NATIONAL TECHNICAL UNIVERSITY OF ATHENS

SCHOOL OF CIVIL ENGINEERING

DEPARTMENT OF WATER RESOURCES & ENVIRONMENT

A 2D SMOOTHED PARTICLE HYDRODYNAMICS MODEL FOR FREE SURFACE APPLICATIONS

Diploma Thesis

George Pouliasis

Supervisor: P. N. Papanicolaou, Assoc. Professor, NTUA

Athens, October 2017

“Doubt thou the stars are fire,

Doubt that the sun doth move,

Doubt truth to be a liar,

But never doubt I love.”

Letter of Hamlet to Ophelia

Hamlet, William Shakespeare

“Ντροπή στον εργάτη στο σκλάβο ντροπή

αν στο αίμα δεν πνίξει μια τέτοια ζωή”

Παιδιά Σηκωθείτε, Μαρία Δημητριάδη

Αυτή την εργασία την αφιερώνω στους γονείς μου,
τους μεγαλύτερους αγωνιστές που έχω γνωρίσει.

Ευχαριστίες / Acknowledgments (In Greek)

Με την ολοκλήρωση και παράδοση αυτής της διπλωματικής εργασίας έρχεται στο τέλος του ένας μεγάλος κύκλος της φοίτησής μου στη Σχολή Πολιτικών Μηχανικών του ΕΜΠ και της ζωής μου στην Αθήνα. Κλείνοντας αυτό το κύκλο, δεν μπορεί κανείς παρά να αναστοχαστεί τα χρόνια του σαν φοιτητής, να αναμετρηθεί με τις επιλογές του και να απολογήσει τις πράξεις του. Σκεπτόμενος τα παραπάνω, συνειδητοποίησα ότι ακόμα μέσα στους συλλογισμούς μου δεν κατέχουν κυρίαρχο λόγο τα γεγονότα αλλά τα συναισθήματα που μου άφησε ο χώρος που έζησα τα τελευταία πέντε χρόνια. Συναισθήματα χαράς και λύπης, αγάπης και στενοχώριας, ενθουσιασμού και σφοδρής απογοήτευσης. Όλα τυλιγμένα με ένα πέπλο γλυκιάς νοσταλγίας και βαθιά ριζωμένα. Συναισθήματα όμως, τα οποία αποτελούν τεκμήριο ότι όποιο και αν είναι το αποτέλεσμα του απολογισμού μου, ο χρόνος μου δεν ξοδεύτηκε, αλλά χρησιμοποιήθηκε στο έπακρο, παρήγαγε εμπειρίες που με διαμόρφωσαν καθοριστικά και των οποίων τα διδάγματα θέλω να θυμάμαι στο υπόλοιπό της ζωής μου.

Φέρνοντας στο νου όλα αυτά τα χρόνια δεν μπορώ να μην σκεφτώ και να ευχαριστήσω όλους τους ανθρώπους που με τις πράξεις τους και τη δράση τους με επηρέασαν και με ενέπνευσαν, με στήριξαν και με διαμόρφωσαν και προφανώς όσους συνέβαλαν στην εκπόνηση αυτής της διπλωματικής.

Πρώτα και κύρια θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Παναγιώτη Παπανικολάου για την ανάθεση αυτού του εξαιρετικά ενδιαφέροντος θέματος, για την αδιάκοπη υποστήριξη του, την εμπιστοσύνη του και την ελευθερία που μου έδωσε παρόλο το «επιστημονικό θράσος» των επιλογών μου. Όμως η συμβολή του δεν περιορίζεται στα πλαίσια της εκπόνησης αυτής της διπλωματικής. Με τον κ. Παπανικολάου γνωριστήκαμε κατά το 4^{ο} εξάμηνο των σπουδών μου στα πλαίσια του μαθήματος της Μηχανικής των Ρευστών και έκτοτε με τις συμβουλές του καθόρισε σημαντικά την πορεία των σπουδών μου. Τον ευχαριστώ θερμά για αυτό και την υπομονή που έδειξε όλο αυτό τον καιρό.

Περαιτέρω θα ήθελα να ευχαριστήσω τον κ. Δημήτρη Κουτσογιάννη, καθηγητή και κοσμήτορα της Σχολής για τη εμπειρία του μαθήματος που μας προσέφερε, τις ώρες που αφιέρωσε απαντώντας σε απορίες και συζητώντας και για την «ανορθόδοξη πλευρά» της επιστήμης που μας δίδαξε. Ακόμα θα ήθελα να ευχαριστήσω τον κ. Νικόλαο Μαμάση, αναπληρωτή καθηγητή για την εμπειρική σκοπιά της επιστήμης που μας μύησε. Σαφώς δεν θα μπορούσα να ξεχάσω τον κ. Ανδρέα Ευστρατιάδη, Δρ Πολιτικό Μηχανικό για την αγάπη στους φοιτητές και τη παροιμιώδη αφοσίωσή του στο ρόλο του δασκάλου και στο ίδρυμα.

Θα ήθελα επίσης να ευχαριστήσω τον Παναγιώτη Δημητριάδη, Δρ. Πολιτικό Μηχανικό για τη πολύτιμή βοήθεια του τον τελευταίο χρόνο των σπουδών μου. Ο Παναγιώτης αποτελεί πρότυπο νέου επιστήμονα, πηγή έμπνευσης για εμάς που είχαμε τη τιμή να συνεργαστούμε μαζί του.

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω όλους τους συντρόφους και τις συντρόφισσές μου από τον Εγκέλαδο Πολιτικών Μηχανικών. Μέσα από τις μάχες που δώσαμε και από τη καθημερινή μας συλλογική δράση, έλαβα ένα από τα σημαντικότερα διδάγματα των φοιτητικών μου χρόνων, ότι η Δημοκρατία δεν αποτελεί απλά μία διακήρυξη αλλά αποτελεί πράξη καθημερινή για την οποία πρέπει να δίνεται μάχη τόσο σε συλλογικό όσο και σε προσωπικό επίπεδο. Εύχομαι το αίσθημα της περηφάνιας που γεμίζει εμένα σήμερα για τη συλλογική μου δράση μέσα στα πλαίσια του

Εγκέλαδου να γεμίζει και τους νεότερους συντρόφους για τα επόμενα χρόνια και το σχήμα να συνεχίζει να διαμορφώνει συνειδήσεις.

Σε πιο προσωπικό επίπεδο θα ήθελα να ευχαριστήσω τον φίλο και συμφοιτητή μου Χαράλαμπο Ντιγκάκη, για την κοινή μας πορεία και τη στήριξή του. Η πρώτη μέρα που έκατσα δίπλα του στο αμφιθέατρο ήταν από τις πιο τυχερές μέρες των σπουδών μου. Ακόμα θα ήθελα να ευχαριστήσω τη Μαρία Νέζη για τη στήριξη και την έμπνευση της καθόλη τη διάρκεια αυτής της διπλωματικής. Δεν θα μπορούσα να παραλείψω να ευχαριστήσω του φίλους και συντρόφους μου Γιάννη Μουστάκη, Νικολάου Τάσου, Παναγιώτη Δήμα, Χρήστο Σκάντζα αλλά και τις συντρόφισσες και φίλες Δέσποινα Θεοδωροπούλου και Άννα Κοντίνη για τις στιγμές που ζήσαμε εντός και εκτός σχολής.

Ακόμα νιώθω την ανάγκη να ευχαριστήσω τους παιδικούς μου φίλους και συναθλητές Παύλο Τσουκιά, Αλέξη Ντελή, Σπύρο Γελάτσορα, Σπύρο Γραμμένο και Δημήτρη Κορακιανήτη που αποτελέσαν για εμένα παράδειγμα πίστης και φιλίας.

Επίσης θα ήθελα να ευχαριστήσω βαθιά το Δάσκαλό πολεμικών τεχνών κ. Γεράσιμο Αλεξάκη. Χάρη στην υπομονή του, την αδιάκοπη προσπάθειά του και την αγάπη του για τους μαθητές του βρέθηκα έτοιμος να αντιμετωπίσω τις καταστάσεις που ήρθαν στο δρόμο μου. Μου δίδαξε αξίες οι οποίες έπαιξαν καίριο ρόλο τόσο στη περάτωση των σπουδών μου όσο και συνολικά στη διαμόρφωση του χαρακτήρα μου.

Τέλος το μεγαλύτερο ευχαριστώ όλων το αξίζω στους γονείς μου Αιμίλιο Πουλιάση και Μαρία Φρουζάκη. Δίχως την αμέριστη στήριξή τους και τη καθοδήγησή τους δεν θα ήταν δυνατό να σπουδάσω. Έθεσαν τα θεμέλια μέσα από εξαιρετικά δύσκολες καταστάσεις ώστε να σπουδάσω και εργάστηκαν με αυταπάρνηση ώστε να μου προσφέρουν τις καλύτερες δυνατές συνθήκες για να ακολουθήσω το δρόμο που επέλεξα. Για αυτό τους ευχαριστώ και τους θαυμάζω.

Πουλιάσης Γιώργος

Αθήνα, Οκτώβριος 2017

Abstract

Currently numerical simulations have become one of the approaches to tackle engineering and scientific problems. They are used in almost every discipline of science and engineering such us solid mechanics, fluid mechanics, climatic models etc. In addition the rapid development of computers, it has enabled the use of complicated numerical tools in professional engineering offices. Most methods are based on grids or meshes such as finite element method (FEM), finite difference method (FDM) and finite volume method (FVM). Particularly the latter is widely applied in computational fluid mechanics packages. However application of these methods, when considering problems with large deformations such as fracture, moving boundary, free surface flows and multi-phase flows, has shown difficulties inherited from their grid nature. Because the entire formulation is based on the grid/mesh, a time-consuming and costly process of generating/regenerating a quality grid/mesh is necessary. Moreover these grids/meshes in many cases need to vary in space and to time, thus inserting high complexities on the algorithm, consisting the development of one a formidable task.

In order to tackle these problems researchers have focused on the development of mesh-free methods. Smoothed Particle Hydrodynamics (SPH) is the most widely established method and has been applied in various fields of science and engineering such us astrophysics, computational solid mechanics, computational fluid dynamics, etc. Despite this method possesses serious advantages and strong conservation properties, a full consensus on its formulation has not yet been reached. Moreover, to the extent of the author's knowledge there has not been a thorough investigation of the key aspects of classical SPH formulation.

The purpose of this thesis is to develop a computer code based on the SPH method and conduct a thorough investigation of the response of this method to the change of various parameters of the method and numerical techniques. The thesis is structured as follows. First a brief presentation of the fundamental descriptions of fluids is given. Then the theoretical foundations of the SPH are analyzed and the basic equations comprising the method are derived. Subsequently the basic numerical components, such as boundary representation techniques and time integration schemes, are presented and discussed. Then the steps of the development of an algorithm are laid and the applications are presented.

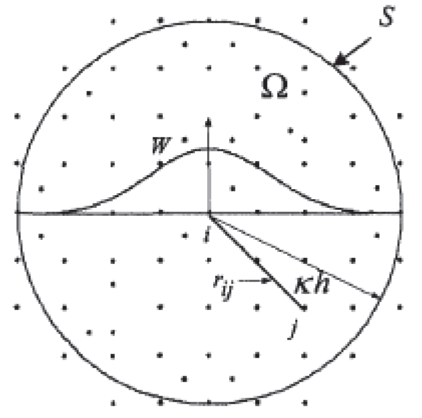
In order to check the validity of our code we have employed a classic benchmark case for the SPH, the collapse of a water column over a dry bed. After the validation we have conducted a thorough investigation on the behavior and performance of the method comprising of thirteen different parameter formulations over the same test case. In addition we have laid the foundations of imposing hydrostatic pressure as initial condition.

Finally the results are discussed and conclusions are drawn as optimal guidelines for the use of the method, and suggestions for further improvements of the method and future work.

Εκτενής περίληψη/ Extended abstract in Greek

Στις μέρες μας οι αριθμητικές αναλύσεις κατέχουν εξέχουσα στην επίλυση προβλημάτων και έχουν βρει εφαρμογή σε πολλά επιστημονικά πεδία όπως η μηχανική του συνεχούς μέσου, μηχανική ρευστών, τα κλιματικά μοντέλα κ.α. Ειδικά με την ραγδαία ανάπτυξη της τεχνολογίας των προσωπικών υπολογιστών η χρήση τους έχει πια καταστεί εκτεταμένη ακόμα και στα μελετητικά γραφεία μηχανικών. Οι πιο διαδομένες μέθοδοι βασίζονται στη διακριτοποίηση του χώρου επίλυσης με τη βοήθεια πλέγματος, μερικές από τις οποίες είναι η μέθοδος πεπερασμένων διαφορών (FDM), η μέθοδος πεπερασμένων στοιχείων (FEM) και η μέθοδος πεπερασμένων όγκων (FVM). Ειδικά η τελευταία έχει γνωρίσει ευρεία χρήση σε εφαρμογές υπολογιστικής ρευστομηχανικής και βρίσκεται σε αρκετά εμπορικά πακέτα. Παρόλη την διάδοση των παραπάνω μεθόδων, αυτές εμφανίζουν σημαντικά μειονεκτήματα στην επίλυση προβλημάτων τα οποία περιλαμβάνουν μεγάλες παραμορφώσεις, όπως είναι η θραύση, τα κινούμενα όρια, οι ροές με ελεύθερη επιφάνεια και οι πολυφασικές ροές. Πηγή αυτών των μειονεκτημάτων αποτελεί η πλεγματική τους προσέγγιση. Για τη χρήση αυτών των μεθόδων απαιτείται μία δαπανηρή διαδικασία δημιουργίας και βελτιστοποίησης του πλέγματος επίλυσης. Περαιτέρω για την μεγιστοποίηση της ακρίβειας τα πλέγματα πολλές φορές μεταβάλλονται τόσο χωρικά όσο και χρονικά με αποτέλεσμα να εισάγονται εξαιρετικές πολυπλοκότητες στους αλγορίθμους, καθιστώντας έτσι την δημιουργία ενός ένα δύσκολο έργο.

Για την αντιμετώπιση αυτών των μειονεκτημάτων οι ερευνητές στράφηκαν στη δημιουργία μη πλεγματικών μεθόδων. Τη πιο διαδεδομένη από αυτές αποτελεί η μέθοδος Υδροδυναμικής Ρεόντων Σωματιδίων (Smoothed Particle Hydrodynamics), η οποία έχει βρει ευρεία εφαρμογή σε πεδία όπως η αστροφυσική, η υπολογιστική μηχανική και η υπολογιστική ρευστομηχανική. Αυτή η μέθοδος βασίζεται στη διακριτοποίηση του χώρου με σωματίδια τα οποία μεταφέρουν όλες τις ιδιότητες του ρευστού και μεταβάλλονται στο χώρο και χρόνο. Χρησιμοποιεί την ολοκληρωματική προσέγγιση για την διακριτοποίηση των εξισώσεων με τη βοήθεια συναρτήσεων πυρήνα.



Εικόνα 1:Συνεχής και διακριτός χώρος μέσα σε πυρήνα επιρροής. Σχηματική απεικόνιση της συνάρτησης πυρήνα

Η SPH φέρει κάποια σημαντικά πλεονεκτήματα:

i. Η διακριτοποίηση των εξισώσεων κίνησης με βάση τη μέθοδο προκύπτει άμεσα από την Λαγκραντζιανή έκφραση των ρευστών, δίνοντας της έτσι ισχυρές ιδιότητες διατήρησης μάζας και ορμής.
ii. Η Λαγκραντζιανή προσέγγιση της προσφέρει ένα ισχυρό θεωρητικό υπόβαθρο.
iii. Είναι εύκολο να αναπτυχθεί και να εφαρμοστεί σε αλγόριθμο.

iv. Η Υδροδυναμική Ρεόντων Σωματιδίων αποτελεί την παλιότερη μη πλεγματική μέθοδο και φτάνει σε ένα επίπεδο ωριμότητας.

v. Με τις συνεχείς βελτιώσεις η ακρίβεια, η σταθερότητα και η προσαρμοστικότητα της μεθόδου φτάνουν σε ένα ικανοποιητικό επίπεδο που να επιτρέπει τη χρήση της πρακτικά προβλήματα μηχανικού.

vi. Το φάσμα των εφαρμογών της είναι ιδιαίτερα ευρύ. Μπορεί να αντιμετωπίσει προβλήματα κλασικής ρευστομηχανικής, πολυφασικές ροές και να συνδυαστεί με μεθόδους μοριακής δυναμικής.

Παρόλα τα παραπάνω πλεονεκτήματα της μεθόδου και το επίπεδο ωριμότητας το οποίο έχει ανέλθει, ακόμα δεν υπάρχει μία ευρέως αποδεχομένη διατύπωση της και στον ορίζοντα της γνώσης του συγγραφέα δεν έχει υπάρξει μία εξαντλητική αποτύπωση της συμπεριφοράς της μεθόδου σε διαφορετικές παραμέτρους και προσεγγίσεις.

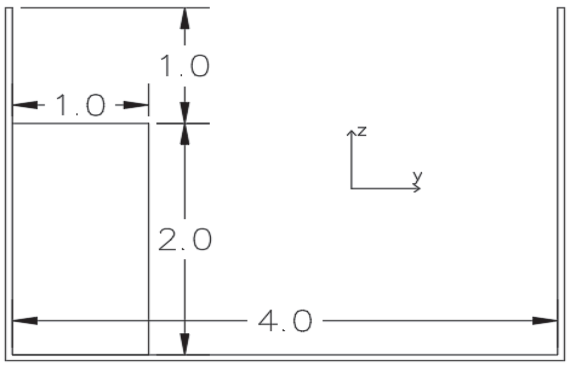
Σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη υπολογιστικού κώδικα με βάση τη μέθοδο της Υδροδυναμικής Ρεόντων Σωματιδίων και η εκ βαθέων διερεύνηση και σύγκριση διαφορετικών αριθμητικών συνιστωσών και παραμέτρων της μεθόδου.

Αρχικώς παρουσιάζονται οι πιο διαδεδομένες εξισώσεις πυρήνα στη βιβλιογραφία και στη συνέχεια το θεωρητικό υπόβαθρο της μεθόδου και προκύπτουν οι βασικές εξισώσεις της μεθόδου όπως αυτές φαίνονται στο παρακάτω πίνακα.

Μεταβλητή	*Εξίσωση*
Πυκνότητα	$p_i = \sum_{j=1}^{N} m_j W_{ij}$
Συνέχεια	$\frac{dp_i}{dt} = \frac{1}{p_i} \sum_{j=1}^{N} m_j \overrightarrow{u_{ij}} \nabla_i W_{ij}$
Ορμή	$\frac{d\overrightarrow{u_i}}{dt} = -\sum_{j=1}^{N} m_j \left(\frac{P_i}{p_i^2} + \frac{P_j}{p_j^2} + \Pi_{ij} \right) \nabla_i W_{ij} + \vec{g} + \vec{f}_{ij}^{LJ}$
Ενέργεια	$\frac{d\overrightarrow{e_i}}{dt} = \frac{1}{2} \sum_{j=1}^{N} m_j \left(\frac{P_i}{p_i^2} + \frac{P_j}{p_j^2} \right) \overrightarrow{u_{ij}} \nabla_i W_{ij}$
Ταχύτητα	$\frac{d\vec{x}_i}{dt} = \vec{v}_i + \varepsilon \sum_{j=1}^{N} \frac{m_j}{\bar{p}_{ij}} \vec{u}_{ji} W_{ij}$
Πίεση	$P_i = \frac{p_o c^2}{\gamma} \left[\left(\frac{p_i}{p_o} \right)^{\gamma} - 1 \right]$

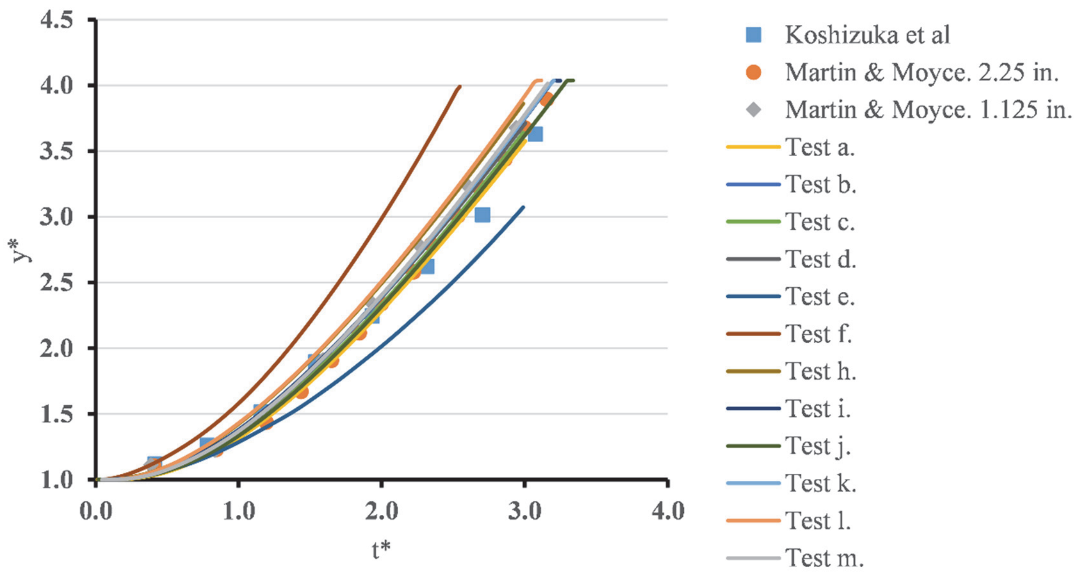
Στην συνέχεια αναλύονται οι αριθμητικές συνιστώσες τις μεθόδου. Αυτές είναι οι τεχνικές προσομοίωσης των ορίων, οι τεχνικές αριθμητικής ολοκλήρωσης, τεχνικές για την κανονικοποίση της πυκνότητας και για την ομαλή κίνηση των σωματιδίων απουσία συνεκτικότητας. Ειδικότερα, όσον αναφορά τις τεχνικές προσομοίωσης των ορίων παρουσιάζονται πέντε εξ αυτών και συζητιόνται ενδελεχώς τα μειονεκτήματα και τα πλεονεκτήματα αυτών. Στη συνέχεια παρουσιάζονται οι διάφορες εκδόσεις του υπολογιστικού κώδικα και δίνονται συγκρίσεις μεταξύ των επιδόσεών τους. Σημειώνεται ότι παρότι κατά την πορεία ανάπτυξης οι επιδόσεις βελτιώθηκαν πάνω από 10 φορές γίνεται αντιληπτό ότι ταχύτητα αποτελεί ένα από τα μειονεκτήματα της μεθόδου.

Για να επικυρωθεί η απόδοση του υπολογιστικού κώδικα που αναπτύχθηκε, χρησιμοποιήθηκε ένα από τα βασικά προβλήματα επικύρωσης στη βιβλιογραφία της μεθόδου, η κατάρρευση στήλης νερού μέσα σε δεξαμενή με στεγνό πυθμένα. Τα αποτελέσματα μας συγκρίθηκαν με πειραματικά δεδομένα από τη βιβλιογραφία. Στην εικόνα που ακολουθεί φαίνεται η διάταξη του προβλήματος.



Εικόνα 2: Διάταξη προβλήματος κατάρρευσης στήλης

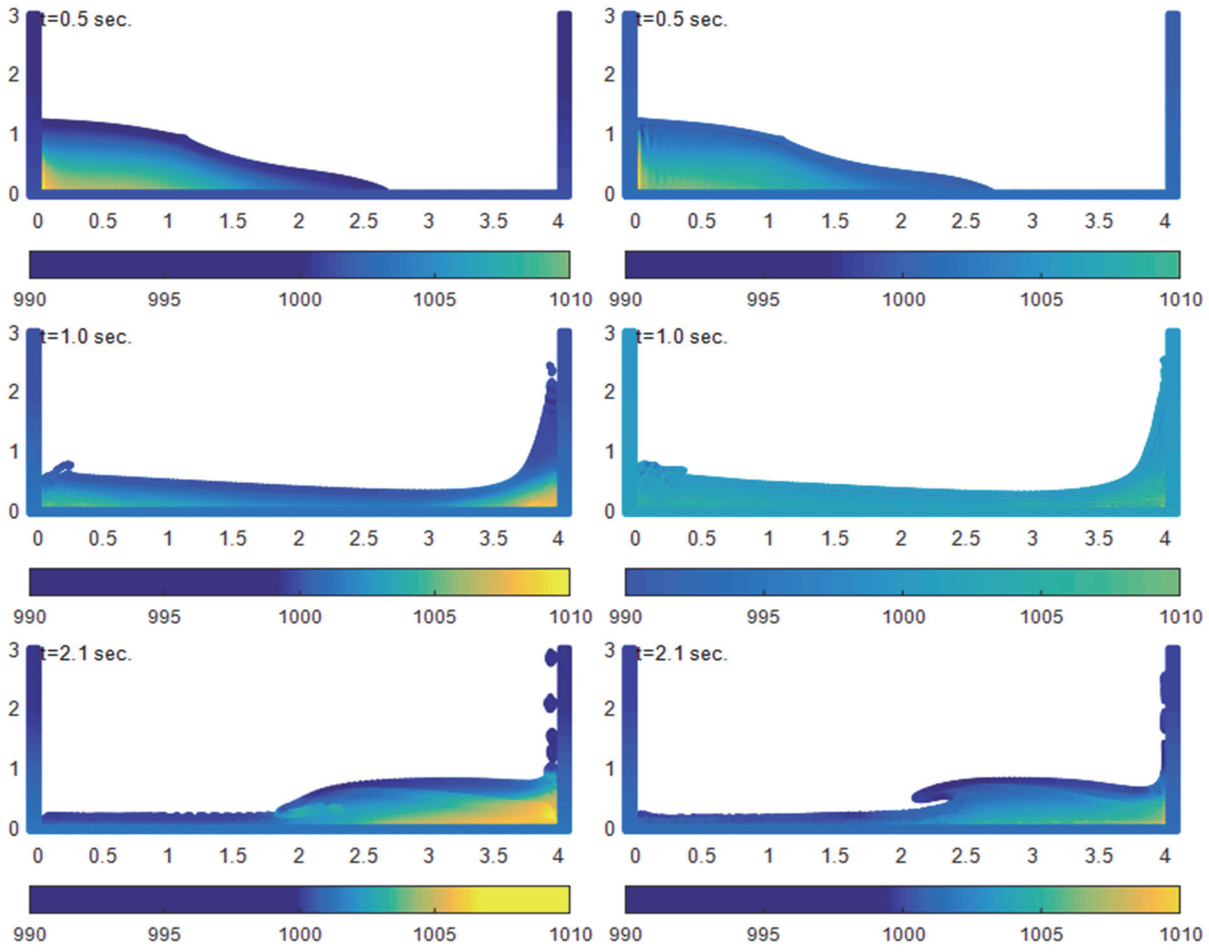
Για την διερεύνηση της μεθόδου σχεδιάστηκαν 13 διατάξεις παραμέτρων, έτσι ώστε να διαπιστωθεί η απόκριση της σε διαφορετικές διακριτοποιήσεις, διαφορετικά μήκη εξομάλυνσης, διαφορετικές τεχνικές προσομοίωσης του ορίου, διαφορετικά χρονικά βήματα καθώς και τεχνικές χρονικής ολοκλήρωσης. Συνολικά φαίνεται ότι η μέθοδος αποδίδει καλώς σε σχέση με τα πειραματικά δεδομένα για τις περισσότερες διατάξεις.



