conditions at the *y*-axis (*x*=0). Before the impact the wedge was assumed to be influenced only by gravity. Thus, the wedge is initially positioned just above the water free surface with an impact velocity of $\sim\sqrt{2gH}$. During the impact, the wedge motion is governed by gravity and pressure forces. The problem is a fluid-rigid body interaction, were the only degree of freedom is the wedge velocity/motion on the vertical axis. The acceleration of the wedge is given by:

$$a_y = g_y + \frac{F_p}{m_{wedge}} \tag{7.40}$$

The simulated medium was water with a density of 998.2kg/m^3 and the speed of sound was set to c_0=100m/s.

Particle dependence study

Three different particle resolutions were used, in order to determine the effect of the particle size in the simulation results. The particles size resolution was increased by a factor of two; particle sizes 10mm, 5mm and 2.5mm, involving 15000, 57000 and 220000 particles, respectively.

In fig. 7.13 the wedge *v*-velocity and the vertical slamming force, are shown. From these two graphs it becomes apparent that the intermediate and fine particle resolutions are able to reproduce practically the same macroscopic results, regarding the wedge motion. Deviation of the wedge vertical velocity between these two resolutions at the end of the simulation is less than 1%. On the other hand the coarse resolution (10mm particle size) calculates the same trends but deviates by ~5%. Some differences are found in the vertical force among the different particle resolutions, but generally in all cases, the same behavior is reproduced.

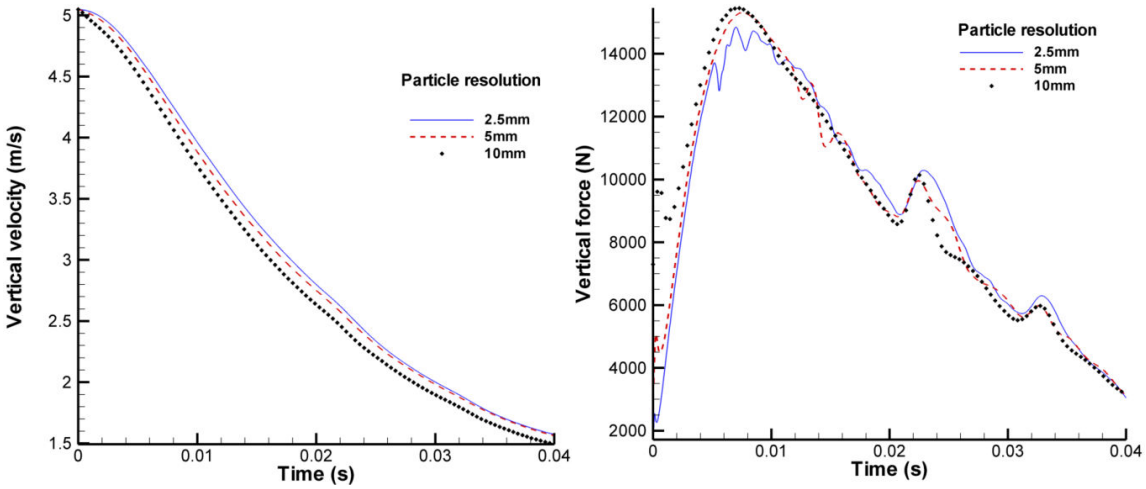


Fig. 7.13. Vertical velocity (left) and vertical force (right) during the impact for different particle resolutions.

In fig. 7.14, an indicative view of the wedge slamming is shown, towards the end of the simulation. Even if the macroscopic results of the wedge motion are accurately predicted, some fine details are not properly captured; the water sheet formed on the wedge side is not predicted unless a fine resolution is used. However, even in that case, the resolution of the water sheet might not be adequate, since its thickness is described by ~4 particles.

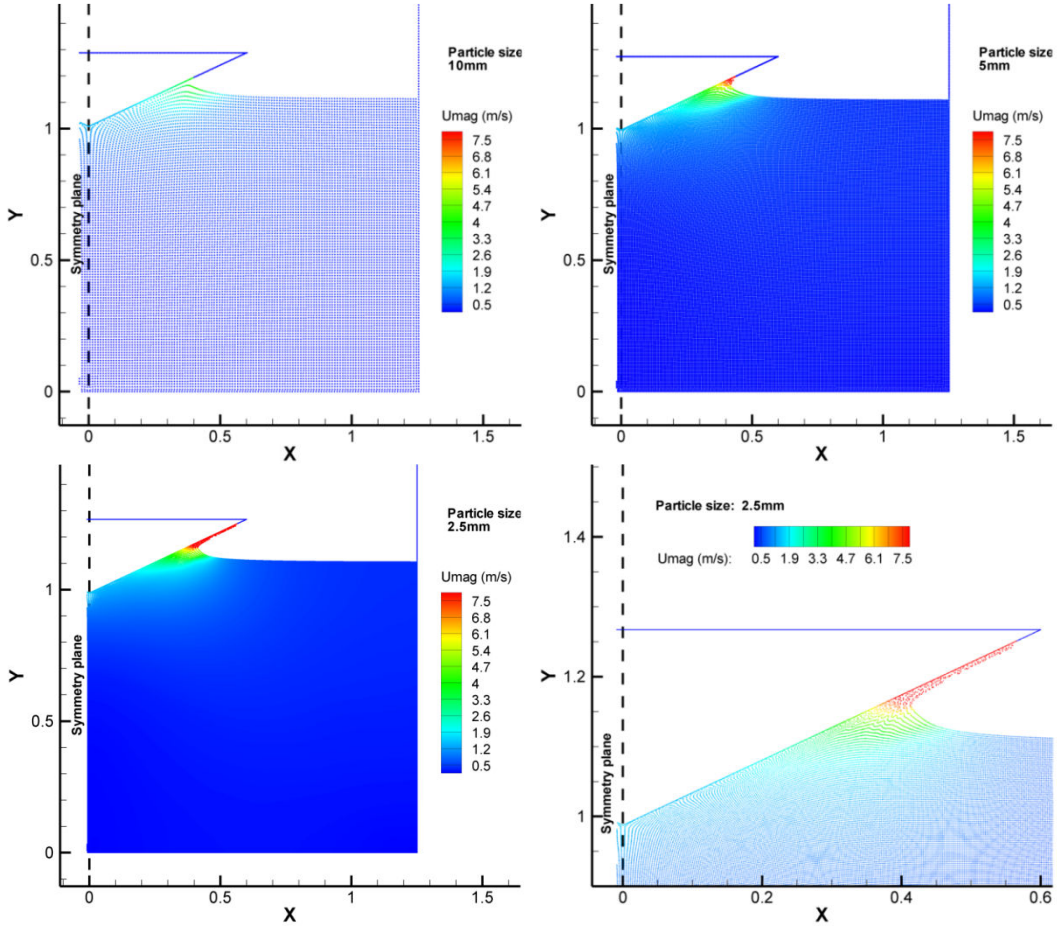


Fig. 7.14. Indicative results of the wedge slamming for different particle resolutions. The image at the down right corner shows a close view of the water sheet on the wedge side. Particle coloring according to velocity magnitude.

Comparison of the results with the experimental/literature data

In fig. 7.15 the vertical velocity of the wedge is shown, in comparison to experimental data [10], mesh based CFD results [11] and an empirical model, Zhao's model [10], from the literature. A notable remark is that SPH-ALE does not reproduce accurately the wedge velocity from 13ms to 25ms. The fact that this discrepancy is found in Lewis' CFD results [11] too, confirms that it might be caused by 3D effects, which is impossible to capture in a 2D simulation. Indeed, such discrepancies have been experienced by other researchers too [14]. Nevertheless, numerical results are close and overall agreement is good. The SPH-ALE method is also close to the analytical model of Zhao, with the exception of a slight velocity underestimation, towards the end of the simulation.

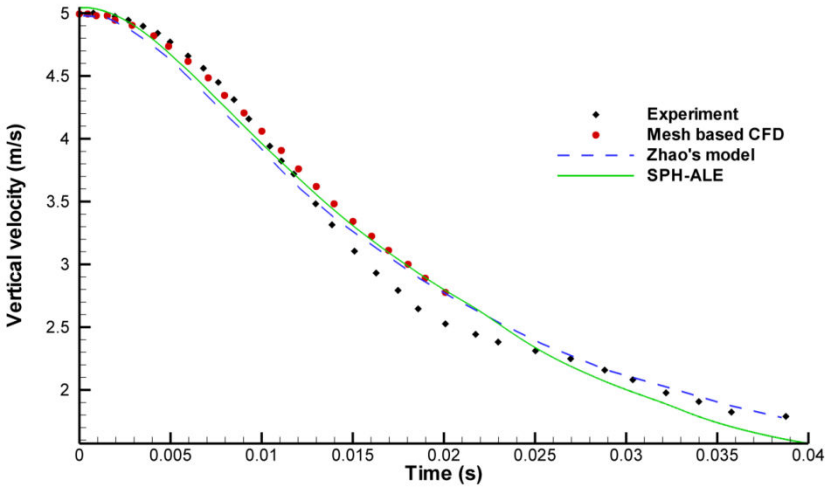


Fig. 7.15. Wedge vertical velocity.

Yettou et al. [10] provided the vertical velocity of the wedge from experimental data and Zhao's model, for several more wedge impacts:

1. Wedge angle 25°, drop height 1.3m, wedge mass 130kg
2. Wedge angle 25°, drop height 1m, wedge mass 130kg
3. Wedge angle 20°, drop height 1.3m, wedge mass 89kg
4. Wedge angle 20°, drop height 1.3m, wedge mass 143kg
5. Wedge angle 15°, drop height 1.3m, wedge mass 143kg

These cases have been simulated using a uniform particle resolution of 5mm, since it was found to be adequate to describe the macroscopic wedge motion. In the following figures (fig. 7.16-7.18) the vertical velocity in respect to time is shown along with the experimental results and Zhao's model results for each case respectively. The SPH-ALE method is able to reproduce properly the velocity deceleration during the impact, for all conditions tested.

Figure 7.15 shows the results for cases 1 and 2. The two cases are identical apart from the drop height, which eventually results to a different impact velocity. Even if the initial impact velocity is different, the final wedge velocity at the ending of the simulation is approximately equal. Figure 7.16 shows the results for case 3 and 4, where the influence of the wedge mass is examined. As it is expected, the increased mass of the wedge results to increased wedge inertia and eventually slower deceleration. Figure 7.17 shows the results for case 4 and 5, where the influence of the wedge angle is examined. Case 5 conditions are the same with case 4, apart from the wedge deadrise angle, which has been reduced to 15°. A smaller wedge angle results to a faster deceleration due to the increased drag.

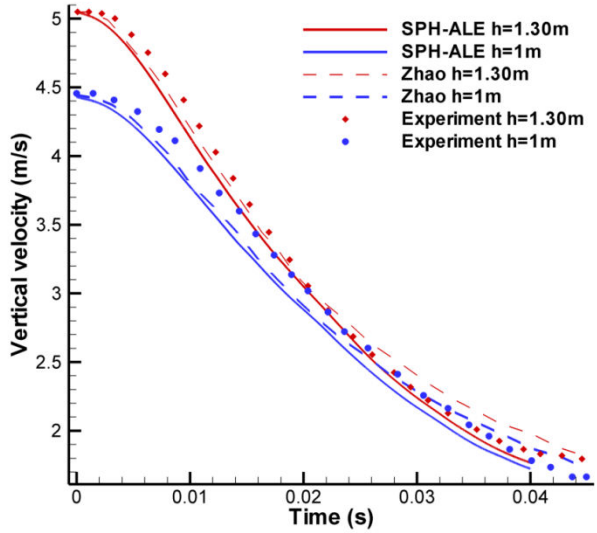


Fig. 7.16. Wedge vertical velocity, case 1 and 2: Different drop heights.

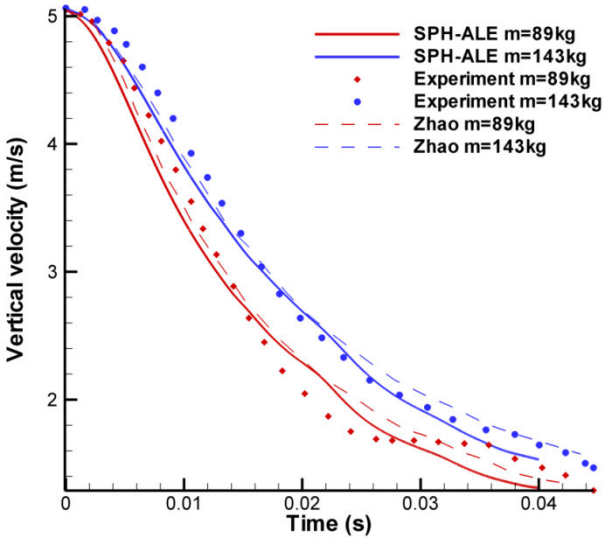


Fig. 7.17. Wedge vertical velocity, case 3 and 4: Different masses.

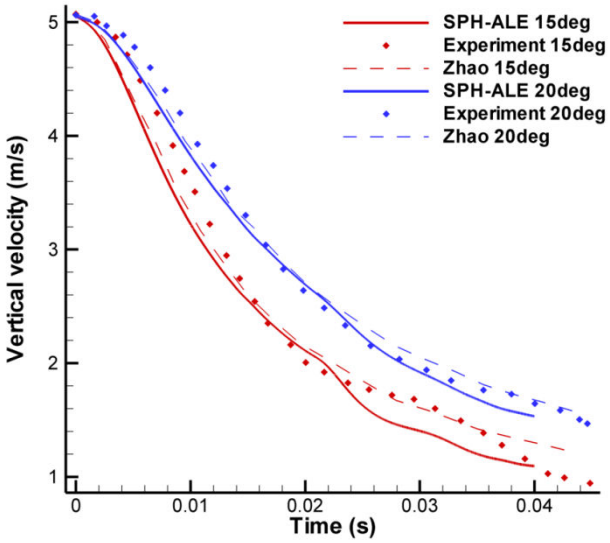


Fig. 7.18. Wedge vertical velocity, case 4 and 5: Different wedge angles.

Variable particle resolution

Further refining resolution uniformly to a particle size of 1.25mm (which is half of the fine resolution used at the particle dependence analysis) would result to a particle number of ~850000particles. This would result to a very cumbersome simulation, where the increased resolution would be redundant in most areas, since most of these particles would be far from the impact zone. Thus, it is beneficial to use a variable resolution in order to capture the fine details near the impact zone, while keeping the overall particle number to a minimum.

The SPH method is able to handle variable resolution, by adjusting properly the particle size and the smoothing length [15, 16]. In the present work, two different particle distributions have been tested. One resembles the distribution used by Oger et al. [15], where particle resolution follows a smooth transition. The other is a telescopic refinement, where particles near the impact zone are split to four smaller particles [17]. The smoothing length in all cases is assumed to be constant in time and equal to 1.4*dx* where *dx* is the particle size. In both cases particle interactions are kept symmetric using the average value of the smoothing length of the interacting particles, i.e.:

$$h_{ij} = \frac{h_i + h_j}{2} \tag{7.41}$$

and this averaged smoothing length is used for the kernel function and its derivative calculation.

For the validation of the multi-resolution implementation of the algorithm, a simulation was performed using both variable resolution methods, in the wedge impact case used for the particle dependence analysis (m=94kg, θ=25°, h=1.3m). For the smooth transition of particle resolution, the particle size at the point of impact was 2.5mm and near the simulated water tank boundary 5mm. Particle size was increased following a geometric sequence with ratio of 1.002. For the telescopic refinement case, all particles had initially a size of 5mm. Then particles lying within a square near the impact point were split in four, eventually leading to a resolution of 2.5mm at the specific zone. Results were compared with the uniform fine resolution (particle size of 2.5mm everywhere). In fig. 7.19 an indicative view of the particle distribution for both cases is shown. Here it must be highlighted that both refinements lead to a total number of particles ~80000, whereas the uniform particle distribution leads to 220000 particles.

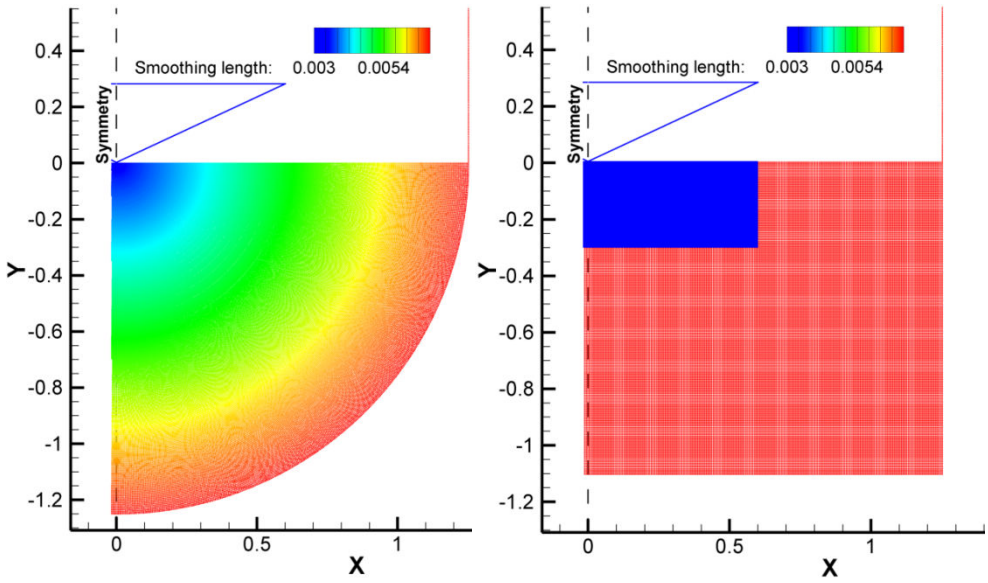


Fig. 7.19. Left: Smooth transition. Right: telescopic refinement.

Macroscopic results (i.e. velocity and forces) of the simulations are similar (fig. 7.20). However it is important to highlight that the telescopic refinement causes some artifacts due to the sudden increase in particle resolution. As shown in fig. 7.21, pressure waves are reflected at the transition between the low and high resolution regions, which is behaving as a semi-permeable interface.

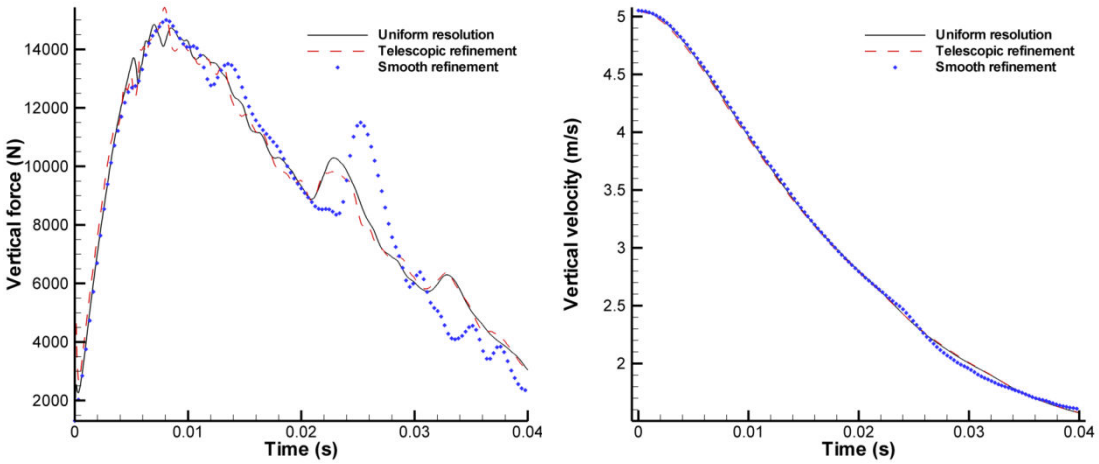


Fig. 7.20. Left: Vertical force. Right: Vertical velocity.

The above observations are explained considering Oger et al. [15] work. Indeed, when variable smoothing length is used, then more terms have to be considered in the derivative calculation formula, such as the gradient of the smoothing length (∇h) and the derivative of the smoothing length in respect to time (dh_i/dt). The derivative of the smoothing length in respect to time is zero, since it does not change throughout the simulation. However, the smoothing length gradient is not zero, and since it was not considered in the SPH approximations used, artifacts appear in areas of sudden smoothing length changes. On the other hand, using a smooth transition minimizes the significance of the gradient term, reducing its effects and enabling the benefits of the particle refinement near the free surface with good accuracy.

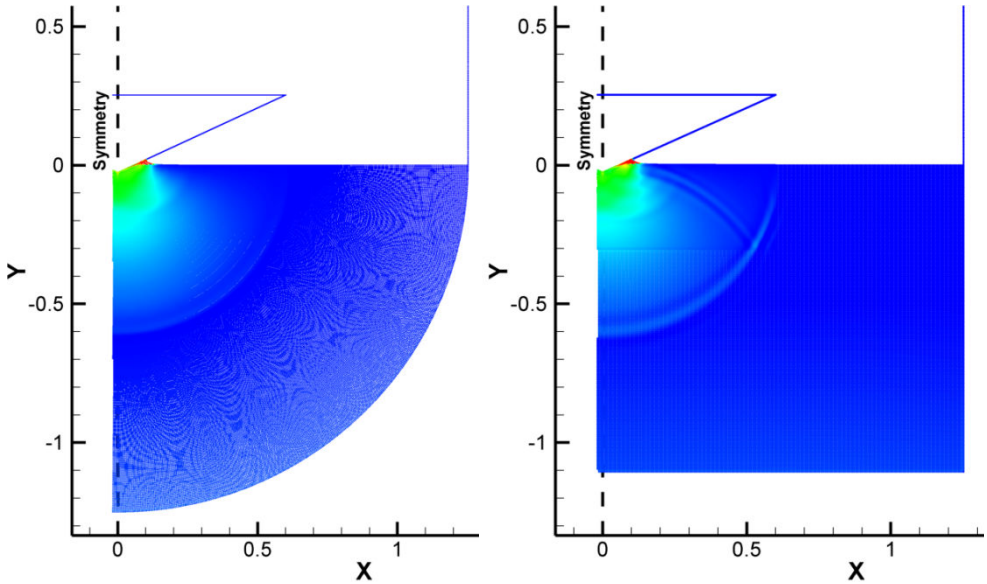


Fig. 7.21. Left: Smooth transition. Right: telescopic refinement. Pressure field. Note the pressure wave reflection at the telescopic refinement.

Final results using finer and variable resolution

The previous case was reconsidered using an even higher resolution near the impact point (dx=1.25mm). Again near the wall the particle resolution was 5mm (growth ratio 1.003). This ratio is small enough to prevent unphysical artifacts. Oger et al. [15] have found that up to a maximum ratio of 1.03, unphysical results are prevented. In fig. 7.22 indicative results of the simulation are shown at the end of the simulation: the simulation using refinement is able to capture better the water sheet evolution as long as the details of the highly curved region. Moreover, the simulation with the uniform resolution has about twice particles (~220000) than the simulation with refinement (~120000).