Εικόνα 3: Σύγκριση πειραματικών δεδομένων με αποτελέσματα όλων των προσομοιώσεων

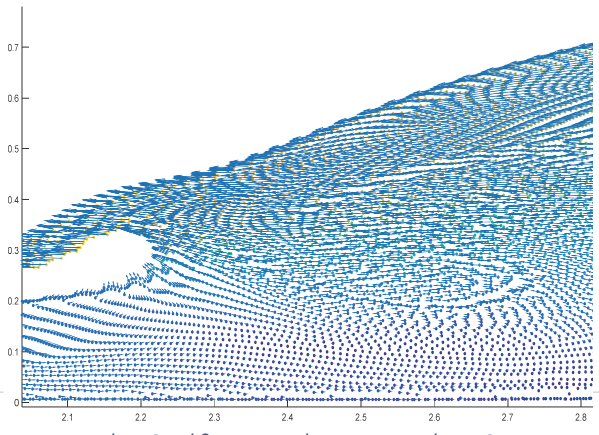
Από τις προσομοιώσει φάνηκε ότι η ακρίβεια της μεθόδου δεν εξαρτάται μόνο από την αρχική διακριτοποίηση του υπολογιστικού χώρου αλλά από τον συνδυασμό αυτής και μήκους εξομάλυνσης του πυρήνα, καθότι έτσι καθορίζεται πλήρως ο αριθμός των σωματιδίων μέσα στο πυρήνα επιρροής. Όσο υψηλότερος είναι ο αριθμός αυτός τόσο βελτιώνεται η ακρίβεια της μεθόδου. Παρόλα αυτά η αναλυτική έκφραση του σφάλματος αυτής της προσέγγισης αποτελεί και σήμερα ανοικτό πρόβλημα.

Περαιτέρω δείχθηκε ότι όσο αυξάνεται το χρονικό βήμα τόσο αποκλίνουν τα αποτελέσματα από τα πειραματικά. Επίσης με την χρήση μεθόδου χρονικής ολοκλήρωσης πρόβλεψης – διόρθωσης η απόκλιση αυτή μειώνεται. Πρέπει να σημειωθεί όμως ότι αυτή η μέθοδος απαιτεί δύο φορές υπολογισμό των παραγώγων σε κάθε χρονικό βήμα επομένως διπλασιάζει ουσιαστικά το χρόνο εκτέλεσης. Ακόμα φαίνεται ότι το χρονικό βήμα και η μέθοδος χρονικής ολοκλήρωσης επηρεάζουν και τη μορφή της ελεύθερης επιφάνειας και την κατανομή των πιέσεων.

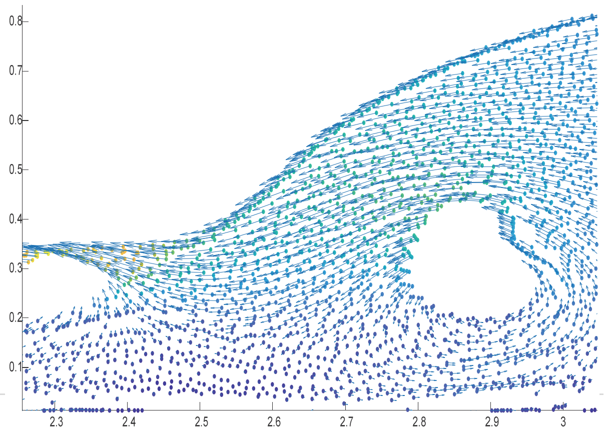


Εικόνα 4: Κατανομή πυκνότητας σε διάφορα στιγμιότυπα. Στην αριστερή στήλη έχει χρησιμοποιηθεί σχήμα πρόβλεψης διόρθωσης και στην δεξιά σχήμα Verlet. Χρονικό βήμα 0.0001 δευτ. και στις δύο περιπτώσεις

Από τις προσομοιώσεις προέκυψε επίσης ότι το σχήμα διόρθωσης ταχύτητας XSPH επηρεάζει σημαντικά τη διασπορά της ροής και βύθιση του κύματος. Όπως φαίνεται από τις παρακάτω



Εικόνα 6: Βύθιση του κύματος με χρήση XSPH



Εικόνα 5: Βύθιση του κύματος χωρίς τη χρήση XSPH

εικόνες αντί το κύμα κατά ένα μέρος να ανακλάται και κατά ένα μέρος να βυθίζεται, ανακλάται ολοκληρωτικά.

Αυτή η μέθοδος είχε εισαχθεί έτσι ώστε τα σωματίδια να κινούνται πιο ομαλά απουσία συνεκτικότητας. Τα αποτελέσματα αυτά, κατά την άποψη του συγγραφέα, καθιστούν αναγκαία την ενσωμάτωση μοντέλου τύρβης στη μέθοδο για την προσομοίωση τέτοιων προβλημάτων.

Περαιτέρω από την εξέταση των μεθόδων ορίου προέκυψαν πλεονεκτήματα και τα μειονεκτήματα αυτών καθώς και τα πεδία εφαρμογής τους. Επίσης φάνηκε ότι η επιλογή της μεθόδου προσομοίωσης ορίου επηρεάζει τη διατήρηση ενέργειας. Συγκεκριμένα η χρήση απωστικών δυνάμεων Lenard – Jones εμποδίζει τη διατήρηση της ενέργειας. Σε αντίθεση η χρήση δυναμικών οριακών σωματιδίων φαίνεται να διατηρεί την ενέργεια. Σχετικά με την διατήρηση ενέργειας φαίνεται να παίζει ρόλο η απουσία μοντέλου τύρβης και η αντικατάστασή του με όρο τεχνητής συνεκτικότητας.

Τέλος, διερευνήθηκε το πρόβλημα της υδροστατικής δεξαμενής. Ένα τέτοιο πρόβλημα αρχικώς μπορεί να φαντάζει συνηθισμένο, όμως καλείτε να αντιμετωπίσει ένα από τα βασικά προβλήματα της μεθόδου SPH το οποίο είναι η επιβολή αρχικών συνθηκών. Λόγο της συμπιεστής φύσης της μεθόδου, για να εφαρμοστεί μία συνθήκη αρχικών πιέσεων θα πρέπει να σωματίδια να βρίσκονται σε εκείνες τις θέσεις όπου θα επιβάλλουν την επιθυμητή θέση. Κάτι τέτοιο όμως είναι αδύνατο να προβλεφθεί εξ αρχής. Για αυτό το λόγω προτάθηκε και διερευνήθηκε η εφαρμογή υδροστατικών συνθηκών μέσω ηρεμίας του ρευστού εφαρμόζοντας μειωτικό συντελεστή στη δύναμη της βαρύτητας. Αυτή η προσέγγιση έδωσε ικανοποιητικά αποτελέσματα.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

It is well known that all phenomena in nature can be described using mathematical tools. Researchers, from Kepler and Newton to Laplace and Kolmogorov, have unlocked the mysteries of nature and presented them in the form of partial differential equations (PDEs). Numerous people including the author, believe that mathematics is the language of the universe. However, describing the nature in terms of equations is far from easy. It requires a great understanding of physical phenomena. But this is not all, as other factors such as the socio-economical background and the political and religious beliefs of the scientist play an important role. In this manner mathematics are very similar to a language. Their study can be as exhilarating as the reading of a novel. A mathematical equation is able to travel the imagination of the researcher to great lengths, from the microscale of the atoms to the vastness of the galaxies. Very few things so compact can give birth to such an emotional and liberating journey.

However, despite how difficult it may be to capture and describe a physical phenomenon in a form of a PDE, acquiring an exact solution of the PDE can be impossible. Many of them to this day remain unsolved. However researchers tried to find approximate solutions to the task at hand. This quest gave birth to numerical analysis. Numerical analysis methods are employed to approximate the solutions from equations describing natural phenomena. Such methods existed since the beginning of 20th century. However the development of computers has started a new era for approximation methods. Computers gave mathematicians and engineers a platform to implement computationally expensive numerical methods efficiently.

The idea behind these methods is to transform the original equations of the the continuum to a simpler discretized form. To obtain this in the present thesis, we will develop an algorithm for the Smoothed Particle Hydrodynamics (SPH) method and apply it in free surface flows.

1.1. Theoretical formulation of fluid motion

1.1.1. Eulerian approach

The Eulerian approach to fluids assumes an observer standing on a set a fixed points x,y,z of the flow field (i.e. grid) and measuring the velocity of particles passing through these points. By this assumption the position of the particle $\vec{x} = (x, y, z)$ is an independent variable and the velocity is a function of the position.

$$\vec{V} = f(\vec{x}, t) \qquad (1)$$

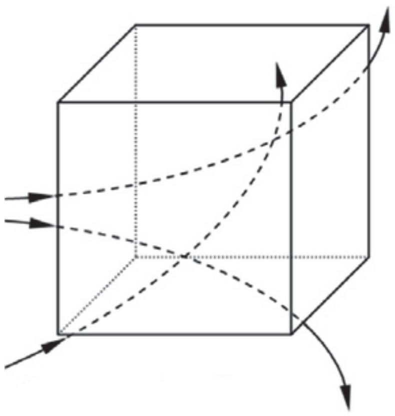


Figure 1: Eulerian notion of media

According to the Euler method, the acceleration is interpreted as the change in the velocity between two known positions (x, y, z) and $(x + dx, y + dy, z + dz)$ in given time dt. By taking the change in velocity to the x axis $u = u(x, y, z, t)$ we get:

$$du = \frac{\partial u}{\partial t} dt + \frac{\partial u}{\partial x} dx + \frac{\partial u}{\partial y} dy + \frac{\partial u}{\partial z} dz \qquad (2)$$

However, the movement (dx, dy, dz), is due to the velocity, therefore we can write:

$$dx = udt, \qquad dy = vdt, \qquad dz = wdt \tag{3}$$

Thus, the acceleration in x axis is written as:

$$a_x = \frac{du}{dt} = \frac{\partial u}{\partial t} + u\,\frac{\partial u}{\partial x} + v\,\frac{\partial u}{\partial y} + w\,\frac{\partial u}{\partial z} \tag{4}$$

The same procedure can be followed for the acceleration in the other directions.

1.1.2. Lagrangian approach

The Lagrangian approach assumes an observer who follows the trajectory of every fluid particle. In other words in Lagrangian the successive positions of particles through time is observed and through them their velocity and acceleration are deduced. For the Lagrangian description of motion, at time t_0 of fluid motion the position of all fluid particles is known. Then for a particle with known initial position $(x_0, y_{0,} z_0)$ the position at a time t is given by:

$$\vec{x} = \varphi(x_0, y_{0,} z_0, t) \tag{5}$$

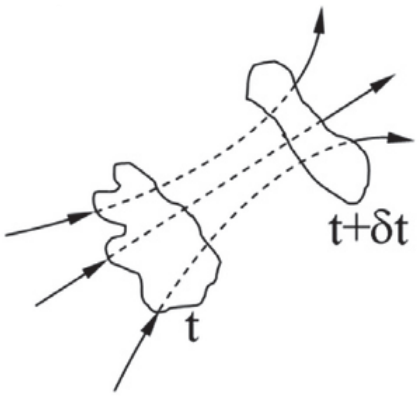


Figure 2: Lagrangian notion of media

Thus, the velocity and acceleration of the particle are deduced using the first and the second derivatives of position vector respectively:

$$\vec{V} = \frac{d\vec{x}}{dt}, \text{ and} \tag{6}$$

$$\vec{a} = \frac{d^2\vec{x}}{dt^2} \tag{7}$$

It can be shown that the Euler and Langrange approaches are theoretically equivalent, but the first one has been used extensively if compared the last one. Further details regarding Lagrangian dynamics of fluid motion one can refer to Bennett (2006).

1.2. Numerical Methods for approximating PDEs

1.2.1. Grid based Euler Methods

Grid based Euler methods refer to a class of methods that use a grid to represent the computational domain. This grid is usually fixed in time and space and thus large deformations do not affect it. The flux of mass, momentum and energy across mesh cell boundaries are simulated to compute the distribution of mass, velocity, energy, etc. in the problem domain. The shape and volume of the mesh cells remain unchanged in the entire process of the computation. The non-moving grid makes these methods suitable for hydrodynamic problems. However, certain problems arise related to the difficulty to analyze the time history of field variables, to the difficulty addressing problems with complex geometries, to computationally 'expensive' grids etc.

1.2.2. Meshfree Lagrangian Particle Methods (MPMs)

Meshfree Particle Methods (MPMs) generally refer to the class of meshfree methods that use a finite set of discrete particles to represent the computational domain. The particles can range from very small (atoms) to very large objects such as planets. Each particle 'carries' the quantities (variables) that represent the media such as mass, position in space, velocity, acceleration and momentum. The evolution of the physical system is determined by the set of conservation equations for mass, momentum, energy as well as some equation of state if necessary.

Most MPMs naturally follow the Lagrangian description of fluid motion, with the particles being part of the physical system acting under the influence of internal and external forces, and thus evolving in time. This is a significant difference from the mesh Eulerian methods because, as shown in eq (5), the field variables do not depend on their current position, and therefore it is easy handling extremely large deformations and free surface problems. However, particle methods exist, which make use of some kind of mesh either for interpolation or for other purposes.

The particle approximation involves the discretization of the governing equations with respect to the set of particles that form the computational domain. The approximation of the field variable quantities of a particle uses information from neighboring particles. The neighborhood of a particle is defined as the influence domain and is an essential parameter for the problem formulation. The neighboring particles within the support domain of a particle, provide via a shape function all necessary and sufficient information for the approximation of the field variables. For example the acceleration of the particle i can be computed as:

$$\frac{du_i}{dt} = \sum_{j=1}^{N} \varphi_j(x_{ij}) \frac{du_j}{dt} \tag{8}$$

Where N in the number of particles inside the influence domain of i, φ_j is the shape function of particle i and x_{ij} the distance of the particles i and j. This formulation using shape functions can be used to create a set discretized equations that represent the dynamics of the system.

In this thesis out of all the MPMs, we are going to employ the Smoothed Particle Hydrodynamics method in fluid flows with free surface due to some of its advantages namely:

i. SPH expressions of the equations of motion can be derived directly from a Lagrangian of the particle system, thus providing the method with very strong conservation properties
ii. Its Lagrangian formulation gives a very robust physical background
iii. It solves exactly the particle continuity equation
iv. It is easy to formulate and program, as it does not require a mesh
v. The SPH method, as the oldest MPM, is quickly approaching to its mature stage
vi. With the continuing improvements and modifications, the accuracy, stability and adaptivity of the SPH method have reached an acceptable level for practical engineering applications
vii. The applications of the method are very wide ranging from CFD to microfluidics. The method can easily simulate multi-phase flows and it can be coupled with molecular processes. .

1.3. Historical Background of the SPH method

The Smoothed Particle Hydrodynamics Method (SPH) is a Lagrangian method that employs a set of finite particles to represent the computational domain. These particles carry all the properties of the fluid. It uses an integral function approximation in order to discretize the governing PDEs. This method was first introduced by Lucy (1977) and Gingold RA (1977) for the study celestial dynamics. The study of such phenomena can be modeled with classical Newtonian dynamics. Although at that time Eulerian methods such as finite element (FEM) and finite difference methods (FDM) existed, their difficulty to deal with extremely large deformations and open boundaries has lead the researchers in search for an alternative.

SPH, as a mesh free method, has proved to be a good choice. Since then it has expanded to solve many problems of hydrodynamics, as well as solids with large deformations such as fracture problems. It has also been used for the modeling of shocks and explosions. Application of SPH to a wide range of problems has led to significant extensions and improvements of the original SPH method. The numerical aspects have been gradually improved, some inherent drawbacks of SPH were identified, and modified techniques or corrective methods were proposed, as it will be discussed later.

Research with SPH has also led to the development of alternative variations of the method. Some of them along with their main characteristics are presented in table 1Hopkins (2014).

Table 1: Variations of SPH method and their main characteristics

Method Name	Consistency /Order	Conservative? (Mass/Energy /Momentum)	Numerical Dissipation	Long-Time Integration Stability?	Number of Neighbors	Known Difficulties
Smoothed-Particle Hydro. (SPH)						
“Traditional” SPH (GADGET, GASOLINE, TSPH)	0	✓	artificial viscosity (AV)	✓	~ 32	fluid mixing, noise, E0 errors
“Modern” SPH (P-SPH, SPHS, PHANTOM, SPHGal)	0	✓	AV+conduction +switches	✓	~ 128 – 442	excess diffusion, E0 errors
“Corrected” SPH (rpSPH, Integral-SPH, Morris96 SPH, Moving-Least-Squares SPH)	0-1	×	artificial viscosity	×	~ 32	errors grow non-linearly, “self-acceleration”
“Godunov” SPH (GSPH, GSPH-I02, Cha03 SPH)	0	✓	Riemann solver	✓	~ 300	instability, expense, E0 errors

We note that despite the recent improvements some of the inherent disadvantages such as low order errors, noise and unstable pressure field are not cured. Overall, all of these improvements have only reduced, as opposed to eliminated, the associated errors. And all have costs, either in terms of CPU time and complexity, numerical noise, excessive diffusion, or slow convergence. Therefore in this thesis we will follow the traditional formulation of the SPH method with some numerical improvements.

2. The Smoothed Particle Hydrodynamics Model

In this chapter the essential components of the SPH method will be presented. The integral approximation upon which the method relies is discussed first. Then, the properties of the smoothing kernels will be surveyed in depth and some popular functions found in literature will be presented. Afterwards a method for constructing function derivatives using integral interpolation will be formulated. Furthermore using the theory and technics described above the basic SPH equations will be derived. Finally some other numerical elements of the method, such as boundary treatment and time step, will be presented. Note that because the method is relatively new, there is not yet a universal formalization regarding the symbols used. Therefore the author will follow the symbols presented by Liu G.R. (2003) andMonaghan (2005).

2.1. Integral Interpolation

The integral interpolation is a method for the numerical approximation of functions. The continuous form of this method reads:

$$f(\vec{x}) = \int_{\Omega} f(\vec{x}')\delta(\vec{x}-\vec{x}')d\vec{x}' \tag{9}$$

Where $\vec{x}$ is a three dimensional position vector, Ω is the volume of the integral that contains $\vec{x}$, and δ is the Dirac function.

$$\delta(\vec{x}-\vec{x}') = \begin{cases} 1 & \vec{x}=\vec{x}' \\ 0 & \vec{x}\neq\vec{x}' \end{cases} \tag{10}$$

Now if we replace the Dirac function with a smoothing function (or smoothing kernel) $W(\boldsymbol{x}-\boldsymbol{x}',h)$ we get the following expression of the integral representation:

$$f(\vec{x}) = \int_{\Omega} f(\vec{x}')W(\vec{x}-\vec{x}',h)d\vec{x}' \tag{11}$$

where h is the smoothing length that determines the support domain of the function.

2.2. Particle Approximation

In the SPH method the fluid is represented by a set of finite particles carrying all the properties of it. Therefore we must transform the integral representation in a particle representation. This means that we must express the equation above in a discretized form. To this aim we note that the

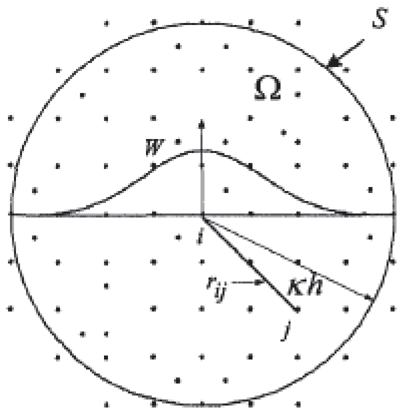


Figure 3: The continuous and the discretized space of the support domain. Note that the support domain is circular with a radius of kh. Figure by Liu G.R. (2003)

infinitesimal volume $d\boldsymbol{x}'$ can be approximated by the finite volume of a particle j ΔV_j. That volume is connected to the particle mass as

$$m_j = \Delta V_j \rho_j \tag{12}$$

Where ρ_j is the particle density. Therefore we can wright the discretized form of the function integral as follows:

$$\begin{aligned} f(\vec{x}) &= \int_{\Omega} f(\vec{x}')W(\vec{x}-\vec{x}',h)d\,\vec{x}' \\ &\cong \sum_{j=1}^{N} f(\vec{x_j})W(\vec{x}-\vec{x_j},h)\,\Delta V_j \\ &= \sum_{j=1}^{N} f(\vec{x_j})W(\vec{x}-\vec{x_j},h)\frac{1}{\rho_j}\rho_j \Delta V_j \\ &= \sum_{j=1}^{N} f(\vec{x_j})W(\vec{x}-\vec{x_j},h)\frac{1}{\rho_j}m_j \end{aligned}$$

Or by bringing the mass and density forward

$$f(\vec{x}) = \sum_{j=1}^{N} \frac{m_j}{\rho_j} f(\vec{x_j})W(\vec{x}-\vec{x_j},h) \tag{13}$$

2.3. The Smoothing function

The SPH method employs the integral representation of a function. Therefore we can understand that the smoothing kernel is one of the key components of the method. It is essentially the medium that transfers the information from the neighboring particles to the value being approximated. In this chapter we will present some basic properties of the smoothing functions as well as some popular ones found in the literature.

2.3.1. Basic properties of smoothing functions

The kernel functions must obey some basic rules emerging from the initial definition of the integral representation as well as from physical considerations. The major properties derived from these principles areLiu G.R. (2003):

i. The smoothing function must be normalized over its support domain.

$$\int_{\Omega} W(\vec{x}-\vec{x}',h)d\vec{x}' = 1 \tag{14}$$

ii. The smoothing function must be compactly supported.

$$W(\vec{x}-\vec{x}',h) = 0, |\vec{x}-\vec{x}'| > kh \tag{15}$$

where h is the smoothing length and k determines the spread of the smoothing function, kh defines the support domain where $|\vec{x}-\vec{x}'| \leq kh$. This transforms the integral approximation from a global operation to a local one. Such a property is not imposed by any mathematical definition but has physical meaning, because particles far away from each other do not interact. Also it significantly reduces the number of particles that contribute to the summation of the approximations, thus resulting in a much faster algorithm.

iii. The smoothing function must remain positive inside the support domain.

$$W(\vec{x}-\vec{x}',h) > 0, |\vec{x}-\vec{x}'| \leq kh \tag{16}$$

This property is also not mathematically necessary but has physical meaning in hydrodynamic simulations. Violation of the property could result in unnatural results such as negative density.

iv. The smoothing function should be monotonically decreasing with respect to $|\vec{x}-\vec{x}'|$. This means that a more adjacent particle will have a bigger influence.

v. The smoothing function should yield the Dirac δ-function as the smoothing length approaches zero.

$$\lim_{h\to 0} W(\vec{x}-\vec{x}',h) = \delta(\vec{x}-\vec{x}') \tag{17}$$

This condition results from the definition of the integral representation and ensures that as the smoothing length decreases, the approximation will tend to the exact value of the function.

vi. The smoothing function should be symmetric. This ensures that particles with the same distance from a given particle will have the same effect.

vii. The smoothing function should be sufficiently smooth.

Using the above properties anyone could construct a sufficient smoothing function to be used in SPH simulations.

2.3.2. Popular smoothing functions

The first smoothing kernel introduced by Lucy (1977) is a bell shaped function

$$W(R,h) = a_d \begin{cases} (1+3R)(1-R)^3, & R \leq 1 \\ 0, & R > 1 \end{cases} \tag{18}$$

where a_d is a coefficient used in order to satisfy the unity condition. It takes 5/4h, 5/πh^2 and 105/16πh^3 in one, two and three dimensions respectively. R is the relative distance between two

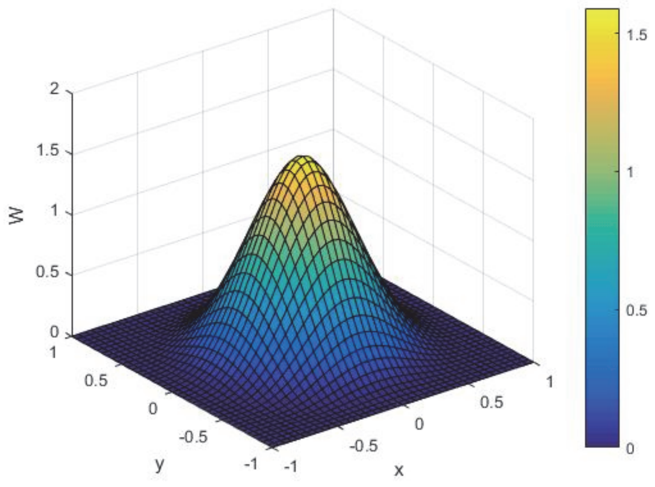


Figure 4: Smoothing Kernel Lucy (1977)

particles given by: $R = \frac{r}{h} = \frac{|\vec{x} - \vec{x'}|}{h}$. Gingold and Monaghan in their original paper Gingold RA (1977) used the following Gaussian kernel

$$W(R, h) = a_d e^{-R^2} \tag{19}$$

where ad is $1/\pi^{1/2}h$, $1/\pi h^2$ and $1/\pi^{3/2}h^3$ in one, two and three dimensions respectively.

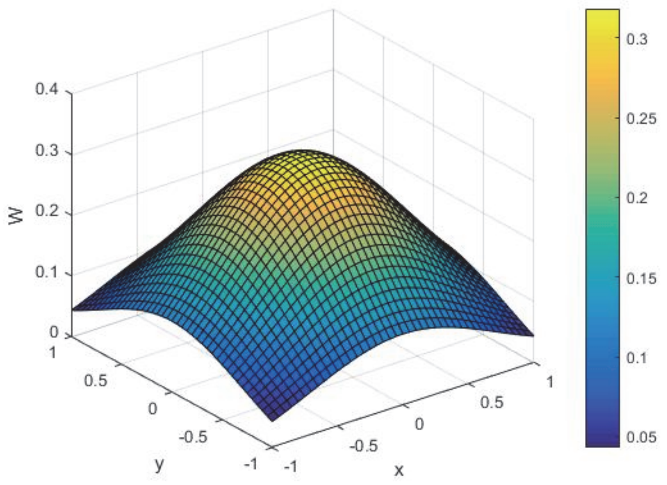


Figure 5: The Gaussian kernel Gingold RA (1977)

As we can see from figure (5) this kernel although it approaches zero fast, it is not really compact, therefore making the computations more expensive. However it is sufficiently smooth for high derivatives and that why it has been an appealing choice.

A very popular family of smoothing functions are based on Mn splines. Perhaps the most commonly used in SPH literature is the cubic B-spline which was originally used by Monaghan JJ (1985).

$$W(R, h) = a_d \begin{cases} \frac{2}{3} - R^2 + \frac{1}{2}R^3, & 0 \le R < 1 \\ \frac{1}{6}(2 - R)^3, & 1 \le R, 2 \\ 0, & R \ge 2 \end{cases} \tag{20}$$

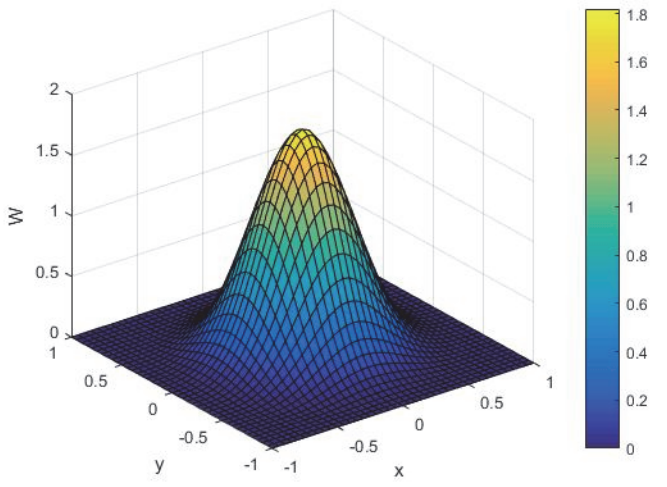


Figure 6: Cubic spline kernel Monaghan JJ (1985)

where a_d is 1/h, 15/πh^2 and 3/2πh^3 in one, two and three dimensions respectively. Its advantages lie in its close resemblance to the Gaussian kernel as well as the compact support domain. However this function, as well as all the spline family kernels, is not in one piece, thus reducing its effectiveness and increasing the computational effort.

Morris J.P. (1996) has introduced a quartic spline function that is reportedly more stable than the cubic spline.

$$W(R,h) = a_d \begin{cases} (R+2.5)^4 - 5(R+1.5)^4 + 10(R+0.5)^4, & 0 \le R < 0.5 \\ (2.5-R)^4 - 5(1.5-R)^4, & 0.5 \le R < 1.5 \\ (2.5-R)^4, & 1.5 \le R < 2.5 \\ 0, & R > 2.5 \end{cases} \tag{21}$$

where a_d is 1/24h, 96/1199πh^2 and 1/20πh^3 in one, two and three dimensions respectively.

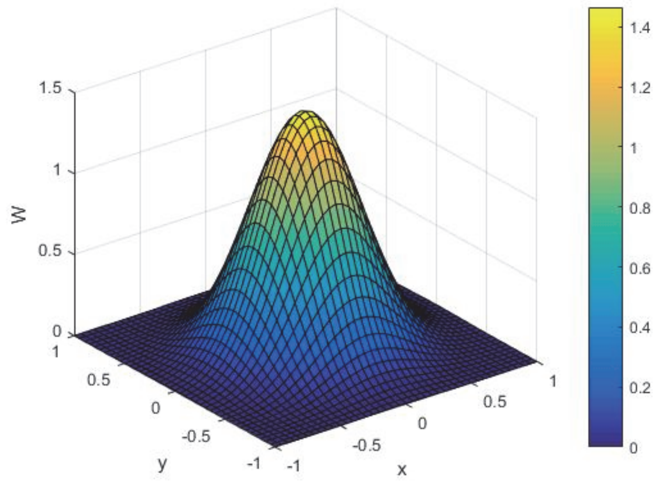


Figure 7: Quartic spline kernel J.P. (1996)

The problems arise by multi-piece kernels have made some researchers to create simpler and more effective kernels with one equation. In here we will present two of them.

First the Wendland kernels, presented by Wendland (1995). We will only present the fourth order kernel of them.

$$W(R,h) = a_d \begin{cases} \left(1 - \frac{R}{2}\right)^4 (1+2R), & 0 \le R \le 2 \\ 0, & R > 2 \end{cases} \tag{22}$$

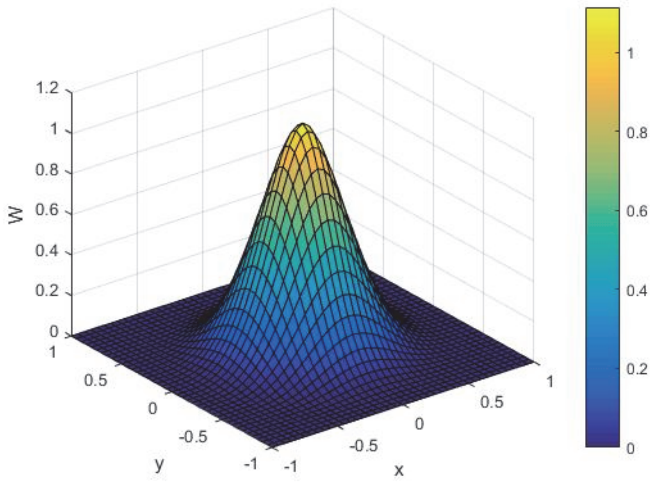


Figure 8: Wendland 4rth order kernel Wendland (1995)

where a_d is 3/4h, 7/4πh^2 and 21/16πh^3 in one, two and three dimensions respectively.

Finally we present a quartic smoothing function constructed byLiu MB (2003).

$$W(R,h) = a_d \begin{cases} \frac{2}{3} - \frac{9}{8}R^2 + \frac{19}{24}R^3 - \frac{5}{32}R^4, & 0 \leq R \leq 2 \\ 0, & R > 2 \end{cases} \tag{23}$$

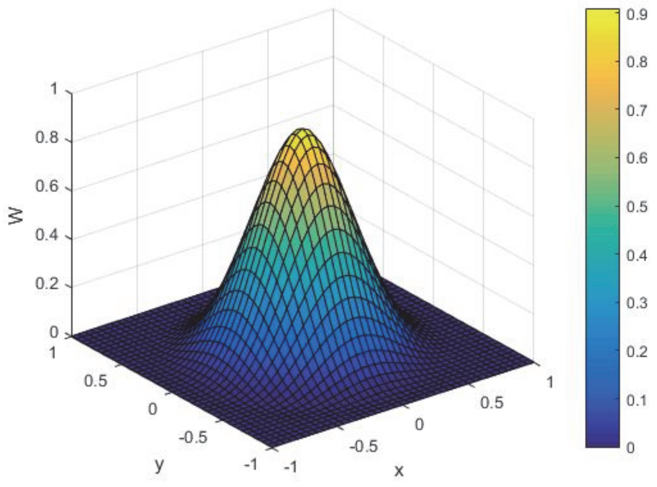


Figure 9: Quartic kernel Liu MB (2003)

where a_d is 1/h, 15/7πh^2 and 315/208πh^3 in one, two and three dimensions respectively.

In figures 10 and 11 a comparison of the above kernels and their first derivatives is shown, for two dimensions (2D) and smoothing length $h = 0.5$.

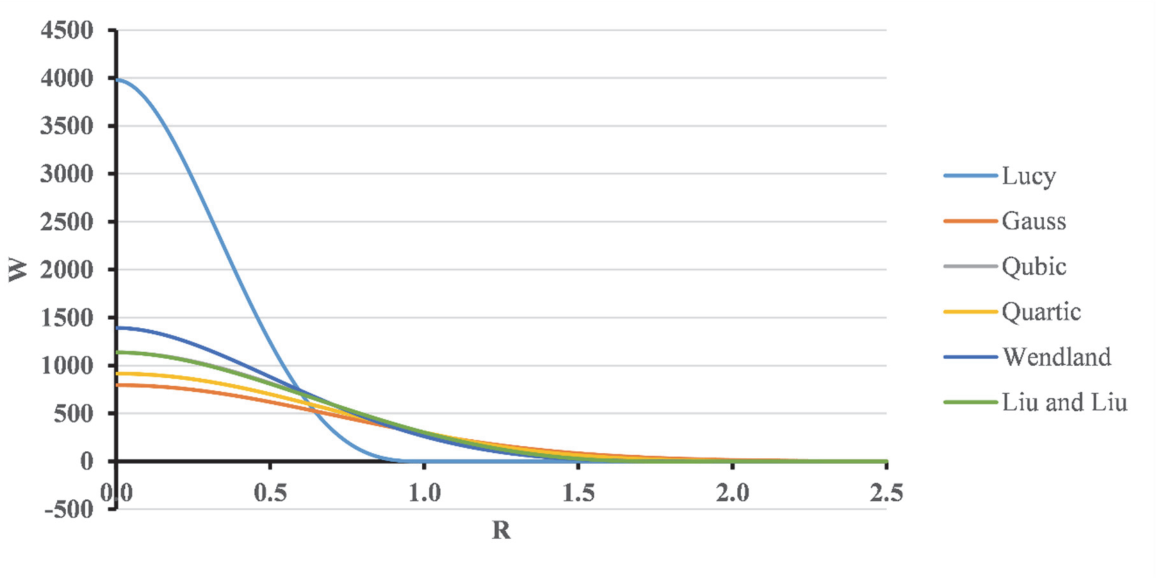


Figure 10: Comparison between smoothing kernel

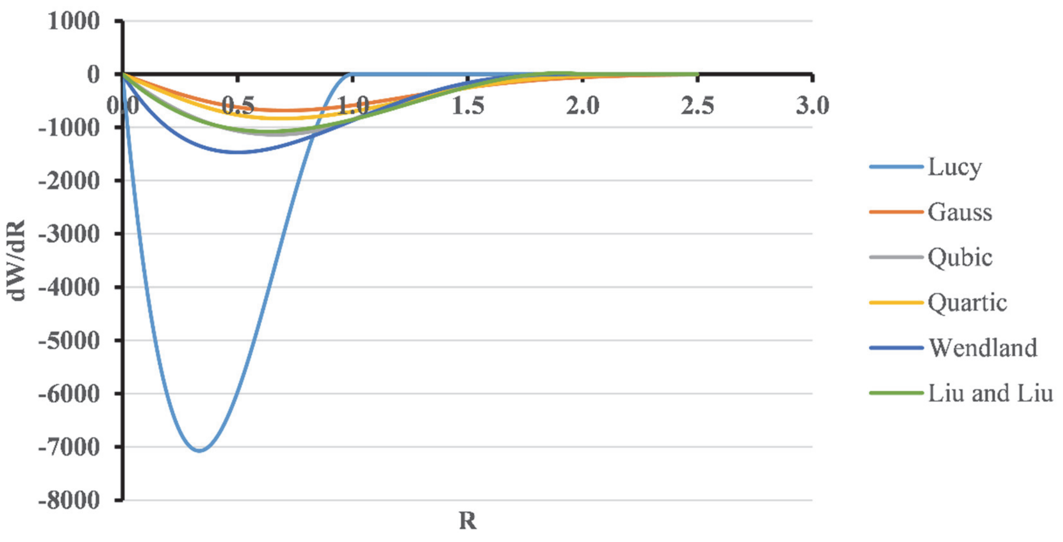


Figure 11: Comparison between kernel derivatives

From the above figures we deduce that the Lucy smoothing function is unsuitable for the simulations. Also we observe that the quartic function from Liu MB (2003) is very similar to the cubic spline smoothing function. However it does not approach asymptotically zero and it needs to be truncated in order to obey rule 3. In summary, all the kernels appear to be similar except for the one introduced by Lucy (1977). Finally we must note that the smoothing length is of outmost importance in the simulations. If it is too big the simulation time increases significantly, while if it is too small will cause very few particles to be inside the support domain which makes the simulation of poor quality.

2.4. Discretization of the equations of motion

The equations of motion that will be used are the Euler equations (momentum equations) and the continuity equation, namely:

$$\frac{d\vec{u}}{dt} = -\frac{1}{\rho}\nabla P + \vec{g} \tag{24}$$

$$\frac{d\rho}{dt} = -\rho\nabla\vec{u} \tag{25}$$

It is clear that in order to obtain a discretized approximation of the above set of equations a methodology for constructing derivatives using integral approximation must be implemented.

One way to obtain a derivative is through direct differentiation of the equation (13). Given that the kernel function W is differentiable we get:

$$\frac{df(x)}{dx} = \sum_{j=1}^{N} \frac{m_j}{\rho_j} f(x_j) \frac{dW(x - x_j, h)}{dx} \tag{26}$$

By generalizing we obtain the gradient:

$$\nabla f(\vec{x_i}) = \sum_{j=1}^{N} \frac{m_j}{\rho_j} f(x_j) \nabla_i W(\vec{x_i} - \vec{x_j}, h) \tag{27}$$

Applying the above derivation to the Euler equation we get:

$$\frac{d\vec{u_i}}{dt} = -\frac{1}{\rho_i} \sum_{j=1}^{N} \frac{m_j}{p\rho_j} P_j \nabla_i W_{ij} + \vec{g} \tag{28}$$

However this form of the derivative does not conserve linear or angular momentum because the force that the particle i applies to particle j is not equal to the one of j to i.

$$\frac{m_i m_j P_i}{\rho_i \rho_j} \nabla_i W_{ij} \neq \frac{m_i m_j P_j}{\rho_i \rho_j} \nabla_j W_{ij}, P_i \neq P_j$$

In order to obtain a form of the derivative that helps conserve the momentum we will employ the Lagrangian of the particle system as presented byVioleau (2012). We get

$$L = \sum \frac{1}{2} m_j u_j^2 - \sum m_j e_{int,j} |\vec{x_i}| - \sum m_j g z_b \tag{29}$$

where the additive terms represent the kinetic energy per unit mass, the internal energy per unit mass and the potential of the gravity force respectively. The discrete equation of motion of each particle will then be provided by the Lagrange equations:

$$\frac{d}{dt} \frac{\partial L}{\partial \vec{u}_i} = \frac{\partial L}{\partial \vec{x}_i} \tag{30}$$

It can be shown that the derivative of the Lagrangian with respect to $\vec{u_i}$ is the momentum of the particle I, meaning:

$$\frac{\partial L}{\partial \vec{u}_i} = m_i \vec{u_i} \tag{31}$$

Inserting (32) to (31) we get:

$$\frac{d}{dt}(m_i\overrightarrow{u_i}) = \frac{\partial L}{\partial \vec{x}_i} \rightarrow m_i \frac{d\overrightarrow{u_i}}{dt} = \frac{\partial L}{\partial \vec{x}_i} \tag{32}$$

Also by derivation of the Lagrangian with respect to the position of the particle i we can obtain:

$$\frac{\partial L}{\partial \overrightarrow{x_i}} = -\sum m_j \frac{\partial e_{int,j}}{\partial \overrightarrow{x_i}} - \sum m_j g \frac{\partial z_j}{\partial \overrightarrow{x_i}} = -\sum m_j \left(\frac{\partial e_{int}}{\partial p}\right)_j \frac{\partial p_j}{\partial \overrightarrow{x_i}} + m_i g$$

Also it can be shown that $\left(\frac{\partial e_{int}}{\partial \rho}\right)_j = \frac{P_j}{\rho_j^2}$ resulting in:

$$\frac{\partial L}{\partial \overrightarrow{x_i}} = -\sum m_j \frac{P_j}{\rho_j^2}\frac{\partial \rho_j}{\partial \overrightarrow{x_i}} + m_i g \tag{33}$$

We know need a SPH expression of the density. From equation (13) we get:

$$\rho_i = \sum_{j=1}^{N} \frac{m_j}{\rho_j}\rho_j W_{ij} \rightarrow \rho_i = \sum_{j=1}^{N} m_j W_{ij} \tag{34}$$

By derivation of the above equation we get:

$$\frac{d\rho_j}{d\overrightarrow{x_i}} = m_i \nabla_i W_{ij} + \delta_{ij}\sum_{j=1}^{N} m_j \nabla_i W_{ij} \tag{35}$$

Substitution of (34) and (36) to (33) and some algebraic manipulations yield:

$$m_i \frac{d\overrightarrow{u_i}}{dt} = -\sum_{j=1}^{N} m_i m_j \frac{\rho_j^2 P_i + \rho_i^2 P_j}{(\rho_i \rho_j)^2}\nabla_i W_{ij} + m_i \vec{g} \rightarrow$$

$$\frac{d\overrightarrow{u_i}}{dt} = -\sum_{j=1}^{N} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2}\right)\nabla_i W_{ij} + \vec{g} \tag{36}$$

For further explanation and details regarding Lagrangian dynamics and SPH, one may refer to Violeau (2012) and Gingold RA (1982). This form of the Euler equation of motion conserves linear and angular momentum since the force that particle i bears to particle j is $m_i m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho}\right)\nabla_i W_{ij}$ which is equal and opposite to the force particle j bears to particle i. This result is crucial from a physical point of view and constitutes one of the main advantages of the method.

To express the continuity equation in SPH terms, Monaghan in Monaghan (2005) used a different approach compared to equation (27) to approximate the derivative of the discretized integral representation i.e.:

$$\nabla f(\vec{x_i}) = \frac{1}{\rho_i}\sum_{j=1}^{N} m_j \left(f(x_j) - f(x_i)\right)\nabla_i W_{ij} \tag{37}$$

The benefit of this form is that when f is constant the derivative vanishes. Therefore we obtain an expression for continuity equation:

$$\frac{d\rho}{dt} = \frac{1}{\rho_i}\sum_{j=1}^{N} m_j \overrightarrow{u_{ij}}\nabla_i W_{ij} \tag{38}$$

Where $\overrightarrow{u_{ij}} = \vec{u_i} - \vec{u_j}$.

2.5. Thermal Energy equation

The rate of change of thermal energy per unit mass is given by:

$$\frac{de}{dt} = -\frac{P}{\rho}\nabla\vec{v} \tag{39}$$

Therefore a similar procedure used for the derivation of the equation of momentum yields the SPH formulation of the thermal energy equation:

$$\frac{d\vec{e_i}}{dt} = \frac{1}{2}\sum_{j=1}^{N} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2}\right)\overrightarrow{u_{ij}}\nabla_i W_{ij} \tag{40}$$

2.6. Viscosity treatment

So far we have addressed the equations of motion without considering any kind of diffusion terms. However, when simulating problems of hydrodynamics some treatment is necessary in order to take account the viscosity of the fluid. A natural option would be the implementation of a turbulence model. But such models are complicated and are out of the scope of this thesis. Therefore we have implemented an artificial viscosity. As the name suggests, this category of viscosity terms bear no relation with real viscosity. However, they are designed to provide numerical stability to the algorithm over a range of hydrodynamical problems, and they are also very simple to use.

The artificial viscosity was introduced by Lattanzio J. C. (1986) and Monaghan (1992a). It is added to the pressure terms in the momentum and thermal energy equations. Therefore:

$$\frac{d\vec{u_i}}{dt} = -\sum_{j=1}^{N} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \Pi_{ij}\right)\nabla_i W_{ij} + \vec{g} \tag{41}$$

$$\frac{d\vec{e_i}}{dt} = \frac{1}{2}\sum_{j=1}^{N} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} + \Pi_{ij}\right)\overrightarrow{u_{ij}}\nabla_i W_{ij} \tag{42}$$

The Π_{ij} of course denote the viscous terms. This has the general form of:

$$\Pi_{ij} = \begin{cases} \dfrac{-\alpha \bar{c}_{ij}\mu_{ij} + \beta\mu_{ij}^2}{\bar{\rho}_{ij}}, & \overrightarrow{u_{ij}}\,.\overrightarrow{x_{ij}} < 0 \\ 0, & \overrightarrow{u_{ij}}\,.\overrightarrow{x_{ij}} > 0 \end{cases} \tag{43}$$

Where

$$\mu_{ij} = \frac{h\overrightarrow{u_{ij}}\,.\overrightarrow{x_{ij}}}{\overrightarrow{x_{ij}}^2 + n^2}$$

$$\bar{c}_{ij} = \frac{c_i + c_j}{2}$$

$$\bar{\rho}_{ij} = \frac{\rho_i + \rho_j}{2}$$

$$\overrightarrow{u_{ij}} = \overrightarrow{u_i} - \overrightarrow{u_j}$$

$$\overrightarrow{x_{ij}} = \overrightarrow{x_i} - \overrightarrow{x_j}$$

$$n = 0.1\frac{h_i + h_j}{2}$$

The coefficients α and β are typically set to 1, although in usual fluid dynamical problems β can be omitted (Monaghan (1992a, Gomez-Gesteira (2010). Also c represents the artificial speed of sound. It is noted that the above relationship for viscosity conserves both, the linear and angular momentum.

A simpler form of viscosity was introduced by Monaghan:

$$\Pi_{ij} = 8\frac{v_i + v_j}{\rho_i + \rho_j}\frac{\overrightarrow{u_{ij}}\,.\overrightarrow{x_{ij}}}{\overrightarrow{x_{ij}}^2} \tag{44}$$

Where $v_i = \frac{\mu_i}{\rho_i}$ is the kinematic viscosity of the fluid. This form also conserves momentum. All the vector multiplications are dot products.

2.7. Calculating the Pressure

So far we have addressed the discretization of the equations of motion (continuity and momentum), but we also have to look at the pressure field. The easiest way to deal with the calculation of pressure is by assuming quasi-compressibility and employing an equation of state. The most common used in SPH related literature is the Tait equation of state.

$$P_i = \mathrm{B}\left[\left(\frac{\rho_i}{\rho_o}\right)^\gamma - 1\right] \tag{45}$$

Where $B = \frac{\rho_o c^2}{\gamma}$ is the reference pressure and γ is usually set to 7. Parameter c is the artificial speed of sound and affects the fluctuations of density. It can be shown that the relative density deviations are approximately:

$$\frac{\Delta\rho}{\rho_o} \sim \frac{U}{c} \tag{46}$$

Where U is the characteristic flow velocity. In order to keep the fluctuations around 1%, c is set to be approximately 10 times the maximum speed of the fluid. However if gravity is important for the simulation a different rule applies:

$$c = 10\max\left(U_{max}, \sqrt{gL}\right) \tag{47}$$

The value $\sqrt{gL}$ corresponds to the wave celerity and L to the characteristic depth.

In this state equation it is assumed that water is an ideal gas under immense pressure and demonstrates adiabatic behavior. For more information the reader can refer toCole (1948).

2.8. Boundary treatment

The treatment of the boundaries is one of the main difficulties that arise in SPH method. The most common way to contain the fluid particles inside the computational domain is by representing the boundary with a set of particles that exhibit repulsive forces to them and thus creating a “wall”. Naturally these repulsive forces can have the form of Lenard-Jones molecular forces. Monaghan in Monaghan (1992a) introduced the following relationship:

$$\vec{f}_{ij}^{LJ} = \begin{cases} D\left(\left(\frac{r_o}{r_{ij}}\right)^{12} - \left(\frac{r_o}{r_{ij}}\right)^{6}\right)\frac{\vec{x}_{ij}}{r_{ij}^2}, & r_{ij} < r_o \\ 0, \quad r_{ij} \geq r_o \end{cases} \tag{48}$$

Where D is a coefficient in the order of square of the maximum velocity in the problem, r_o is the cutoff distance of the repulsive force and r_{ij} the distance between particles i and j. The cutoff distance is usually considered slightly less than the initial fluid particle spacing.

Another form of the Lenard-Jones forces was also introduced by Monaghan (1992a) and modified by Violeau (2012):

$$\vec{f}_{ij}^{LJ} = \begin{cases} D\left(\left(\frac{r_o}{r_{ij}}\right)^{4} - \left(\frac{r_o}{r_{ij}}\right)^{2}\right)\frac{\vec{x}_{ij}}{r_{ij}}, & r_{ij} < r_o \\ 0, \quad r_{ij} \geq r_o \end{cases} \tag{49}$$
$$D = CmgH$$

where C is a problem dependent coefficient ranging usually from 5 to 10, m is the average mass of

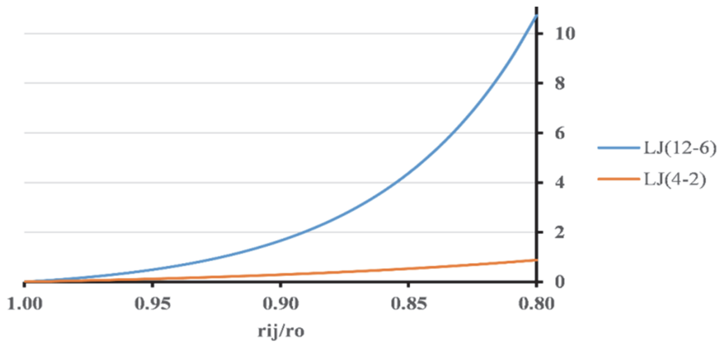


Figure 12: Graph of $\left(\frac{r_o}{r_{ij}}\right)^{12} - \left(\frac{r_o}{r_{ij}}\right)^{6}$ *and* $\left(\frac{r_o}{r_{ij}}\right)^{4} - \left(\frac{r_o}{r_{ij}}\right)^{2}$ *respectivelly*

the fluid particles and H the water depth. The latter equation seems more appealing for free surface flows because it incorporates the gravitational influence. It should be noted that these repulsive forces act pairwisely along the center of the particles. Then they are added to the momentum equation. A quantitative comparison of the two relationships follows in figure (12).

We can see that the first realization of the Lenard-Jones forces grows much faster than the second one. This could mean that it is unsuitable for use in problems were the driving force is gravity. However, numerical experiments will follow in order to test this hypothesis.

Another form of repulsive boundary force was introduced by Monaghan JJ. (1999). This formulation aimed to exhibit a constant repulsive force for a particle travelling parallel to the boundary. It was applied to the problem of a solitary wave traveling towards a beach. Perhaps for simple boundaries (i.e. a straight line) this form is superior. But in the test cases that follow the boundary geometry is not simple, therefore this form is not considered for evaluation. The repulsive forces help prevent the fluid particle from unnatural boundary penetration. This approach though has a serious

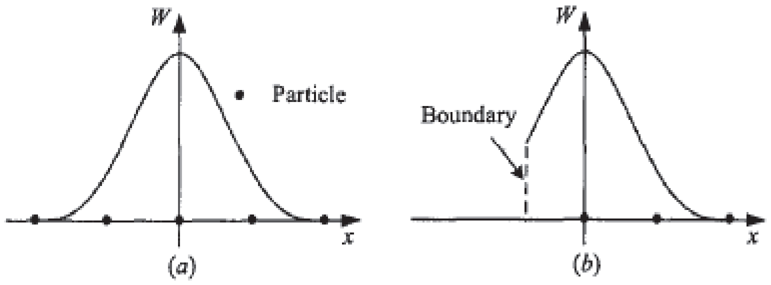


Figure 13: Particle approximation in one dimension, a) Kernel summation for a particle far from the boundary, b) Kernel summation for a particle near the boundary Liu G.R. (2003)

drawback. Naturally when integrating through space the kernels have particles in all the support domain. But when the particles approach the boundaries, only particles from inside the computational domain contribute to the summation of the kernel. This one sided integration can cause serious deficiencies near the boundaries, because the velocity might be near zero depending on the problem, but the pressure and density are not.

To this aim many researchers (see Marone (), Libersky L. D. (1993),Libersky L. D. (1995)) have introduced layers of particles, beyond the boundary, in order to complete the kernel integration. In this thesis we will implement the method of the dummy particles. This is a very simple method that consists of layers of particles outside the boundary that carry the fluid properties but do not evolve in time. Their only action is to contribute to the summation of the kernel. Although the number of layers theoretically depends on the kernel radius, experience shows that two or three layers are

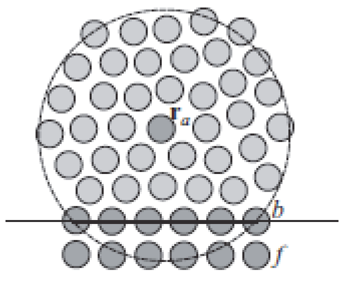


Figure 14: One layer of dummy particles and the LJ particles in the boundary, Violeau (2012)

enough Violeau (2012).

However, assigning the dummy particles the proper values of pressure can sometimes be difficult. Usually the assignment is done using the expected values of pressure along the boundary. Although in some problems the pressure along the boundary from theory or experiments (i.e. hydrostatic pressure in a tank), in some other problems this information is not available. Furthermore structural instabilities in the pressure field of the particles can cause unnatural spikes in pressure, greater than the assigned dummy particle pressure, thus resulting in a pressure gradient towards the boundary. This causes the particles to stick to the boundary and disturb the pressure field even more. In order to take care of this situation Adami et al Adami S. (2012) developed a generalized framework to deal with the boundaries. This approach consists of a series of fixed boundary particles placed in a Cartesian grid representing the boundary. However their property values progress in time in order to satisfy a force equilibrium at the fluid-boundary interface. Moreover the boundary particles must cover the entire support domain of the fluid particles.

Assuming a body force balance at the fluid-boundary interface and integrating through the support domain of the boundary particle we get

$$P_i^B = \frac{\sum_{j=1}^{N} P_j^F W_{ij} + (\vec{g} - \vec{a}_B) \sum_{j=1}^{N} \rho_j^F \vec{x}_{ij} W_{ij}}{\sum_{j=1}^{N} W_{ij}} \tag{50}$$

where B and F denote boundary and fluid particles respectively and $\vec{a}_B$ is the acceleration of the boundary in the case of a moving one.