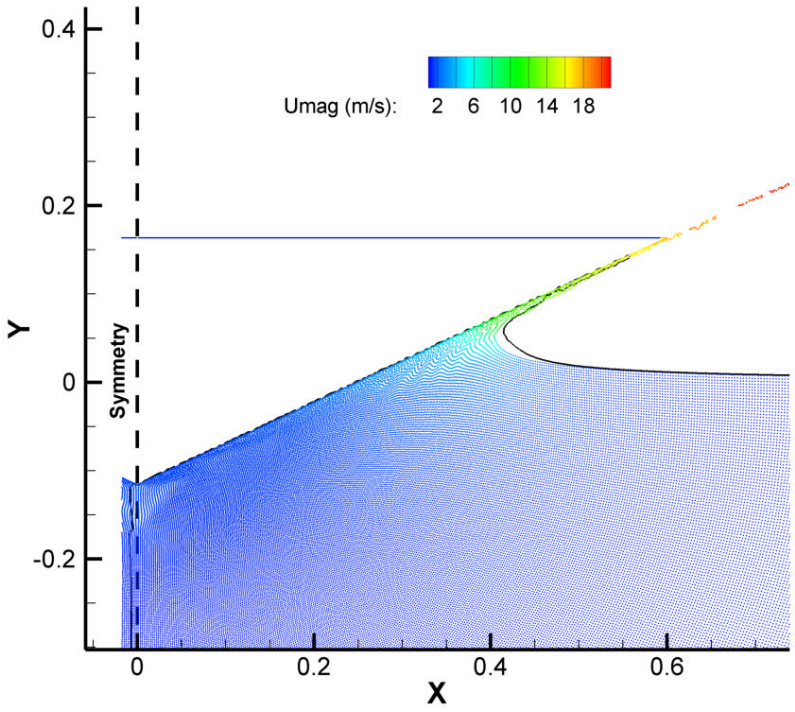


Fig. 7.22. Results of the refined simulation (particles). The free surface calculated with the uniform resolution is visible as a continuous line.

In fig. 7.23 the results of the pressure history on the sensors are shown. At all pressure transducers, a pressure peak appears at the time of contact of the specific area with the water surface. Since pressure sensors do not come to contact with water all together, but sequentially, one after the other, it is expected to have a lag in the pressure peak. After the peak, pressure gradually reduces. The simulation with refinement gives results which exhibit some scattering, but tends to predict better the general trend of the pressure history, especially for the last sensors.

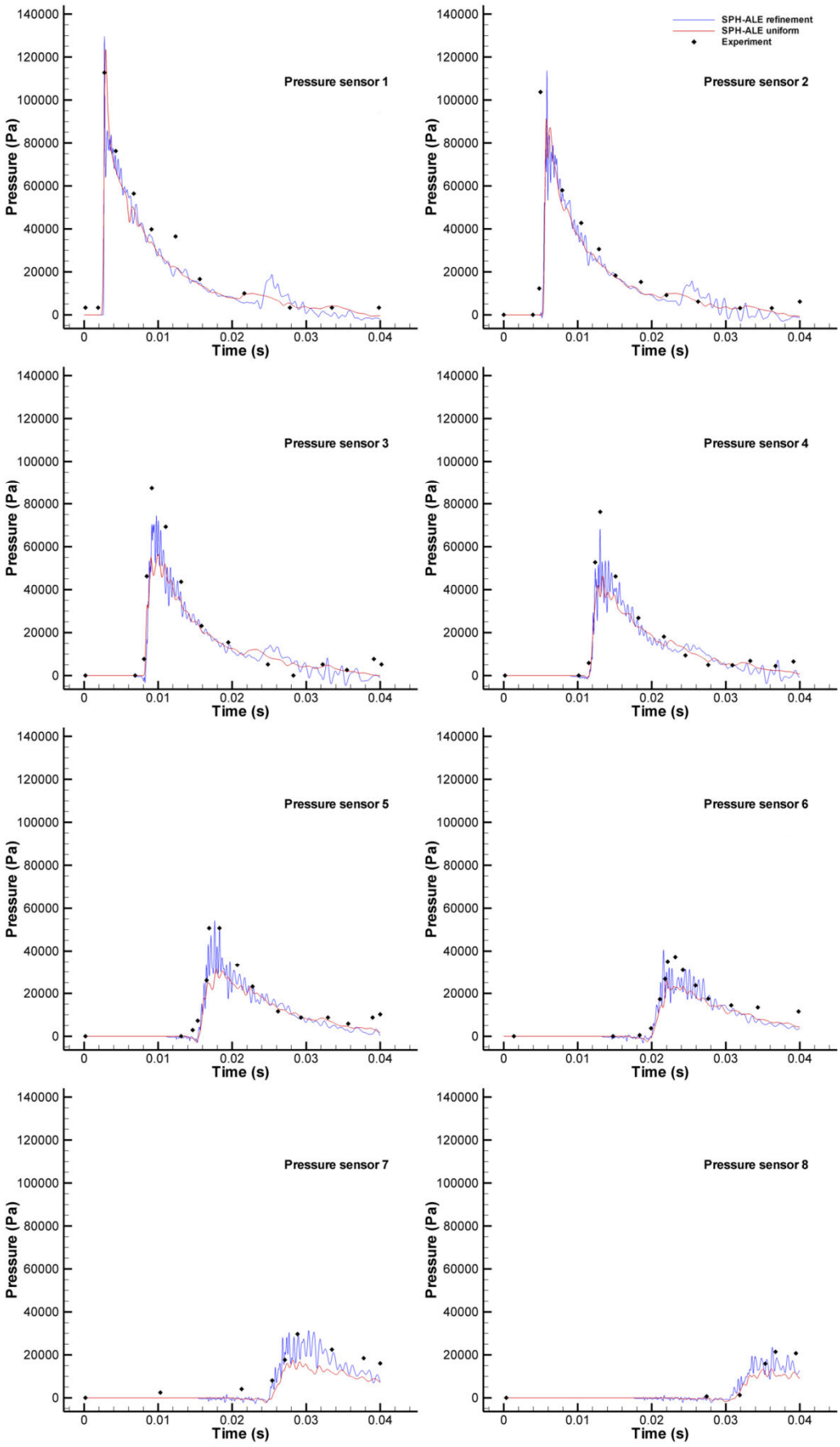


Fig. 7.23. Pressure history at the pressure sensors. Results for uniform and refined resolutions.

3D jet impingement

A 3D jet impingement was also simulated using the SPH-ALE method. The simulation was performed for the same conditions as the studied SPH simulation in section 4.3 and for various impingement angles, in order to compare the results. First of all, in fig. 7.24 and 7.25 a general view of the flow is shown for the 90° and 45° jet impingement using the 2nd order SPH-ALE methods. The velocity and pressure distribution is shown at the YZ symmetry plane; fields in the SPH-ALE method are much smoother than those of the traditional SPH method. Also the effects of the numerical viscosity on the SPH-ALE method are observable, even when using the 2nd order MUSCL scheme; indeed the velocity of the water sheet formed after the impingement is lower than the velocity of the water jet impinging on the plate. The situation is even worse when using the 1st order SPH-ALE method, since excessive numerical viscosity leads to great underestimation of the formed water sheet velocity (by about 45%). However underestimation in the water jet velocity happens also with the SPH method, in areas of few particles, where the SPH particle approximations tend to degrade.

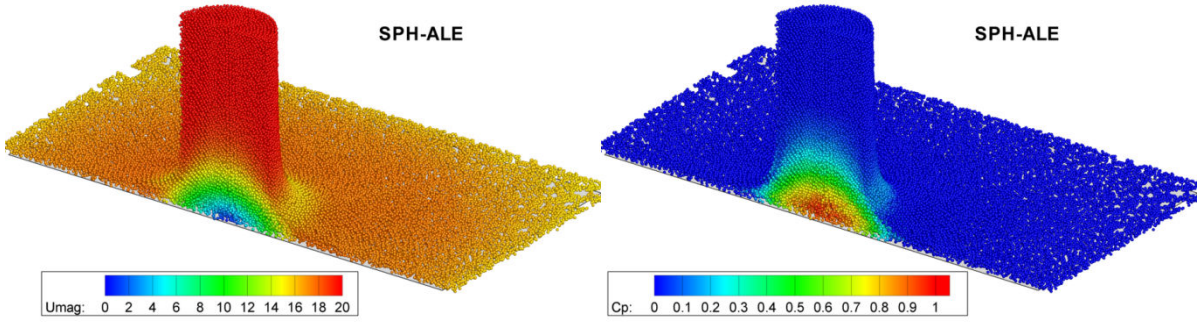


Fig. 7.24. 90° jet impingement. Left: Velocity magnitude distribution. Right: pressure coefficient distribution.

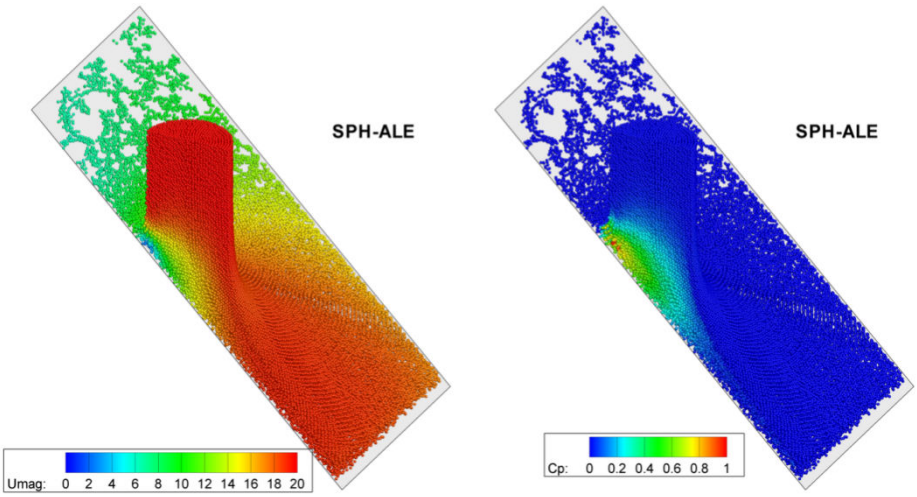


Fig. 7.25. 45° jet impingement. Left: Velocity distribution. Right: pressure coefficient distribution.

Regarding the free-surface location, the 2nd order SPH-ALE method is able to capture accurately the free surface (fig. 7.26). The 2nd order SPH-ALE method slightly overestimates the water layer thickness due to the inherent numerical viscosity, as observed in [4] too, but the difference with the experimentally found free-surface is negligible. On the other hand, the 1st order Godunov method greatly overestimates the free surface thickness.

In fig. 7.26 and 7.27 the pressure coefficient distribution on the flat plate is shown. Here it must be highlighted that the SPH-ALE results are instantaneous, but the standard SPH results are time averaged. The 2nd order SPH-ALE method accurately predicts the pressure distribution. Only in the

case of the 30° jet impingement the 2nd order SPH-ALE method slightly underestimates the pressure near the stagnation point, but this is mainly attributed to the resolution used. In this case, the stagnation point occurs at a very narrow region (fig. 7.27), which would need a very fine resolution to capture accurately. Nevertheless the pressure integral on the plate calculated using Kvicinsky's [18] results and the 2nd order SPH-ALE results differs by ~0.7%.

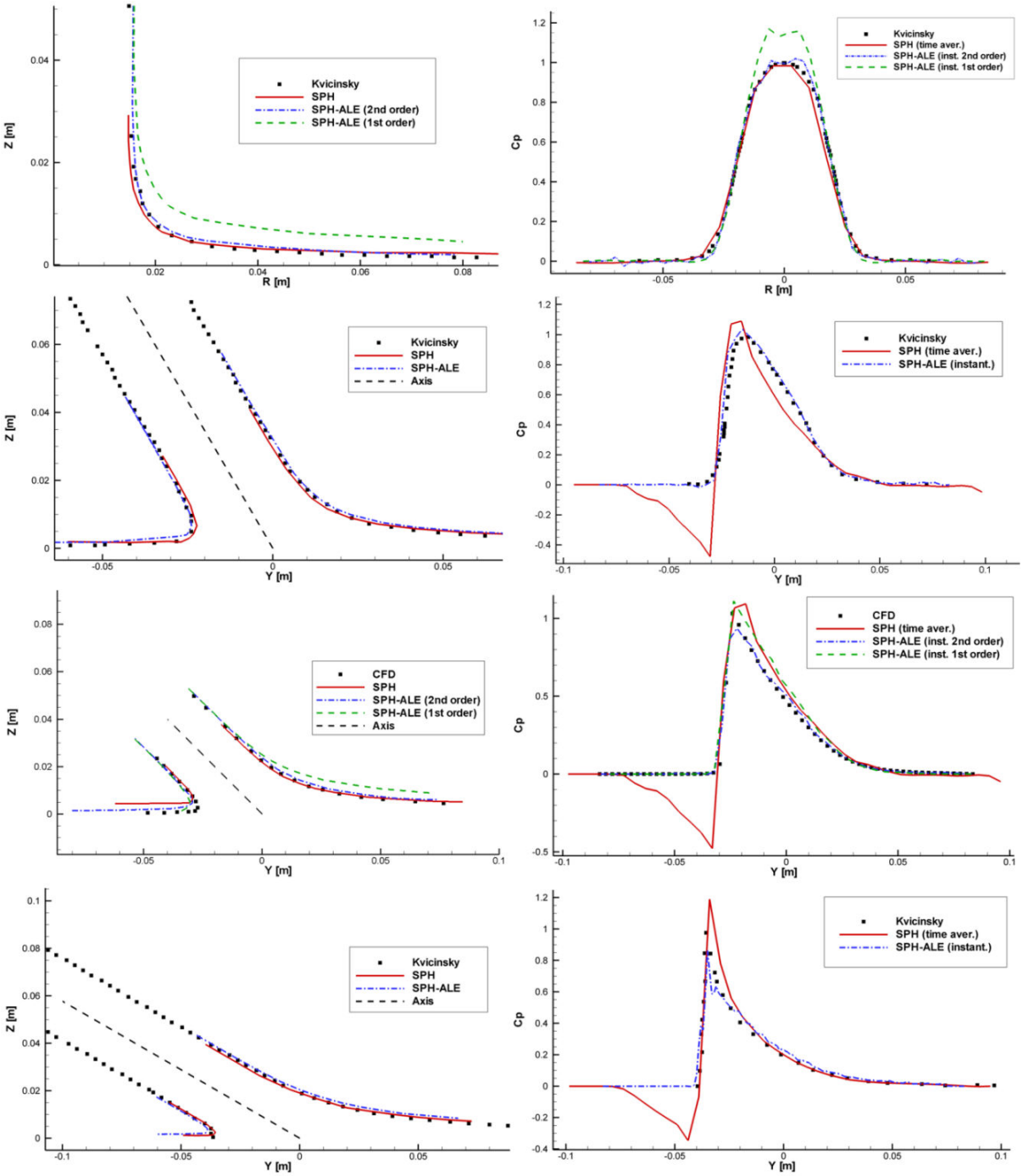


Fig. 7.26. Comparison between SPH and SPH-ALE methods. Left: Free surface location. Right: pressure coefficient distribution for X = 0. From top to bottom: 90°, 60°, 45° and 30° jet impingement

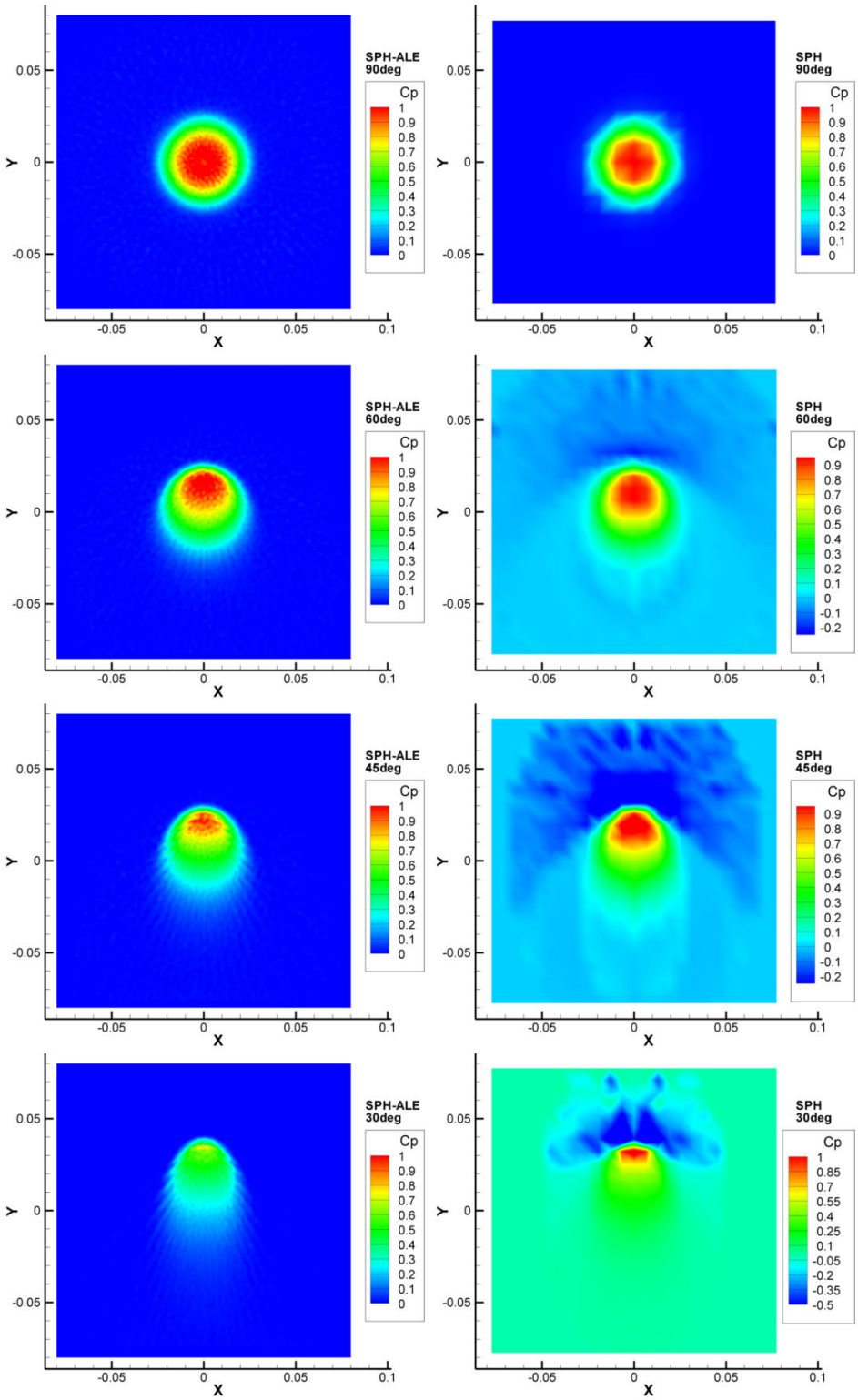


Fig. 7.27. Comparison of the pressure coefficient maps on the plate surface. Left: Instantaneous SPH-ALE. Right: time averaged SPH. From top to bottom: 90°, 60°, 45° and 30° jet impingement

3D interaction with Turgo blade and Pelton bucket

As a final application, the 2nd order SPH-ALE method was tested in impulse turbine geometries, at impingements on the stationary Turgo blade (30° impingement) and Pelton turbine bucket (90° impingement). It must be highlighted here that in order to prevent particle alignment issues, particles were given a random shift, with maximum amplitude equal to *dx/3* (*dx* being the inter-particle spacing) from their normal positions. In fig. 7.28 the general view of the flow is shown for the two impingement cases. Again the effects of numerical viscosity are visible, since velocity of the outflowing water sheets is ~85% of the water jet velocity and in certain areas even lees. Similar performance has been found by Marongiu et al. [7]. Also comparing to the 2nd order SPH-R solution, it appears that the numerical viscosity is larger in the SPH-ALE method.

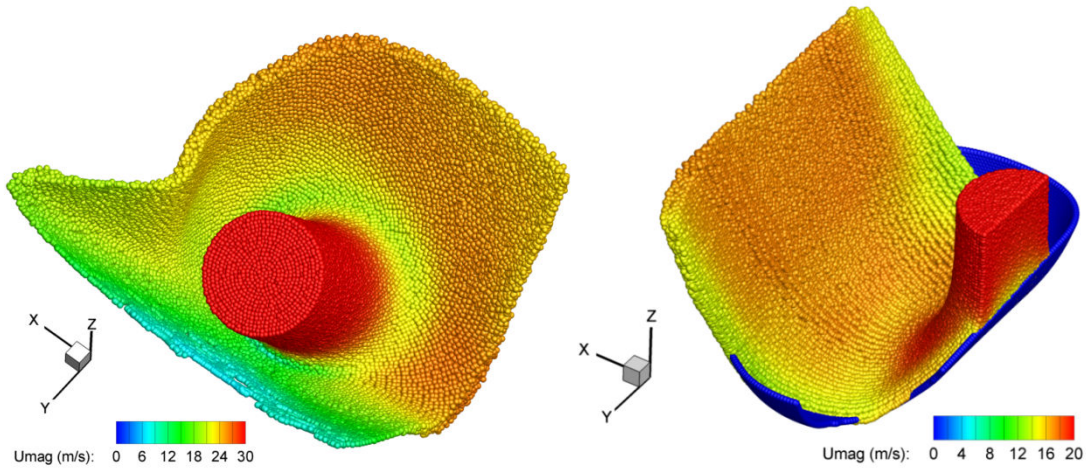


Fig. 7.28. General view of the stationary Turgo impingement (left) and the Pelton impingement (right). Simulation using the SPH-ALE method.

However, forces calculated by the SPH-ALE method are somewhat closer to the reference solution of SPH and Fluent program. Considering the fact that further increasing resolution (see table 7-II) improves the calculated forces, at least in respect to the F_x and F_z components, this implies that the particle independence of the results might have not been reached.

Table 7-I. Comparison of the calculated forces with SPH and SPH-ALE. Turgo jet impingement.

Calculated forces	F_x [N]	F_y [N]	F_z [N]
Fluent	5425.575	-8614.56	-25216.4
SPH	5338.143	-8476.81	-24802.1
SPH-ALE	4846.31	-7960.04	-23887.9

Table 7-II. Comparison of the calculated forces with SPH and SPH-ALE. Pelton jet impingement.

Calculated forces	F_x [N]	F_y [N]	F_z [N]
Fluent	-21.9	-32.1	-278.8
SPH	-17.7	-30.5	-260.6
SPH-ALE 1mm	-24.39	-27.17	-251.28
SPH-ALE 0.75mm	-21.27	-26.91	-256.04

In fig. 7.29 and 7.30 the pressure maps on the Turgo blade and Pelton bucket are shown, as it was calculated from the SPH-ALE method and the Fluent program. The general patterns of the results are

similar; pressure is maximum near the stagnation point and at areas where the direction of the flow changes. However there are some notable differences:

- The pressure map on the Turgo blade surface exhibits some ripples. These are an artifact of the blade surface triangulation and are pronounced at the Fluent solution too, at a lesser extent.
- Pressure is overestimated at the splitter of the Pelton turbine. This was also experienced with the SPH method in chapter 5 and it is caused by the steep angle at the splitter. However pressure overestimation at the splitter is significantly lower with the SPH-ALE method.

In both cases a slight under prediction of pressure is visible, resulting to the forces underestimation shown in table 7-I and 7-II.