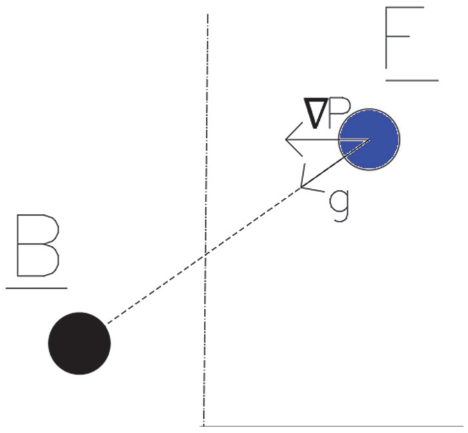


Figure 15: Forces acting upon a boundary particle from a single fluid particle

Note that the summation is performed over the fluid particles domain, thus projecting the pressure field in the boundary. Equation (46) can be simplified in the case of stationary boundaries and Cartesian coordinates as

$$P_i^B = \frac{\sum_{j=1}^{N} P_j^F W_{ij} - g \sum_{j=1}^{N} p_j^F z_{ij} W_{ij}}{\sum_{j=1}^{N} W_{ij}} \tag{51}$$

For each time step the pressure of the boundary is calculated using equations (46) or (47), then the particle density is updated through the equation of state.

$$\rho_i^B = \rho_o \left(\frac{P_i^F}{B} + 1 \right)^{1/\gamma} \tag{52}$$

Using the same method a free-slip boundary condition can be imposed, by omitting the dummy particles in the calculation of the viscous terms. A non-slip boundary condition can be imposed by extrapolating the velocity field of the fluid particles in the boundary particles and inserting them in the calculation of the viscous terms. The velocity of the boundary particles is given by:

$$\vec{v}_i^B = 2\vec{v}_i^{B,p} - \bar{v}_i \tag{53}$$

where $\bar{v}_i$ is the extrapolated velocity:

$$\bar{v}_i = \frac{\sum_{j=1}^{N} \vec{v}_i^F W_{ij}}{\sum_{j=1}^{N} W_{ij}} \tag{54}$$

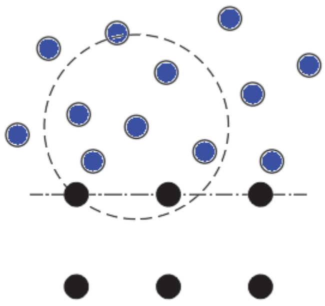


Figure 16: Representation of Adami method boundary particles.

Another method, called dynamic boundary particles method, was developed by Dalrymple R. A. (2001) and it was studied in depth by Crespo A. J. C. (2007). According to the proposed approach the boundaries are represented by a set of particles, such as the dummy particles mentioned above, but this time the density and pressure evolve in time with the same equations as the fluid. The repulsive mechanism is quite simple. When a fluid particle approaches the boundary the density and pressure rise, consequently creating a pressure gradient towards the computational domain. Crespo (2007) has indicated that Crespo A. J. C. (2007) this is essentially an elastic collision. The benefits of this approach include the reduced computational overhead and the simplification of the algorithm, as the boundary particles are treated in the same manner as the fluid ones. Numerical experiments have shown that the optimal placement of the boundary particles is in staggering form with spacing less than the initial fluid particle spacing.

In practice, both, the Lenard–Jones repulsive forces and the dummy particle methods can be used independently as well as simultaneously.

2.9. Normalization of the flow

Some of the main drawbacks of the SPH method include unstable pressure and velocity fields. To overcome this two simple techniques may be used, density reinitialization and XSPH. The X in the later stands for the unknown factor.

2.9.1. Density reinitialization

Density fluctuations occurring in SPH simulations create an inherent problem. Many solutions for this have been proposed by different researchers such as kernel correction. However the simplest and most computationally effective one is the zeroth-order density correction or Shepard filter. This technique was implemented by Colagrossi Andrea (2003) and Gomez-Gesteira (2010) as follows. Every n steps (usually $n = 30$) the density of the particles is corrected with:

$$\rho_i^{new} = \sum_{j=1}^{N} \rho \widetilde{W_{ij}} \frac{m_j}{\rho_j} = \sum_{j=1}^{N} m_j \widetilde{W_{ij}} \tag{55}$$

where the kernel has been corrected by zeroth-order correction:

$$\widetilde{W_{ij}} = \frac{W_{ij}}{\sum_{j=1}^{N} W_{ij} \frac{m_j}{\rho}} \tag{56}$$

When the mass is the same for all particles the above relationships yield:

$$\rho_i^{new} = \sum_{j=1}^{N} m_j \frac{W_{ij}}{\sum_{j=1}^{N} W_{ij} \frac{m_j}{\rho_j}} = \sum_{j=1}^{N} m_j \frac{W_{ij}}{m_j \sum_{j=1}^{N} W_{ij} \frac{1}{\rho}} \rightarrow$$

$$\rho_i^{new} = \sum_{j=1}^{N} \frac{W_{ij}}{\sum_{j=1}^{N} \frac{W_{ij}}{\rho_j}} \tag{57}$$

It must be noted that in case a dummy particle method is used to represent the boundaries, the density reinitialization takes into account only the fluid particles. Another density correction of the first order called the Moving List Squares, has been implemented by Colagrossi Andrea (2003). However when implemented momentum is not conserved and therefore this method will not be further discussed in this thesis.

2.9.2. XSPH

This technique was introduced by Monaghan (1992b). It aims to move the particles closer to the average velocity of the neighboring particles and thus to reduce the velocity fluctuations. The particle velocity is written as

$$\frac{d\vec{x}_i}{dt} = \vec{v}_i + \varepsilon \sum_{j=1}^{N} \frac{m_j}{\bar{\rho}_{ij}} \vec{u}_{ji} W_{ij} \tag{58}$$

where $0 \leq \varepsilon \leq 1$ and $\bar{\rho}_{ij} = \frac{\rho_i + \rho_j}{2}$.

The XSPH variant has proven to be useful in the simulation of nearly incompressible fluids such as water, where it keeps the particles orderly in the absence of viscosity. Another interesting feature of the XSPH variant is that if pressure and viscous forces are set to zero, it simulates the Burgers equation with very large effective Reynolds number Monaghan (1992b). This method with some modifications can represent a turbulence model as noted by Violeau (2012).

It is very important that XSPH does not break conservation of linear and angular momentum.

2.10. Time integration

The numerical time integration scheme used is a very important factor for the accuracy and the stability of the method. The scheme employed must be time reversible and conservative for linear and angular momentum. The schemes to be presented include these properties and are suitable for SPH simulations. One of the most commonly used schemes in the literature is the Predictor-Corrector algorithm as described by Monaghan (1989). Superscripts t, $t+1/2$ and $t+1$ denote the time step, where t is the present, whereas the subscripts i denote the particle.

First the values at half the time step are predicted using Euler's method:

$$\vec{u}_i^{t+1/2} = \vec{u}_i^t + \frac{\Delta t}{2}\frac{\overrightarrow{du_i^t}}{dt}, \vec{\rho}_i^{t+1/2} = \vec{\rho}_i^t + \frac{\Delta t}{2}\frac{\overrightarrow{d\rho_i^t}}{dt}, \vec{x}_i^{t+1/2} = \vec{x}_i^t + \frac{\Delta t}{2}\frac{\overrightarrow{dx_i^t}}{dt} \tag{59}$$

Then these values are corrected using the forces at the half step.

$$\vec{u}_i^{t+1/2} = \vec{u}_i^t + \frac{\Delta t}{2}\frac{\overrightarrow{du_i^t}}{dt}, \vec{\rho}_i^{t+1/2} = \vec{\rho}_i^t + \frac{\Delta t}{2}\frac{\overrightarrow{d\rho_i^t}}{dt}, \vec{x}_i^{t+1/2} = \vec{x}_i^t + \frac{\Delta t}{2}\frac{\overrightarrow{dx_i^t}}{dt} \tag{60}$$

Finally the values of the field variables are calculated at the end of the time step using:

$$\vec{u}_i^{t+1} = 2\vec{u}_i^{t+1/2} - \vec{u}_i^t, \vec{\rho}_i^{t+1} = 2\vec{\rho}_i^{t+\frac{1}{2}} - \vec{\rho}_i^t, \vec{x}_i^{t+1} = 2\vec{x}_i^{t+1/2} - \vec{x}_i^t \tag{61}$$

Note that in order to use this method, we must calculate the derivatives of the field variables again for the half step, meaning essentially for each time steps we will have to do the calculations twice. Therefore this method increases significantly the computational effort.

Another well-known method in the literature is the Verlet scheme introduced by Verlet (1967). According to this method the calculations consist of two parts. In general the field variables are calculated according to:

$$\vec{u}_i^{t+1} = \vec{u}_i^{t-1} + 2\Delta t\frac{\overrightarrow{du_i^t}}{dt}, \vec{\rho}_i^{t+1} = \vec{\rho}_i^{t-1} + 2\Delta t\frac{\overrightarrow{d\rho_i^t}}{dt}, \vec{x}_i^{t+1} = \vec{x}_i^t + \Delta t\frac{\overrightarrow{dx_i^t}}{dt} + 0.5\Delta t^2\frac{\overrightarrow{du_i^t}}{dt} \tag{62}$$

once every 50 steps, in order to prevent the integration from diverging, the calculations are done from equation:

$$\vec{u}_i^{t+1} = \vec{u}_i^t + \Delta t\frac{\overrightarrow{du_i^t}}{dt}, \vec{\rho}_i^{t+1} = \vec{\rho}_i^t + \Delta t\frac{\overrightarrow{d\rho_i^t}}{dt}, \vec{x}_i^{t+1} = \vec{x}_i^t + \Delta t\frac{\overrightarrow{dx_i^t}}{dt} + 0.5\Delta t^2\frac{\overrightarrow{du_i^t}}{dt} \tag{63}$$

This scheme needs only one force calculation per time step, but requires smaller time steps in general. For a more in depth look of the time integration schemes the reader can refer to Leimkuhler Benedict J. (1996).

2.11. Time stepping

The time step of the simulation is as important as the numerical integration scheme employed, depends on it, and greatly affects the stability, the accuracy and the computational effort. Therefore some criteria have been set for its evaluation that have also physical meaning. According to Monaghan JJ. (1999) time step is set using two criteria. The first is the Courant-Friedrichs-Levy (CFL) Criterion. It states that in a given time step Δt the particle must not travel further than the spatial resolution of the computational domain, i.e.

$$\Delta t_{cv} = min_i \left(\frac{h_i}{c_i + max\left(\left| \frac{h\overrightarrow{u_{ij}} \cdot \overrightarrow{x_{ij}}}{\overrightarrow{x_{ij}}^2} \right| \right)} \right) \tag{64}$$

One may note that application of the CFL criterion takes into account the viscous forces as well.

The second criterion is the force per unit of mass. It states that that for a given time step Δt the particle must not travel a distance greater than that which is imposed by the smallest acceleration

$$\Delta t_f = min_i \left(\sqrt{\frac{h_i}{|f_a|}} \right) \tag{65}$$

where $f_a = \sqrt{\left(\frac{dv}{dt}\right)^2 + \left(\frac{dw}{dt}\right)^2}$ in 2D space. Finally the time step is calculated by:

$$\Delta t = 0.3min(\Delta t_{cv}, \Delta t_f) \tag{66}$$

A failure to regulate the time step correctly can cause great instabilities. While a very short time step increases the computational effort, a very large one can cause instabilities. In order to depict the above let us consider two cases. The first when two fluid particles approach each other, and the second when a fluid particle approaches the boundary.

In the first case a very large time step will cause the particles to just pass by each other resulting in no interaction between them, whereas the correct time step will result in a progressive interaction.

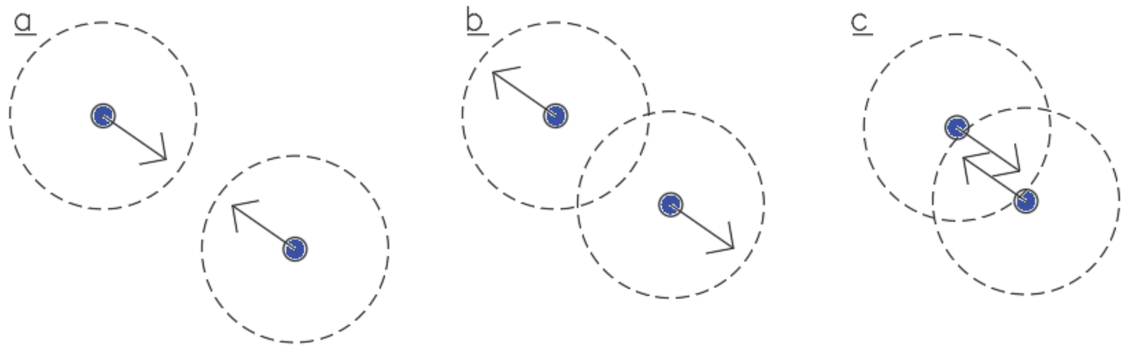


Figure 17 Fluid particles interacting. a) the particles approaching, b) large time step, c) correct time step

In the second case a large time step will result in two situations. Either the fluid particle will end up very close to the boundary particle which will cause extremely large Lenard-Jones repulsive forces thus making the fluid unnaturally bump, or the fluid particle will pass through the boundary and thus the Lenard-Jones forces will push it outside the computational domain. A correct time step will result in a progressive interaction between the boundary and the fluid.

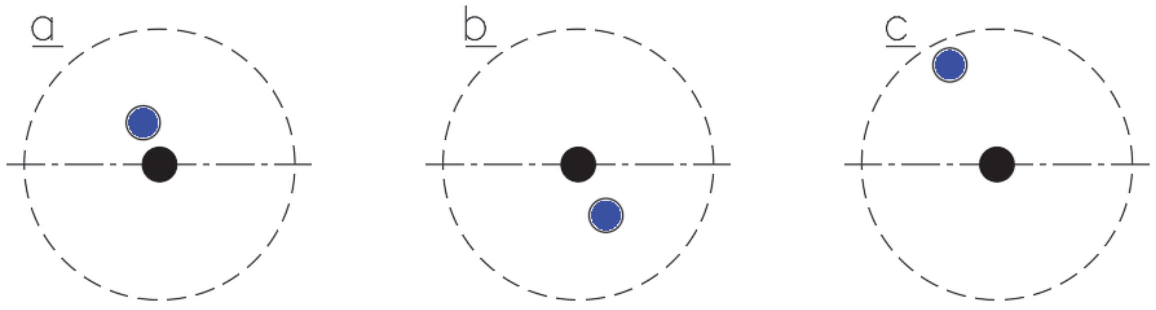


Figure 18: Fluid and boundary particles interaction. a) fluid particle very close to boundary, b) fluid particle passing by the boundary, c) fluid particle smoothly enters the support domain of the boundary

2.12. Summary

In this chapter the governing equations of SPH method have been derived and their main characteristics were discussed in depth. The basic components of the method are summarized in the tables below.

Table 2: Summary of basic SPH Kernels

Kernel Name	*Equation*
Gaussian	$W(R,h) = a_d e^{-R^2}$
Cubic Spline	$W(R,h) = a_d \begin{cases} \frac{2}{3} - R^2 + \frac{1}{2}R^3, & 0 \le R < 1 \\ \frac{1}{6}(2-R)^3, & 1 \le R, 2 \\ 0, & R \ge 2 \end{cases}$
Quartic Spline	$W(R,h) = a_d \begin{cases} (R+2.5)^4 - 5(R+1.5)^4 + 10(R+0.5)^4, & 0 \le R < 0.5 \\ (2.5-R)^4 - 5(1.5-R)^4, & 0.5 \le R < 1.5 \\ (2.5-R)^4, & 1.5 \le R < 2.5 \\ 0, & R > 2.5 \end{cases}$
Wendland	$W(R,h) = a_d \begin{cases} \left(1 - \frac{R}{2}\right)^4 (1+2R), & 0 \le R \le 2 \\ 0, & R > 2 \end{cases}$
Liu and Liu	$W(R,h) = a_d \begin{cases} \frac{2}{3} - \frac{9}{8}R^2 + \frac{19}{24}R^3 - \frac{5}{32}R^4, & 0 \le R \le 2 \\ 0, & R > 2 \end{cases}$

Table 3: Summary of basic SPH model equations

Variable	*Equation*
Density	$p_i = \sum_{j=1}^{N} m_j W_{ij}$
Continuity	$\frac{dp_i}{dt} = \frac{1}{p_i} \sum_{j=1}^{N} m_j \overrightarrow{u_{ij}} \nabla_i W_{ij}$
Momentum	$\frac{d\overrightarrow{u_i}}{dt} = -\sum_{j=1}^{N} m_j \left(\frac{P_i}{p_i^2} + \frac{P_j}{p_j^2} + \Pi_{ij} \right) \nabla_i W_{ij} + \vec{g} + \vec{f}_{ij}^{LJ}$
Energy	$\frac{d\overrightarrow{e_i}}{dt} = \frac{1}{2} \sum_{j=1}^{N} m_j \left(\frac{P_i}{p_i^2} + \frac{P_j}{p_j^2} \right) \overrightarrow{u_{ij}} \nabla_i W_{ij}$
Velocity	$\frac{d\vec{x}_i}{dt} = \vec{v}_i + \varepsilon \sum_{j=1}^{N} \frac{m_j}{\bar{p}_{ij}} \vec{u}_{ji} W_{ij}$
Pressure	$P_i = \frac{p_o c^2}{\gamma} \left[\left(\frac{p_i}{p_o} \right)^{\gamma} - 1 \right]$

Table 4: Summary of problem depended parameters

Parameter	*Use*	*Equation*
h	Smoothing length	18-23,43
a	Viscosity parameter	43
b	Viscosity parameter	43
B	Reference density	45
γ	Equation of state parameter	45
c	Speed of sound	43,45,47
ρ_o	Initial density	45
D	Parameter of Lenard-Jones	48
C	Parameter of Lenard-Jones coefficient	49
ε	XSPH parameter	58

3. Algorithm implementation of SPH method

In the previous chapter we have presented the key components of the SPH method. In this chapter the method will be implemented and a computer program will be assembled. The algorithm was created to meet the following basic requirements:

- The algorithm needs to be robust. In order to solve different problems the user should only be required to change the geometry, the initial conditions and some parameters. Nothing more.
- Robustness means simplicity. A simple code is easy to troubleshoot and easier to read from different users.
- The code should be as fast as possible. Simple problems could require a number of particles of order 10^4 and run steps of order 10^5, therefore a slow code would make computations last for weeks, thus rendering the code useless.
- The machine-user communication is very important. Visualization routines must be implemented for quantitative assessment of the results.

Matlab has been chosen to be the programming language. Matlab is an interpreted high level language developed to be used by scientists and engineers. Naturally as an interpreted language it is considerably slower than a compiled low level language, but offers some great advantages in return. First of all it enables interactive work with the data, thus making the development of the computer program much easier. Secondly it can run commands instantly without waiting to compile, therefore making it possible to access the results instantaneously. A significant drawback realized by the author is its limitations when it comes to parallelization of the code in Matlab. There is a limitation of 12 cores when using conventional parallelization techniques offered with Matlab suite. This problem could be overcome by using the pMatlab package developed by Ms. Nadya T. Bliss and Dr. Jeremy Kepner of the MIT Lilcoln Laboratory. This package incorporates MPI protocol in Matlab and cancels the core limitation. However it requires a very advanced level of programming. The later was not used in this thesis.

Parallelization was the only way to achieve a high performance code. Because all the algorithm is structured in matrices and calculation method is explicit, parallelization was relatively easy. This attribute enabled us to divide the matrices within the cores (referred in Matlab as workers) independently, without the need to communicate with each other. This is done by Matlab using the parfor function instead of the classic for. Apparently a very well structured code and some special considerations are required.

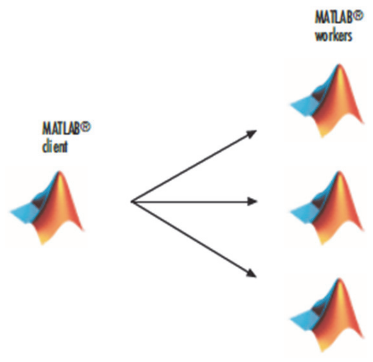


Figure 19: Matlab parallelization scheme

Development of the code was by no means a linear procedure. For educational purposes all the different versions of the code will be presented, discussed and finally compared in the chapters to follow.

In this chapter we will present the algorithm in steps, then each step will be discussed in depth. The full code is available in Appendix C and online in GitHub.

3.1. Implementation #1

The first approach is shown in the following flowchart:

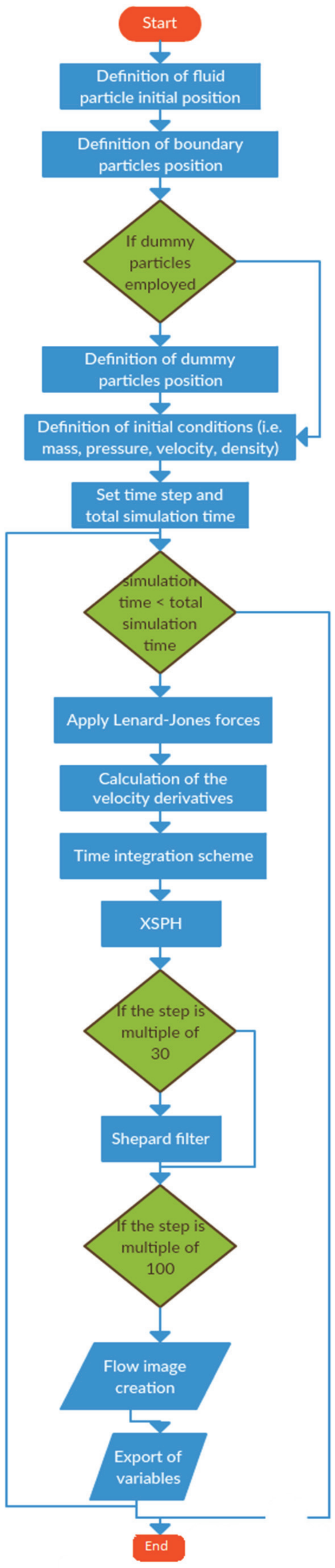

Start
Definition of fluid particle initial position
Definition of boundary particles position
If dummy particles employed
Definition of dummy particles position
Definition of initial conditions (i.e. mass, pressure, velocity, density)
Set time step and total simulation time
simulation time < total simulation time
Apply Lenard-Jones forces
Calculation of the velocity derivatives
Time integration scheme
XSPH
If the step is multiple of 30
Shepard filter
If the step is multiple of 100
Flow image creation
Export of variables
End

Figure 20: Implementation #1 flowchart

Basically the algorithm consists of four parts:

i. Definition of geometry and initial conditions
ii. Calculation of the equations of motion
iii. Time integration and flow normalization
iv. Post processing

i. Definition of geometry and initial conditions

This part is the simplest of the algorithm. First the geometry is defined. The fluid particles are placed in a regular Cartesian grid according the initial spacing. Their coordinates are assigned in matrices YF and ZF, position along y axis and z axis respectively. Next, the boundaries are formed using boundary particles. Their coordinates are assigned in matrices YB and ZB, position along y axis and z axis respectively. Subsequently the user enters the initial values for the mass, velocity, pressure, density and the values of the problem parameters shown in table 4. The field variables are stored in a one-dimensional (1D) matrices with length equal to the number of fluid particles. Therefore each particle is immediately assigned with a unique identification (ID) which corresponds to the number of the cell where its properties are stored. Note that if we use the dummy particles boundary method along with the Lenard-Jones repulsive forces, these particles are added at the end of the fluid particle matrices.

ii. Calculation of the equations of motion

After defining the problem parameters in the first step a while loop is set in order to check if the simulation time has exceeded the total simulation time. It is stated earlier that only neighboring particles contribute to the summation for the calculation of equations (38) , (41) (42). Therefore a parfor loop is set from 1 to the number of fluid particles. For each particle the distance from other particles is determined. If we use the dummy particle boundary method, then these particles are treated as fluid particles, rather than treated separately. The result is low computational overhead when using this boundary approach. Then a logical operation to find the particles inside the support domain follows. This is the following piece of code:

The matrix *c* is the result of the logical operation and consists of zeroes and ones, if particles are

```
r=sqrt((YF-YF(i)).^2+(ZF-ZF(i)).^2);
k=r./h(i);
c=bsxfun(@le,k,2);
```

located outside or inside the support domain respectively. For a detailed list of all the variable names the reader can refer to appendix A. Next, the kernel derivatives, the viscosity terms and the pressure-density terms for all particles are calculated.

```
partsum=(P(i)/p(i)^2+P./p.^2).*dWy+ArV.*dWy;
```

The variable *partsum* stands for partial summation and is one of many generic variable names used in the code. The reason behind using generic names is that they essentially perform the same task, in this case to keep a partial summation temporarily, and therefore multiple names would unnecessarily complicate the code.

Now we have formed *partsum,* a matrix with the contribution of all the particles. If we multiple this matrix with *c* the following operations will occur:

$$\begin{matrix} 0 \\ 1 \\ 0 \end{matrix} .* \begin{matrix} partsum(1) \\ partsum(2) \\ partsum(3) \end{matrix} = \begin{matrix} 0 \\ partsum(2) \\ 0 \end{matrix}$$

The resulting table contains only the contributions of the particles located inside the computational domain. Finally adding the rows of the resulting matrix we end up with the velocity derivative. In the same manner the particles inside the influence of the boundary are located and the Lenard-Jones forces are calculated.

iii. Time integration and flow normalization

Once the derivatives of velocity, density and energy are calculated, the new velocities and particle positions are updated using one of integration schemes presented in chapter 2.10. If dummy (boundary) particles are involved, there is no need for special consideration because the derivatives assigned to them are set to zero and they do not evolve. Subsequently we use the XSPH scheme. Note that the neighboring particles are located using the old position of the particles and not the ones calculated in the time integration step. The same procedure followed in step ii for the calculations is used here. More specifically, the neighboring particles are determined and the same logical operations involving zeros and ones are conducted in order to retain only the matrices' positions of the neighboring particles. Finally the Shepard filter is applied if necessary, following again the same procedure. If dummy particles are used, in the calculations only the section of the matrices containing the fluid particles must participate.

iv. Post processing

In this part of the code we export the position, velocity, acceleration, density, pressure and smoothing length of the particles in '.mat' file format every 100 steps. Off course the number of steps may vary accordingly. A scatter plot of the computational space is created with the color depicting the velocity of the particles. The color can be easily changed in order to demonstrate other fluid variables such as density or pressure. Also a sub-routine named 'EnergyCalc' was created to calculate and monitor the total energy of the fluid in each time step. Finally a sub-routine is incorporated to create a '.gif' image of the simulation. Apparently one can employ more elaborate visualization tools. However, from our point of view these are not essential for the engineers and researchers.

After the code was completed, it run in 12 Xeon E5-2630 v3 2.4 GHz cores. It performed poorly despite the good computing power available. It took the computer approximately 9 minutes to complete a single time step for 90000 particles, which is not an acceptable performance. The profiler was used to identify the slowest portions of the code. Part ii of the code was found to be the most time consuming. To test and improve the performance of the code a toy model called 'ToyModel2D_1' was set up containing only part ii. Multiple runs of the script were performed for 90000 particles. These resulted in an average run time of 270 seconds. Note that an increase in the number of particles does not result in a linear increase of computational time. For example, the ToyModel2D_1 was run several times for 175000 particles resulting in an average run time of 2805 seconds per step, which is 10 times that of 90000 particles.

3.2. Implementation #2

The first version of the algorithm performed computations with matrices of size equal to the number of fluid particles, or fluid particles and the dummy ones, henceforth called NF. Usually, the number of particles located within the support domain is less than 100. This means that a large amount of time is spent in unnecessary calculations. In order to perform only the necessary calculations, an extra step was added to the algorithm before the calculation of the derivatives. In this step, for a given particle, once the calculation of the distance from other particles is obtained, the neighboring particle ID's are defined in a new matrix. Using this matrix, new matrices containing only the field variables of the particles within the support domain are created and denoted with subscript k (i.e. for velocity along y axis vk). The size of these matrices depends on the spatial resolution of each problem. In typical SPH problems this number is less than 100. Consequently whenthese matrices are used for the computation of derivatives for each particle the computational effort is significantly reduced.

To test the performance of the new version of the algorithm a new toy model script 'ToyModel2D_2' was created. Multiple runs of the script were performed for 90000 particles resulting in an average run time of 50 seconds. This is 5.4 times faster than the previous version. The same script was run multiple times for 175000 particles resulting in an average run time of 312 seconds. If compared to ToyModel2D_1 it is 9 times faster. It is important to note that this algorithm is not affected by the number of particles as much as the previous one, because only neighboring particles contribute to the matrix calculations. The flow chart is following in figure 22.