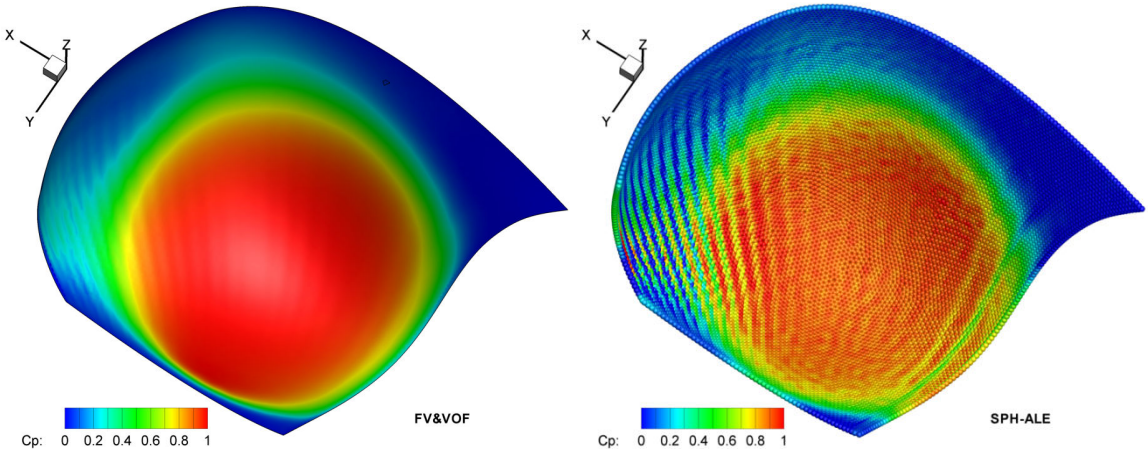


Fig. 7.29. Comparison of the pressure map on the Turgo blade. Left: Fluent solution. Right: SPH-ALE instantaneous solution.

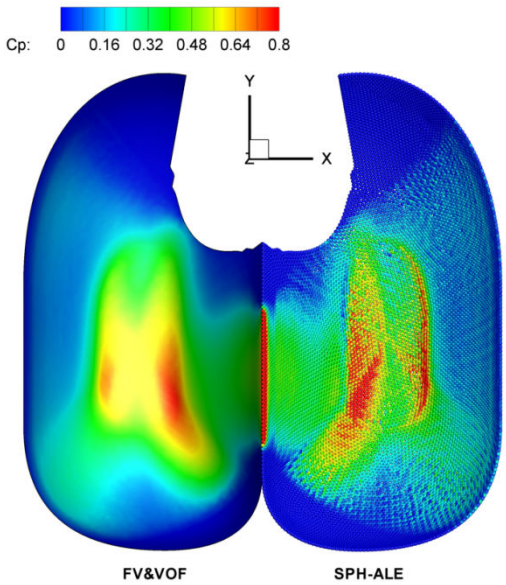


Fig. 7.30. Comparison of the pressure map on the Pelton bucket. Left: Fluent solution. Right: SPH-ALE instantaneous solution.

7.8. Concluding remarks

This chapter focuses on the implementation and the behavior of the SPH-ALE method, a hybrid method based on the Euler equations, under ALE perspective, solved using Finite Volume framework, but with SPH approximations. Moreover it has a robust mathematic background and is able to treat boundary conditions efficiently. Also it is able to handle moving geometries easier than the SPH/SPH-R methods, due to its inherent ALE perspective. Riemann solvers are used as the heart of the algorithm, in order to determine the interactions between adjacent particles.

The method was used for various cases, examining both Eulerian and Lagrangian perspectives. The Eulerian perspective enables the solution of problems, such as the Taylor-Green vortex, since particle motion would deteriorate approximations, result to clumping and eventually would prevent the simulation from completing due to instabilities [19]. Other simulations, such as implosions/explosions or the shock tube are also easier handled than the SPH-R method. On the other hand, the Lagrangian perspective enables the simulation of moving jets or moving geometries, such as the wedge impact.

The 2nd order scheme has been examined in various cases; however in practical cases, such as the Turgo/Pelton impingements, forces are underestimated. A similar underestimation was experienced with the SPH-R method too, indicating that higher particle resolution might be required, or even higher order variants should be used. Even if the SPH-ALE method is somewhat faster than the SPH-R method, further increasing resolution does not seem possible with the given algorithm structure/parallelization in feasible time.

Variable resolution simulations have been performed using the SPH-ALE method; the variable resolution enables high accuracy/detail in areas of interest, while keeping the total particle number at acceptable levels. However there are constraints to the particle distribution, since new terms have to be included in the SPH approximations.

References

[1] J. P. Vila, “On particle weighted methods and smooth particle hydrodynamics”, Mathematical models and methods in applied sciences, World scientific publishing company, 1997.

[2] E. F. Toro, “Riemann solvers and numerical methods for fluid dynamics”, 2nd edition, Springer, 2009.

[3] M. J. Ivings, D. M. Causon and E.F. Toro (1998), “On Riemann solvers for compressible fluids”, International Journal for Numerical Methods in Fluids, vol. 28, Issue 3, December 1998, pp.395-418.

[4] J. C. Marongiu, E. Parkinson, “Riemann solvers and efficient boundary treatments: an hybrid SPH-finite volume numerical method”, Proceedings of the 3rd Spheric workshop, Lausane, June 2008.

[5] F. Dubois, “Partial Riemann problem, boundary conditions and gas dynamics”, Absorbing boundaries and layers, domain decomposition methods: application to large scale computations, p. 16-77, New York 2001, Nova Science Publishers.

[6] Y. Jang, J.C. Marongiu, E. Parkinson, “Analysis of SPH and mesh based simulations using point based post processing tool”, Proceedings of the 3rd Spheric Workshop, Lausanne, Switzerland 2008.

[7] J.C. Marongiu, F. Leboef, J. Caro, E. Parkinson, “Low Mach number numerical schemes for the SPH – ALE method: Application in free surface flows in Pelton turbines”, 4th Spheric Workshop, Nantes France, May 2009.

[8] D. Drikakis, W. Rider, “High resolution methods for Incompressible and low speed flows”, Springer, 2005.

[9] E. Turkel (1987), “Preconditioned methods for solving the incompressible and low speed flow equations”, Journal of Computational Physics, Vol. 2, Issue 2, October 1987, Pages 277-298.

[10] E. M. Yettou, A. Desrochers, Y. Champoux, “Experimental study on the water impact of a symmetrical wedge”, Fluid Dynamics Research, vol. 38, pp. 47-66, 2006.

[11] S.G. Lewis, D.A. Hudson, and S.R. Turnock, “Simulation of a free falling wedge into water using 2D CFD with applications in the prediction of high speed craft motions”, 10th Numerical Towing Tank Symposium (NuTTS'07), Hamburg, Germany, 23 - 25 Sep 2007.

[12] S.G. Lewis, D.A. Hudson, S.R. Turnock, J.I.R. Blake, and R.A. Shenoi, “Predicting the motions of high speed RIBs: a comparison of non-linear strip theory with experiments”, Proceedings of the 5th International Conference on High Performance Marine Vehicles, Launceston, Australia, 8-10 November 2006.

[13] A. Colagrossi, M. Antuono, S. Marrone, “A 2D+t SPH model with enhanced solid boundary treatment”, Proceedings of the 4th Spheric workshop, Nantes, France, 27-29 May 2009.

[14] G. Oger, M. Doring, B. Alessandrini, P. Ferrant, “Two-dimensional SPH simulations of wedge water entries, Journal of Computational Physics”, Vol. 213, Issue 2, pp. 803-822, April 2006. http://dx.doi.org/10.1016/j.jcp.2005.09.004.

[15] G.R. Liu, M. B. Liu, “Smoothed Particle Hydrodynamics: a meshfree particle method”, 1st edition, World Scientific Publishing, 2003.

[16] J.J. Monaghan, “Smoothed particle hydrodynamics”, Reports on Progress in Physics, vol. 68, pp. 1703-1759, 2005.

[17] P. Omidvar, P. Stanby, B. Rogers, “Wave body interaction in 2D using smoothed particle hydrodynamics (SPH) with variable particle mass”, International Journal for Numerical Methods in Fluids, vol. 68, Issue 6, pp.686-705

[18] S. Kvicinsky, “Method d’analyse des ecoulements 3D a surface libre: application aux turbines Pelton”, These N°2526, Ecole Polytechnique Federale de Lausanne, 2002.

[19] R. Xu, C. Moulinec, P. Stansby, B. Rogers, D. Laurence, “Simulation of vortex spindown and Taylor-Green vortices with incompressible SPH method”, 3rd Spheric workshop, Lausanne, June 2008, Switzerland.

Chapter 8

Epilogue

The aim of the present thesis is the development of SPH-based algorithms for the simulation of free surface flows. The main interest is the application of the method on impulse turbine components and turbine runners, including the design optimization of a Turgo turbine runner. Another objective of the present thesis is the validation of the developed algorithms in a wide range of problems, involving viscous and inviscid, free surface and enclosed flows.

The SPH algorithm is based on the standard, weakly compressible SPH developed by Monaghan, Gingold and Lucy, enhanced with several corrections such as density filter, XSPH correction, monotone upwind flux, etc. Truly incompressible SPH was also examined, but its practical implementation involves the efficient solution of the Poisson equation, which is rather computationally expensive using SPH approximations. Various implementations were developed and examined, highlighting their strengths and weaknesses. Eventually, the SPH method was used for the simulation of viscous flows for low and high Reynolds numbers. At low Reynolds numbers the SPH algorithm predicted results comparable to the reference results from literature or other numerical solutions. At high Reynolds numbers, a periodic redistribution of particles in the problem domain is required in order to control the approximation errors and avoid instabilities which tend to form voids and give unrealistic flow fields. The implementation of turbulence models is possible within the SPH framework.

The SPH method was used in various free surface flows too. For free surface flows, wall boundary conditions were described using boundary particles, since this type seems to be well adapted for describing complicated geometries. Generally, the resulting scheme was able to produce very accurate results regarding the free-surface location, comparing with the experimental data, other CFD programs, or theoretical solutions. However, the time history of forces was oscillatory, due to the noisy pressure field, thus time averaging was required. Indicative applications of the SPH method are the simulation of a Pelton turbine injector, a Pelton turbine deflector and the operation of Pelton and Turgo impulse turbines. Pelton and Turgo impulse turbines were tested both while stationary and rotating. Efficiency of the turbines estimated with the SPH method was found to be under predicted. Increasing particle resolution did not seem to improve the efficiency. Comparison with the results of a commercial CFD program showed that the implementation of boundary forces is unable to describe properly all flow effects during the Turgo turbine operation, such as the attachment of water behind each blade. Also the torque curve of the Pelton simulation was underestimated during the emptying of the runner. Optimization of the Turgo turbine was carried out using the combination of the SPH method and a simplified algorithm, the FLS method. The latter algorithm uses some simplifications and requires parameter tuning to describe properly the flow patterns of an impulse turbine. SPH was used to provide data for the tuning and also was used to perform the final evaluation of the elite blade geometries. Despite the underestimation of the absolute efficiency value, the SPH method was able to identify correctly relative differences between designs and eventually provided an optimized geometry. Regarding the performance of the SPH algorithm, it requires approximately one day for the

evaluation of a design; it is less expensive than the Fluent mesh based two-phase solver (which requires several days.

In order to solve some issues of the SPH method, the SPH-R variant was developed and examined. This variant involves the integration of Riemann solvers for the particle interactions, in combination with a Godunov-like scheme. The resulting scheme succeeds in damping the pressure field oscillations providing much smoother fields, but suffers from excessive numerical viscosity. A novel MUSCL scheme was developed, based on the proven MSUCL-Hancock scheme, using forward and backward derivatives. The scheme was used in test cases and applications, were it proved that it performed considerably better than the 1st order Godunov method. Even if it gave better results in the turbine simulations, comparing with the reference, it still exhibits considerable diffusion for the particle resolutions examined.

Another variant which was examined was the SPH-ALE method. This method shifts to the conservative Euler equations using ALE perspective, it is Godunov based and uses Riemann solvers too. The ALE feature enables proper implementation of boundary conditions. Also simulations such as the Green Taylor vortex are performed easily; such simulations would be impossible with a pure Lagrangian perspective or at least impossible without particle redistribution. A novel high order extension was developed, based on the MUSCL-Hancock scheme already discussed. Again, several cases have been examined, including practical applications in impulse turbines. However, as with the SPH-R method, the numerical viscosity is considerable for the particle resolutions used. In both SPH-R and SPH-ALE algorithms, the 2nd order scheme introduces additional computational cost, for the calculation of field variable derivatives and the limiter function. Thus, simulations using these methods are considerably slower than the SPH algorithm. This fact renders high resolution simulations very computationally intensive, requiring other types of parallelization in order to produce results in reasonable times.

An interesting alternative of the SPH-R and SPH-ALE methods is the novel SPH/SPH-R hybrid. The latter method combines the robustness of dealing boundaries accurately using partial Riemann problems, while using the standard SPH relations for the interactions of the fluid particles. Pressure field is smoothed by using the upwind monotone flux formulation. The resulting scheme employs the Godunov method only when a fluid particle interacts with a boundary, thus reducing the overall numerical diffusion. On the other hand, since fluid particle interactions are handled using the SPH method, there is no need for the costly MUSCL scheme, including derivative and limiter function computation. The SPH/SPH-R hybrid has been tested on several benchmark cases showing fast performance and a quality comparable with the 2nd order SPH-R/SPH-ALE schemes.

To sum-up the findings of the present thesis, the following points are highlighted:

- The SPH algorithm, including SPH-based variants developed in the present thesis, was validated in several test cases, involving the reproduction of the wave structure of the Euler equations, conservation of angular momentum and incompressibility.

- Moreover the SPH algorithm was used to simulate viscous flows of high and low Reynolds numbers. The flow field is well reproduced at low Reynolds numbers, comparing with theoretical solutions (Couette, Poisseuille, etc.) and numerical or experimental data (backward facing step case). Particle redistribution was implemented for controlling the particle distribution, reducing approximation errors and consequently enhancing the accuracy of the method. High Reynolds flows are possible to be simulated with SPH, provided that the particle distribution is controlled.

- Free surface flows are easily modeled with the SPH method. The free surface is accurately captured. However, the pressure field is noisy, requiring time averaging of the data for production of meaningful results.

- The flow at the nozzle of a Pelton turbine was simulated using the SPH method. The SPH algorithm was able to capture accurately the evolution of the jet, the shape of the free surface and predict the mass flow rate of the nozzle for a specific needle stroke and pressure difference.

- An impulse turbine deflector of cylindrical shape was evaluated for throttling the turbine load, by cutting part of the jet. The 2D and 3D simulations of the deflector showed that it induces jet distortion which is expected to affect greatly the turbine efficiency, especially if interaction with air is considered. Thus, such a shape is not advisable for a deflector used for turbine throttling.

- The flow in impulse turbine runners was simulated using the SPH method. Again, for all simulated cases the free surface location was similar to the one calculated by the Fluent software. The method estimated a similar torque curve of the turbine runner, with the Fluent software, for both the Turgo and Pelton turbine simulations.

- Furthermore, the SPH method was used in design optimization of a Turgo turbine, in combination with the FLS algorithm; the FLS algorithm was used, after tuning, as an inexact evaluator for the turbine geometry. The final evaluation was performed by the SPH method. Also the SPH method predicted the forces on the axial and radial direction of the turbine runner. These forces were confirmed using velocity triangles and the conservation of momentum theorem and were taken into account for the bearing selection of the laboratory turbine installation.

- The SPH method was used for the simulation of the complete runner of a Pelton and Turgo turbine. In the latter case, it proved that the Turgo turbine is able to operate without significant jet interference from the outflowing water sheets.

- Riemann-based SPH variants were examined. The SPH-R and SPH-ALE methods were developed in order to treat the inherent issues of the standard SPH method, namely the noisy pressure field and the boundary handling. A high order variant of the methods, based on the MUSCL-Hancock scheme, was developed. Both SPH-R and SPH-ALE were used in hydrodynamic problem simulations, such as jet impingements and simulation of the flow in impulse turbines.

- The SPH-ALE method enjoys a significant advantage in respect to the rest SPH-based methods, since it employs an Arbitrary Lagrangian Eulerian perspective, which enables better reproduction of the simulated phenomena, since particles move with an arbitrary transport field. Proper selection of the transport field helps avoiding/preventing two severe instabilities inherent to the SPH method, the tensile instability and the particle alignment, which ruin or deteriorate the solution accuracy.

- A new SPH-based method was developed, formed as a hybrid of the standard SPH with upwind flux formulation and the SPH-R with partial Riemann problem solution for boundary handling. The new SPH/SPH-R hybrid method combines the strengths of SPH and SPH-R, namely the low diffusion of the SPH method and the accurate boundary implementation in SPH-R.

8.1. Comparison of the SPH-based methods

Table 8-I summarizes the strengths and weaknesses of the SPH methods developed and used in the present thesis, having in mind the experience obtained through the presented applications. PR represents the partial Riemann problem (implying that it is used for boundary handling), IC represents initial conditions and BC the boundary conditions. Execution time in all cases is expressed in respect to the SPH method without any corrective schemes.

-Pressure field reproduction, represents the accuracy and smoothness of the pressure field

-Boundary handling, represents the capability to implement accurate boundary conditions

-Stability, implies the tendency of the method to produce unphysical values, which ruin the simulation and probably lead to the algorithm crashing (in Fluent software implies divergence).

-Diffusive, implies the inherent numerical diffusion of the method and schemes used.

-Prone to particle alignment implies the particle clumping which increases particle approximation errors (explained in section 2.3).

-Parallelization efficiency implies the percentage of the algorithm which may be parallelized

Table 8-I. Characteristics of the numerical methods used in the present thesis

Method	*Feature*						
	Pressure field reproduction	*Boundary handling*	*Stability*	*Execution time*	*Diffusive*	*Prone to particle alignment*	*Parallelization efficiency*
SPH without corrections	Bad	Problematic	Unstable	1x	No	Yes	Near 100%
SPH with corrections	Reasonable only with time averaged results	Problematic	Prone to instabilities	1.5x	Yes, depending on the correction	Yes	Near 100%
SPH-R (2nd order)	Good	Accurate (PR)	Prone to tensile instability	3-5x	Yes	Yes	Near 100%
SPH-ALE (2nd order)	Good	Accurate (PR)	Prone to tensile instability only if pure Lagrangian perspective is used	2-4x	Yes	No, if an appropriate transport field is selected	Near 100%
SPH/SPH-R hybrid	Good	Accurate (PR)	Prone to tensile instability	1-2x	Limited diffusion, considerable near walls	Yes	Near 100%
Fluent	Good, assuming convergence is achieved	Accurate	Possible convergence difficulties, depending on the numerical schemes/ICs /BCs	If there is steady state solution, trivial Unsteady multiphase problems: 10x	Diffusive especially near the free surface	N/A	Depending on the implementation of the linear solver

8.2. Contribution of the present thesis

Novel elements

Regarding the SPH method:

- Development of SPH-based algorithms. SPH algorithms were developed in the course of the study, along with variants based on the most recent SPH models (such as SPH with upwind flux formulation, SPH-R and SPH-ALE). Algorithms are parallelized using shared memory parallelization with OpenMP directives.

- Development of a new 2nd order scheme based on the MUSCL-Hancock method. Riemann based SPH methods suffer from excessive dissipation due to the nature of the Godunov scheme which is inherently 1st order. A new 2nd order scheme is formulated and proposed, based on the proven MUSCL-Hancock scheme used in the Finite Volumes method.

- Development of a new hybrid SPH method, combining the standard SPH with upwind flux formulation and accurate boundary handling, solving partial Riemann problems. In order to treat the inherent numerical viscosity of the Riemann based SPH methods and the weak boundary handling of the standard SPH method, a hybrid method, combining the strengths of each method, is formulated. The hybrid SPH/SPH-R method treats particle interactions using the standard SPH scheme, whereas boundaries are treated by solving the partial Riemann problem. In this way the numerical viscosity of the method is reduced, while boundaries are handled accurately.

Regarding the simulation of free surface flows:

- Implementation of the SPH method for the solution of the flow field in impulse turbines. The majority of simulations in impulse turbines are performed using traditional CFD techniques, employing Eulerian perspective and moving meshes. In the present study the SPH meshfree particle method is used instead.

- Simulation of Turgo turbine geometries and design optimization of a Turgo turbine. Turgo turbines are built by very few companies worldwide and the turbine design is based on experimental data or empirical relations. In the present study the SPH method is used for the simulation and the design optimization of a Turgo turbine for maximizing the turbine efficiency. The resulting optimal blade geometry was used to construct an actual turbine model runner.

Journal publications

- Koukouvinis. P., Anagnostopoulos J., Papantonis. D., «Turbulence Modeling in Smoothed Particle Hydrodynamics Methodology: Application in Nozzle Flow», AIP Conference Proceedings vol. 1168, p. 248-251, 2009. DOI: 10.1063/1.3241439

- Koukouvinis. P., Anagnostopoulos J., Papantonis. D., «SPH method used for flow predictions at a Turgo impulse turbine: comparison with Fluent», World Academy of Science Engineering and Technology, vol. 79, p. 659-666, 2011.

- Koukouvinis. P., Anagnostopoulos J., Papantonis. D., "An improved MUSCL treatment for the SPH-ALE method: comparison with the standard SPH method for the jet impingement case" Journal of Numerical Methods in Fluids. In press. DOI: 10.1002/fld.3706.

- Koukouvinis. P., Anagnostopoulos J., Papantonis. D., "Simulation of 2d wedge impacts on water using the SPH-ALE method" European Journal of Mechanics B/Fluids. Under review.

Participation in conferences

- Stamatelos F., Koukouvinis P., Anagnostopoulos J., Papantonis D., «Numerical simulation of incompressible flow using the Smoothed Particle Hydrodynamics method», Flow 2008 conference, Kozani, Greece, 28 November 2008.

- Koukouvinis. P., Anagnostopoulos J., Papantonis. D., «Flow modelling in the injector of a Pelton turbine», Proceedings of the 4th Spheric Workshop, Nantes, France, 27 - 29 May 2009.

- Koukouvinis. P., Anagnostopoulos J., Papantonis. D., «Flow modeling in a Turgo turbine using SPH», Proceedings of the 5th Spheric Workshop, Manchester, United Kingdom, 23 - 25 June 2010.

- Koukouvinis. P., Anagnostopoulos J., Papantonis. D., «Flow analysis inside a Pelton turbine bucket using smoothed particle hydrodynamics», Hydro 2010 conference, Lisbon, Portugal, 27 - 29 September 2010.

- Koukouvinis. P., Anagnostopoulos J., Papantonis. D., «Development and application of the SPH method for impulse turbines», Flow 2010 conference, Thessaloniki, Greece, 12-13 November 2010.

- Koukouvinis. P., Anagnostopoulos J., Papantonis. D., «Numerical Computation of the performance curve of a Pelton turbine using the smoothed particle hydrodynamics method», 7th GRACM International Conference on Computational Mechanics, Athens, Greece, 30 June - 2 July 2011.

- Anagnostopoulos J., Koukouvinis P., Stamatelos F., Papantonis D., «Optimal design and experimental validation of a Turgo model Hydro turbine», Proceedings of the 11th Conference on Engineering Systems Design and Analysis (ESDA 2012), Nantes, France, 2012.