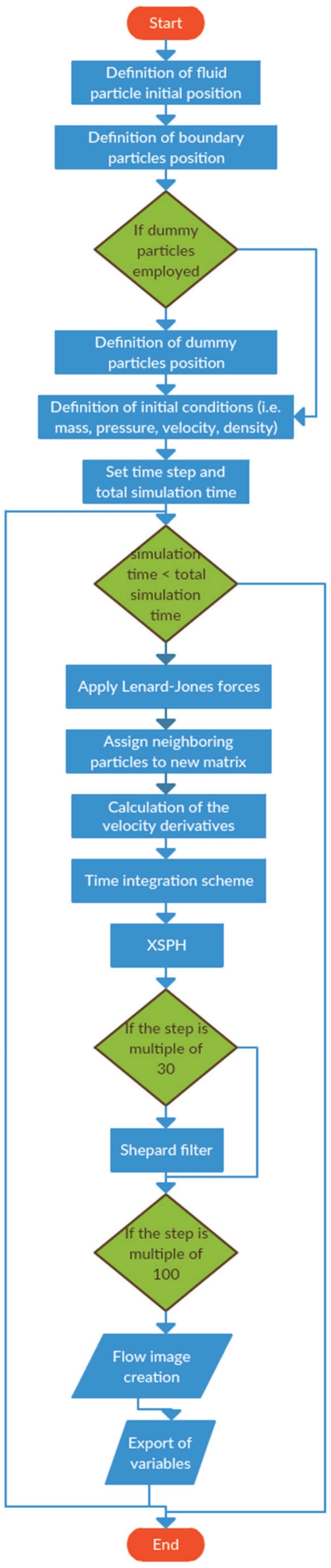

Start
Definition of fluid particle initial position
Definition of boundary particles position
If dummy particles employed
Definition of dummy particles position
Definition of initial conditions (i.e. mass, pressure, velocity, density)
Set time step and total simulation time
simulation time < total simulation time
Apply Lenard-Jones forces
Assign neighboring particles to new matrix
Calculation of the velocity derivatives
Time integration scheme
XSPH
If the step is multiple of 30
Shepard filter
If the step is multiple of 100
Flow image creation
Export of variables
End

Figure 21: Implementation #2 flowchart

Despite the significant improvement in the performance, the new version is still not adequate. As mentioned earlier, a typical simulation of 10^5 steps would require around 130 days to complete, making the code useless. At this point three choices were considered. The first one was to compile the heavier (slower) parts of code into MEX (Matlab Executable) functions that was done using built in functions of Matlab along with a C or C++ compiler. However this approach yielded no results. The second solution was to employ sparse matrices calculations, but that wasn't adequate as well. The third solution was to further refine the algorithm, as shown in the following section.

3.3. Implementation #3

In order to locate the weak link of the algorithm, the code was profiled again. It was found that the most time consuming procedure was the calculation of the distance of a given particle from the particles inside the support domain, as well as the logical operations to locate these particles. Until now this procedure required NF^2 operations during each time step, which has been noted by other researchers as well. Liu (2003) Liu G.R. (2003)proposed a solution according to which the particles are placed in a spatial grid and the search for the particles inside the support domain is conducted only on neighboring cells rather than the whole computational domain thus significantly reducing the size of the search domain. The proposed method is an improvement of the earlier one. The new algorithm consists of five parts.

i. Definition of geometry and initial conditions
ii. Grid creation and assignment of the particles in the cells
iii. Calculation of the equations of motion
iv. Time integration and flow normalization
v. Post processing

The former parts remain essentially the same, with some minor adjustments to the new method.

ii. Grid creation and assignment of particles to the cells

As the name suggests this method consists of a spatial grid to place the particles in order to reduce the search effort. In order to optimize the computational effort this grid has cells with size equal to the diameter of the smoothing kernel. For example, if we use the kernel function (24) which has a support domain of 2, the cells will have a side of $2h$. This reduces the search from the entire computational domain to the adjacent cells.

Each step the outlying positions along y and z axes are found. The grid is created and the particles are assigned to each cell. Numbering of the cells starts from the bottom left of the grid. If a grid has width w columns and height h lines it contains $(w-1)(h-1)$ cells. The first implementation of this method involved a fixed grid over the computational domain. The proposed implementation involves a dynamic grid in time, so that there are as few empty cells as possible, thus reducing the computational effort. That is achieved by the creation of a new grid each time step. In figure (22) the evolution of the grid is shown according to the movement of the outlying particles.

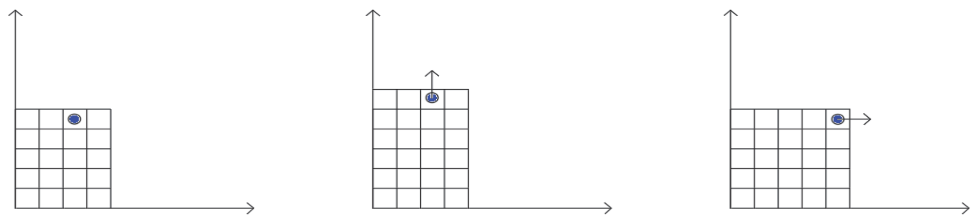


Figure 22: Evolution of the spatial grid in time

However we cannot avoid having empty cells in all problems. Therefore the algorithm is designed to omit the empty cells that occur, thus reducing the iterations needed. For performance improvement of the algorithm no 'if' statements were included and comparisons are performed by logical operations.

The procedure will be explained step by step. First the particles are assigned to the grid cells using the code below:

```
for i=1:l1-1
        for j=1:l2-1
            I=find(and(YF>GY(i+1,j) & YF<=GY(i+1,j+1),ZF>GZ(i,j+1) &
ZF<=GZ(i+1,j+1)));
            GP(I)=(i-1)*(l2-1)+j;
            npic((i-1)*(l2-1)+j)=size(I,1);
        end
    end
```

where GY and GZ are matrices containing the coordinates of the cells. GP is a matrix with the size of NF that stores the cell number each particle is contained. Note that if the particle (i) is not placed inside a cell then $GP(i) = 0$. This will cause an error to the code.

Once cell numbers that contain particles are located, their adjacent cells are found and assigned in the matrix pc. This matrix has 9 columns and number of rows equal to the number of cells containing particles. In each column, the adjacent cell number is assigned according to the figure (23):

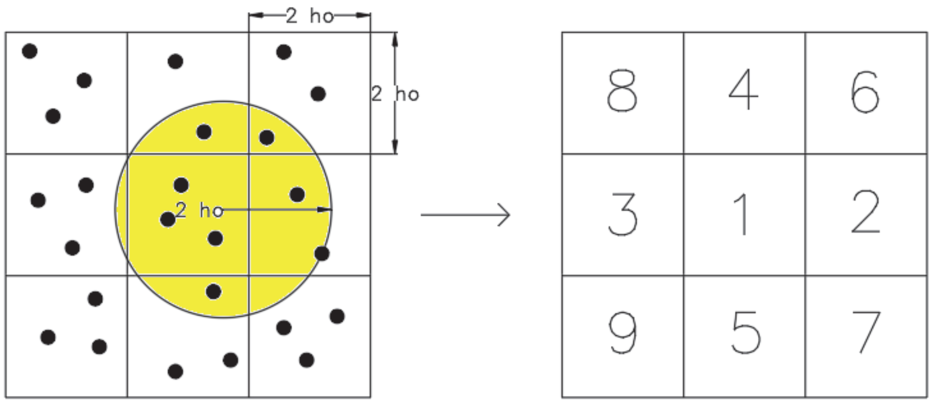


Figure 23: Neighboring cells assignment method

The code that executed the previous is:

```
for i=1:size(uGP,1)
        gp=uGP(i);
        ipc(gp)=i;
        pc(i,1)=gp;
        gate=gp/(size(GY,2)-1)==int32(gp/(size(GY,2)-1));
        pc(i,2)=min((1-gate)*(gp+1),(l1-1)*(l2-1));
        pc(i,2)=(pc(i,2)>gp)*pc(i,2);
        gate=(gp-1)/(size(GY,2)-1)==int32((gp-1)/(size(GY,2)-1));
        pc(i,3)=(1-gate)*(gp-1);
        pc(i,4)=((gp+l2-1)/(l2-1)<(l1-1))*(gp+l2-1);
        pc(i,5)=((gp-(l2-1))/(l2-1)>0)*(gp-(l2-1));
        pc(i,6)=(pc(i,2)>0)*(pc(i,4)>0)*(gp+1+l2-1);
        pc(i,7)=(pc(i,2)>0)*(pc(i,5)>0)*(gp+1-(l2-1));
        pc(i,8)=(pc(i,3)>0)*(pc(i,4)>0)*(gp-1+l2-1);
        pc(i,9)=(pc(i,3)>0)*(pc(i,5)>0)*(gp-1-(l2-1));
end
```

Finally when the step 3 is executed for its particle a matrix containing the ID's of all neighboring particles must be created. That is done by a for loop

To test the performance of the new implementation a new toy model script ToyModel2D_7 was created. The versions between ToyModel2D_2 and ToyModel2D_7 followed the MEX approach. Multiple runs of the script were performed for 90000 particles. These resulted in an average run time of 15 seconds. That is 3.3 times faster if compared to the ToyModel_2. The same script was run multiple times for 175000 particles resulting in an average run time of 46 seconds.

Therefore, one may note that the last implementation not only performs better as far as the overall speed is concerned, but it is also less affected by the increase of the number of particles.

The flow chart is given below:

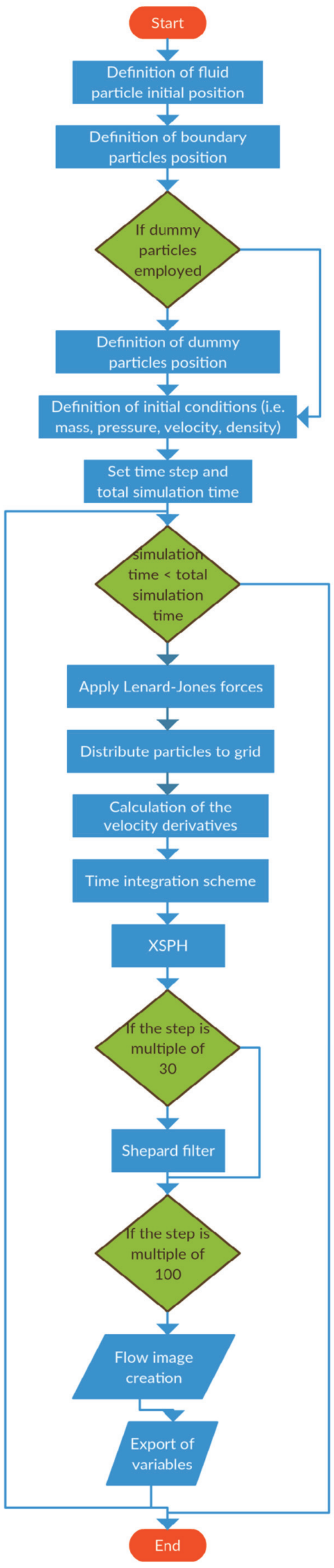

Start
Definition of fluid particle initial position
Definition of boundary particles position
If dummy particles employed
Definition of dummy particles position
Definition of initial conditions (i.e. mass, pressure, velocity, density)
Set time step and total simulation time
simulation time < total simulation time
Apply Lenard-Jones forces
Distribute particles to grid
Calculation of the velocity derivatives
Time integration scheme
XSPH
If the step is multiple of 30
Shepard filter
If the step is multiple of 100
Flow image creation
Export of variables
End

Figure 24: Implementation #3 flowchart

3.4. Improved Implementation #3

According to the third implementation of the method, when the ID's of the neighboring particles are located, they are concatenated in a single vector. However it's cells do not contain the same number of particles. Consequently when using a matrix to store the ID's of them, there will be empty spaces expressed as zeros. In order to remove them we employ a 'for' loop from 1 to 9 removing the zeros and concatenating. However this method is suboptimal considering that this procedure is applied twice, once for the calculation of the derivatives and once for the calculation of the XSPH correction.

In order to find a better algorithm to tackle the 'for' loop we employed a structure to store the ID's of the particles contained in each cell. The structure has a length equal to the number of cells and can store matrices of different sizes in each sub-structure. Consequently a simple concatenation takes the place of the for loop.

```
I1=[s(I0(1)).a.b;s(I0(2)).a.b;…s(I0(9)).a.b
```

To test the performance of the new implementation a new toy model script ToyModel2D_9 was created. Multiple runs of the script were performed for 90000 particles. These resulted in an average run time of 13 seconds. That is 2 sec. faster than ToyModel_7. The same script was run multiple times for 175000 particles resulting in an average run time of 28 seconds. The last improvement might not have yielded such large difference in time execution as previous ones, however it enables us to run XSPH velocity correction scheme without recreating the grid, thus saving significant computational time which does not show in these tests.

3.5. Version Comparison

The performance results of the various versions of the computer code built are presented in table 3.

Table 5: Performance comparison between versions

	Particles	Time (sec.)	ToyModel2D_1	ToyModel2D_2	ToyModel2D_7
ToyModel2D_1	90000	270	-	-	-
	175000	2805	-	-	-
ToyModel2D_2	90000	50	5.4	-	-
	175000	312	9	-	-
ToyModel2D_7	90000	15	18	3.3	-
	175000	46	61	6.8	-
ToyModel2D_9	90000	13	20.7	3.8	1.15
	175000	28	100	11	1.6

Moreover, it must be noted that for the number of particles used in each simulation there is an optimal number of cores to distribute the load. As an example, of the flow of 20000 fluid particles was executed in a Xeon E5-2630 v3, 2.4 GHz processor. In the figure below we have plotted the timed needed to complete each time step versus the number of cores.

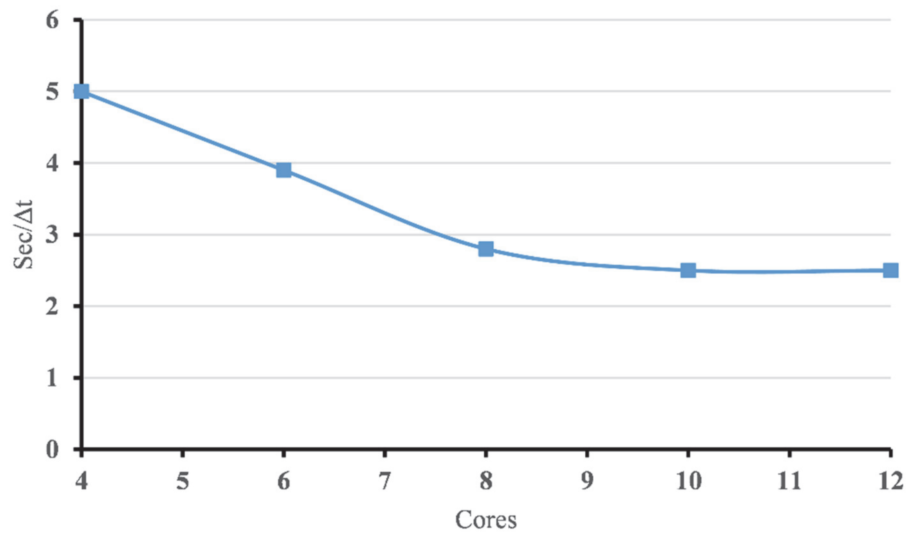


Figure 25: Plot of time/Δt versus number of cores

One may observe that for more than 8 cores the increase in performance is negligible. That is due to the additional overhead computations needed to divide and transport the matrices to the additional cores. Furthermore, use of additional cores can sometimes make the performance even worse because of this additional overhead. Consequently, some tests must be conducted to determine the optimal settings for the use of our algorithm.

4. Case studies

In this chapter we will present a number of free surface flow problems that arise in civil engineering applications tested with our algorithm. First we will validate the algorithm in the problem of water column collapse over dry bed. Next, we will test the behavior of the algorithm using different settings (i.e. with or without density filter, different time steps etc). Then we will use the results in order to propose an optimal algorithm which will be applied in a number of cases.

The free surface flow case studies to be computed using SPH in this thesis are the following:

i. Water column collapse over dry bed
ii. Hydrostatic Tank
iii. Flow under a sluice gate

4.1. Water column collapse over dry bed

Our algorithm will be used to study the collapse of a water column over dry bed under gravity. The results will be compared to the experimental data provided by Koshizuka S. (1996) and Martin J.C. (1952). This case study has been proposed by many researchers (Liu G.R. (2003), Violeau (2012), Gomez-Gesteira (2010) for the validation of SPH algorithms. The setup consists of a 4 meter wide tank and a column of water 2 meters high and 1 meter wide.

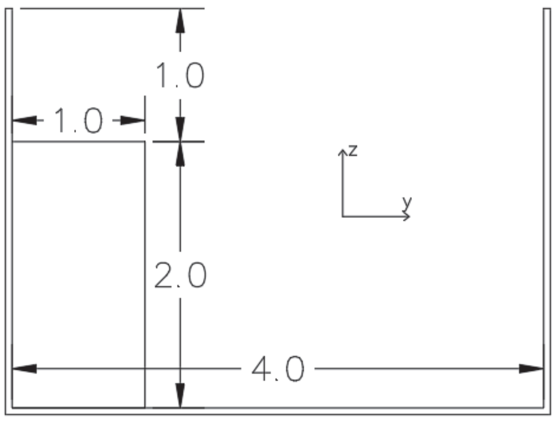


Figure 26: Setup of column collapse case study

The experimental data provide the toe evolution over the dry bed through time. These data are given in a non-dimensional form. Specifically the non-dimensional position of the toe is given by $y^* = y/L$ and the non-dimensional time is given by $t^* = t\sqrt{2g/L}$, where L is the column's initial width. It is schematically shown in Fig. 27.

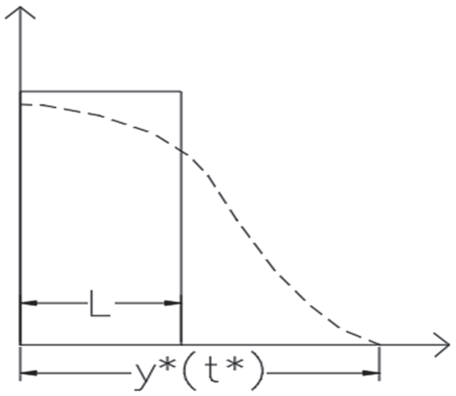


Figure 27: Schematic representation of the toes evolution in column collapse

The experimental data are presented below:

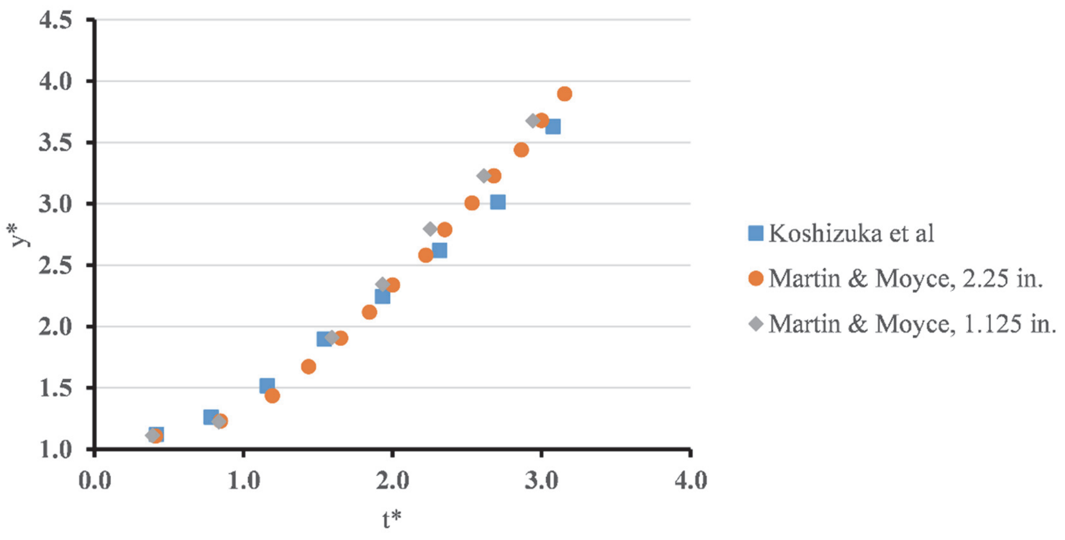


Figure 28: Experimental data of toe evolution over dry bed (non-dimensional)

First we will make a preliminary run of the code in order to test its performance, which will follow the ideas of Violeau D. (2007). In this case we will use dummy boundary particles set with hydrostatic pressure and repulsive Lenard-Jones boundary particles following equation (49). Through numerical experiments we found that when using the equation (49) the cut-off distance must be set so that the fluid particles feel a small repulsive force at the beginning of the simulation. For the time integration we will employ a Verlet scheme as presented in 2.10. Also the kernel function used is the quartic function proposed by Liu MB (2003) in equation (23). The fluid particles are placed in regular Cartesian grid. The test run parameter setup is given below:

Table 6: Parameter setup for preliminary run in column collapse

Smoothing length	2 cm
Discretization	1 cm
Number of fluid particles	20000
Boundary Discretization	1 mm
Number of boundary particles	14000
D coefficient of Lenard-Jones forces	1.2
Boundary cut-off distance	1.2cm
XSPH coefficient ε	0.5
a coefficient of artificial viscosity	0.3
Mass of fluid particles	0.01kg
Time step	0.000025 sec
Initial density	1000 kg/m^3
Artificial speed of sound	120 m/s

The total simulation time was 2.5 seconds which is equivalent to 10^5 time steps. The total run time was 117 hours. After the simulation the evolution of the toe over the bed was monitored using a routine called 'ToeCalc'. The results are subject of +- 1cm error, due to the discretization. The results are shown below:

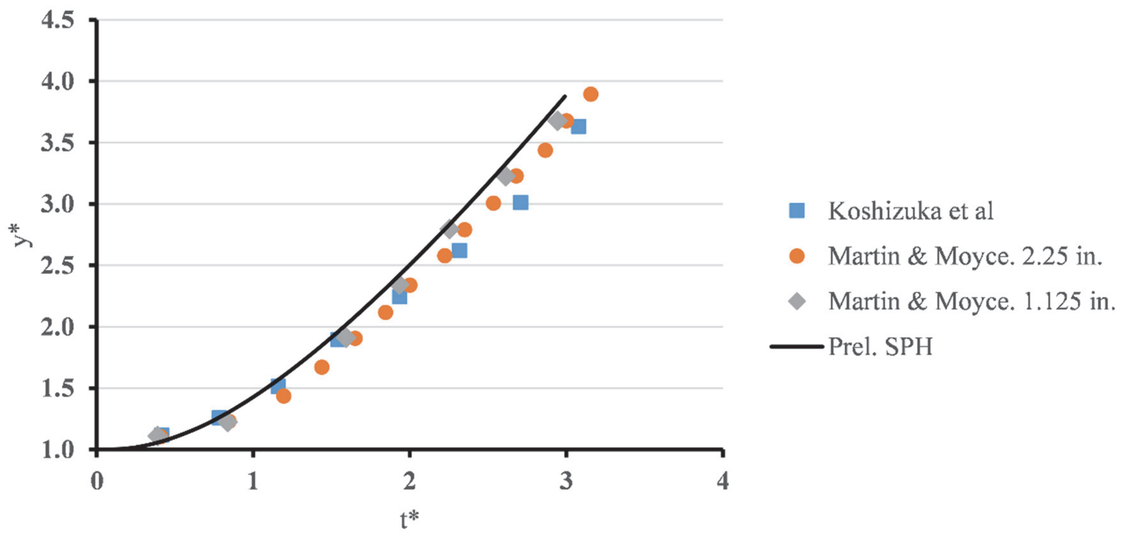


Figure 29: Toe evolution over dry bed for preliminary test run (non-dimensional).

The results are in very good agreement with the experimental measurements. Moreover the accuracy of the results can be quantified using the square root error from the measurements.

$$error = \sqrt{\sum (x_{meas} - x_{sim})^2 \,/ \sum {x_{meas}}^2} \tag{67}$$

Perfect agreement between the experimental data and the simulation would yield $error \to 0$. Results are given in the table below:

Table 7: Preliminary simulation errors

	Koshizuka et al	Martin & Moyce, 2.25 in.	Martin & Moyce, 1.125 in.
Preliminary simulation	0.214	0.260	0.103

Overall, we have obtained very good agreement with the experimental data. However the deviation from the experimental data of Martin & Moyce, 1.125 in. is almost half of that compared to the other experiments, because we have used a slip free boundary condition omitting essentially any friction from the wall container.

From the figure 28 one may see that our simulation starts with small velocity which subsequently builds up. This is a result of the initial condition of zero pressure, while naturally in the experiment a hydrostatic pressure is present. At this point we compare our results to those obtained by Violeau D. (2007) in figure 29. It appears that we have obtained better results. This may be explained from the fact that in the simulation we have use a higher order kernel than the cubic function used by Violeau. However this assumption has not been tested.

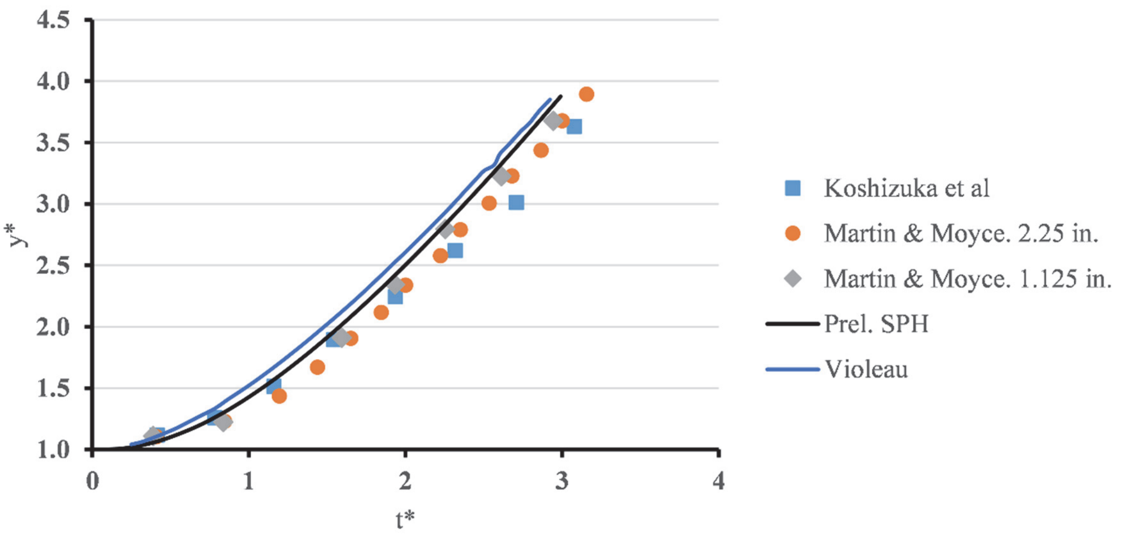


Figure 30: Toe evolution over dry bed for preliminary test run compared with Violeau (non-dimensional).

Next we present time snapshots of the flow.