8.3. General thoughts about the SPH-based methods / suggestions for further research.

Perhaps the greatest advantage of the SPH-based methods is the fact that they do not need a computational mesh for performing interpolations or derivative approximations. Performing approximations using a set of arbitrarily distributed particles renders simulations simpler, while large deformations are easily described. However the quality of the approximations is greatly dependent on the particle distribution. Uneven particle distributions or partially filled support domains greatly affect the accuracy of the interpolations. Moreover, particle clumping, which may result from ordered particle distributions, is also a distortion which impacts accuracy. Another important remark is that, since particle distribution changes over time, the errors in the approximations change over time too. Eventually, uneven particle distribution is likely to affect the results too, through error accumulation.

SPH approximations are not equally prone to interpolations errors. For example eq. 2.17 has reduced interpolation errors in comparison with eq. 2.12, since it employs the difference of a function f instead of its value. Actually eq. 2.17 is able to reproduce accurately zeroth order functions from arbitrary distribution points, whereas eq. 2.12 or eq. 2.18 fail to do so. A way to improve the accuracy of the interpolations is by using the kernel gradient procedure. In this way it is ensured that particle approximations are able to reproduce first order functions accurately. But even in that case, additional cost is introduced, in order to correct the kernel and there are limitations at areas with few particles, since the correction might be unphysically large. Higher order corrections have not been explored yet;

such corrections might be even more computationally expensive and exhibit instabilities in areas of particle disorder though.

The Lagrangian nature of SPH is another attractive characteristic, since particles move following the flow patterns and move naturally preserving interfaces and free surfaces. However, a pure Lagrangian perspective might not always be beneficial, since the particle motion according to the local flow velocity might cause particle disorder. Such cases have already been presented; particle clumping occurs near the stagnation point of a jet impingement, or at the corner of the shear cavity. In the high Reynolds number flows it was shown that instabilities might occur, which alter drastically the solution behavior. For that reason techniques such as particle redistribution are employed. On the other hand particle redistribution adds diffusion, due to the approximate interpolations needed to find field variables at the new particle locations. One topic of further research is the development of accurate and conservative interpolation schemes used for particle redistribution. Another possibility to control the particle distribution is by using the ALE perspective. Indeed, the ALE perspective allows particles to be moved following an arbitrary transport field, enabling some particles to remain motionless (Eulerian perspective) and other move following the flow (Lagrangian perspective) in the same simulation. Further exploration of the ALE capabilities is suggested; instead of utilizing either a pure Lagrangian or Eulerian view, it might be beneficial to move particles using an intermediate transport field, avoiding the aforementioned problems of pure Lagrangian motion, while still being able to track important flow features. Keeping the particle distribution controlled is also necessary in variable resolution simulations. In such simulations it is important to keep the fine particles at the area of interest, instead of them being dragged away by the flow. Alternatively methods for splitting and merging particles should be considered.

Another important remark has to do with the accuracy of the SPH method in relation to the computational cost. The SPH method involves two parameters which affect the accuracy of the approximations, one is the particle spacing dx and the other is the smoothing length h. Both play an important role: dx affects the summation approximation and h the integral approximations (practically represents the resolution of the simulation). If the smoothing length is large, the important features might be smeared out, whereas if it is too small the accuracy will suffer (the previous are valid for constant particle spacing). In practical situations in 3D it is suggested to choose a value the smoothing length, which results to a particle interacting with ~50 neighbors. This results to considerable computational effort. Generally in SPH, particle approximations are rather expensive to obtain. In order to simulate realistic applications, parallel algorithms are mandatory, especially for 3d simulations. Variable resolution is a way to reduce computational effort, since it is beneficial to use a fine resolution in areas of interest and keep a coarse resolution at the rest areas. However, variable resolution introduces some additional terms which have to be considered in the summation approximations - these terms have to do with the spatial gradient and the time derivative of the smoothing length. If these additional terms are not considered it is still possible to use variable resolution, but there are strict restrictions; the resolution should change slowly and smoothly. Thus, another subject for research is the implementation of SPH approximations including the additional terms, in conjunction with the additional cost which will be introduced.

From the previous paragraph, it becomes apparent that SPH methods are computationally intensive and require extensive parallelization for practical problems. Fortunately, the heart of the SPH interpolations enables easy parallelization, since particle interactions may be calculated independently for each individual particle. In the present study, the algorithm was parallelized using OpenMP directives (see also appendix B) which practically instruct the CPU to execute in parallel specific computationally intensive loops. However, the way of shared memory parallelization has its own limitations; it can only be used on single multi-core machines, where all CPU cores share the same memory. Implementation of distributed memory parallelization, with e.g. MPI would allow the

utilization of multiple multi-core machines, thus enabling larger problems to be simulated. This type of parallelization requires a different strategy to implement, since problem distribution is necessary among the computers involved in the simulation. Communication between computers is required for transferring data, at the boundaries of the distributed problem domains, and transferring particles from one domain to another, since they are moving in space. Algorithms for the efficient distribution and communication between collaborating processes are an ongoing subject of research. Apart from parallelization using CPUs, another ongoing subject of research is the utilization of modern GPGPUs as massive parallel computers, for the SPH simulation; such parallelization seems very promising, since massive speed-ups of even 100x (in comparison to the CPU algorithm execution) have been obtained for various cases.

Since most industrial application involve high Reynolds flows, it is necessary to use a turbulence model for the flow description. Adaptation of traditional turbulence models within the SPH framework is possible but there are some issues which have to be considered. First of all, the particle distribution has to be controlled, in order to avoid instabilities, as already mentioned. Periodic particle redistribution, using high order interpolation schemes might be an option; however the extension to free surface flows is not straightforward. Whereas the SPH-ALE model would be able to control particle distribution easily, it is applicable using Riemann solvers and thus suffers from considerable numerical diffusion, which might mask the turbulent structures. SPH-specific turbulence models have been developed by Monaghan, however they have not been explored in practical cases.

Implementation of Riemann solvers within the SPH method allowed treating some of the inherent weaknesses of the standard SPH; pressure oscillations are smoothed out, whereas partial Riemann problems enable the accurate handling of boundaries. On the other hand, Riemann solvers introduce numerical viscosity, enforcing the use of 2^{nd} order schemes; however, even with 2^{nd} order schemes the effects of numerical viscosity are observable. Implementation of even higher order schemes might be possible, but it is questionable whether they will perform well in practical simulations, where particle distribution might get seriously distorted. The hybrid SPH/SPH-R method might be a solution, combining the best features of SPH and a robust boundary treatment, using partial Riemann problems, from the SPH-R method. Implementations of various advanced boundary treatments and modifications of the SPH method is another ongoing subject of research. Apart from the SPH-ALE method, another interesting method with ALE capabilities is the Finite Volume Particle Method. The latter method departs from the standard SPH approximations, using a sort of Shepard's kernel instead, which enables better conservation and accuracy properties.

Appendix A

Euler equations and Riemann solvers – Godunov method [1, 2, 3]

The Riemann problem is the initial value problem for a hyperbolic system of conservation laws, written in the following form

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = 0 \tag{A.1}$$

with initial conditions defining a discontinuity between two different states:

$$\mathbf{U}(t=0) = \begin{cases} \mathbf{U}_L & x < 0 \\ \mathbf{U}_R & x \geq 0 \end{cases} \tag{A.2}$$

In the above equations **U** is a vector of variables, *F* is the flux vector and the indices *t* and *x* represent partial derivatives.

$$\mathbf{U} = \begin{bmatrix} u_1 \\ \dots \\ u_m \end{bmatrix} \text{ and } \mathbf{F}(\mathbf{U}) = \begin{bmatrix} f_1 \\ \dots \\ f_m \end{bmatrix}$$

The system of equations may be written also in the following form:

$$\mathbf{U}_t + \mathbf{A}(\mathbf{U})\mathbf{U}_x = 0 \tag{A.3}$$

where **A**(**U**) is called the jacobian matrix of the flux vector *F*, defined as:

$$\mathbf{A}(\mathbf{U}) = \begin{bmatrix} \partial f_1/\partial u_1 & \dots & \partial f_1/\partial u_m \\ \partial f_2/\partial u_1 & \dots & \partial f_2/\partial u_m \\ \dots & \dots & \dots \\ \partial f_m/\partial u_1 & \dots & \partial f_m/\partial u_m \end{bmatrix} \tag{A.4}$$

If the matrix **A** has *m* real eigenvalues and a set of *m* corresponding eigenvectors, then it is said to be hyperbolic.

For the 3D Euler equations using the Tait equation of state, the system of conservation laws (eq. A.1) is written as:

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x + \mathbf{G}(\mathbf{U})_y + \mathbf{H}(\mathbf{U})_z = 0 \tag{A.5}$$

where:

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \end{bmatrix} \mathbf{F}(\mathbf{U}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \end{bmatrix} \mathbf{G}(\mathbf{U}) = \begin{bmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vw \end{bmatrix} \mathbf{H}(\mathbf{U}) = \begin{bmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \end{bmatrix}$$

Pressure is linked *only* to density through the Tait equation of state:

$$p = k\rho^{\gamma} - B \tag{A.6}$$

where $B = \dfrac{\rho_0 {c_0}^2}{\gamma}$ and $k = \dfrac{B}{{\rho_0}^{\gamma}}$.

Note that, due to the fact pressure is only dependent on density, the Euler equations are decoupled from the energy equation [3]. Practically, when solving the Euler equations in 3 dimensions, one has to solve the split 3D Euler equations [1]:

$$\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\end{bmatrix}_t + \begin{bmatrix}\rho u\\ \rho u^2+p\\ \rho uv\\ \rho uw\end{bmatrix}_x = 0 \overset{Tait}{\Rightarrow} \begin{bmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\end{bmatrix}_t + \begin{bmatrix}\rho u\\ \rho u^2+k\rho^{\gamma}\\ \rho uv\\ \rho uw\end{bmatrix}_x = 0 \tag{A.7}$$

Here only density ρ and velocity u, play an important role defining the solution of the Riemann problem. The rest velocity components (v, w) are simply advected by the flow and are treated as passive scalars. The jacobian matrix of the system of equations in A.7 is:

$$\mathbf{A}(\mathbf{U}) = \begin{bmatrix}0 & 1 & 0 & 0\\ \gamma k\rho^{\gamma-1}-u^2 & 2u & 0 & 0\\ -uv & v & u & 0\\ -uw & w & 0 & u\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 & 0\\ c^2-u^2 & 2u & 0 & 0\\ -uv & v & u & 0\\ -uw & w & 0 & u\end{bmatrix} \tag{A.8}$$

Finally the eigenvalues are $\lambda_1 = u\text{-}c$, $\lambda_{2,3} = u$ and $\lambda_4 = u + c$ and the respective right eigenvectors of the jacobian matrix are:

$$\mathbf{K}_1 = \begin{bmatrix}1\\ u-c\\ v\\ w\end{bmatrix} \quad \mathbf{K}_2 = \begin{bmatrix}0\\ 0\\ 1\\ 0\end{bmatrix} \quad \mathbf{K}_3 = \begin{bmatrix}0\\ 0\\ 0\\ 1\end{bmatrix} \quad \mathbf{K}_4 = \begin{bmatrix}1\\ u+c\\ v\\ w\end{bmatrix}$$

where $c(\rho) = \sqrt{\gamma k\rho^{\gamma-1}}$ the speed of sound, i.e. the speed an event is transmitted inside the described medium. Each eigenvalues is associated with a wave; eigenvalues λ_1 and λ_4 are associated with non-linear waves, rarefactions or shock waves, and λ_2, λ_3 are associated with contact discontinuity. The general form of the Riemann problem solution is the following:

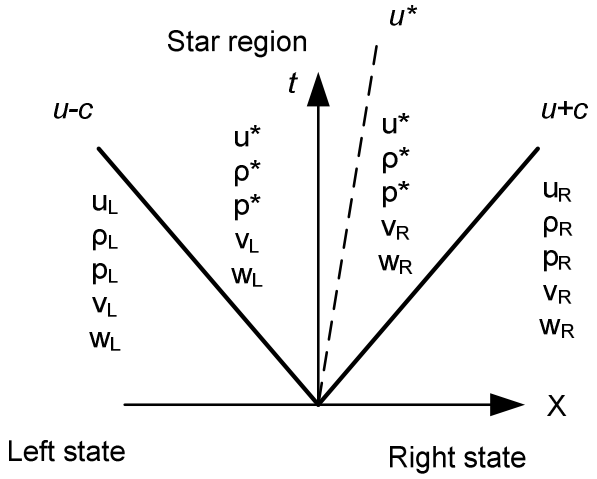


Fig. A.1. Solution of the Riemann problem

The solution of the Riemann problem is self-similar, as shown in fig. A.1 since the solution is dependent on the wave speeds only. The waves divide the solution in four distinct states. Between the two non-linear waves there is an intermediate state denoted with a star, where the x-velocity component and pressure are the same. However these non-linear waves do not change the passive scalars, such as the y and z velocity components. On the other hand, the tangential velocity

components and any other passive scalars change before and after the linearly degenerate contact discontinuity. Here it is highlighted that in our case, which uses the Tait equation of state, density is the same throughout the whole star region. However this is not valid for other equations of state, such as the ideal gas equation of state; in the latter case, the density field is divided in four regions: ρ_L, ρ_L^*, ρ_R^* and ρ_R. It must also be mentioned that the speeds of the waves is, generally, not equal to the characteristic speeds given by the eigenvalues; nevertheless in approximate Riemann solvers it may be assumed so.

Exact Riemann solver

The exact solution of the Riemann problem described above may be obtained by solving a non-linear algebraic equation in respect to density only. The equation has the following general form:

$$g(\rho^*) = g_L(\mathbf{U}_L, \rho^*) + g_R(\mathbf{U}_R, \rho^*) + \Delta u = 0 \tag{A.9}$$

Where:

- $\Delta u = u_R - u_L$
- g_L, g_R are functions depending on whether the respective non-linear wave is a shock wave or a rarefaction wave.

The g_L and g_R functions are calculated using *Rankine-Hugoniot* conditions for the shock wave or *Riemann invariants* for the rarefaction wave (the reader is referred to the excellent work of E.Toro [2] and [3]). Eventually it is proved that [3]:

$$g_K(\rho_*) = \begin{cases} \left[\dfrac{k\left(\rho_*^{\gamma} - \rho_K^{\gamma}\right)\left(\rho_* - \rho_K\right)}{\rho_K \rho_*} \right]^{1/2} & if\ \rho_* > \rho_K \\ \dfrac{2c_K}{(\gamma\text{-}1)} \left[\left(\dfrac{\rho_*}{\rho_K} \right)^{(\gamma-1)/2} - 1 \right] & if\ \rho_* \le \rho_K \end{cases} \tag{A.10}$$

$$c(\rho_K) = \sqrt{\gamma\, k\, \rho_K^{\gamma\text{-}1}} \tag{A.11}$$

The previous relation may be used for *K=R* or *L*.

Equation A.10 is solved efficiently using the Newton – Raphson iterative method:

$$\rho_*^{new} = \rho_*^{old} - \frac{g\left(\rho_*^{old}\right)}{g'\left(\rho_*^{old}\right)} \tag{A.12}$$

until the difference between the new and old ρ_* is below a specified tolerance:

$$\left| \rho_*^{new} - \rho_*^{old} \right| \le tol \tag{A.13}$$

The derivative of the $g(\rho_*)$ function is complicated, especially for the shock relation. Instead of using the exact derivative function $g'(\rho_*)$, it might be simpler to use a derivative approximation:

$$g'(\rho_*) \approx \frac{g(\rho_* + e) - g(\rho_*)}{e} \tag{A.14}$$

As an initial estimation of density ρ_*, the average value of ρ_L and ρ_R might be used.

After obtaining ρ_* with the Newton-Raphson method, it is possible to find u_*:

$$u_* = \frac{1}{2}(u_L + u_R) + \frac{1}{2}\left[g_R(\rho_*) - g_L(\rho_*) \right] \tag{A.15}$$

The u_* is assumed to be the velocity of the contact discontinuity wave (also denoted with S^*).

Completing the solution

Once ρ_* and u_* are found it is possible to calculate the complete distribution of fields. First the wave speeds must be calculated.

- In case of a shock wave:

$$S_L = u_L - \frac{Q_L}{\rho_L} \text{ shock wave towards left state} \tag{A.16}$$

$$S_R = u_R + \frac{Q_R}{\rho_R} \text{ shock wave towards right state} \tag{A.17}$$

$$Q_K = \left[\frac{k\left(\rho_*{}^{\gamma} - \rho_K{}^{\gamma}\right)\rho_K \rho_*}{\rho_* - \rho_K} \right]^{1/2}, \text{ for } K = L, R \tag{A.18}$$

- In case of rarefaction wave:

$$S_{HL} = u_L - c_L,\ S_{TL} = u_* - c_* \text{ rarefaction wave towards left state} \tag{A.19}$$

$$S_{TR} = u_* + c_*,\ S_{HR} = u_R + c_R, \text{ rarefaction wave towards right state} \tag{A.20}$$

Here it is highlighted that the rarefaction wave is not a true discontinuity, such as the shock wave, but rather a smooth transition from one state to another (see also fig. A.2). Each rarefaction wave consists of a head and a tail (denoted with *H* and *T* respectively).

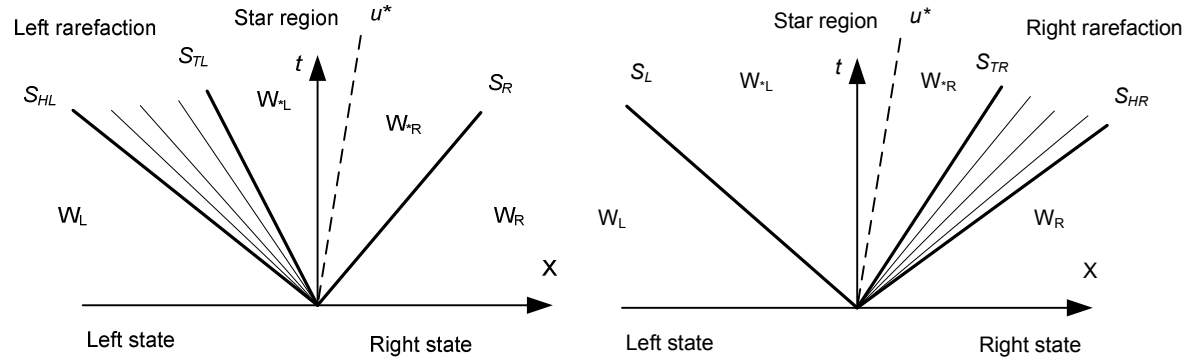


Fig. A.2. Rarefaction towards left and right. The head and tail of the rarefaction wave are visible.

Characteristics (ρ, u) inside the rarefaction wave may be calculated, using Riemann invariants, as follows:

- Left rarefaction:

$$u = \frac{2}{\gamma+1}\left[\frac{u_L(\gamma\text{-}1)}{2} + c_L + \frac{x}{t} \right] \tag{A.21}$$

Using the Riemann invariant:

$$u_L + \frac{2c_L}{\gamma-1} = u + \frac{2c}{\gamma-1} \tag{A.22}$$

the sound speed, c, is calculated and then using the definition of sound speed, density is calculated:

$$\rho = \left[\frac{c^2}{\gamma k} \right]^{\frac{1}{\gamma-1}} \tag{A.23}$$

- Following the similar procedure for the right rarefaction:

$$u = \frac{2}{\gamma+1}\left[\frac{u_R(\gamma\text{-}1)}{2} - c_R + \frac{x}{t}\right] \tag{A.24}$$

and again:

$$u - \frac{2c}{\gamma-1} = u_R - \frac{2c_R}{\gamma-1} \tag{A.25}$$

$$\rho = \left[\frac{c^2}{\gamma k}\right]^{\frac{1}{\gamma-1}} \tag{A.26}$$

Regarding the passive scalars, such as the tangential velocity components, their distribution is determined by the contact discontinuity. The contact discontinuity divides the solution in two areas, left and right; each area has the same passive scalar value as the respective state.

As a test of the exact Riemann solver examined above, the following case is considered:

$$\begin{cases} \rho_L = 1100, \ u_L = 200 & x < 0 \\ \rho_R = 1000, \ u_R = 0 & x > 0 \end{cases}$$

The parameters of the Tait equation of state are: ρ_0=997.04kg/m^3, γ=7.15 and B=300MPa.
Results are shown in the following figure:

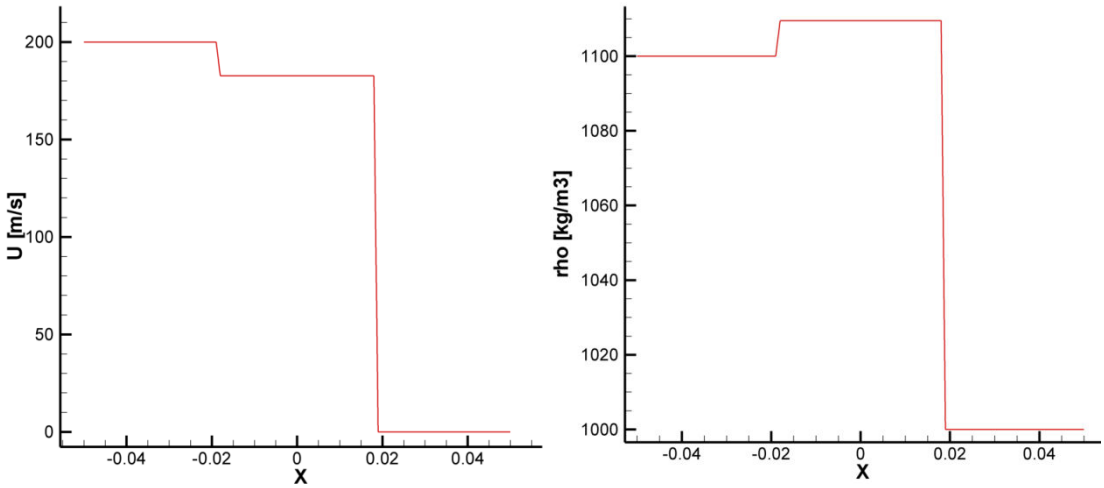


Fig. A.3. Solution of the Riemann problem.

From the solution:

- $\rho_* = 1109.53$kg/m^3
- $u_* = 182.63$m/s
- $p_* = 344306912$Pa

Using these results it is possible to calculate the fluxes, which are identical to the calculated fluxes in Ivings work [3]. The solution of the exact Riemann problem is used for testing SPH algorithms presented in the present work and for testing other approximate Riemann solvers.