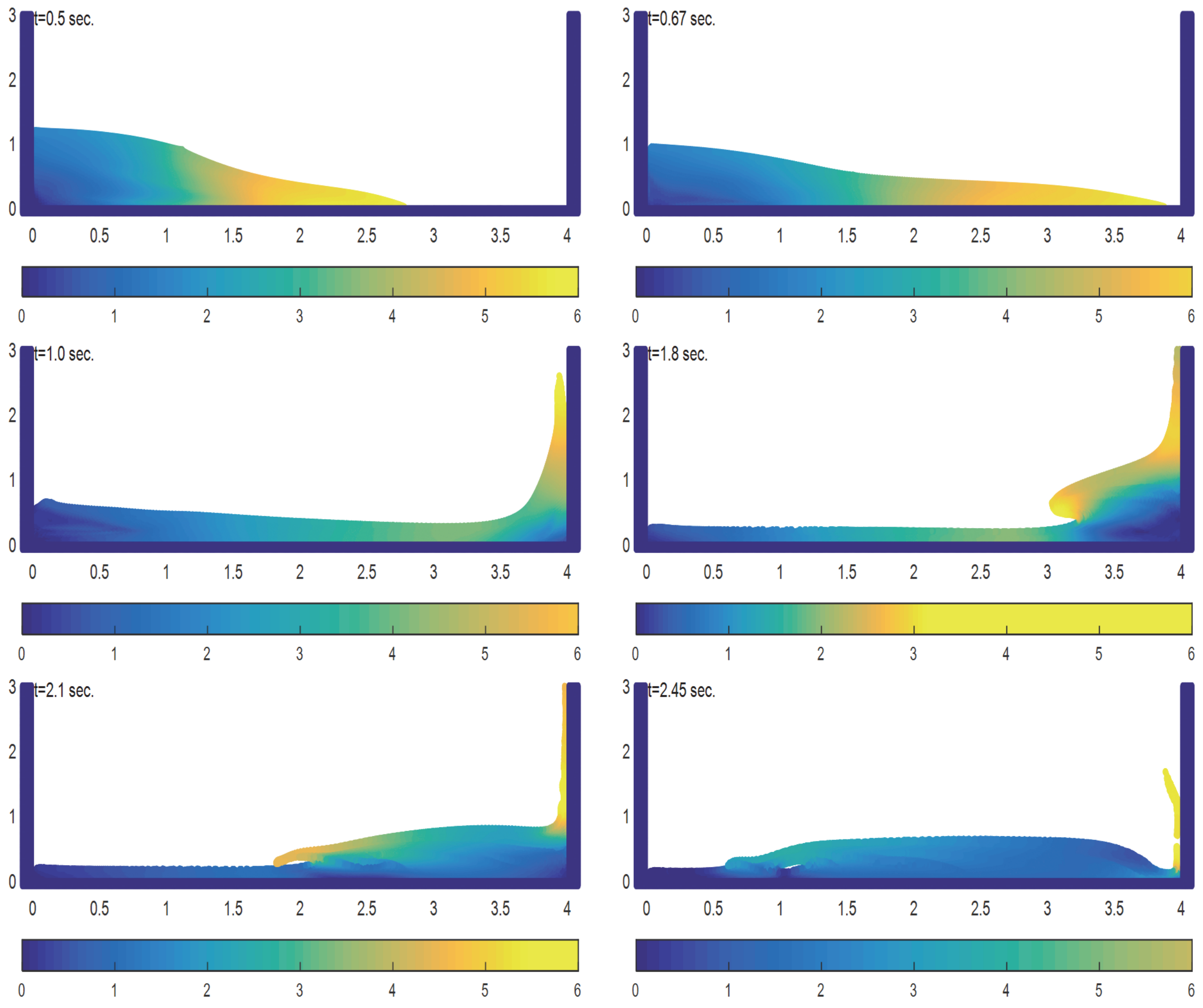


Figure 31: Preliminary Test Snapshots. Colorbar denotes the norm of the velocity

One may observe that the wave propagates to the other side of the container and starts climbing up until it overturns and plunges. The plunge however appears to be drowned. We will focus on this matter during later tests. We proceed presenting the density distribution.

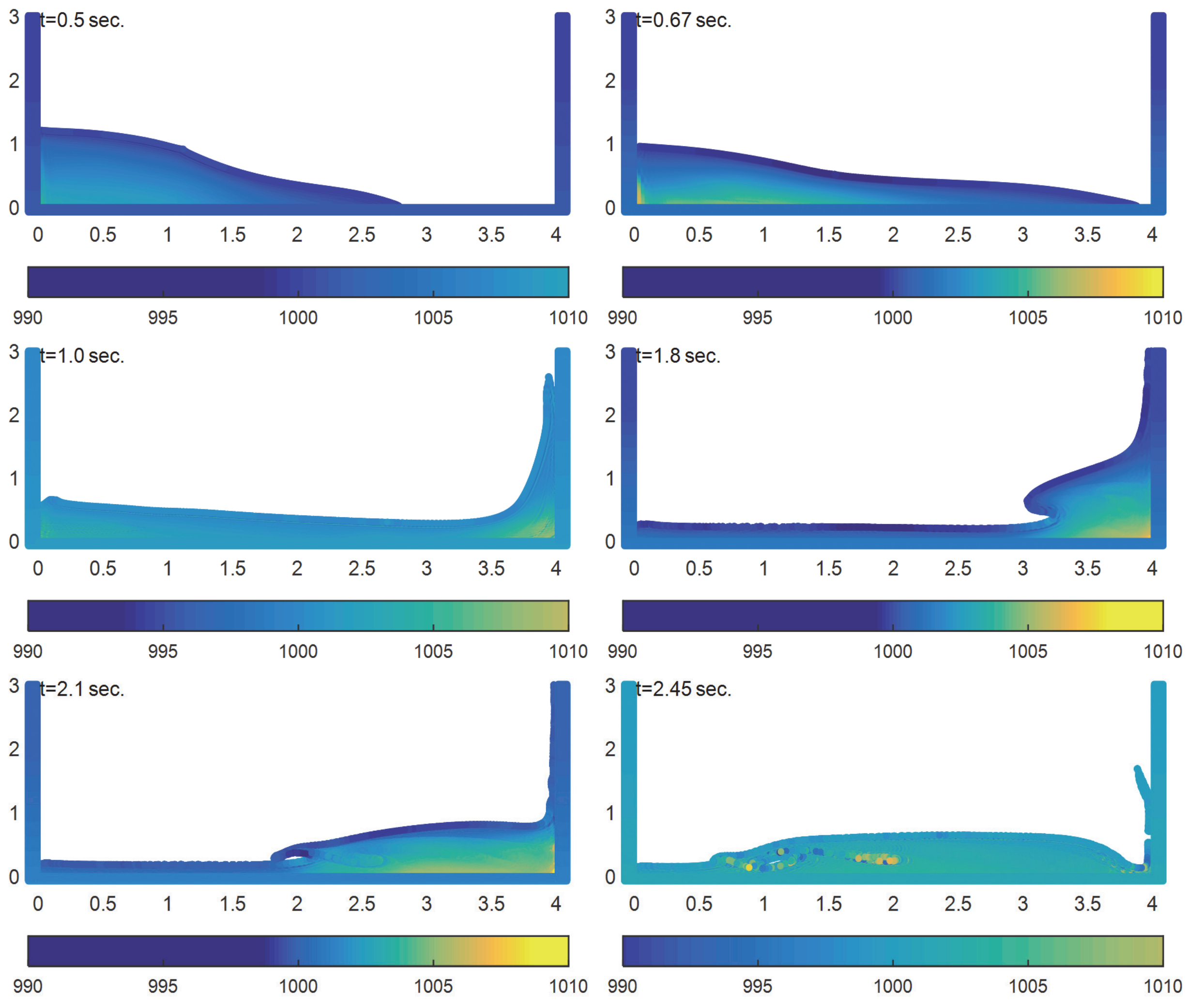


Figure 32: Preliminary Test Snapshots. Colorbar denotes the density distribution

We observe that in general a smooth density profile is obtained. However, two distinct anomalies are found. First, in all snapshots the layer of particles that is adjacent to the free surface has constantly lower density than the reference. This is caused by the incomplete integration of the kernel, as it was presented in the case of boundary particles (figure 13). Special treatment must be employed in order to tackle that problem, such as the insertion of air (two phase flow). However in this thesis we will not apply this approach. The reader may refer to Colagrossi Andrea (2003). Secondly we observe some irregularities in t=2.45 sec caused by the force exerted during the plunge.

At this point we focus in the time frames were the overturning wave plunges into the bottom.

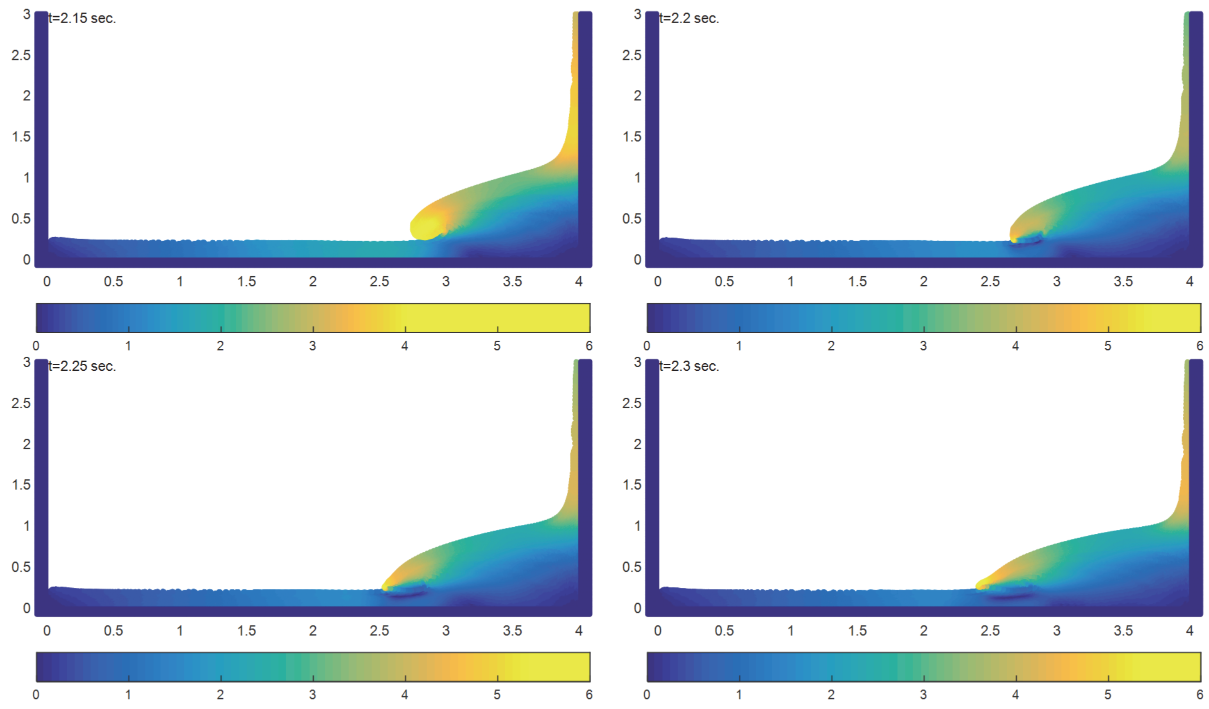


Figure 33: Snapshots of plunging wave in preliminary test. Color bar denotes the magnitude of velocity

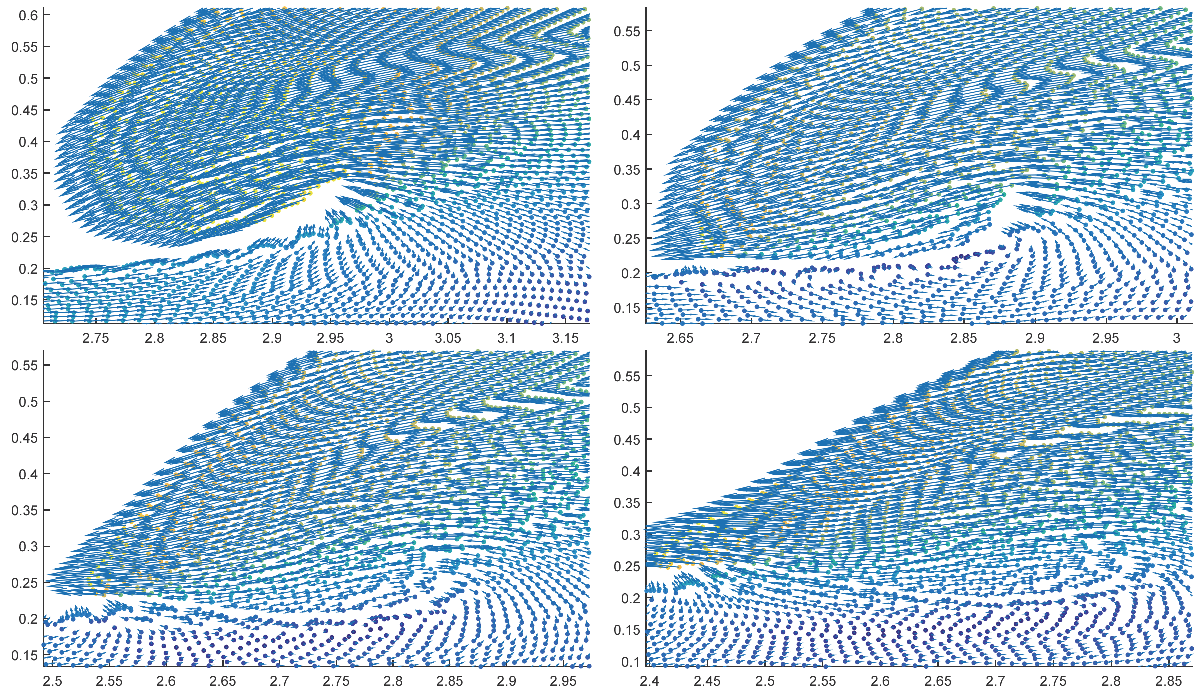


Figure 34: Snapshots of plunging wave in preliminary test with velocity arrows

It is observed that the wave does not really plunge, it is rather reflected from the plunging surface. This behavior along with the “drowned” plunging will be discussed later.

Having validated our SPH implementation we will now start a thorough search on the impact of different parameters used on the SPH method. More specifically we will conduct the following tests:

a. Discretization of 1.5 cm, with same parameters as the preliminary run
b. Discretization of 2 cm and a smoothing length of 2 cm.

c. Discretization of 2 cm and a smoothing length of 3 cm.
d. Discretization of 1.5 cm and use of dynamic boundary particles
e. Discretization of 2 cm and XSPH method applying on boundary particles.
f. Preliminary run setup but only with Lenard-Jones boundary particles method as in equation (48)
g. Preliminary run setup but only with Lenard-Jones boundary particles method as in equation (49)
h. Discretization of 1.5 cm without the Shepard density filter.
i. Discretization of 1.5 cm with time step of 0.00005 second.
j. Discretization of 1.5 cm with time step of 0.00005 second and a predictor-corrector time integration scheme.
k. Discretization of 1.5 cm with time step of 0.0001 second and a predictor-corrector time integration scheme.
l. Discretization of 1.5 cm with time step of 0.0001 second and a Verlet time integration scheme.
m. Discretization of 1.5 cm with time step of 0.0001 second and a Verlet time integration scheme with smoothing length of 4 cm.

4.2. Test a.

In this case we will increase the discretization of the simulation from 1 to 1.5 cm resulting in 10000 fluid particles. Also we will increase the smoothing length in order to attain better results. The complete setup is presented below.

Table 8: Parameter setup for test a.

Smoothing length	4 cm
Discretization	1.5 cm
Number of fluid particles	10000
Boundary Discretization	1 mm
Number of boundary particles	14000
D coefficient of Lenard-Jones forces	1.2
Boundary cut-off distance	1.8cm
XSPH coefficient ε	0.5
a coefficient of artificial viscosity	0.3
Mass of fluid particles	0.02kg
Time step	0.000025 sec
Initial density	1000 kg/m^3
Artificial speed of sound	120 m/s

The run time was 50 hours, less than one half of the preliminary run. Again we will use the toe evolution to compare the performance of the new setup with the previous one.

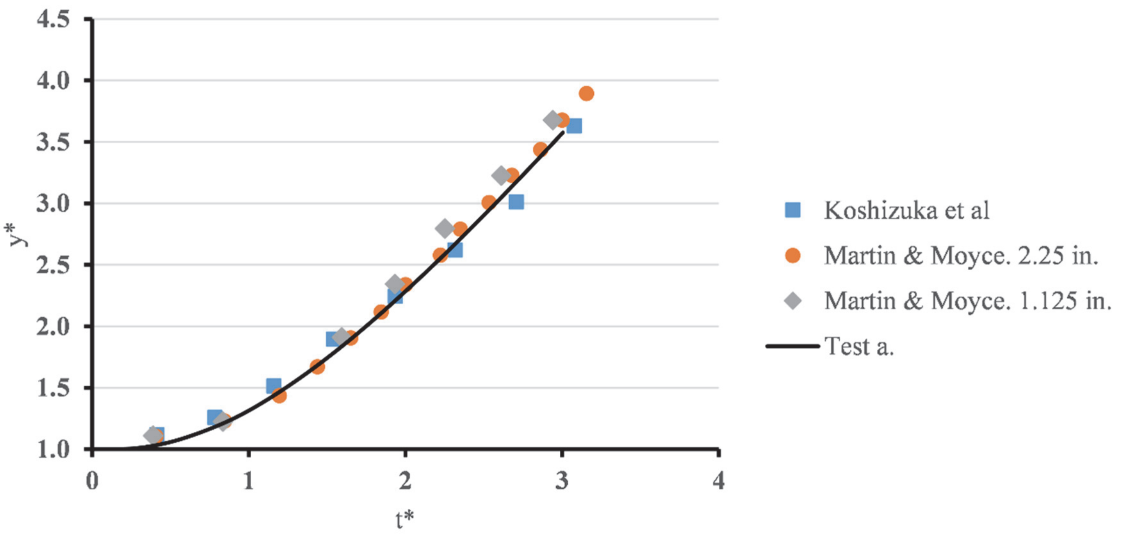


Figure 35: Toe evolution over dry bed for test a (non-dimensional).

Table 9: Test a errors

	Koshizuka et al	**Martin & Moyce, 2.25 in.**	**Martin & Moyce, 1.125 in.**
Test a.	0.138	0.086	0.147

This simulation performs much better than the preliminary simulation and the results are in agreement with the experiments.

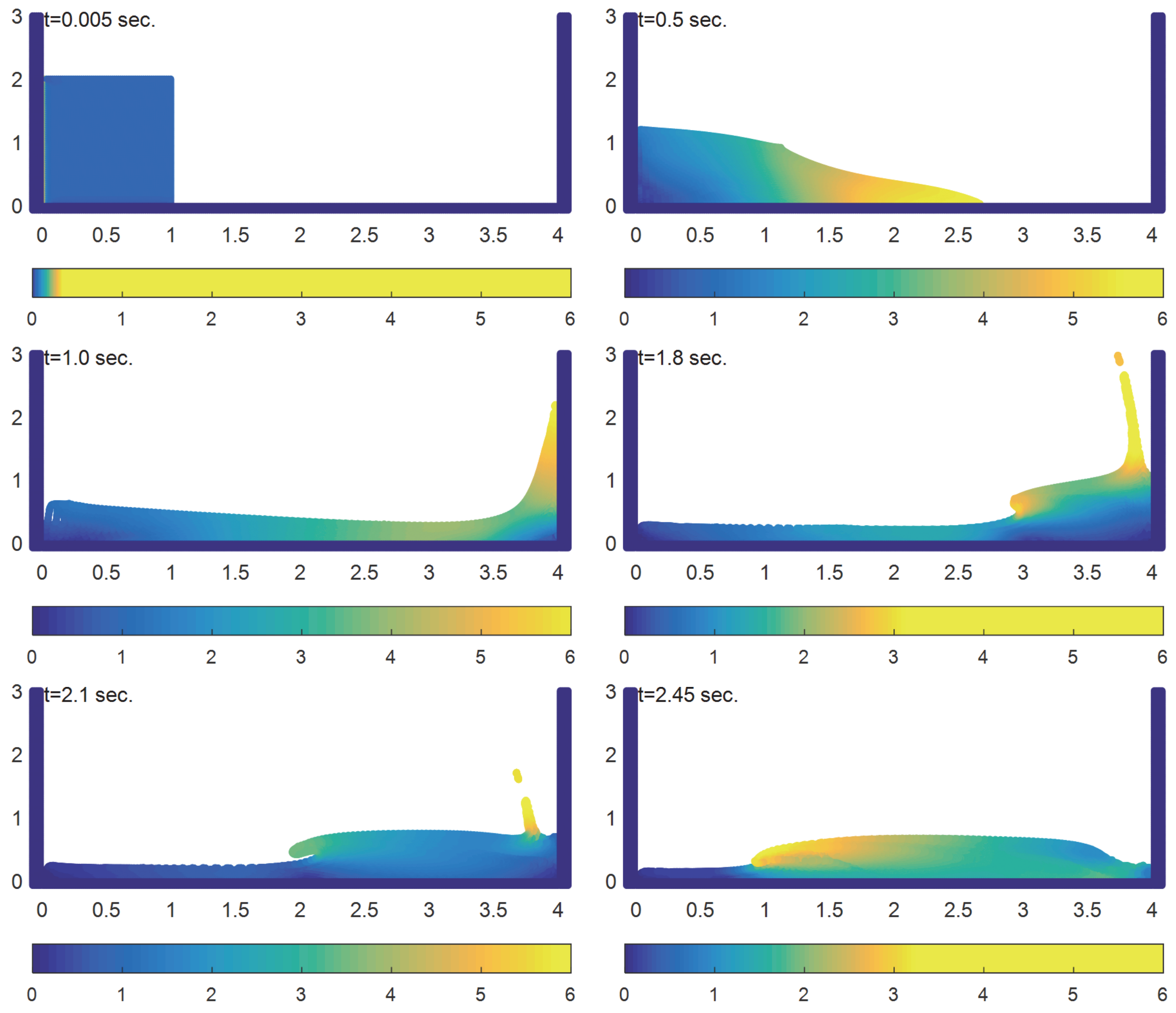
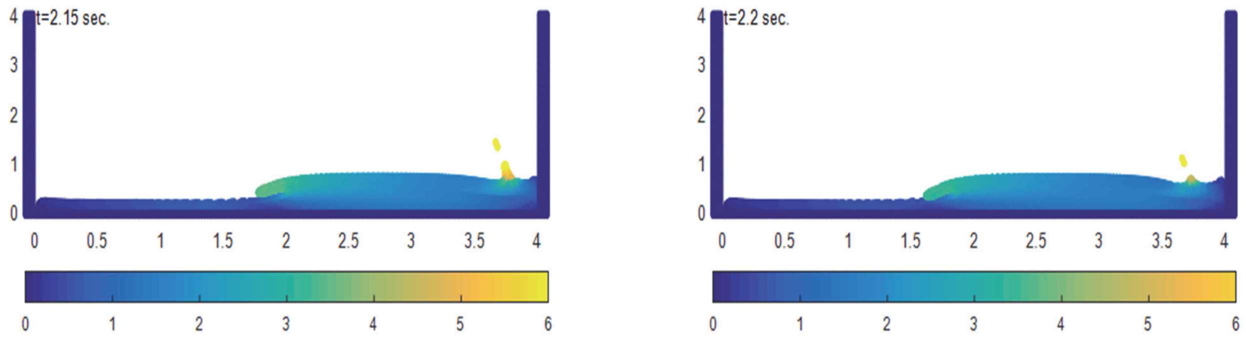


Figure 36: Test a. Snapshots. Colorbar denotes the norm of the velocity

This simulation presents almost identical behavior if compared to the preliminary case until the overturn and plunge of the climbing wave on the right wall. There the toe that was formerly distinguished while collapsing is now "drowned". Let us take a closer look in that time window.



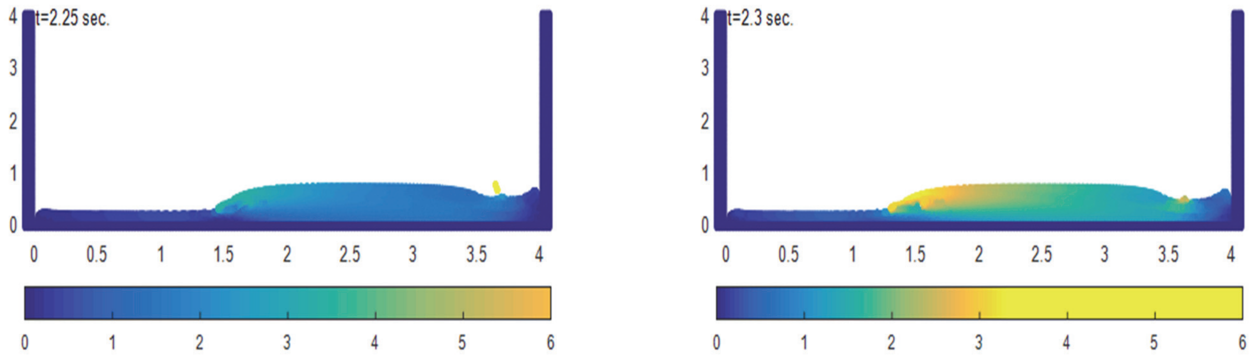


Figure 37: Snapshots of plunging wave in test a. Colorbar denotes the norm of the velocity

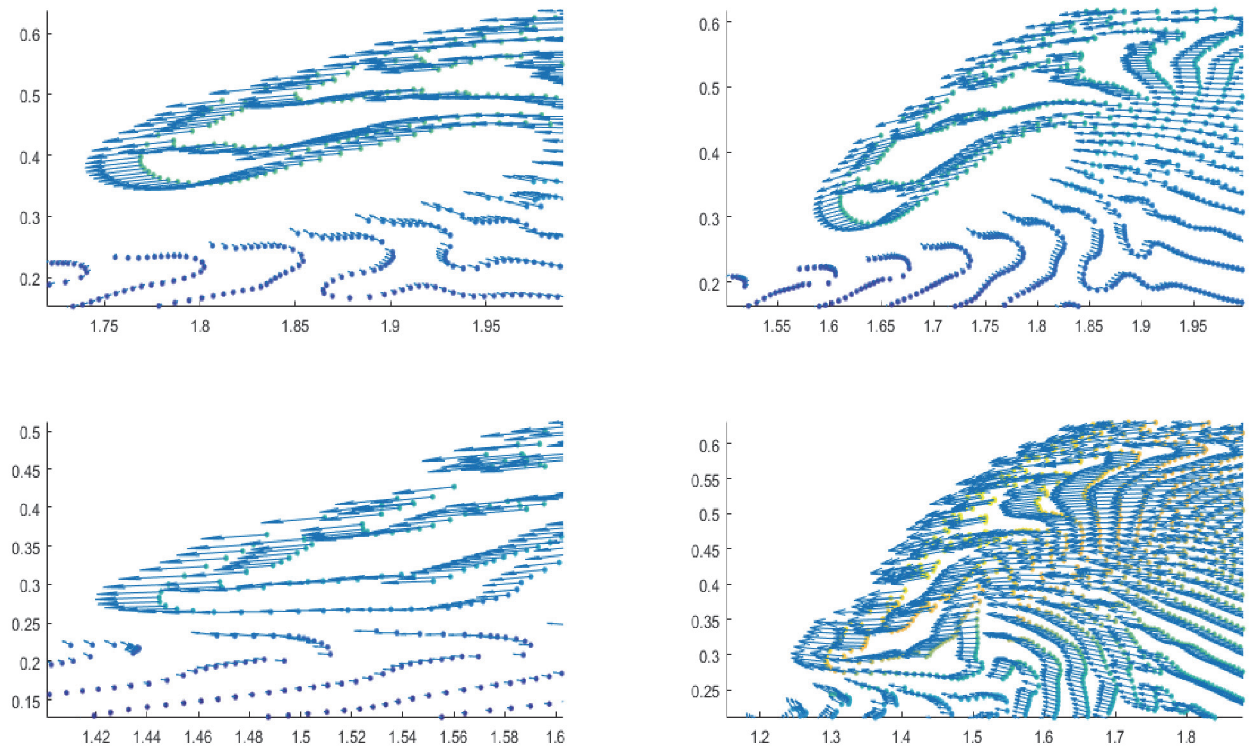


Figure 38: Snapshots of the wave toe with velocity arrows

From the above figures we observe that the toe of the wave loses gradually the vertical component of velocity while the fluid below the toe gradually gains horizontal velocity. Consequently the toe never collapses, but rather deflects. In this simulation while lowering the discretization we have increased the smoothing length from 2 cm to 4cm. In the same time we have employed the XSPH scheme which corrects the particle velocities relative to their neighboring particles. It appears that since we have increased the smoothing length the XSPH unnaturally “corrects” the velocities of the particles below the toe offering them velocity from the toe particles and vice versa. In order to test this hypothesis we will initialize the simulation again this time without employing the XSPH correction.

The results are presented below:

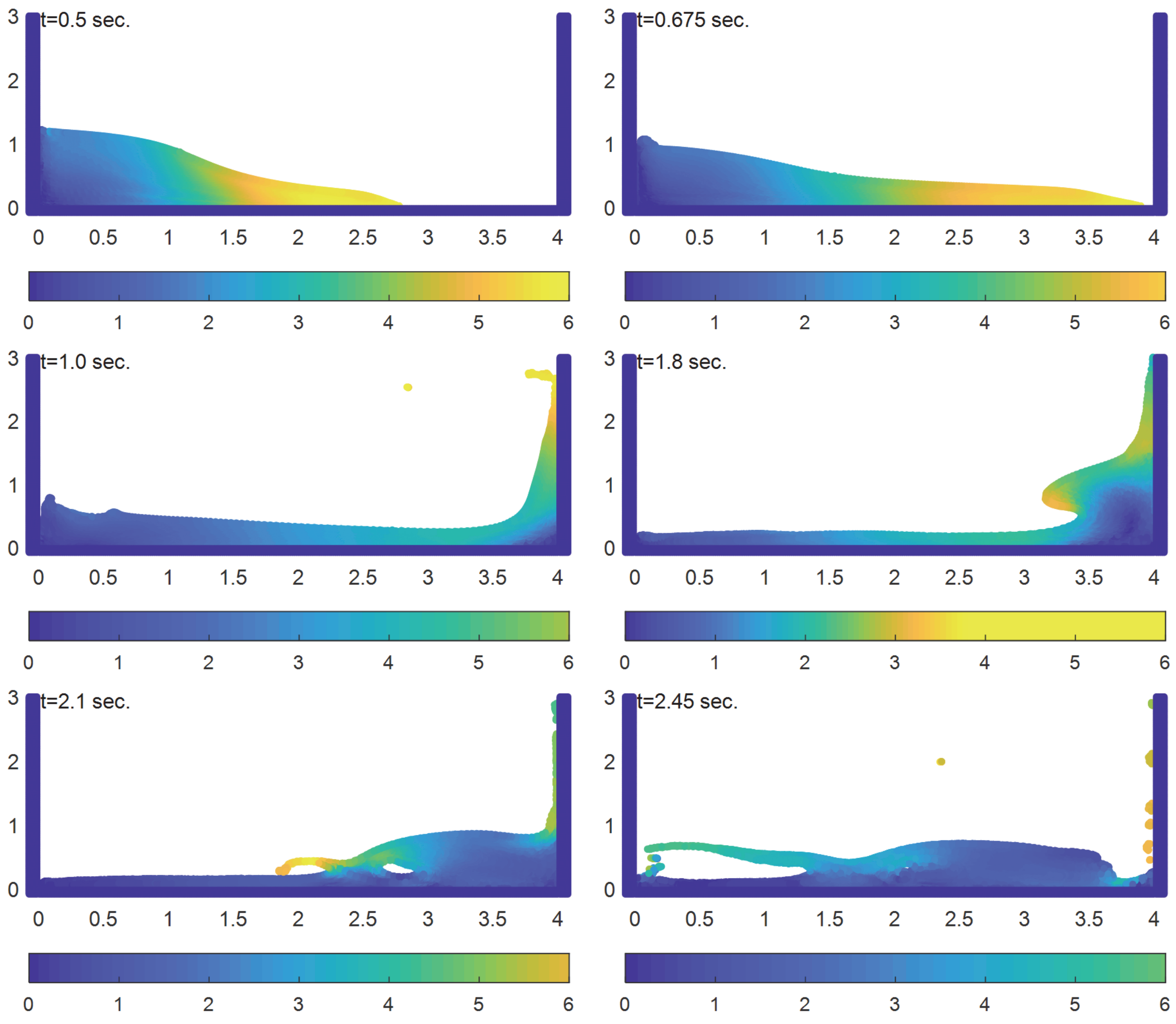


Figure 39: Snapshots of collapsing wave in test a without the XSPH. Color bar denotes the norm of the velocity

We can immediately observe the difference in the overturn of the wave. In order to test the toe propagation we will compare the results with the ones collected with the use of the XPSH correction method and experiments.

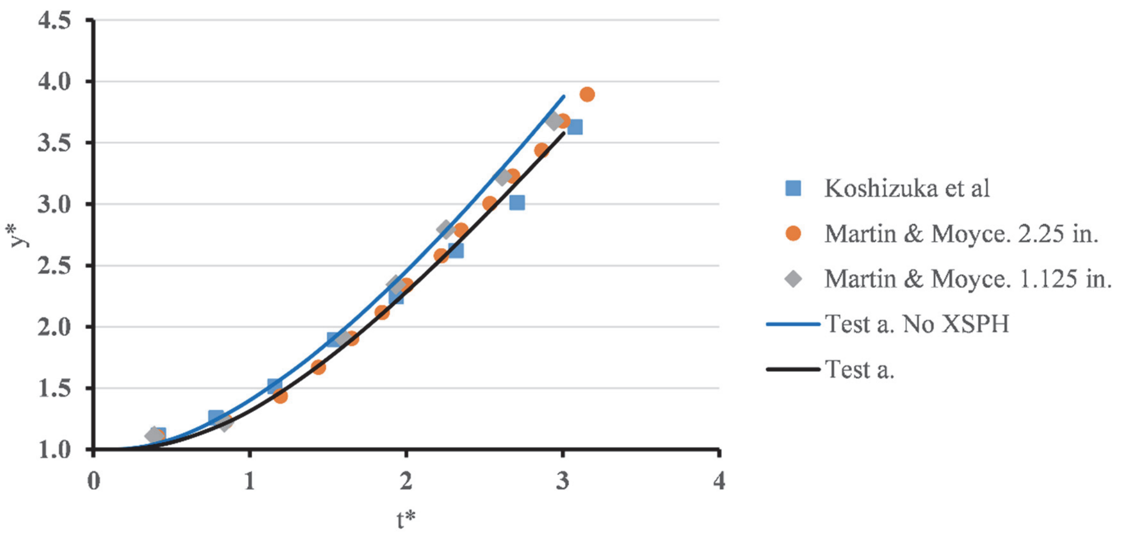


Figure 40: Toe evolution over dry bed for test a with and without XSPH (non-dimensional).

It appears that the performance is affected without the XSPH correction. More specifically the curve of the toe evolution appears higher, meaning that greater velocities evolved during the simulation. However the error is acceptable compared to the other simulations.

Table 10: Test a with and without XSPH errors

	Koshizuka et al	Martin & Moyce, 2.25 in.	Martin & Moyce, 1.125 in.
Test a.	0.138	0.086	0.147
Test a. No XSPH	0.247	0.295	0.123
Preliminary simulation	0.214	0.260	0.103

Next, let us focus on the plunging of the overturned wave. We present the snapshots of the same time steps as the test a.

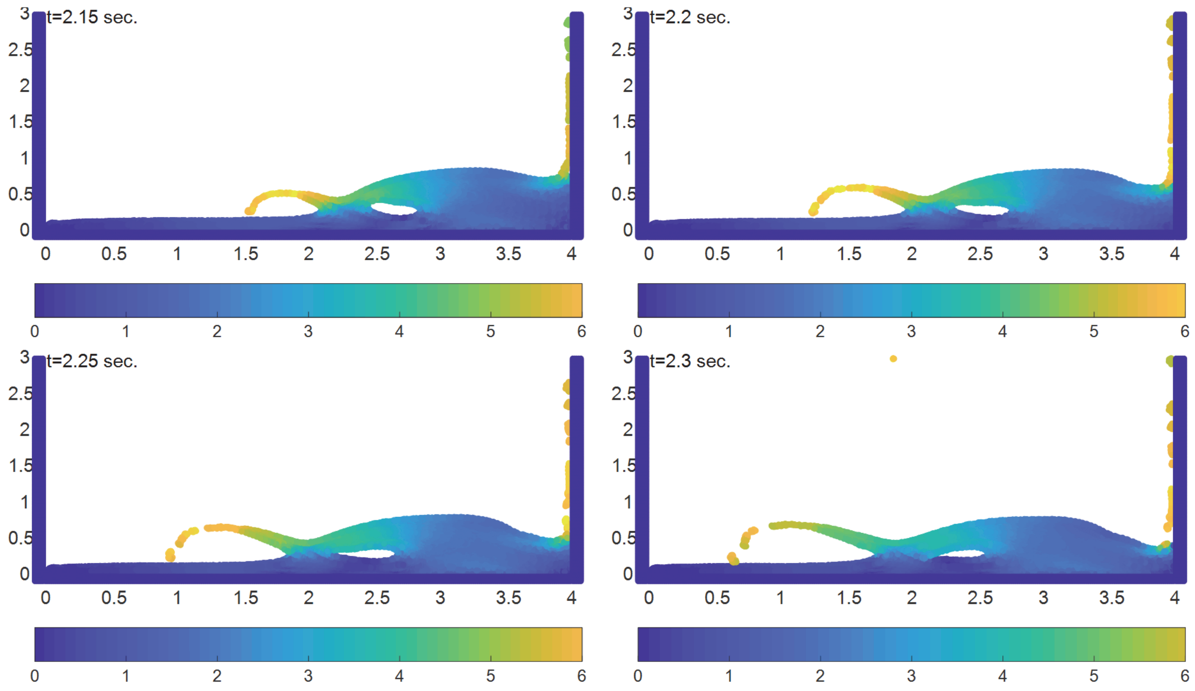


Figure 41: Snapshots of collapsing wave in test a without XSPH. Colorbar denotes the norm of the velocity

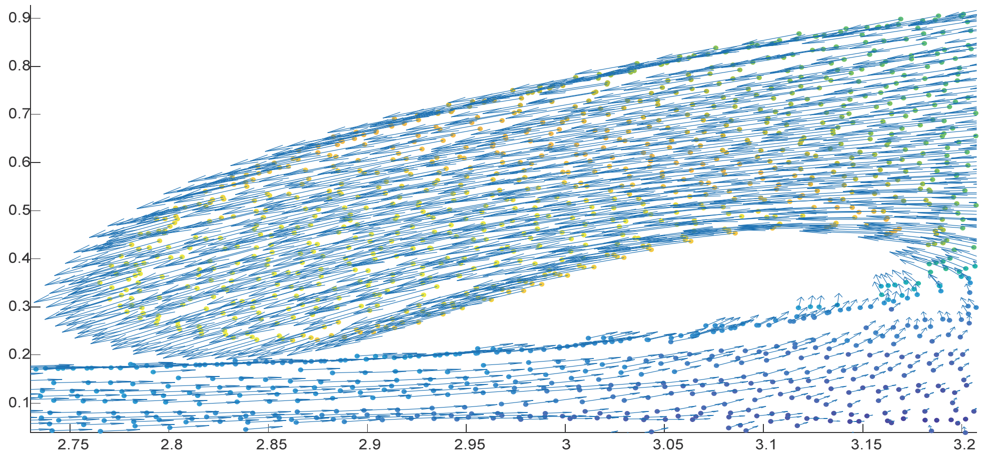


Figure 42: Snapshot of the plunging wave at t=1.95 sec. with velocity arrows

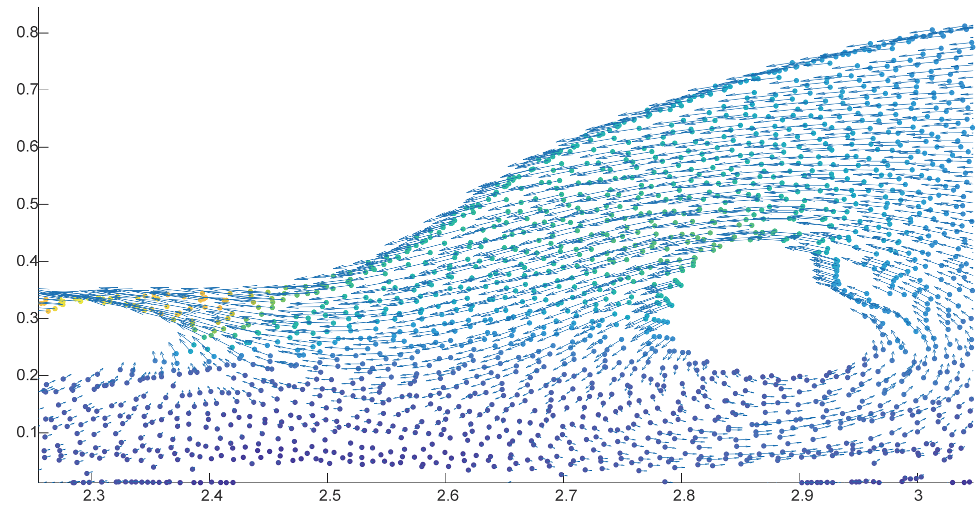


Figure 43: Snapshot of the plunging wave at t=2.05 sec. with velocity arrows

The difference between the plunging with and without XSPH correction is obvious. As suspected, without XSPH correction the velocities at the toe and below it are unaffected by each over and the plunging occurs naturally. Consequently our initial hypothesis was correct. Moreover we will compare the plunging at t=2.05 sec with the one obtained by our preliminary simulation.

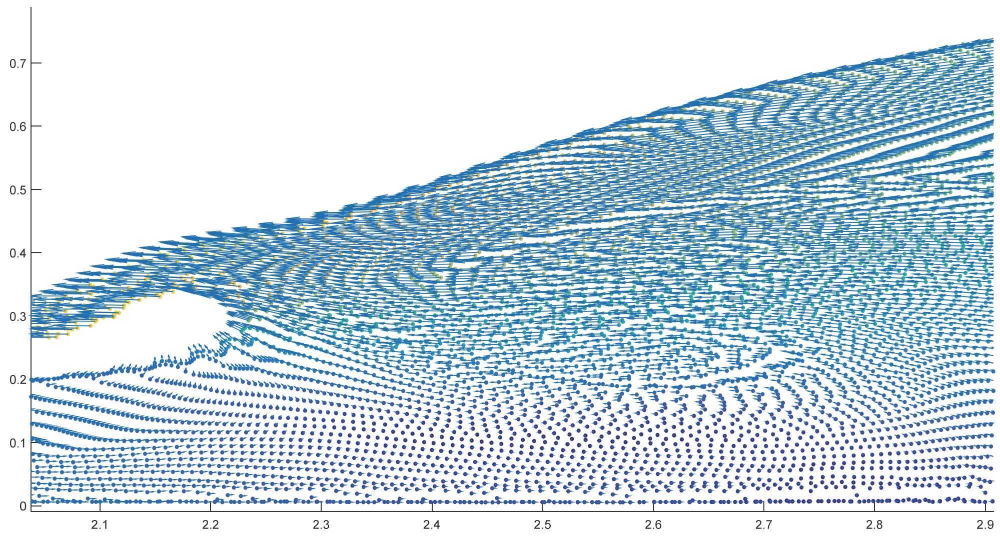


Figure 44: Snapshot of the plunging wave at t=2.05 sec. from preliminary simulation with velocity arrows

The difference in the behavior of the plunging wave is significant. In figure 43 part of the plunging wave reflects on the surface of water while the other part disperses. In contrast we can observe that the wave reflects almost entirely while part of it behaves as a drowned vortex. This result is significant and must raise questions on the appropriateness of this correction method.

4.3. Test b.

In this test case we will conduct the same experiment with discretization of 2 cm, which is equivalent to 5000 fluid particles. Also, the smoothing length will be lowered to 2 cm in order to assess the impact of the overall reduced resolution. We will use dummy boundary particles set with hydrostatic pressure and repulsive Lenard-Jones boundary particles following equation (49). For the time integration we will employ a Verlet scheme as presented in 2.10. Also the kernel function used is the quartic function proposed by Liu MB (2003) in equation (23). The fluid particles are placed in regular Cartesian grid.

Table 11: Parameter setup for test b.

Smoothing length	2 cm
Discretization	2 cm
Number of fluid particles	5000
Boundary Discretization	1 mm
Number of boundary particles	14000
D coefficient of Lenard-Jones forces	1.2
Boundary cut-off distance	2.4 cm
XSPH coefficient ε	0.5
a coefficient of artificial viscosity	0.3
Mass of fluid particles	0.04kg
Time step	0.000025 sec
Initial density	1000 kg/m^3
Artificial speed of sound	120 m/s

The comparison to the experimental data is given below:

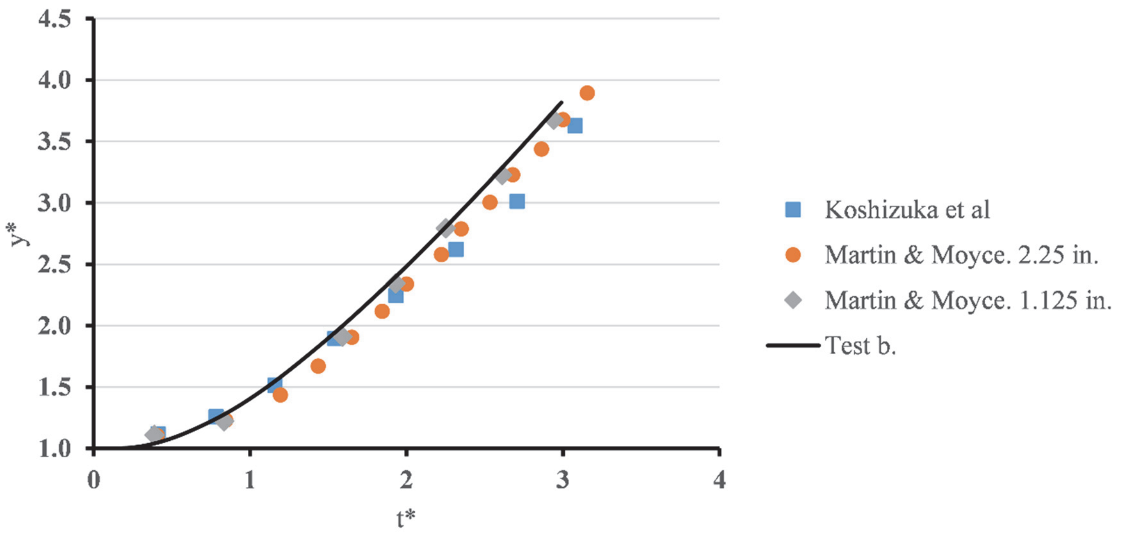


Figure 45: Toe evolution over dry bed for test b (non-dimensional).

It can be seen that we have obtained a very good agreement with the experimental data, similar to case a.

Table 12: Test b. errors

	Koshizuka et al	**Martin & Moyce, 2.25 in.**	**Martin & Moyce, 1.125 in.**
Test b.	0.194	0.226	0.091

Furthermore the behavior of the free surface is examined below:

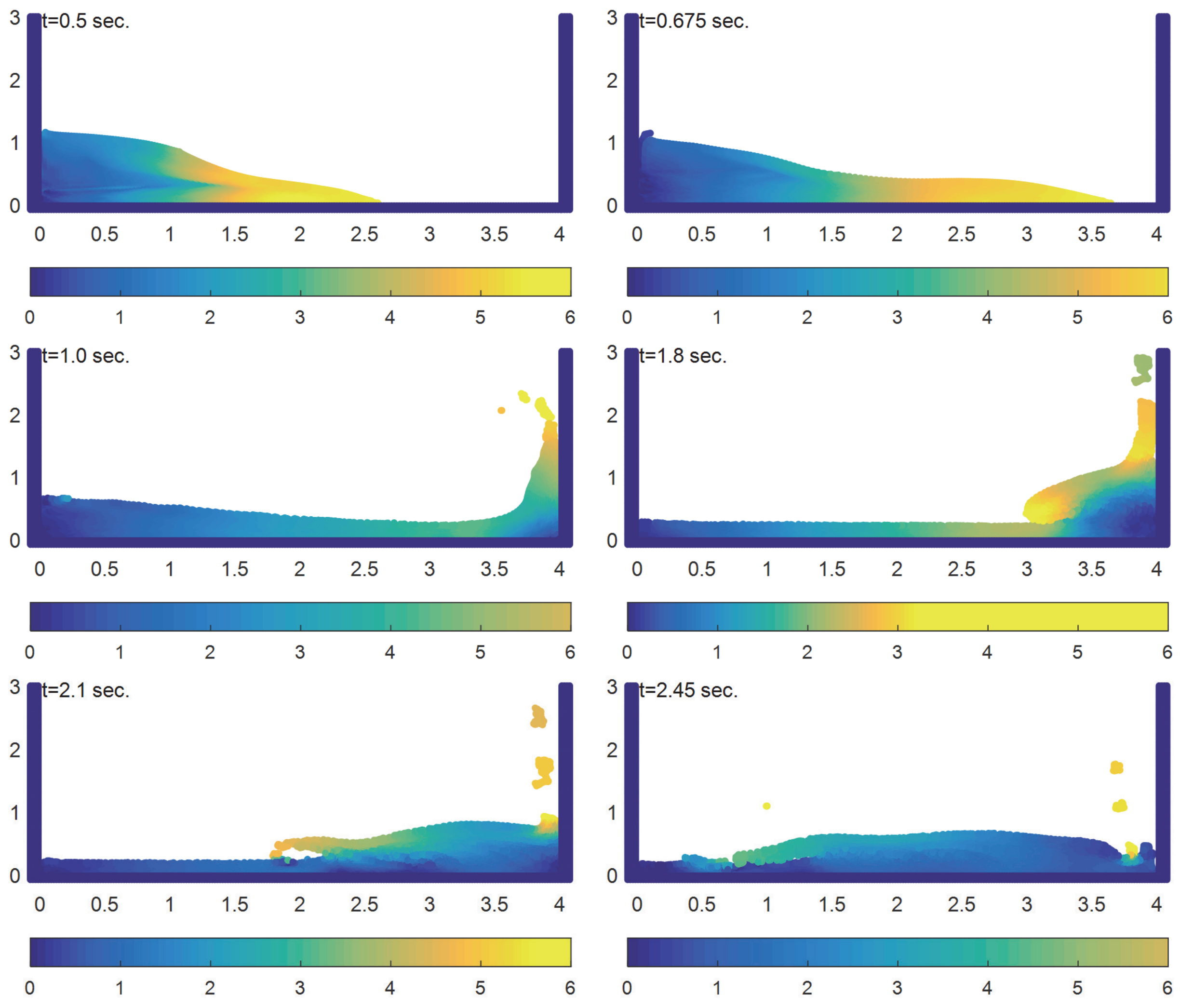
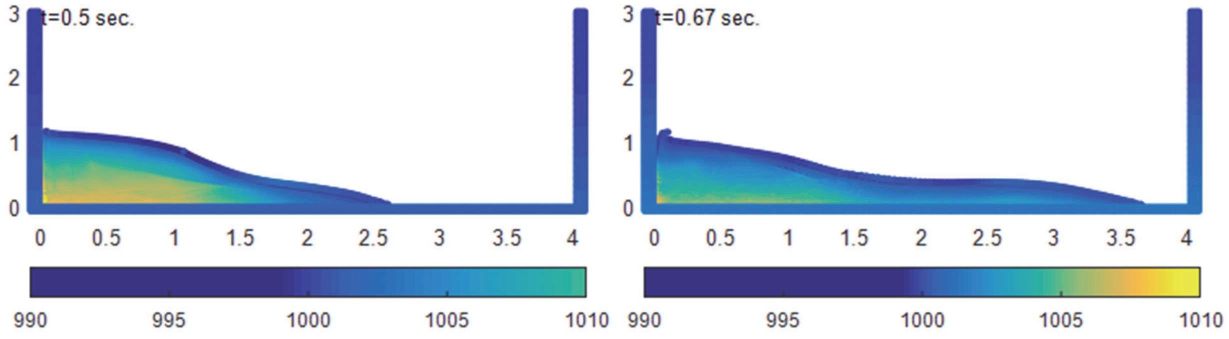


Figure 46: Test b. Snapshots. Colorbar denotes the norm of the velocity

One may observe that the free surface appears to be unstable. Moreover, during the collision of the wave with the wall of the container, there is much more splashing than in test a. Also the distribution of velocities appears less smooth. This evidence compels us to examine the distribution of density over the fluid particles. From the snapshot in figure 47 below it appears that the distribution of density and therefore pressure is not smooth. This may be the result of the very low resolution. More specifically, during this simulation in the initial placement for each particle there are a total of twelve particles inside the kernel. Consequently, attributing this behavior to low resolution is a logical assumption. However this will be tested in test case c.



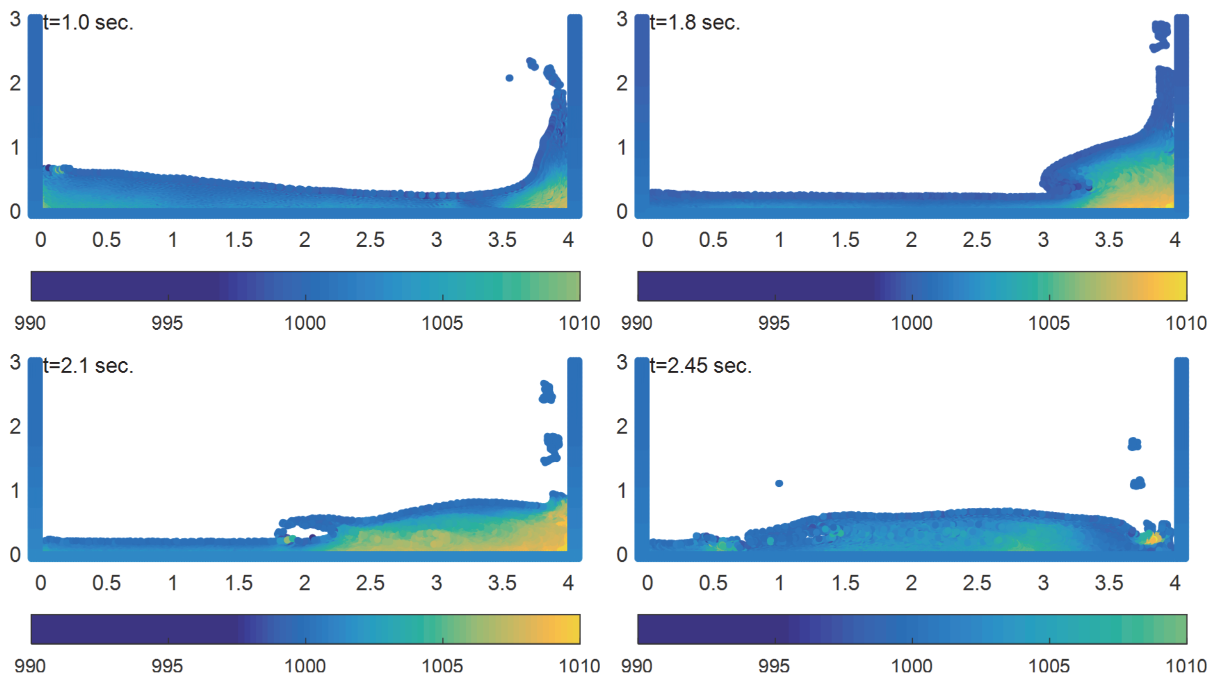


Figure 47: Test b. Snapshots. Colorbar denotes the density distribution

4.4. Test c.

In this test case we will conduct the same experiment with discretization of 2 cm, which is equivalent to 5000 fluid particles. However, in this test a smoothing length of 3 cm will be used. We will use dummy boundary particles set with hydrostatic pressure and repulsive Lenard-Jones boundary particles following equation (49). For the time integration we will employ a Verlet scheme as presented in 2.10. Also the kernel function used is the quartic function proposed by Liu MB (2003) in equation (23). The fluid particles are placed in regular Cartesian grid.

Table 13: Parameter setup for test c.