Expressing Euler equations in different variable sets

The Euler equations may be written in other variable forms, instead of the conservative. Generally such forms may calculate incorrect wave speeds and wave strengths near shock waves [2]. However they are useful for the derivation of approximate Riemann solvers. Such forms are:

- The primitive variables $\mathbf{W}_P=[\rho, u, v, w]^T$:

Beginning from the conservative variable formulation:

$$\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\end{bmatrix}_t + \begin{bmatrix}\rho u\\ \rho u^2 + k\rho^\gamma\\ \rho uv\\ \rho uw\end{bmatrix}_x = 0 \Rightarrow$$

$$\textit{Continuity}:\ \rho_t + \rho_x u + \rho u_x = 0 \tag{A.27}$$

$$\textit{u-momentum}:\ \rho_t u + \rho u_t + \rho_x u^2 + 2uu_x\rho + \gamma k\rho^{\gamma-1}\rho_x = 0 \Rightarrow$$

$$u\left(\rho_t + \rho_x u + u_x\rho\right) + \rho\left(u_t + uu_x + \gamma k\rho^{\gamma-2}\rho_x\right) = 0 \Rightarrow$$

$$u_t + uu_x + \gamma k\rho^{\gamma-2}\rho_x = 0 \tag{A.28}$$

$$\textit{v-momentum}:\ \rho_t v + \rho v_t + \rho_x uv + \rho u_x v + \rho uv_x = 0 \Rightarrow$$

$$\rho\left(v_t + uv_x\right) + v\left(\rho_t + \rho_x u + \rho u_x\right) = 0 \Rightarrow$$

$$v_t + uv_x = 0 \tag{A.29}$$

The derivation is similar for the w-velocity component:

$$w_t + uw_x = 0 \tag{A.30}$$

Eventually:

$$\begin{bmatrix}\rho\\ u\\ v\\ w\end{bmatrix}_t + \begin{bmatrix}u & \rho & 0 & 0\\ c^2/\rho & u & 0 & 0\\ 0 & 0 & u & 0\\ 0 & 0 & 0 & u\end{bmatrix}\begin{bmatrix}\rho\\ u\\ v\\ w\end{bmatrix}_x = 0 \tag{A.31}$$

With eigenvalues: $\lambda_1 = u\text{-}c$, $\lambda_{2,3} = u$ and $\lambda_4 = u + c$ and the respective eigenvectors:

$$\mathbf{K}_1 = \begin{bmatrix}\rho\\ -c\\ 0\\ 0\end{bmatrix}\ \mathbf{K}_2 = \begin{bmatrix}0\\ 0\\ 1\\ 0\end{bmatrix}\ \mathbf{K}_3 = \begin{bmatrix}0\\ 0\\ 0\\ 1\end{bmatrix}\ \mathbf{K}_4 = \begin{bmatrix}\rho\\ c\\ 0\\ 0\end{bmatrix}$$

- The entropic variables $\mathbf{W}_E= [p, u, v, w]^T$:

Beginning from the primitive variable formulation and using the Tait equation of state:

$$p = k\rho^\gamma - B$$

$$\rho = \left(\frac{p}{k} + \rho_0{}^\gamma\right)^{\frac{1}{\gamma}} \tag{A.32}$$

$$\text{and } c^2 = \gamma k\left[\left(\frac{p}{k} + \rho_0{}^\gamma\right)^{\frac{1}{\gamma}}\right]^{\gamma-1} \tag{A.33}$$

one may obtain:

$$\textit{Continuity}:\ \rho_t + \rho_x u + \rho u_x = 0$$

$$\left[\left(\frac{p}{k}+\rho_0{}^{\gamma}\right)^{\frac{1}{\gamma}}\right]_t + \left[\left(\frac{p}{k}+\rho_0{}^{\gamma}\right)^{\frac{1}{\gamma}}\right]_x u + \left[\left(\frac{p}{k}+\rho_0{}^{\gamma}\right)^{\frac{1}{\gamma}}\right]u_x = 0 \Rightarrow$$

$$\frac{1}{\gamma}\left[\left(\frac{p}{k}+\rho_0{}^{\gamma}\right)^{\frac{1}{\gamma}-1}\right]\frac{p_t}{k} + \frac{1}{\gamma}\left[\left(\frac{p}{k}+\rho_0{}^{\gamma}\right)^{\frac{1}{\gamma}-1}\right]\frac{p_x}{k}u + \left[\left(\frac{p}{k}+\rho_0{}^{\gamma}\right)^{\frac{1}{\gamma}}\right]u_x = 0 \Rightarrow$$

$$p_t + p_x u + \gamma k\left(\frac{p}{k}+\rho_0{}^{\gamma}\right)u_x = 0 \Rightarrow$$

$$p_t + p_x u + \gamma k\rho^{\gamma-1}\rho u_x = 0 \Rightarrow$$

$$p_t + p_x u + c^2\rho u_x = 0 \tag{A.34}$$

$$\textit{u-momentum}:\ u_t + uu_x + \gamma k\rho^{\gamma-2}\rho_x = 0$$

$$u_t + uu_x + \gamma k\left[\left(\frac{p}{k}+\rho_0{}^{\gamma}\right)^{\frac{1}{\gamma}}\right]^{\gamma-2}\left[\left(\frac{p}{k}+\rho_0{}^{\gamma}\right)^{\frac{1}{\gamma}}\right]_x = 0 \Rightarrow$$

$$u_t + uu_x + \gamma k\left[\left(\frac{p}{k}+\rho_0{}^{\gamma}\right)^{\frac{1}{\gamma}}\right]^{\gamma-2}\frac{1}{\gamma}\left(\frac{p}{k}+\rho_0{}^{\gamma}\right)^{\frac{1}{\gamma}-1}\frac{p_x}{k} = 0 \Rightarrow$$

$$u_t + uu_x + \gamma k\left[\left(\frac{p}{k}+\rho_0{}^{\gamma}\right)^{\frac{1}{\gamma}}\right]^{\gamma-1}\frac{1}{\gamma}\left(\frac{p}{k}+\rho_0{}^{\gamma}\right)^{-1}\frac{p_x}{k} = 0 \Rightarrow$$

$$u_t + uu_x + c^2\frac{1}{\gamma}\left(\rho^{\gamma}\right)^{-1}\frac{p_x}{k} = 0 \Rightarrow$$

$$u_t + uu_x + c^2\frac{1}{\gamma k\rho^{\gamma-1}}\frac{p_x}{\rho} = 0 \Rightarrow$$

$$u_t + uu_x + \frac{p_x}{\rho} = 0 \tag{A.35}$$

$$v\text{-momentum:}\ v_t + uv_x = 0 \tag{A.36}$$

$$w\text{-momentum:}\ w_t + uw_x = 0 \tag{A.37}$$

$$\begin{bmatrix} p \\ u \\ v \\ w \end{bmatrix}_t + \begin{bmatrix} u & c^2\rho & 0 & 0 \\ 1/\rho & u & 0 & 0 \\ 0 & 0 & u & 0 \\ 0 & 0 & 0 & u \end{bmatrix}\begin{bmatrix} p \\ u \\ v \\ w \end{bmatrix}_x = 0 \tag{A.38}$$

With eigenvalues: $\lambda_1 = u\text{-}c$, $\lambda_{2,3} = u$ and $\lambda_4 = u + c$ and the respective eigenvectors:

$$\mathbf{K}_1 = \begin{bmatrix} \rho c \\ -1 \\ 0 \\ 0 \end{bmatrix} \quad \mathbf{K}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \mathbf{K}_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \mathbf{K}_4 = \begin{bmatrix} \rho c \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Approximate Riemann solvers

The exact solver is accurate but has the disadvantage of needing an iterative method to solve the non-linear equation (eq A.10). Since the Riemann solver is the heart of any numerical method employing the Godunov scheme [1, 2], it is required to solve the Riemann problem between each control volume for each time step, thus making the iterative scheme inefficient in respect to computational cost. It is beneficial to derive simpler non-iterative schemes which give a solution close to the exact Riemann solver, but are less computationally expensive.

Primitive Variable Riemann Solver (PVRS)

The primitive variable Riemann solver is based on the linearization of the Jacobian matrix using the primitive variable formulation. The values of the Jacobian matrix are assumed to be the average values between the Left and Right states. Since the Jacobian is assumed to be constant, it is possible to use the jump relations across the waves:

$$\Delta \mathbf{W} = \sum_i a_i \mathbf{K}_i \tag{A.39}$$

where a_i are the wave strengths and $\mathbf{K}_i$ the eigenvectors respective to each wave. Also $\Delta \mathbf{W} = \mathbf{W}_R - \mathbf{W}_L$.

For the 3D-split Euler equations:

$$\begin{bmatrix} \rho_R - \rho_L \\ u_R - u_L \\ v_R - v_L \\ w_R - w_L \end{bmatrix} = a_1 \begin{bmatrix} \overline{\rho} \\ -\overline{c} \\ 0 \\ 0 \end{bmatrix} + a_2 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + a_3 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + a_4 \begin{bmatrix} \overline{\rho} \\ \overline{c} \\ 0 \\ 0 \end{bmatrix} \tag{A.40}$$

By solving the above system of equations in respect to the wave strengths a_1, a_2, a_3, a_4, it is then possible to derive the relations for the star region in respect to the left state (the results would be same if the right state was used instead):

$$\begin{bmatrix} \rho_* - \rho_L \\ u_* - u_L \\ v_{*L} - v_L \\ w_{*L} - w_L \end{bmatrix} = a_1 \begin{bmatrix} \overline{\rho} \\ -\overline{c} \\ 0 \\ 0 \end{bmatrix} \tag{A.41}$$

Eventually:

$$\rho_* = \frac{\overline{c}(\rho_R + \rho_L) - \overline{\rho}(u_R - u_L)}{2\overline{c}} \tag{A.42}$$

$$u_* = \frac{\overline{\rho}(u_R + u_L) - \overline{c}(\rho_R - \rho_L)}{2\overline{\rho}} \tag{A.43}$$

Another possibility is to use the same procedure but for the entropic variables instead of the primitive variables. In that case:

$$\begin{bmatrix} p_R - p_L \\ u_R - u_L \\ v_R - v_L \\ w_R - w_L \end{bmatrix} = a_1 \begin{bmatrix} \overline{\rho} \cdot \overline{c} \\ -1 \\ 0 \\ 0 \end{bmatrix} + a_2 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + a_3 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + a_4 \begin{bmatrix} \overline{\rho} \cdot \overline{c} \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{A.44}$$

and eventually:

$$p_* = \frac{1}{2}(p_R + p_L) - \frac{\overline{\rho} \cdot \overline{c}}{2}(u_R - u_L) \tag{A.45}$$

$$u_* = \frac{1}{2}(u_R + u_L) - \frac{1}{2\overline{\rho} \cdot \overline{c}}(p_R - p_L) \tag{A.46}$$

There are other possibilities for deriving equivalent approximations based on the same procedure, such as using characteristic equations or using left eigenvectors [2].

Harten, Lax, van Leer solver - HLL

The HLL solver is based on the conservative variables formulation. The solution vectors are given by [3]:

$$\mathbf{U}^*_{HLL} = \frac{S_R \mathbf{U}_R - S_L \mathbf{U}_L + \mathbf{F}_L - \mathbf{F}_R}{S_R - S_L} \tag{A.47}$$

where S_R and S_L are the respective wave speeds towards the left and right state respectively.
Thus:

$$\rho_* = \frac{S_R \rho_R - S_L \rho_L + (\rho u)_L - (\rho u)_R}{S_R - S_L} \tag{A.48}$$

and:

$$(\rho u)_* = \frac{S_R (\rho u)_R - S_L (\rho u)_L + (\rho u + p)_L - (\rho u + p)_R}{S_R - S_L} \tag{A.49}$$

The complete solution is given by (here the HLLC correction is adopted [2]):

$$\mathbf{U} = \begin{cases} \mathbf{U}_L & if \quad x/t < S_L \\ \mathbf{U}_{*L} & if \quad S_L \le x/t < S_* \\ \mathbf{U}_{*L} & if \quad S_* \le x/t < S_R \\ \mathbf{U}_R & if \quad S_R \le x/t \end{cases} \tag{A.50}$$

From the above it is obvious that the HLL Riemann solver needs an estimate for the wave speeds S_L and S_R (for S^* no estimate is required, since it is assumed equal to u_*). There are many different ways for estimating wave velocities and a simple way is:

$$S_L = u_L - c_L \tag{A.51}$$

$$S_R = u_R + c_R \tag{A.52}$$

Other possible estimates are to use the Roe averaged u and c variables. The HLL solver accuracy is greatly dependent on the wave speed estimation, as it will be shown later on.

Roe solver

The Roe solver main idea is to construct an averaged Jacobian matrix and to use it for calculating the jump conditions as in the PVRS solver. The main characteristic of the Roe averaged Jacobian, is that it satisfies several desired properties, such as:

- Hyperbolicity of the system. The Jacobian matrix is required to have real eigenvalues and a complete set of linearly independent eigenvectors.
- Consistency with the exact Jacobian, when used to represent the right or left state.
- Conservation across the discontinuities:

$$\mathbf{F}(\mathbf{U}_R)-\mathbf{F}(\mathbf{U}_L)=\tilde{\mathbf{A}}(\mathbf{U}_R-\mathbf{U}_L) \tag{A.53}$$

The method of Roe involves using a parameter vector $\boldsymbol{Q}$ with which can be used to express the conserved variables **U** and flux variables **F**(**U**). Generally the parameter vector is chosen as:

$$\mathbf{Q}=\frac{\mathbf{U}}{\sqrt{\rho}} \tag{A.54}$$

Then the average vector $\tilde{\mathbf{Q}}$ is defined as:

$$\tilde{\mathbf{Q}}=\frac{1}{2}(\mathbf{Q}_R+\mathbf{Q}_L)=\begin{bmatrix}\tilde{q}_1\\ \tilde{q}_2\\ \tilde{q}_3\\ \tilde{q}_4\end{bmatrix}=\begin{bmatrix}\sqrt{\rho_L}+\sqrt{\rho_R}\\ u_L\sqrt{\rho_L}+u_R\sqrt{\rho_R}\\ v_L\sqrt{\rho_L}+v_R\sqrt{\rho_R}\\ w_L\sqrt{\rho_L}+w_R\sqrt{\rho_R}\end{bmatrix} \tag{A.55}$$

Eventually one has to calculate the $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{C}}$ matrices, which satisfy the following conditions:

$$\left.\begin{aligned}\Delta\mathbf{U}&=\tilde{\mathbf{B}}\Delta\mathbf{Q}\\ \Delta\mathbf{F}&=\tilde{\mathbf{C}}\Delta\mathbf{Q}\end{aligned}\right\} \tag{A.57}$$

Then the Roe averaged Jacobian may be calculated as:

$$\tilde{\mathbf{A}}=\tilde{\mathbf{C}}\tilde{\mathbf{B}}^{-1} \tag{A.58}$$

Calculation of these matrices leads to the following result:

$$\tilde{\mathbf{C}}=\begin{bmatrix}\tilde{q}_2 & \tilde{q}_1 & 0 & 0\\ 2\tilde{c}^2\tilde{q}_1 & 2\tilde{q}_2 & 0 & 0\\ 0 & \tilde{q}_3 & \tilde{q}_2 & 0\\ 0 & \tilde{q}_4 & 0 & \tilde{q}_2\end{bmatrix} \tag{A.59}$$

$$\tilde{\mathbf{B}}=\begin{bmatrix}2\tilde{q}_1 & 0 & 0 & 0\\ \tilde{q}_2 & 2\tilde{q}_1 & 0 & 0\\ \tilde{q}_3 & 0 & \tilde{q}_1 & 0\\ \tilde{q}_4 & 0 & 0 & \tilde{q}_1\end{bmatrix} \tag{A.60}$$

Finally the Roe averaged Jacobian is:

$$\tilde{\mathbf{A}}=\begin{bmatrix}0 & 1 & 0 & 0\\ \tilde{c}^2-\tilde{u}^2 & 2\tilde{u} & 0 & 0\\ -\tilde{u}\tilde{v} & \tilde{v} & \tilde{u} & 0\\ -\tilde{u}\tilde{w} & \tilde{w} & 0 & \tilde{u}\end{bmatrix} \tag{A.61}$$

where:

$$\tilde{\rho} = \sqrt{\rho_R \rho_L} \text{ or } \tilde{\rho} = \left(\frac{\sqrt{\rho_R} + \sqrt{\rho_L}}{2} \right)^2$$

$$\tilde{c} = \sqrt{\gamma \cdot k \cdot \tilde{\rho}^{\gamma-1}}$$

$$\tilde{u} = \frac{u_R \sqrt{\rho_R} + u_L \sqrt{\rho_L}}{\sqrt{\rho_R} + \sqrt{\rho_L}}$$

$$\tilde{v} = \frac{v_R \sqrt{\rho_R} + v_L \sqrt{\rho_L}}{\sqrt{\rho_R} + \sqrt{\rho_L}}$$

$$\tilde{w} = \frac{w_R \sqrt{\rho_R} + w_L \sqrt{\rho_L}}{\sqrt{\rho_R} + \sqrt{\rho_L}}$$

After obtaining the Jacobian values, then the jump relations may be used to determine the star region:

$$\Delta \mathbf{U} = \sum_i a_i \tilde{\mathbf{K}}_i \tag{A.62}$$

$$\begin{bmatrix} \rho_R - \rho_L \\ \rho u_R - \rho u_L \\ \rho v_R - \rho v_L \\ \rho w_R - \rho w_L \end{bmatrix} = a_1 \begin{bmatrix} 1 \\ \tilde{u} - \tilde{a} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} + a_2 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + a_3 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + a_4 \begin{bmatrix} 1 \\ \tilde{u} + \tilde{a} \\ \tilde{v} \\ \tilde{w} \end{bmatrix} \tag{A.63}$$

By solving the system of equations arising from A.63, it is possible to calculate the wave strengths a_1, a_2, a_3, a_4 and then calculate the values at the star region.

Comparison

In this part a small comparison of all the described solvers, exact and approximate, is presented. Several tests are performed for several different initial conditions.

$$Test - A: \begin{cases} \rho_L = 1050, \ u_L = 10 & x < 0 \\ \rho_R = 1000, \ u_R = 0 & x > 0 \end{cases}$$

$$Test - B: \begin{cases} \rho_L = 1100, \ u_L = 200 & x < 0 \\ \rho_R = 1000, \ u_R = 0 & x > 0 \end{cases}$$

$$Test - C: \begin{cases} \rho_L = 2000, \ u_L = 400 & x < 0 \\ \rho_R = 1000, \ u_R = 0 & x > 0 \end{cases}$$

Each test has different discontinuity strength. The results from each solver are shown in the following tables (A-I, A-II, A-III). The Riemann solvers which were used for comparison are:

- The exact Riemann solver
- The PVRS solver
- The entropic variable formulation
- The HLL solver using the simple velocities $S_L=u_L-c_L$ and $S_R=u_R+c_R$
- The HLL solver using the Roe averaged velocities $S_L= \tilde{u} - \tilde{c}$ and $S_R= \tilde{u} + \tilde{c}$

- The Roe solver

From the results it is obvious that for smaller discontinuities the error of the approximate Riemann solvers will be smaller. At large discontinuities (for example see test – C), approximate Riemann solvers may fail, since error exceeds 10%; indeed in such cases the exact Riemann solver might be the only acceptable solution. It has to be highlighted here, that since the described Riemann solvers will be used for weakly compressible (and consequently subsonic) cases, such discontinuities will not be actually simulated. In cases encountered in the present work, with density variations of 1% and velocities at the order of 50m/s, errors are practically negligible (~0.03%). However these tests will help the selection of the most robust approximate solver for the most adverse conditions.

By examining tables A-I, A-II and A-III, the dependency of the accuracy of the HLL approximate Riemann solver on the wave speed estimation is clear; using the simple wave speed estimates results to errors considerably larger than any other approximate Riemann solver. Regarding the rest solvers, the Roe solver followed by the PVRS and the entropic variable solver are the most accurate and for that reason they will be used for the Riemann solver implementations in combination with SPH described in the present work (chapters 6 and 7, focusing on SPH-R and SPH-ALE).

Table A-I. Test A results for different Riemann solvers

Test-A	ρ_*	% error	u_*	% error
Exact	1028.819		43.97095	
PVRS	1028.203	0.06%	44.02399	0.12%
Entropic	1030.016	0.12%	43.99841	0.06%
HLLC - A	1030.091	0.12%	44.42274	1.03%
HLLC -B	1028.21	0.06%	44.09004	0.27%
Roe	1028.21	0.06%	43.93248	0.09%

Table A-II. Test B results for different Riemann solvers

Test-B	ρ_*	% error	u_*	% error
Exact	1109.538		182.6386	
PVRS	1110.616	0.10%	182.4868	0.08%
Entropic	1108.004	0.14%	182.2808	0.20%
HLLC - A	1122.05	1.13%	200.8936	10.00%
HLLC -B	1111.091	0.14%	183.0871	0.25%
Roe	1111.091	0.14%	181.8877	0.41%

Table A-III. Test C results for different Riemann solvers

Test-C	ρ_*	% error	u_*	% error
Exact	1666.162		2142.972	
PVRS	1543	7.39%	2525.591	17.85%
Entropic	1840.327	10.45%	2264.649	5.68%
HLLC - A	1949.815	17.02%	2012.814	6.07%
HLLC -B	1560.047	6.37%	3205.793	49.60%
Roe	1560.047	6.37%	1775.1	17.17%

The Godunov method

The Godunov scheme may be used to solve numerically hyperbolic systems of conservation laws, using the Finite Volume (FV) or other equivalent methods (such as the SPH-R or SPH-ALE methods presented in the present work). The methodology will be briefly described here, using the FV method, assuming a one dimensional problem:

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = 0 \tag{A.64}$$

with initial conditions $\mathbf{U}(t=0)$ not necessarily being discontinuous. The solution may be obtained by time marching using the following formula:

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n + \frac{dt}{dx}\left[\mathbf{F}_{i-\frac{1}{2}} - \mathbf{F}_{i+\frac{1}{2}}\right] \tag{A.65}$$

Where:

- dx is the discretization size
- dt is the time step, which has to satisfy the stability criterion: $dt = \frac{dx}{S_{\max}}$, with S_{max} being the maximum wave velocity
- n represents the current time step, $n+1$ is the next time step
- $\mathbf{F}_{i+\frac{1}{2}}$ is the flux vector calculated at the interface between i and i+1 cells. Practically it is calculated using the solution of the Riemann problem $\mathbf{U}_{i+\frac{1}{2}}$ at the interface between the cells, i.e.:

$$\mathbf{F}_{i+\frac{1}{2}} = \mathbf{F}\left(\mathbf{U}_{i+\frac{1}{2}}(0)\right) \tag{A.66}$$

Cells i and i+1 are treated as the two states of the Riemann problem and the interface between them is the discontinuity. In order to use the described algorithm, one has to find the solution of the Riemann problem and determine the vector of variables **U** at the interface between the cells, or in other words, on the t-axis, for X=0 (or for a velocity equal to zero), in the local coordinate system where cell i is the left state and cell i+1 is the right state. Thus, one has to solve the Riemann problem either using an exact or using an approximate Riemann solver and then has to indentify the wave pattern (see fig. A.4).

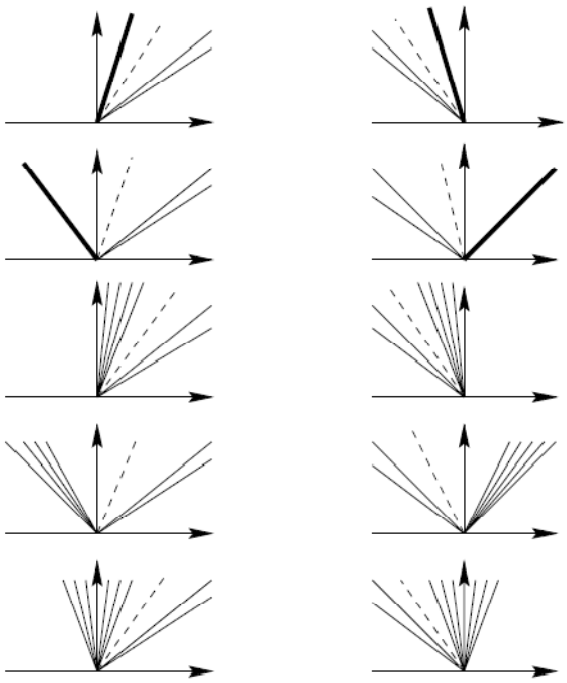


Fig.A.4. Possible wave patterns.

Unfortunately the Godunov method is rather dissipative, since it only 1st order accurate. Thus, higher order schemes must be used to minimize dissipation, such as the MUSCL scheme. An example is shown in the following figure, for initial conditions:

$$\begin{cases} \rho_L = 1100, \ u_L = 100 & x < 0 \\ \rho_R = 1000, \ u_R = 0 & x > 0 \end{cases}$$

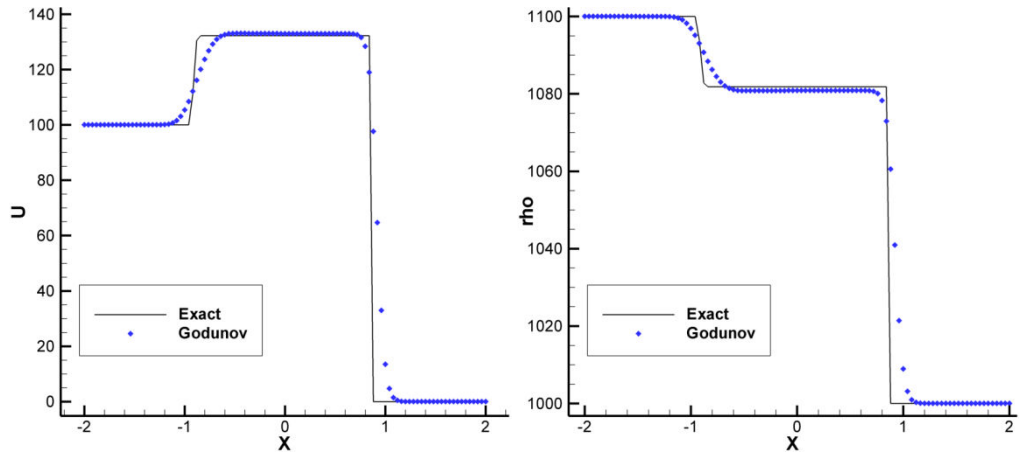


Fig. A.5. Comparison of the exact Riemann solver and the Godunov method. Left: velocity. Right: density.

The dissipation of the method is visible near the rarefaction and the shock waves. An alternative method for solving problems where the variable vector is of the form:

$$\mathbf{U}(t=0) = \begin{cases} \mathbf{U}_L + x\mathbf{U}_L' & x < 0 \\ \mathbf{U}_R + x\mathbf{U}_R' & x > 0 \end{cases}$$

is by solving the Generalized Riemann Problem [4].

It is also possible to solve multi-dimensional problems, by extending the 1d-Godunov method. A multi-dimensional problem may be dealt either by solving the split-*n* dimensional system, where *n*

one dimensional Riemann problems are solved between the adjacent cells, or by solving the *unsplit* approach [2]. The solution of a 2d problem:

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x + \mathbf{G}(\mathbf{U})_y = 0 \tag{A.67}$$

where:

- **U** is the conservative variable vector: $\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \end{bmatrix}$
- **F**(**U**) and **G**(**U**) are the flux vector: $\mathbf{F}(\mathbf{U}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \end{bmatrix}$, $\mathbf{G}(\mathbf{U}) = \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \end{bmatrix}$

with the *unsplit* approach, is the following:

$$\mathbf{U}_{i,j}^{n+1} = \mathbf{U}_{i,j}^{n} + \frac{dt}{dx}\left[\mathbf{F}_{i-\frac{1}{2},j} - \mathbf{F}_{i+\frac{1}{2},j}\right] + \frac{dt}{dy}\left[\mathbf{G}_{i,j-\frac{1}{2}} - \mathbf{G}_{i,j+\frac{1}{2}}\right] \tag{A.68}$$

where $\mathbf{F}_{i-\frac{1}{2},j}, \mathbf{F}_{i+\frac{1}{2},j}, \mathbf{G}_{i,j-\frac{1}{2}}, \mathbf{G}_{i,j+\frac{1}{2}}$ are the solutions of Riemann problems between cells {(i-1,j), (i,j)}, {(i,j), (i+1,j)}, {(i,j-1), (i,j)}, {(i,j), (i,j+1)} respectively. A similar solution is obtained for 3d problems.

In cases where there is an axis (axis-symmetric problem/cylindrical symmetry) or point (spherical symmetry) of symmetry, it is possible to obtain a solution, by solving the one-dimensional inhomogeneous system of PDEs:

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_r = \mathbf{S} \tag{A.69}$$

where:

- **U** is the conservative variable vector: $\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \end{bmatrix}$
- **F**(**U**) is the flux vector: $\mathbf{F}(\mathbf{U}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \end{bmatrix}$
- **S** is the source term: $\mathbf{S} = -\frac{a}{r}\begin{bmatrix} \rho u \\ \rho u^2 \end{bmatrix}$ (r is the radial distance). For cylindrical symmetry a is equal to 1 and for spherical symmetry 2.

According to Toro [2], such problems are treated by solving first the homogenous system (step 1) and then evolving the variable vector in time using the source term (step 2). The procedure is the following:

$$\text{Step 1: } \begin{Bmatrix} PDE: & \mathbf{U}_t + \mathbf{F}(\mathbf{U})_r = 0 \\ IC: & \mathbf{U}(x, t^n) = \mathbf{U}^n \end{Bmatrix} \Rightarrow \overline{\mathbf{U}}^{n+1} \tag{A.70}$$

$$\text{Step 2: } \begin{Bmatrix} PDE: & \mathbf{U}_t = \mathbf{S}(\mathbf{U}) \\ IC: & \overline{\mathbf{U}}^{n+1} \end{Bmatrix} \Rightarrow \mathbf{U}^{n+1} \tag{A.71}$$

High order extensions are available for multidimensional and one-dimensional axis-symmetric problems [1, 2].

References

[1] R. J. Leveque, “Finite volume methods for hyperbolic problems”, Cambridge University Press, ISBN 0-511-04219-1.

[2] E.F. Toro, “Riemann Solvers and Numerical methods for fluid dynamics: a practical introduction”, 3rd edition Springer, ISBN 978-3-540-25202-3, e-ISBN 978-3-540-49834-6, DOI: 10.1007/978-3-540-49834-6

[3] M.J. Ivings, D.M. Causon, E.F.Toro, “On Riemann solvers for compressible liquids”, International Journal for Numerical Methods in Fluids, vol. 28, p.395 – 418, 1998.

[4] M. Ben-Artzi, J. Falcovitz, “Generalized Riemann Problems in computational dynamics”, Cambridge Monographs on Applied and Computational Mathematics, April 14, 2003, ISBN-10: 0521772966, ISBN-13: 978-0521772969.

Appendix B

Implementation details of the SPH-based algorithms

In this section there is a description of the basic operation of the SPH-based algorithms which were developed in the course of the present study. SPH algorithms were developed from scratch, in Fortran programming language, using OpenMP directives for the effective parallelization of the program (see appendix C).

In all SPH algorithms each field variable is represented by a one-dimensional array. The index of the array represents a specific particle. All arrays are declared with a maximum index (*NP*), which represents the maximum number of particles that may be involved in the simulation. On the other hand, the total number of particles simulated is represented as *NTOT* and it is always less than or equal to *NP*. The total number of particles may change during a simulation since more particles may be added simply by increasing the corresponding *NTOT* value. Particle interactions are performed by operations between array elements corresponding to the appropriate field variables of the respective particles involved in interactions.

The algorithm may be divided in several distinct parts (see also fig. B.1 and B.2):

- The case set-up, where fluid parameters and particle discretization are given.
- The definition of initial and boundary conditions. Essentially this part consists of creating particle patches which describe the simulated problem and properly assigning particles to specific types, such as fluid particles, symmetry particles, or wall boundary particles.
- The creation of the neighbor list, by creating a background grid and assigning particles to cells.
- The calculation of the time derivatives of the field variables ($[\rho, u, v, w]^T$ or $[\rho, \rho u, \rho v, \rho w]^T$ depending on the implementation of the SPH algorithm) for each particle. The time derivatives of the field variables are used for the explicit integration scheme. A specific subroutine is used to perform this task, referred as *CALCULATE* subroutine in fig.B.1 and B.2. The same subroutine is used to calculate the time derivative of the turbulent kinetic energy (k) and the turbulence dissipation (ε) or the specific dissipation rate (ω), in case of simulating cases with turbulent effects, such as the turbulent backward facing step.
- Particle renumbering in order to remove particles which move beyond the extents of the described problem domain.
- Implementation specific parts which depend on the nature of the algorithm, such as the density filter, the XSPH correction for the standard SPH method, or the spatial derivative calculation of field variables for the MSUCL scheme.
- Data files are generated periodically containing the coordinates of the particles along with the field variables.

The manipulation of the neighbor list is a crucial part of the whole algorithm, since in this way it is possible to find efficiently neighboring particles. Apart from the dedicated algorithm section creating of the neighbor list, all other parts of the algorithm involving particle interactions also involve neighbor list manipulation. For that reason, it will be the first algorithm element to be analyzed.

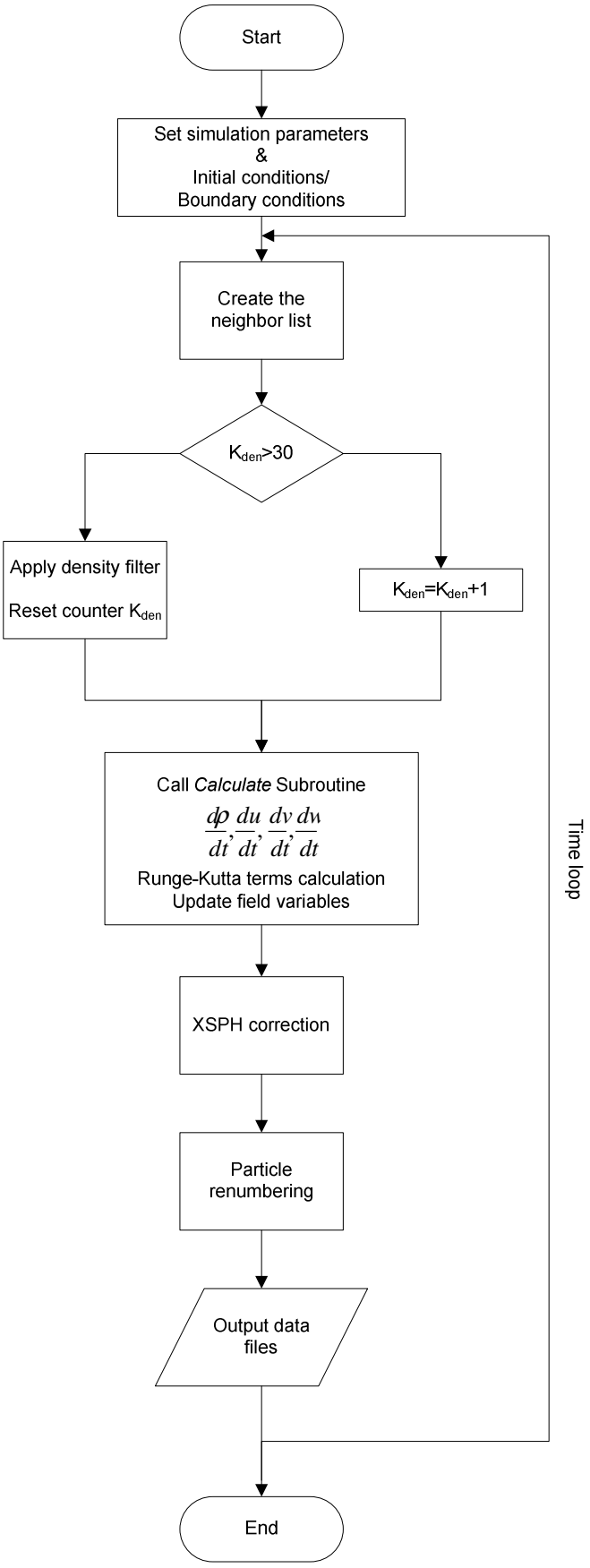


Fig. B.1. Diagram of the SPH algorithm.

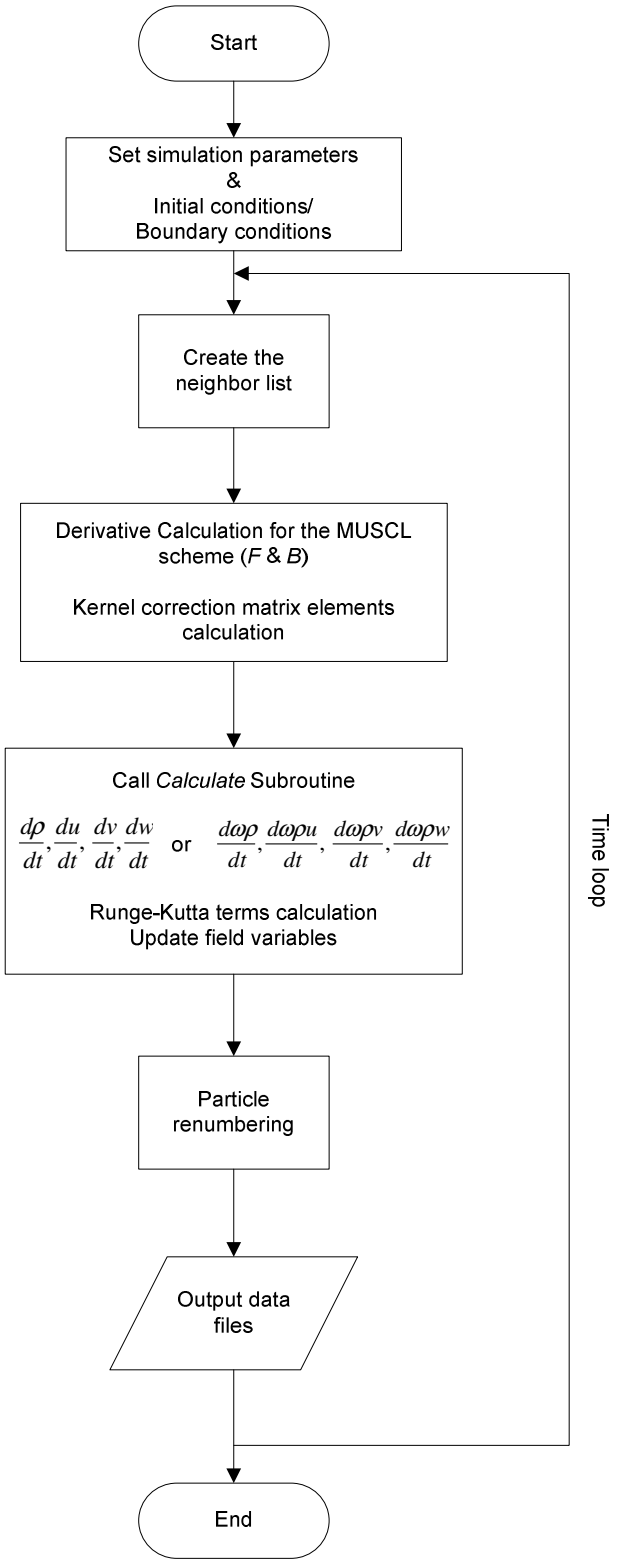


Fig. B.2. Diagram of the SPH-R and SPH-ALE algorithms.

In the following part, there is detailed description of the algorithm used for the neighbor list. The present implementation is possible to handle extension or contraction of the computational domain and the only parameters required to be set are the total number of cells and the maximum number of particles per cell.

The creation of the neighbor list is shown using a simplified diagram in fig. B.3. First the extents of the computational domain are found (X_{max}, X_{min}, …). Then, the number of cells, at each direction, is found as the integer part of the division of (X_{max}-X_{min}) with the cell dimension C_{dx}, which is set equal to the support radius of the kernel function (2.5h for the quartic spline). After that, particles are assigned to the neighbor list, using the following procedure:

- First, particle i is found at which cell it belongs, i.e. the coordinates of this cell (N_{Xgrid}, N_{Ygrid}, N_{Zgrid}) are found (see also fig. B.4).
- From the cell coordinates the cell index number is found (N_{POS} : N-position).
- For this cell index the *NPC* matrix is increased by one (*NPC*: Number of Particles in Cell)
- The particle i is assigned at the *JJJ*(N_{POS}, *NPC*) array element.

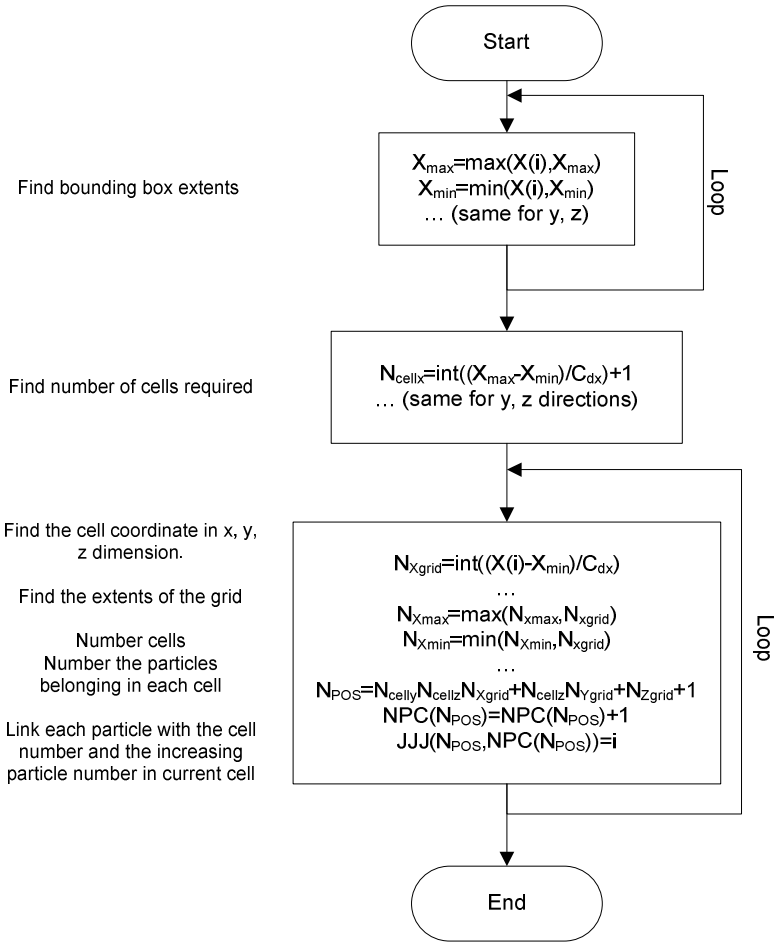


Fig. B.3. Creation of the neighbor-list.

In the previous procedure, there is also a if-statement checking whether the total number of cells (found by $N_{cellX}\ N_{cellY}\ N_{cellZ}$) is greater than the maximum number of cells set as parameter in the algorithm. Exceeding the maximum number of cells would cause a segmentation violation when assigning *NPC*(N_{POS}) values, since N_{POS} would exceed the predefined array size. The other parameter

which has to be considered is the maximum value of $NPC(N_{POS})$ since it affects the size of the *JJJ* array.

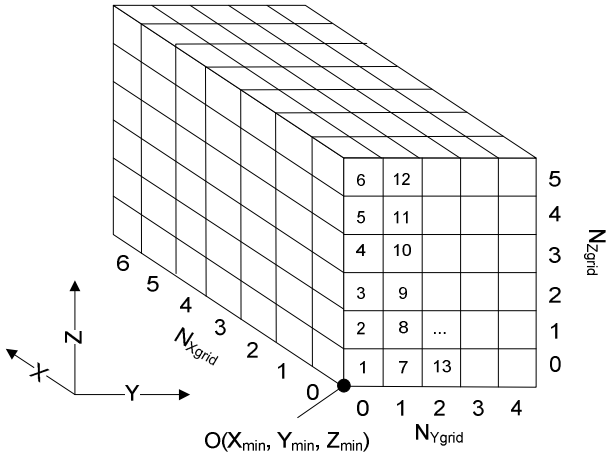


Fig. B.4. Cell index numbering at cell centers (N_{POS}) and cell coordinates N_{Xgrid}, N_{Ygrid}, N_{Zgrid}

To recall particles for calculating the inter-particle interactions of particle *i*, the opposite procedure is used (see also fig. B.5). First, the cell in which particle *i* resides, is found. Then, the neighboring cells are found; these cells are simply found by adding or subtracting 1 from the *X*, *Y*, *Z* cell coordinates in which particle *i* is located. Several loops are performed for all neighboring cells at each *X*, *Y*, *Z* direction, finding the number of particles they contain (*NPC*) and looping again for that *NPC*. The *j* neighboring particle is found using the neighbor list *JJJ* for the neighboring cell position index N_{POS} and for the current value of the particle loop index (from 1 to *NPC*).

Particle position index (N_{POS}) is used in order to avoid calling the neighbor list array *JJJ* using four arguments (in 3D: three arguments for the cell coordinate and one argument for the particle number inside the cell). In this way only one parameter is required for the size of *NPC* and two for the *JJJ* array. Moreover, using only one array argument, instead of three, helps to speed up the execution of the algorithm. Calculation of the N_{POS} is done by counting cells first in the Z direction, then in the Y direction and finally in the X direction.

The described procedure enables great reduction of the computational load especially for large particle numbers. Considering that particles move in respect to each other, throughout a simulation, it is obvious that this procedure takes a significant part of the total computational load and greatly affects the algorithm efficiency. Also, as mentioned earlier, it only requires two parameters, the maximum number of cells and the maximum number of particles per cell. The disadvantage of this method, though, is the increased memory requirement.

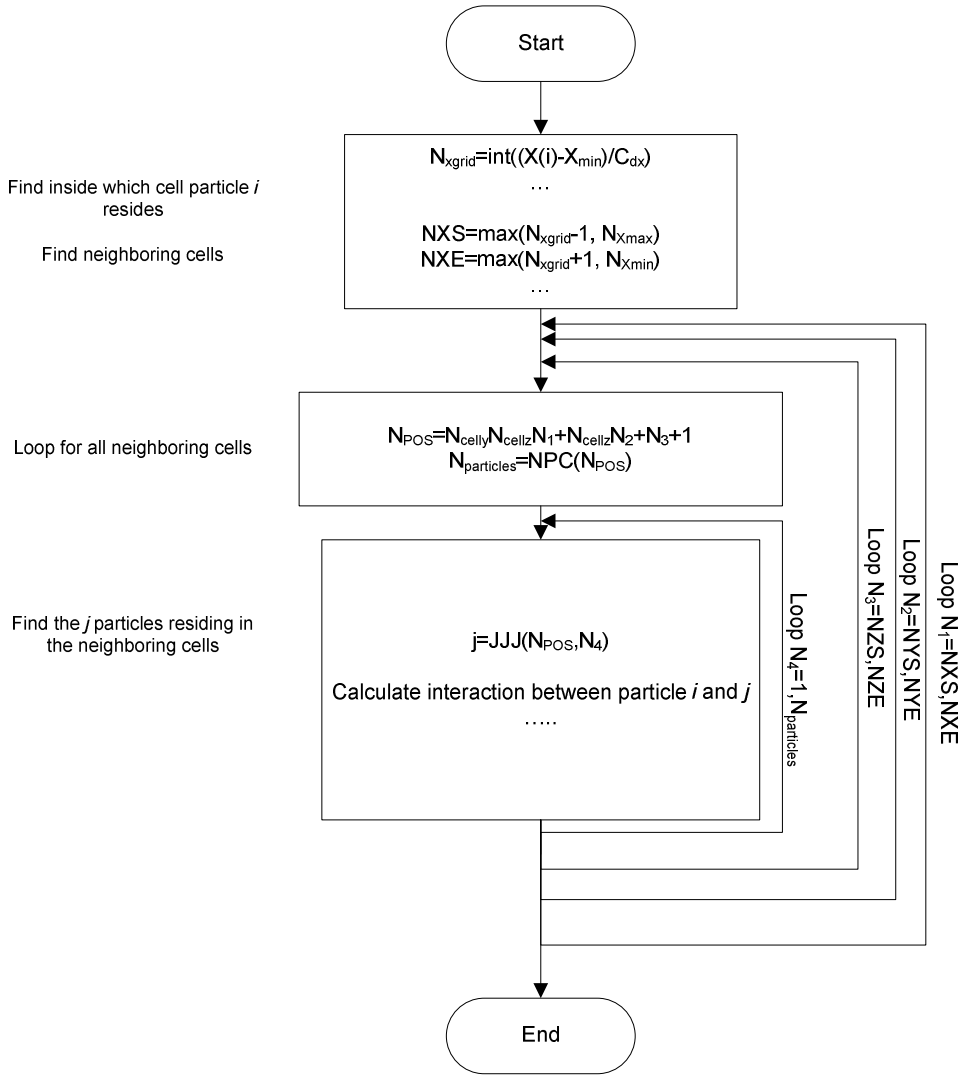


Fig. B.5. Recalling particles from the neighbor list to find interactions.

SPH algorithm

The next step in the SPH algorithm is the density filter. As shown in fig. B.1 the density filter is performed periodically, every 30 time steps; when the K_{den} counter is equal to 30 then the density filter is applied, and also the counter is reset to zero. The density filter is not involved when the density upwind flux formulation term is included (eq. 3.75). It is crucial to highlight that, when using the Shepard or the MLS filter, a temporary array must be involved for storing the filtered density. In other words, eq. 3.67 or eq. 3.69 reads:

$$\rho_i^{temp} = \sum_j m_j \tilde{W}_{ij} \tag{B.1}$$

After calculating the filtered density for all involved particles, the elements of the temporary density array are transferred to the actual particle density array. The reason behind this implementation is that particle density is involved in the calculation of the corrected kernel in eq. 3.68 or 3.72. Replacing density with the filtered density, before it is calculated for all particles, would yield results with increased diffusion, since the particles which were filtered first would employ the unfiltered density values of their neighbors, whereas particles to be filtered last would use filtered density values of their neighbors. Another important remark is the behavior when interacting with

boundary particles; mirror/symmetry/dynamic particles are considered in the density filter, whereas boundary force particles are not.

Once the density field is filtered, the *CALCULATE* subroutine is called for the calculation of the time derivatives of field variables, which are necessary for the explicit integration scheme. This subroutine shares most of its variables, using *common variables*. The subroutine is divided in two separate parts, with each part dealing with two distinct particle types; the actual fluid particles and the boundary particles. For the actual fluid particles, the continuity and momentum equations are solved (and turbulence transport functions in case of turbulent effects). On the other hand, boundary particles are treated according to the boundary implementation, i.e.:

- In case of force boundary particles, the Lennard-Jones acceleration is exerted on the boundary particle and the appropriate reaction on the interacting fluid particle. Forces are summed on the boundary to obtain total forces.
- In case of dynamic boundary particles, the continuity equation is solved, but their velocity is prescribed.
- Mirror particles are generated symmetrically to a plane/axis of symmetry. No equation is solved for these particles.

After the time integration the fluid variables are updated and the XSPH correction is applied. As with the density filter, several additional arrays are required for storing the corrected particle velocity, with each array storing one velocity component. Wall boundary particles are not considered in the XSPH interactions, since this would result to an artificial viscosity effect.

SPH-R and SPH-ALE algorithms

The important difference in the SPH algorithms involving Riemann solvers is the calculation of the spatial derivatives of field variables for the MUSCL scheme. Instead of a density filter, these algorithms include a section calculating the spatial derivatives of field variables at each time step. If partial Riemann boundary conditions are used, then the spatial derivative calculation is divided in two separate parts each one treating the actual fluid particles (volume elements) and the boundary particles (surface elements) using the appropriate formula (see eq. 2.7). It is possible to include in this part the calculation of the correction matrix elements for the kernel renormalization procedure. In this way it is possible to avoid using an additional loop.

After the calculation of the spatial derivatives, field variable time derivatives are calculated for the explicit time integration. A similar to the SPH algorithm *CALCULATE* subroutine is employed, which treats particles appropriately depending on their type. Interactions between fluid particles are treated, by first solving the Riemann problems between them, assuming the one particle is the left state and the other the right state. The solution of the Riemann problem involves the extrapolation and limiting of field variables at the interface between particles. Once the Riemann problem solution is obtained, eq. 6.7-6.8 or eq. 7.4-7.9 are used for the SPH-R and SPH-ALE methods respectively. Interactions between fluid particles and surface elements are treated by solving the respective partial Riemann boundary problem and using eq. 6.30-6.31 and 7.29-7.33 (for the SPH-R and SPH-ALE methods respectively). Low speed preconditioning may be used in the SPH-ALE algorithm in nearly incompressible cases such as the water jet impingements, but not in cases such as the explosion or implosion where compressibility effects become dominant. After calculating all particle interactions, the time derivatives of the field variables are returned to the main program for the calculation of the Runge-Kutta terms.

In all SPH-based algorithms particles exceeding the prescribed domain size are removed by renumbering, after updating field variables at the new time step. The renumbering procedure consists of rewriting particle arrays after checking whether specific conditions are satisfied. If a particle satisfies these conditions, then it is not rewritten; on the other hand the corresponding array elements are overwritten by the next particle which does not satisfy the specified conditions. At the end of this procedure, the total number of particles involved is set to its new value. A simple code fragment of the particle renumbering process, where the condition is a particle's *x*-coordinate should not exceed *xmax*, is shown below:

```
C ------------------------------------------------------------------
C –-------          Begin renumbering           -----------

        inew=0                      ! this is an initial value for the new index
        do i= 1,ntot                ! loop for all particles
        if (X(i).gt.xmax) cycle     ! the condition is if the x-coordinate of a particle exceeds xmax

        inew=inew+1                 ! update the particle index value
        X(inew)=X(i)                ! rewritte array elements
        Y(inew)=Y(i)
        Z(inew)=Z(i)
        rho(inew)=rho(i)
        …

        enddo

        ntot=inew                   ! set the new total maximum number of particles
C ------------------------------------------------------------------
```

Appendix C

Parallelization with OpenMP [1]

OpenMP enables the creation of shared memory parallel (SMP) programs. Actually OpenMP is a *set of directives* which are supported by many Fortran/C/C++ compilers under various platforms (Windows, Linux, etc.) in order to parallelize an algorithm. It must be highlighted here that OpenMP should not be considered as a new programming language, but rather as a notation which can be added in the source code of a sequential program to describe how work will be shared among threads which will execute on different processors or processor cores and how data will be shared.

A thread is defined as the minimum set of instructions which can be issued for execution by an operating system. A process, on the other hand, can be considered as a superset of threads, since it may contain one or many threads. Threads of the same process share the same resources, while processes do not. When executing a program, the operating system creates a process to execute the program. Then the process allocates resources and creates one or more threads for the execution. Multiple threads may be executed on a single processor via context switches; via simultaneous multithreading. On the other hand, threads working concurrently on multiple processors or cores execute a parallel program.

When an algorithm using OpenMP directives is executed, the program starts an *initial thread* (or *master thread*) of execution. In this way the serial part of the program is executed. Once the program reaches to an *OpenMP Parallel* directive, then it creates a team of threads (called as a *fork*, the number of threads depend on other OpenMP directives as it will be mentioned later on). The initial thread becomes the master of the team and collaborates with the rest of the threads to execute the code enclosed in the OpenMP parallel construct. At the end of the construct there is the *OpenMP End Parallel* directive; when it is reached all threads synchronize and then only the master thread continues, the rest threads are terminated (this part is also called as the *join*). The enclosed region, inside the parallel directives, is called a parallel region. A simple schematic of such a program is shown in fig. C.1.

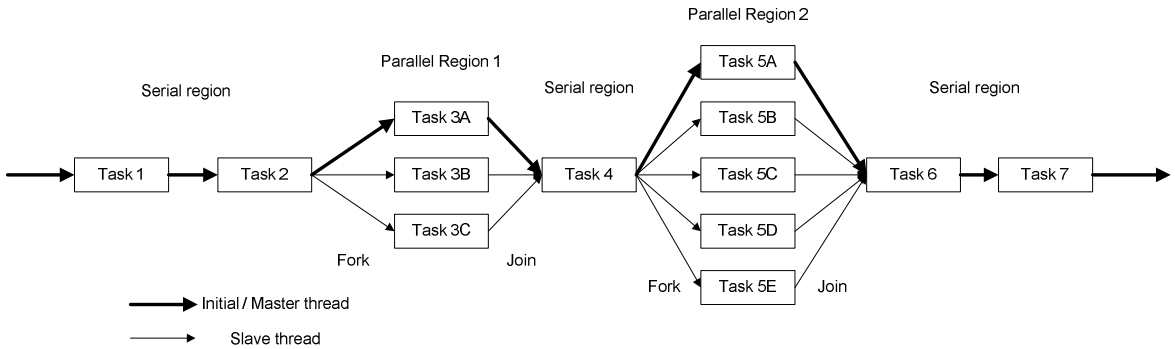


Fig. C.1. A program containing serial and parallel regions

As it was mentioned earlier parallel regions are defined using OpenMP directives. OpenMP directives in Fortran begin with *$OMP*; the *$* character is considered as a comment if the appropriate compiler flags (particularly the –openmp flag, invoking the compiler OpenMP directives) are not given during compilation. This fact enables to use the same source code for parallel and serial

compilation – in serial compilation the OpenMP directives are treated as comments. The directives required, to define a parallel region in Fortran, are:

```
C ----------------------------------------------------------------
$OMP PARALLEL  ….(directive arguments)…

! this is the parallel region
…(Source code to be executed in parallel)…

$OMP END PARALLEL
C ----------------------------------------------------------------
```

In the present SPH algorithm, the regions defined as parallel OpenMP regions are those involving:

- Calculation of the rate of change of particle characteristics (i.e. calculation of acceleration and density rate of change)
- Density filter
- Kernel gradient correction
- XSPH correction
- Spatial derivative calculation, required for the MUSCL procedure

All the aforementioned tasks involve particle interactions and are computationally intensive taking up a significant part of the total program execution, thus running on parallel greatly reduces calculation time. In these cases a loop is performed for all *i* particles, calculating contributions from their neighbors *j*. Thus a parallel *DO* construct is used. When the OpenMP *DO* directive is reached, the loop is divided and each thread is assigned a part of it. Special care must be taken to avoid *data race conditions*. Data race conditions occur due to the shared memory nature of OpenMP. Generally this issue occurs when at least two threads access a shared memory location and at least one modifies it. Thus, the thread reading the memory location might get the old value or the updated one, or some other erroneous value if the update requires more than one store operation. Moreover this issue is not reproducible, since it is affected during run-time by the execution of the threads and it may not always show-up. Due to its definition, data race occurs only for *shared* variables between threads.

Data race is affected by:

- System load, which affects the timing between threads
- Input data set, which may cause load imbalance between threads
- Number of threads used.

Thus, it is imperative to declare the intermediate variables in a parallel DO-loop region as *private* for each thread. In that case each thread keeps a private, individual copy of this variable and only the specific thread is able to access it. Once the parallel region ends, all the private variables are deleted.

A sample code of such a parallel region is given below, including a small part of the neighbor list. Note that the *i* loop counter and the *NXgrid*, *NYgrid*, *NZgrid* variables must be declared as private for each thread. As a general rule of thumb, every variable modified inside a parallel region should be declared as private, or proper synchronization should be used alternatively, to avoid data race conditions. In the sample code below, *NUM_THREADS*() defines how many threads will be created at the parallel region.

```
C ---------------------------------------------------------------
$OMP PARALLEL  PRIVATE (I, NXGRID, NYGRID, NZGRID, …. )  NUM_THREADS (8)
$OMP DO

do i=1,ntot     ! ntot is the N number of particles involved in the simulation
NXgrid=int((X(i)-Xmin)/cdx)
NYgrid=int((Y(i)-Ymin)/cdy)
NZgrid=int((Z(i)-Zmin)/cdz)
….(rest source code)…

enddo

$OMP ENDDO
$OMP END PARALLEL
C ---------------------------------------------------------------
```

Sometimes it is required to sum the values of a private variable for all threads. In such a case, a *REDUCTION* directive is used. Declaring a variable to be used for reduction, results to the creation of a private variable for each thread, but at the end of the parallel region all private values of this variable are reduced according to a specified rule (reduction may be used for addition, finding maximum and minimum etc.). This can be used, for example, to calculate the net forces on all boundary particles, as it is shown on the following source code part.

```
C ---------------------------------------------------------------
$OMP PARALLEL  PRIVATE (I,…) REDUCTION(+ : SFX, …)  NUM_THREADS (8)
$OMP DO

do i=1,ntot     ! ntot is the N number of particles involved in the simulation
….(source code)…

SFX=SFX-acc_x*amass(i)

….(source code)…
enddo

$OMP ENDDO
$OMP END PARALLEL
C ---------------------------------------------------------------
```

However there are cases where it is preferable to know the force on each boundary particle separately, in order to estimate torque, or for plotting the force distribution on the boundary. In such case a matrix is required for storing the boundary forces for each boundary element from the contributions of the fluid particles (see eq. 3.87 in chapter 3). Unfortunately, reduction of a matrix is impossible (or, to be precise, was impossible with OpenMP 2.0 – the newest implementation of OpenMP 3.1, released on July 2011 supports matrix reduction). In that case the following procedure is used:

- A temporary two dimensional matrix is allocated. The one dimension of the matrix is the same as the force matrix, which will be used for storing the forces on each individual particle, and the other dimension is equal to the number of threads.
- Each thread is assigned a unique *ID* number with which stores the calculated forces on the temporary matrix
- Each thread accesses the temporary array with its unique ID, writing forces from interactions.
- After the parallel region, the temporary arrays are manually reduced to their permanent location (without an OpenMP directive) and temporary arrays are de-allocated.

```
C ---------------------------------------------------------------
Num_procs=8
Call omp_set_num_threads(num_procs)          ! this subroutine sets the thread number
Allocate (Fx_temp(1:nwall,1:num_procs),…)    ! allocating memory for the temporary array

$OMP PARALLEL  PRIVATE (I,ID_OMP,…)
Id_omp=omp_get_thread_num()+1                !this function returns a unique number for each
thread

Do i=1,nwall
Fx_temp(i,id_omp)=0.                         ! initializing the temporary array
Enddo

$OMP DO

do i=1,ntot                 ! ntot is the N number of particles involved in the simulation
….(source code)…

Fx_temp(j,id_omp)= Fx_temp(j,id_omp)-acc_x*amass(i)

….(source code)…
enddo

$OMP ENDDO
$OMP END PARALLEL

Do i=1,nwall                ! manual reduction of the array
Do j=1,num_procs
Fx(i)=Fx(i)+Fx_temp(i,j)
Enddo
Enddo

Deallocate (Fx_temp,…)      ! free memory
 C ---------------------------------------------------------------
```

In a similar manner it is possible to parallelize the linked list creation, since it involves an array reduction. However in practical applications it showed insignificant or even negative speed-up. This fact has to do with the number of particles involved in the simulation; if the particle number is small,

the benefit from the parallelization is small whereas there is significant calculation overhead from allocating / de-allocating temporary arrays, the reduction etc.

References

[1] B. Chapman, G. Jost, R. Pas, "Using OpenMP: Portable Shared Memory Parallel Programming", MIT Press, ISBN-10:0-262-53302-2, ISBN-13: 978-0-262-53302-7.

Appendix D

Details on the numerical schemes used by Fluent® software [1]

The Fluent software is a general purpose CFD software, able to simulate various types of flows, involving laminar and turbulent flows, single phase and multi-phase, compressible and incompressible, steady and transient. The Fluent solver is based on the Finite Volume method, thus it employs an Eulerian perspective; however it is possible to employ different frames of motion or sliding or dynamic meshes in order to describe the effects of rotating or moving geometries. The solver handles meshes in an unstructured manner and it is able to handle 2D and 3D meshes with arbitrary element shape (in 2D: quadrilateral / triangular cells and in 3D: tetrahedral / hexahedral / pyramid / wedge / polyhedral cells, the solver also supports mixed, i.e. hybrid, meshes). The computational meshes in the present study were created using the GAMBIT software or the Design Modeler and Mesher of the ANSYS Workbench package.

In the present study Fluent is entirely used for incompressible simulations, thus the *pressure based* implicit solver is used. Fluent simulations on turbulent and laminar flows (see chapter 4) are treated as steady state. The numerical schemes for such simulations are:

- SIMPLE method for pressure-velocity coupling
- Second order discretization for pressure
- Second order upwind discretization for momentum

When turbulent effects are accounted, second order upwind discretization is used for the additional turbulence transport equations for k, ε or ω. Moreover the computational mesh used is properly refined in near wall regions, in order to satisfy the y^+ condition of the employed standard wall functions ($30<y^+<300$).

All free surface flows are treated using the Volume of Fluid (VOF) method simulating two phases: water and air. With the VOF method it is possible to model two or more immiscible fluids by solving a single set of momentum equations and tracking the volume fraction of each of the fluids throughout the domain. In the present study, the *explicit VOF formulation* has been used, since it enables better interface capturing, especially combined with the *Geo-Reconstruct* scheme. According to the Fluent software manual [1], the geometric reconstruction interpolation scheme should be used whenever the main interest is the time-accurate transient behavior of the VOF solution. However the explicit VOF formulation imposes restrictions on the maximum time step, since the CFL number is suggested to be well below unity.

To sum up, free surface flows are treated using the pressure-based unsteady implicit solver and with the following numerical schemes:

- PISO for pressure-velocity coupling
- Body force weighted discretization for pressure
- Second order upwind discretization for momentum
- Explicit VOF formulation with geo-reconstruct discretization for volume fraction

In stationary free surface simulations, after reaching the steady state, mesh adaption is performed, based on the gradient of the volume fraction. In this way the mesh resolution is increased near the free-surface and diffusion of the volume fraction is substantially decreased, increasing the accuracy of the free surface location (see fig. D.1).

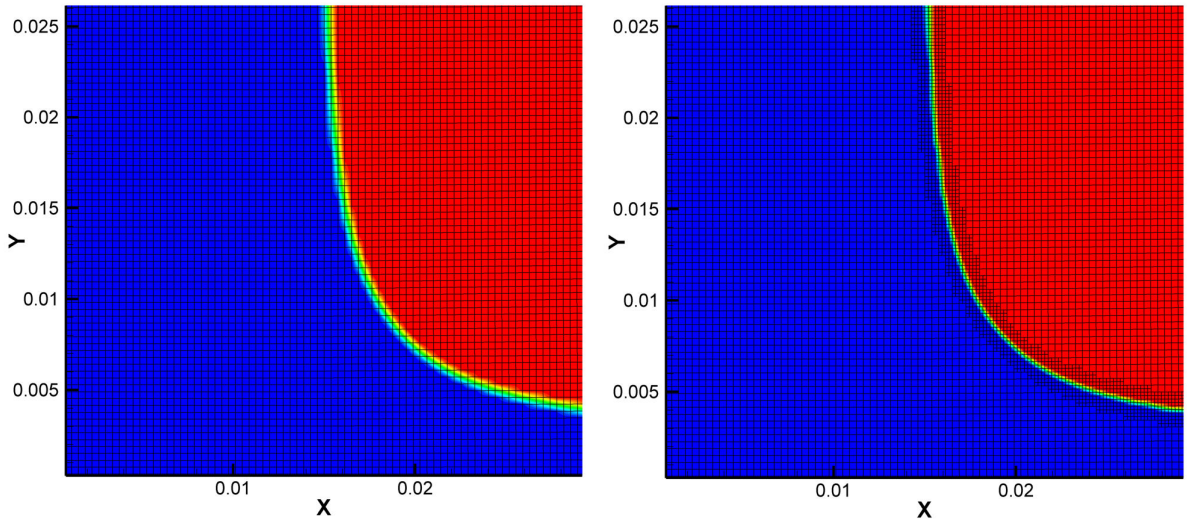


Fig. D.1. Effect of mesh adaption. Left: original mesh. Right: Mesh with adaption based on the free surface gradient. Contours represent the volume fraction.

Cases involving rotating meshes (such as the Turgo or Pelton turbine simulations), are simulated using the sliding mesh approach. Two unconnected computational meshes are constructed with each one representing the moving and the stationary mesh. One face of both meshes is defined as an interface. Interfaces enable information to be passed between the two different computational domains. At each time step, the intersection of the two interfaces is determined; the intersection is considered as an interior face (i.e. a face with fluid cells at its both sides), while the remaining parts of the interface faces are considered as a pre-determined boundary type (such as wall or, in the present study, as pressure outlets in order to allow the fluid to exit).

References

[1] Fluent 14 documentation.

Appendix E

Construction of the Turgo turbine

The surface of the optimal Turgo blade, after the optimization procedure (see chapter 5), is used to construct a 3D blade (fig. E.1). The 3D blade is formed after extruding the surface by a specific thickness at the local normal direction. Runner hub and tip are formed as surfaces of revolution from the turbine blade hub and tip edges (fig. E.2).

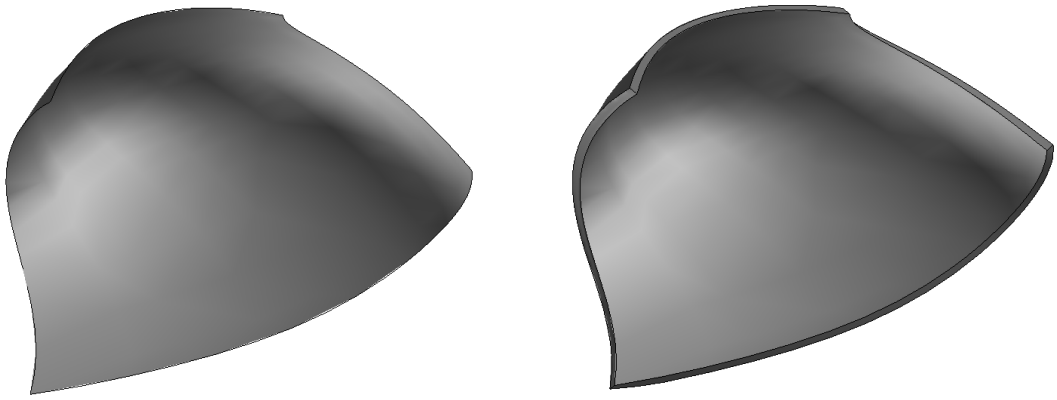


Fig. E.1. Left: Surface representing the blade. Right: Extruded 3D body, forming a full 3D blade.

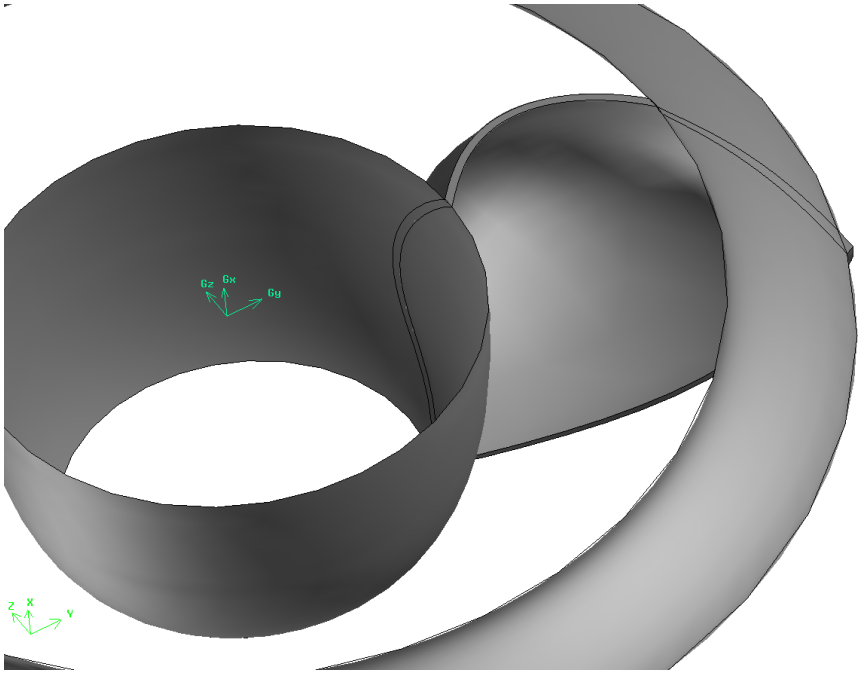


Fig. E.2. Runner hub and tip surfaces.

A small protrusion was created on the blades, in order to fit them on the hub (see fig. E.3). This protrusion had a length of ~3.5mm. The turbine hub was designed with corresponding slots, to accommodate proper fitting (fig. E.3). The 3D drawing of the tip of the runner is shown in fig. E.4.

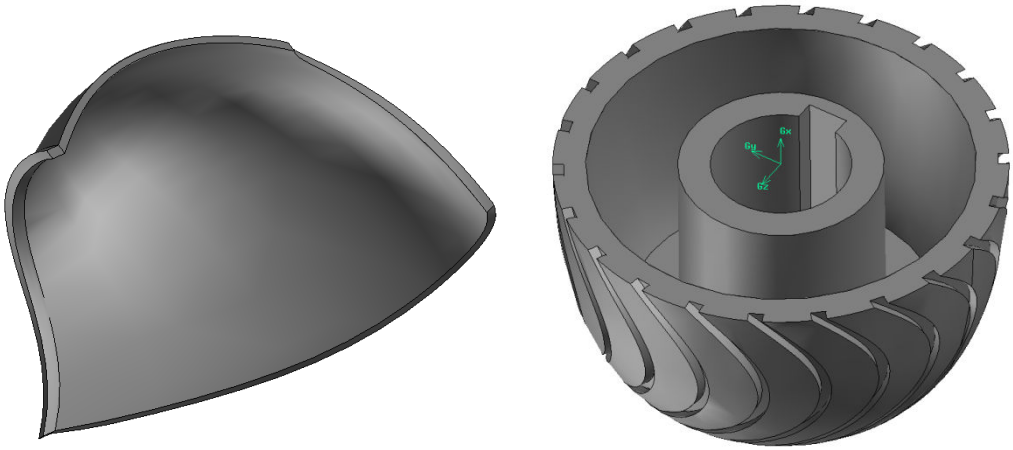


Fig. E.3. Left: 3D blade with protrusion. Right: Turbine hub. Blade slots are visible.

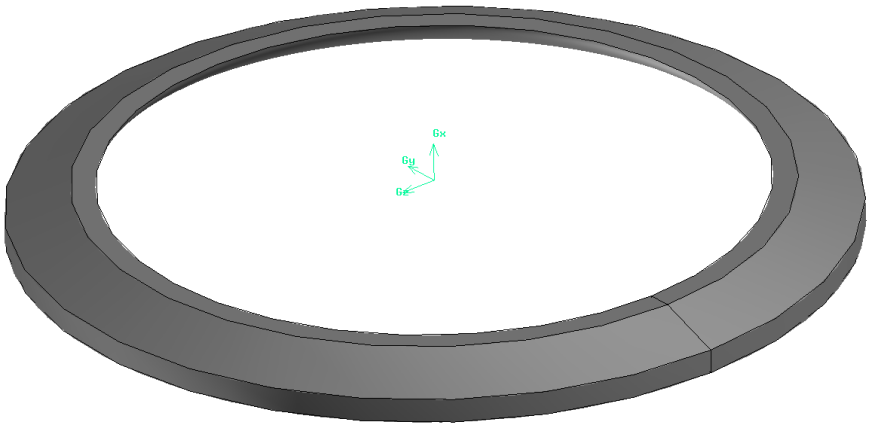


Fig. E.4. Turbine tip.

The turbine runner is made from phosphorus bronze. This particular alloy was selected due to resistance to errosion, durability, and strength. In order to ensure the accuracy of representation at the actual turbine, blades were formed with the following technique:

- A surface triangulation of the 3D blade was made with CAD software (fig. E.5)
- The surface triangulation was used to create a model blade with rapid prototyping. The method of rapid prototyping used was granular material binding. The prototype blade was formed as layers of material binded together. The accuracy of the process was 0.5mm, thus the blade has significant surface roughness (fig. E.5).
- The model blade was used to form a mold for casting. An aluminium cast blade was created and then machined/polished to remove inaccuracies and improve surface roughness.
- The aluminium blade was used to form a new mold, for casting the final blades. The final turbine blades are shown in fig. E.6, after polishing.

The turbine hub was formed by casting. Then, the hub was machined at a 5-axis CNC machining centre, to form the slots shown in fig. E.3. The turbine hub and tip are shown in fig. E.7, before final assembling. The turbine runner was welded with bronze welding and was checked for balancing issues.

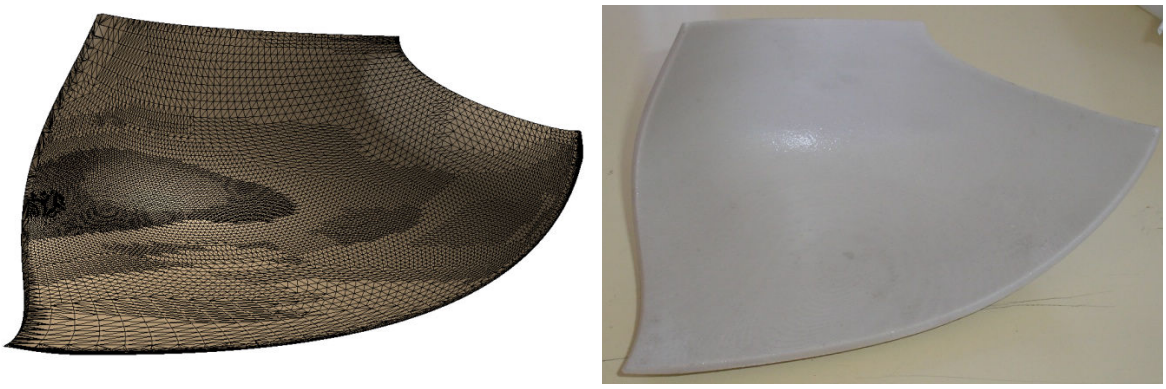


Fig. E.5. Left: STL triangulation of the turbine blade. Right: Composite turbine blade after rapid prototyping.

Fig. E.6. Turgo turbine blades after final casting.

Fig. E.7. Left: Turgo turbine before assembling. Hub slots are visible. Right: Another view of the runner.

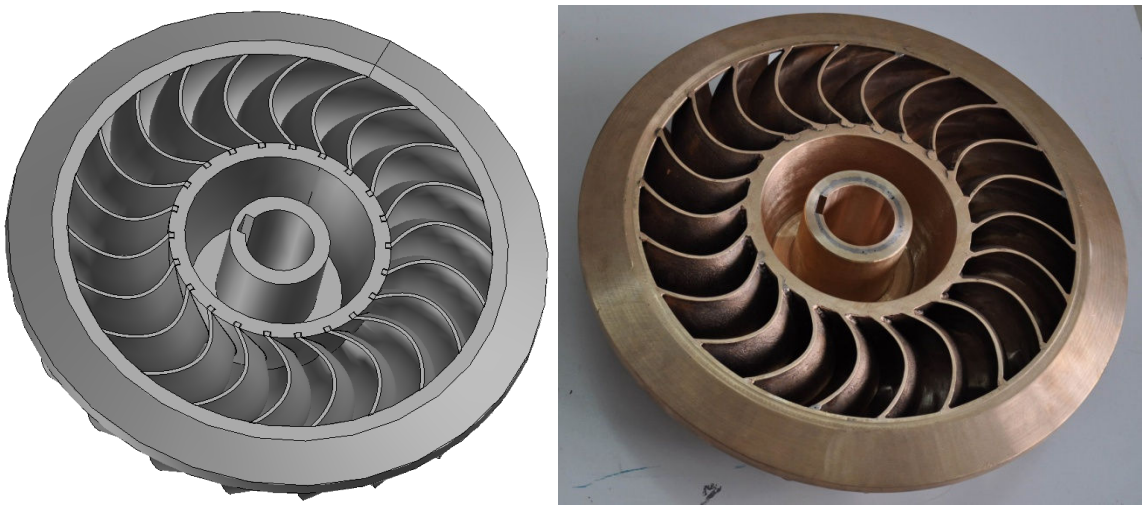
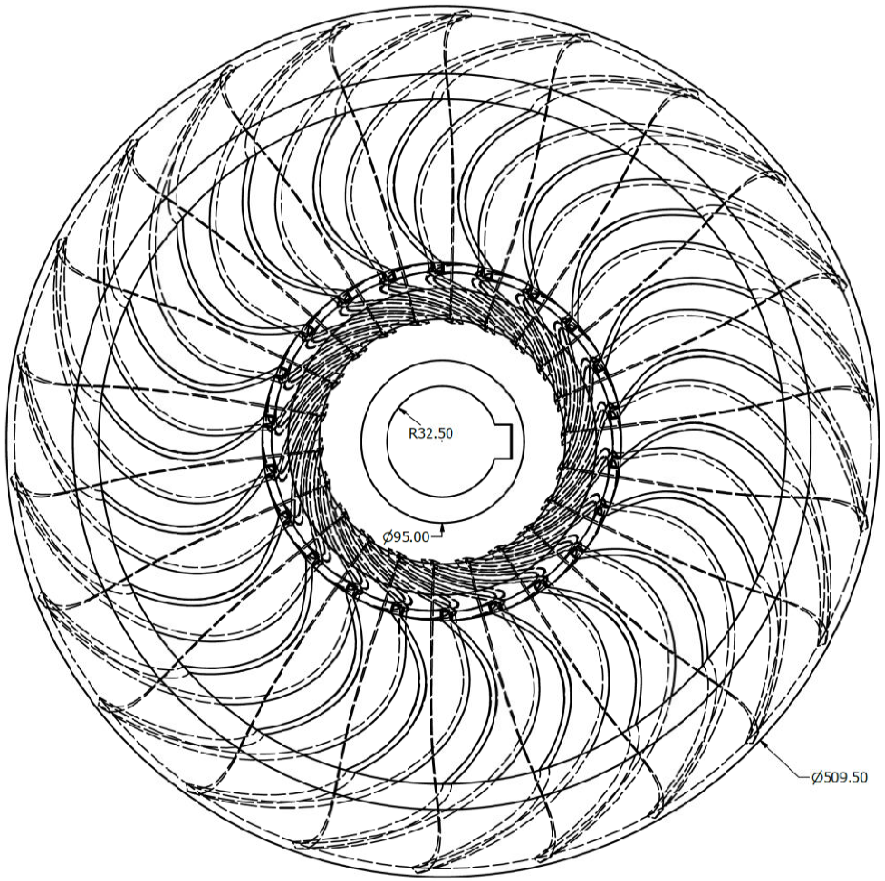
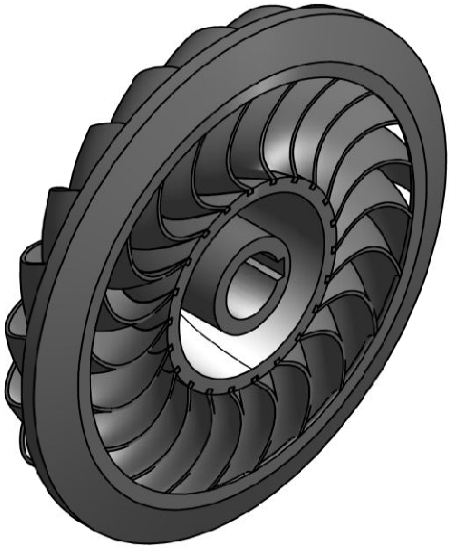
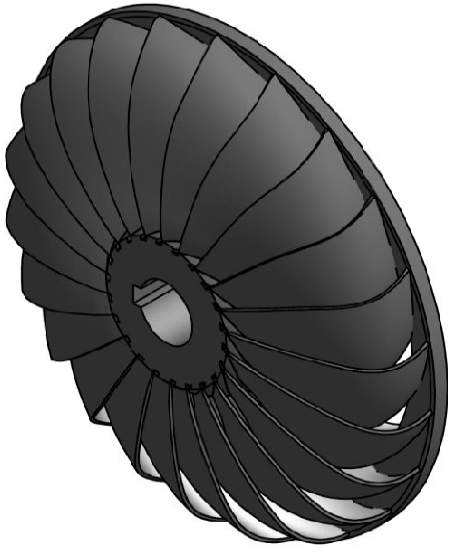


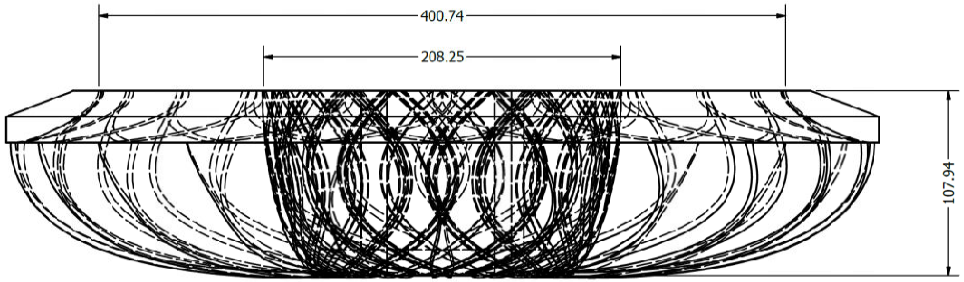
Fig. E.8. Left: Turgo runner 3D CAD drawing. Right: Actual runner after assembling and welding.

In the following pages there are drawings of the turbine runner and casing, along with basic dimensions.

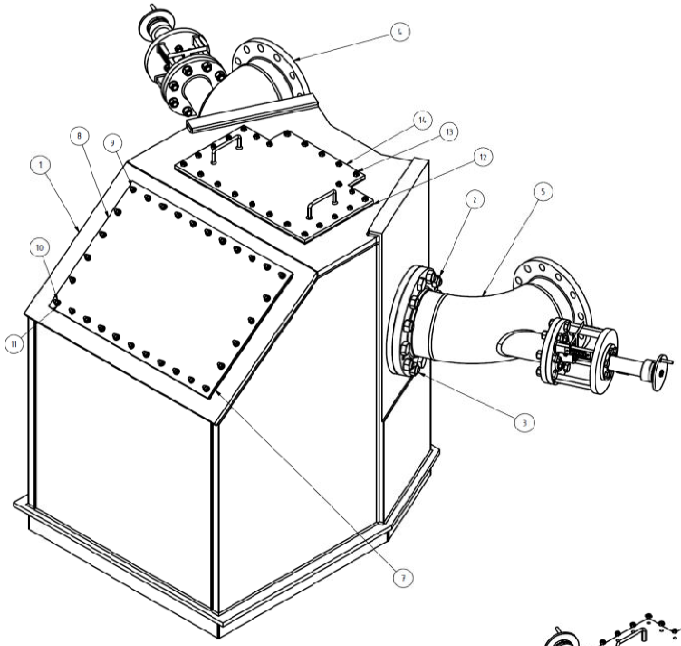


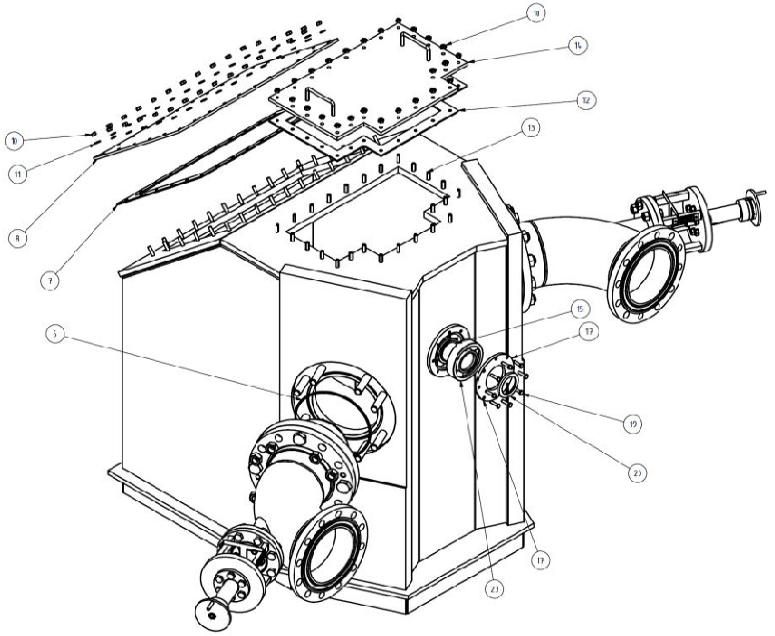


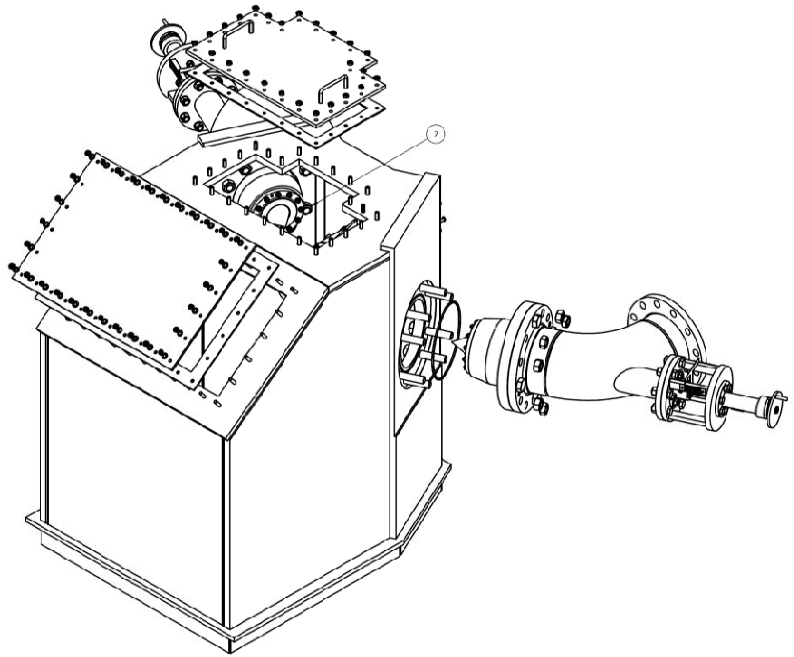




DRAWN Phoevos Koukouvinis	27/04/2011	National Technical University of Athens
CHECKED		TITLE: Turgo turbine runner
QA		
MFG		
APPROVED		
		SIZE / DWG NO / REV
		SCALE







ΠΙΝΑΚΑΣ ΕΞΑΡΤΗΜΑΤΩΝ

α/α	ΠΟΣΟΤΗΤΑ	ΟΝΟΜΑΣΙΑ	ΠΕΡΙΓΡΑΦΗ
1	1	ΚΑΜΠΑΝΑ	
2	32	DIN 555-5 - M36	Hex Nut
3	16	IS 1826 - M30 x 140	Specification for Studs
4	1	ΑΝΩ ΑΚΡΟΦΥΣΙΟ ΥΔΡΟΣΤΡΟΒΙΛΟΥ TURGO	
5	1	ΚΑΤΩ ΑΚΡΟΦΥΣΙΟ ΥΔΡΟΣΤΡΟΒΙΛΟΥ TURGO	
6	2	DIN 5171 - 315 x 7 - N - NBR 70	U-Ring
7	1	ΕΛΑΣΤΙΚΗ ΦΛΑΝΤΖΑ ΣΤΕΓΑΝΟΤΗΤΑΣ ΕΜΠΡΟΣΘΟΥ ΠΑΡΑΘΥΡΟΥ	
8	1	ΥΑΛΟΠΙΝΑΚΑΣ	
9	30	DIN 835 - M12 x 40	Double End Stud
10	56	DIN 555-5 - M12	Hex Nut
11	31	DIN 125 - A 13	Washer
12	1	ΕΛΑΣΤΙΚΗ ΦΛΑΝΤΖΑ ΣΤΕΓΑΝΟΤΗΤΑΣ ΑΝΩ ΚΑΛΥΜΜΑΤΟΣ	
13	25	DIN 835 - M12 x 65	Double End Stud
14	1	ΟΥΡΑΝΙ ΜΕ ΧΕΙΡΟΥΛΙΑ	
15	1	ΑΠΡΑΚΤΟΣ ΠΙΕΣΤΗΡΑ	
17	2	ΚΑΠΑΚ ΦΟΛΙΑΣ ΕΔΡΑΝΩΝ ΚΥΛΙΣΗΣ	
19	16	DIN 931.1 - M10 x 40	Hex Head Bolt
20	2	DIN 5760 - A5 - 80 x 110 x 10 - NBR	Shaft Seal Ring
22	1	SKF NUNU 316 ECJ	Cylindrical Roller Bearings Single Row type NU SKF
23	2	SKF 87316 BE	Single row angular contact ball bearings SKF
24	2	DIN 471 - 80 x 4	Spring Retaining Ring

διαστάσεις χωρίς ανοχές ±0.05 | υλικό

	ημερ.	όνομα	υπογραφή
σχεδ.		Δουκας Χαΐτογλου	
μελετ.			
εγκρ.			

ΜΟΝΤΕΛΟ ΥΔΡΟΣΤΡΟΒΙΛΟΥ TURGO

ΣΥΝΑΡΜΟΛΟΓΗΜΕΝΟ ΣΥΝΟΛΟ

κλίμακα 1:5

Αρ. Σχ. ΤΟΤΑ

Ε.Μ.Π. ΤΟΜΕΑΣ ΡΕΥΣΤΩΝ

ανοχές | αποκλίσεις

PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT

PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT

PRODUCED BY AN AUTODESK EDUCATIONAL PRODUCT