Smoothing length	3 cm
Discretization	2 cm
Number of fluid particles	5000
Boundary Discretization	1 mm
Number of boundary particles	14000
D coefficient of Lenard-Jones forces	1.2
Boundary cut-off distance	2.4 cm
XSPH coefficient ε	0.5
a coefficient of artificial viscosity	0.3
Mass of fluid particles	0.04kg
Time step	0.000025 sec
Initial density	1000 kg/m^3
Artificial speed of sound	120 m/s

The comparison to the experimental data is given below:

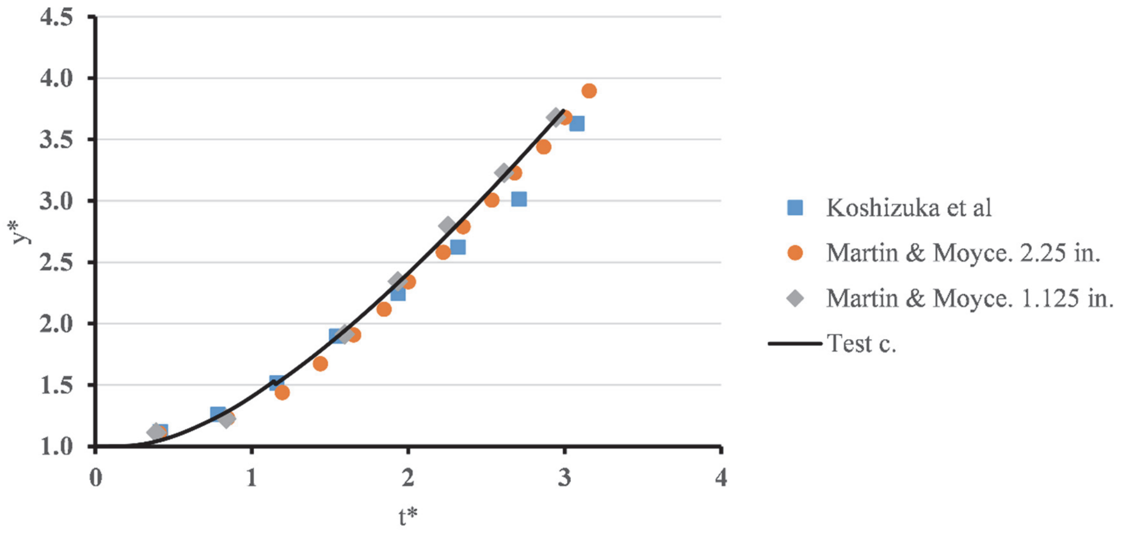


Figure 48: Toe evolution over dry bed for test c (non-dimensional).

We observe that the results are better from the ones obtained in test (b) and they are in agreement with the experimental results.

Table 14: Test c. errors

	Koshizuka et al	Martin & Moyce, 2.25 in.	Martin & Moyce, 1.125 in.
Test b.	0.194	0.226	0.091
Test c.	0.144	0.147	0.084

Moreover we will examine the behavior of the free surface.

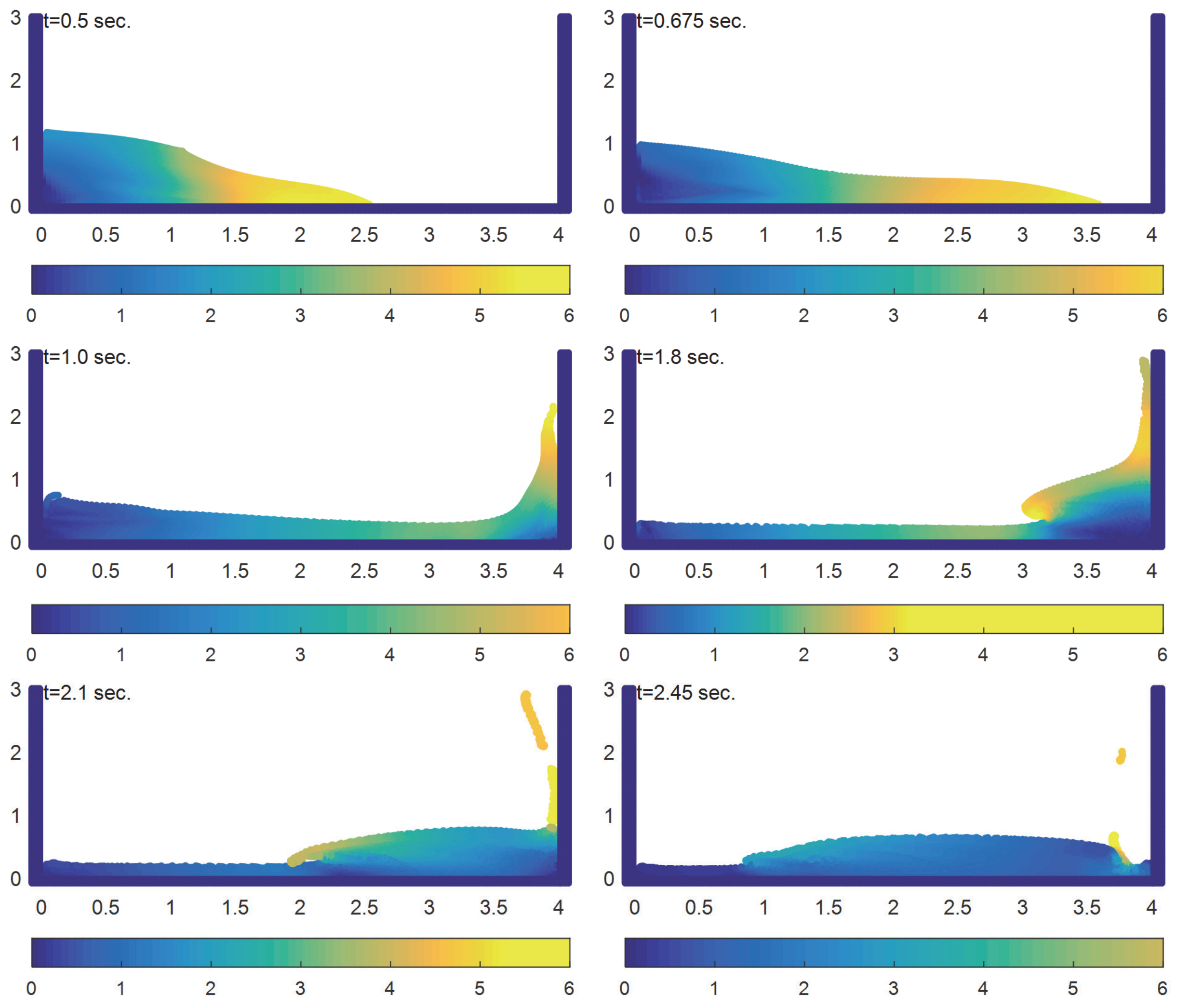
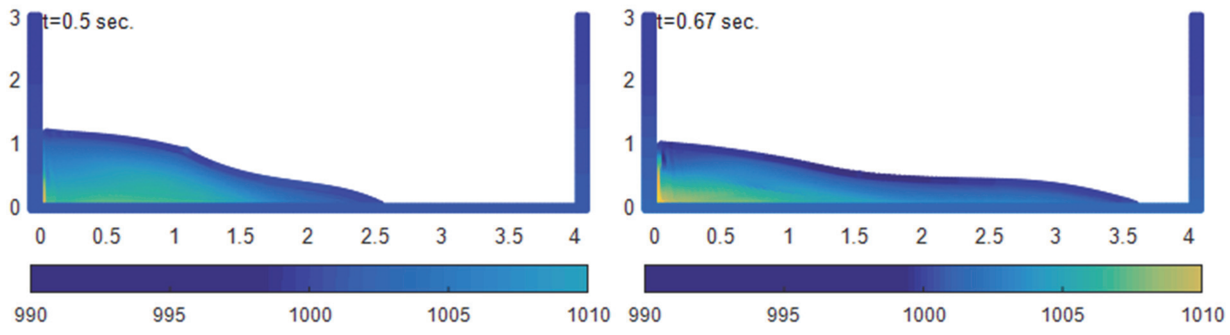


Figure 49: Test c. Snapshots. Colorbar denotes the norm of the velocity

As expected the free surface as well as the velocity distribution are much smoother and the behavior is similar to the one obtained by test case (a). Moreover, from the distribution of density over the fluid particles presented below a smoother profile arises, compared to test case (b). Consequently it appears that our assumption for the cause of the irregularities found in test case (b) is correct.



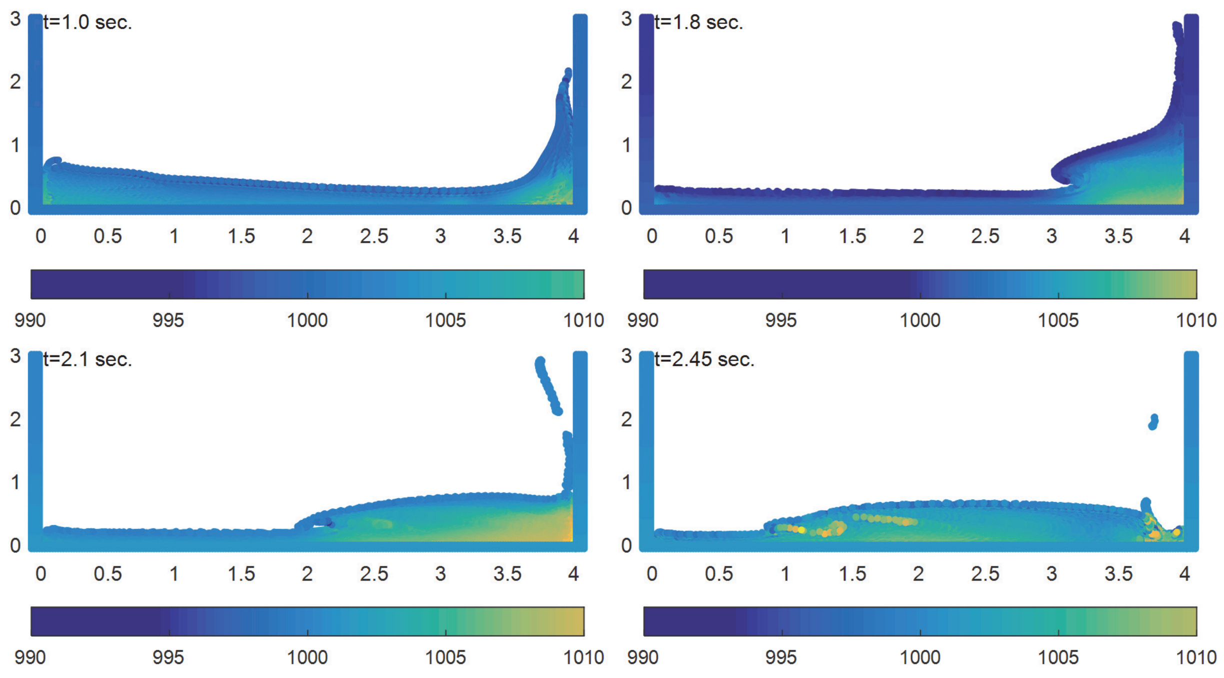


Figure 50: Test c. Snapshots. Colorbar denotes the density distribution

4.5. Test d.

During this case we examined the behavior of the dynamic particles boundary method. The boundary particles were placed in staggered grid with discretization of 1 cm. For the time integration we employed a Verlet scheme as presented in 2.10. Also the kernel function used is the quartic function proposed by Liu MB (2003) in equation (23). The fluid particles are placed in regular Cartesian grid.

Smoothing length	3 cm
Discretization	1.5 cm
Number of fluid particles	10000
Boundary Discretization	1 cm
Number of boundary particles	9731
D coefficient of Lenard-Jones forces	-
Boundary cut-off distance	-
XSPH coefficient ε	0.5
a coefficient of artificial viscosity	0.3
Mass of fluid particles	0.02kg
Time step	0.000025 sec
Initial density	1000 kg/m^3
Artificial speed of sound	120 m/s

The comparison to the experimental data is given below.

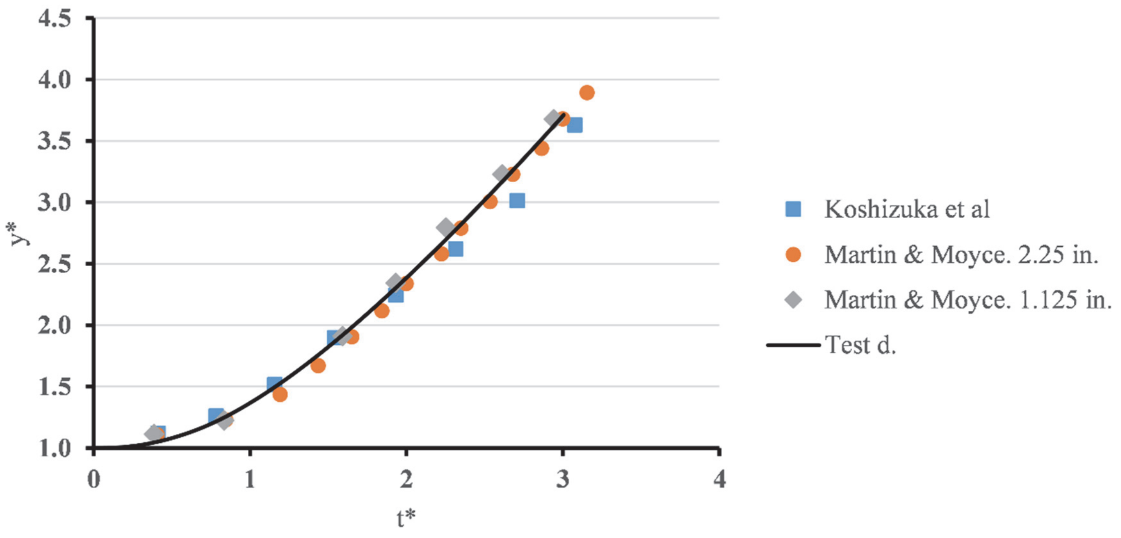


Figure 51: Toe evolution over dry bed for test d (non-dimensional).

We observe very good agreement with the experimental data and computational results from other simulations.

Table 15: Test d. errors

	Koshizuka et al	**Martin & Moyce, 2.25 in.**	**Martin & Moyce, 1.125 in.**
Test d.	0.138	0.124	0.077

Next we present the evolution of the free surface.

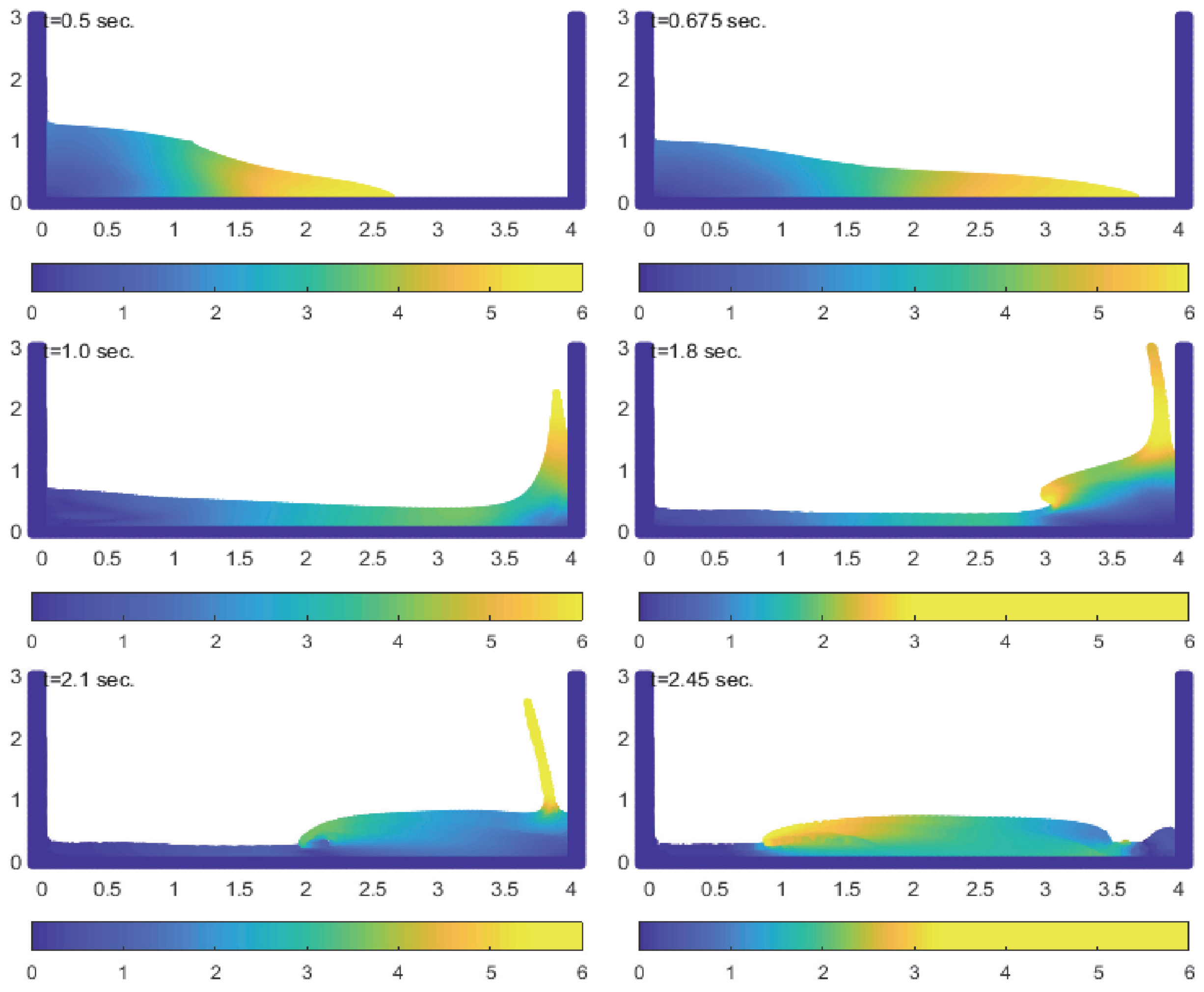
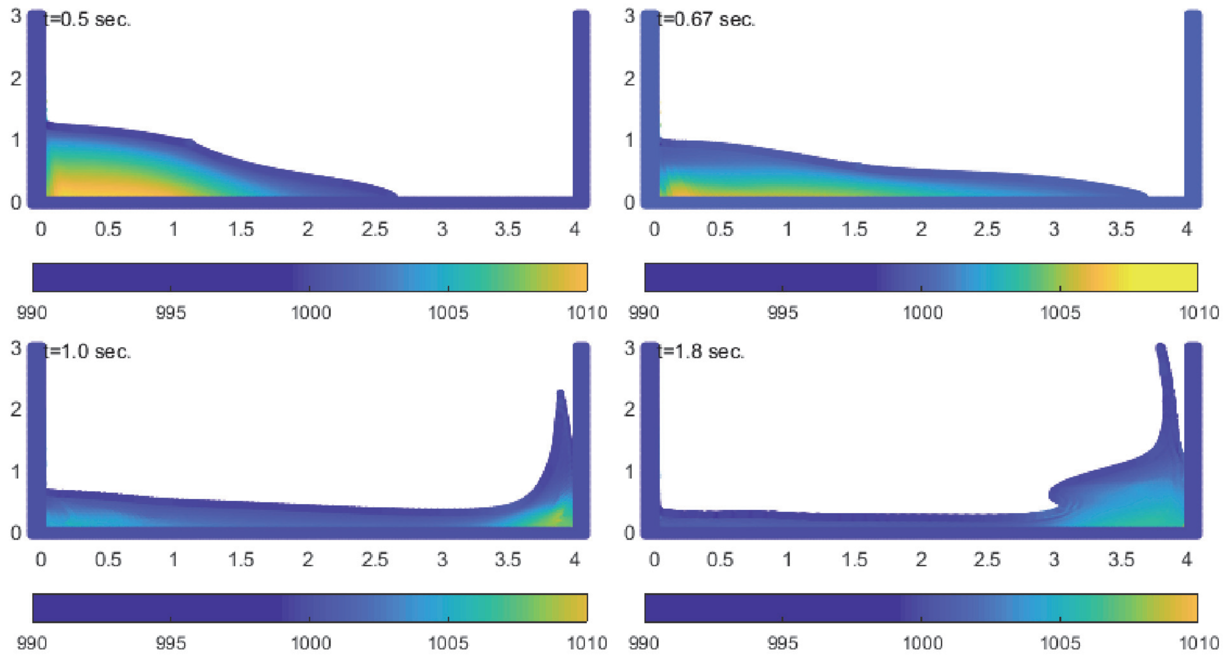


Figure 52: Test d. Snapshots. Colorbar denotes the norm of the velocity

It appears that the free surface is unaffected by the change of the boundary method and is almost identical to the results from test a. We continue with the investigation of the density distribution.



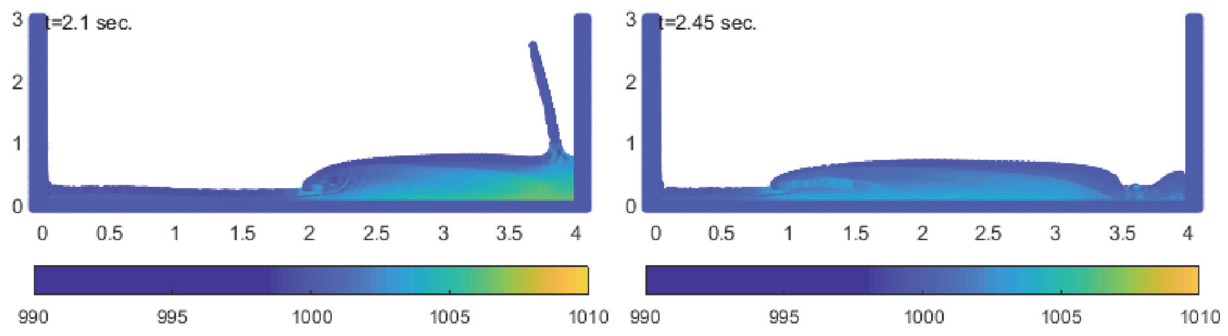


Figure 53: Test d. Snapshots. Colorbar denotes the density distribution

It is observed that we have achieved a smoother density profile than the preliminary case. Even during the plunge in the 2.45 s snapshot we have eliminated the spikes in density.

4.6. Test e.

In this test case we will try to assess the impact of the XSPH correction when set to take into account the dummy particles as well. We will use dummy boundary particles set with hydrostatic pressure and repulsive Lenard-Jones boundary particles following equation (49). For the time integration we will employ a Verlet scheme as presented in 2.10. Also the kernel function used is the quartic function proposed by Liu MB (2003) in equation (23). The fluid particles are placed in regular Cartesian grid.

Table 16: Parameter setup for test e.

Smoothing length	4 cm
Discretization	2 cm
Number of fluid particles	5000
Boundary Discretization	1 mm
Number of boundary particles	14000
D coefficient of Lenard-Jones forces	1.2
Boundary cut-off distance	2.4 cm
XSPH coefficient ε	0.5
a coefficient of artificial viscosity	0.3
Mass of fluid particles	0.02kg
Time step	0.000025 sec
Initial density	1000 kg/m^3
Artificial speed of sound	120 m/s

The comparison to the experimental data is given below:

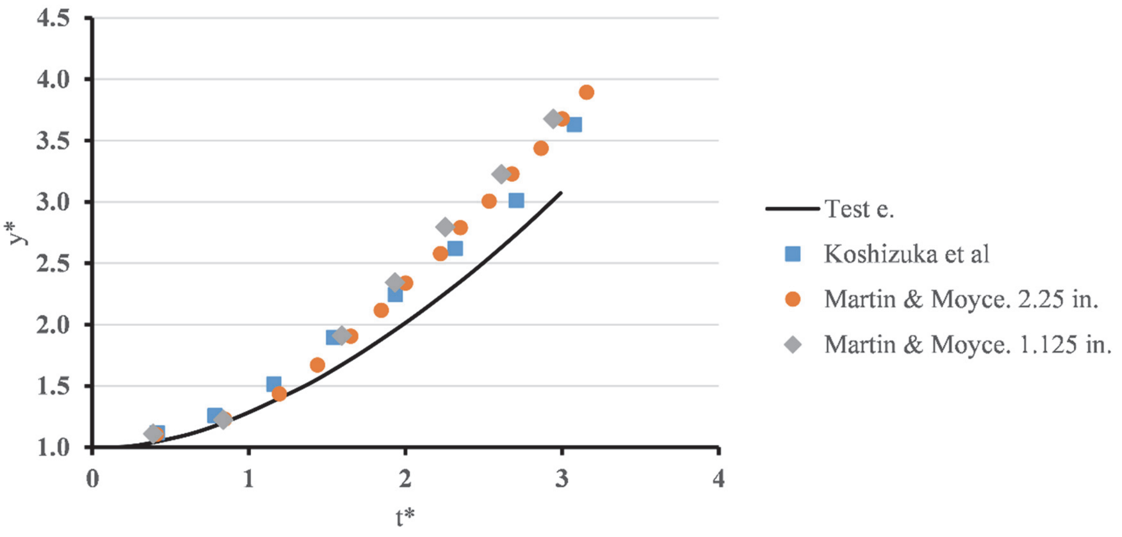


Figure 54: Toe evolution over dry bed for test e (non-dimensional).

As expected the flow was significantly slowed down resulting in poor performance if compared to the experimental data. The error is quantified in the table below. It is evident that this approach yields unnatural results and therefore it is not adequate.

Table 17: Test e errors

	Koshizuka et al	**Martin & Moyce, 2.25 in.**	**Martin & Moyce, 1.125 in.**
Test e.	1.114	1.430	1.131

4.7. Test f.

In this test case we examined the behavior of the method when using only Lenard-Jones repulsive forces at boundary (equation 48). From numerical experiments we have concluded that the cutoff distance of the boundary force must be set equal to the initial particle spacing. For the time integration we employed a Verlet scheme as presented in 2.10. Also the kernel function used is the quartic function proposed by Liu MB (2003) in equation (23). The fluid particles are placed in a regular Cartesian grid.

Table 18: Parameter setup for test f

Smoothing length	2 cm
Discretization	1 cm
Number of fluid particles	20000
Boundary Discretization	1 mm
Number of boundary particles	14000
D coefficient of Lenard-Jones forces	1.2
Boundary cut-off distance	1 cm
XSPH coefficient ε	0.5
a coefficient of artificial viscosity	0.3
Mass of fluid particles	0.01kg
Time step	0.000025 sec
Initial density	1000 kg/m^3
Artificial speed of sound	120 m/s

Comparison with experimental data is given in the graph below:

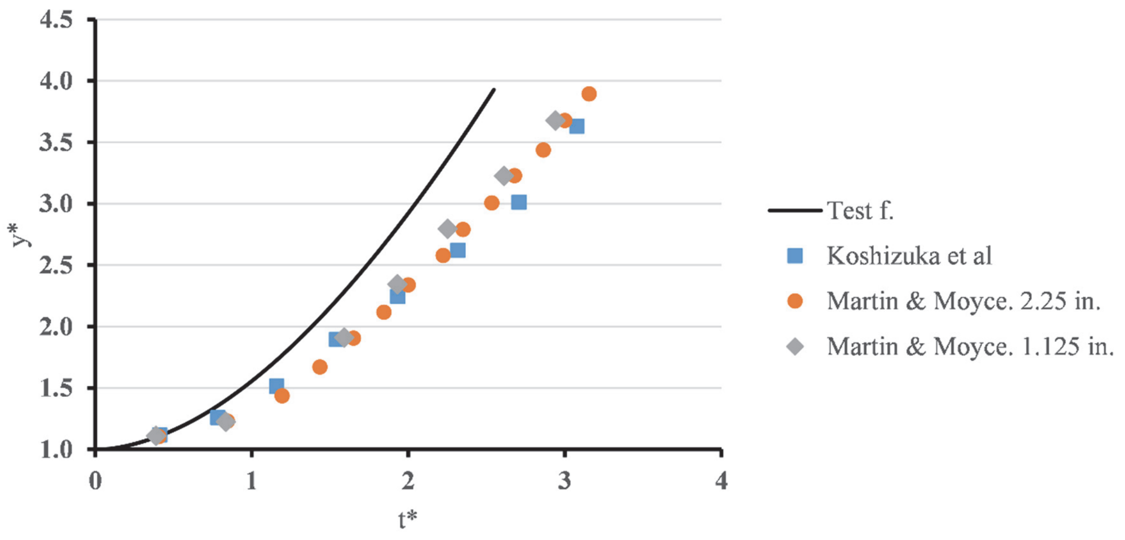


Figure 55: Toe evolution over dry bed for test f (non-dimensional).

We observe the simulation results deviate from the experiments, since the computed normalized y*(t*) takes higher values from the experiment, indicating that essentially the model does not provide any friction.

Table 19: Test f. errors

	Koshizuka et al	Martin & Moyce, 2.25 in.	Martin & Moyce, 1.125 in.
Test f.	1.487	2.132	1.092

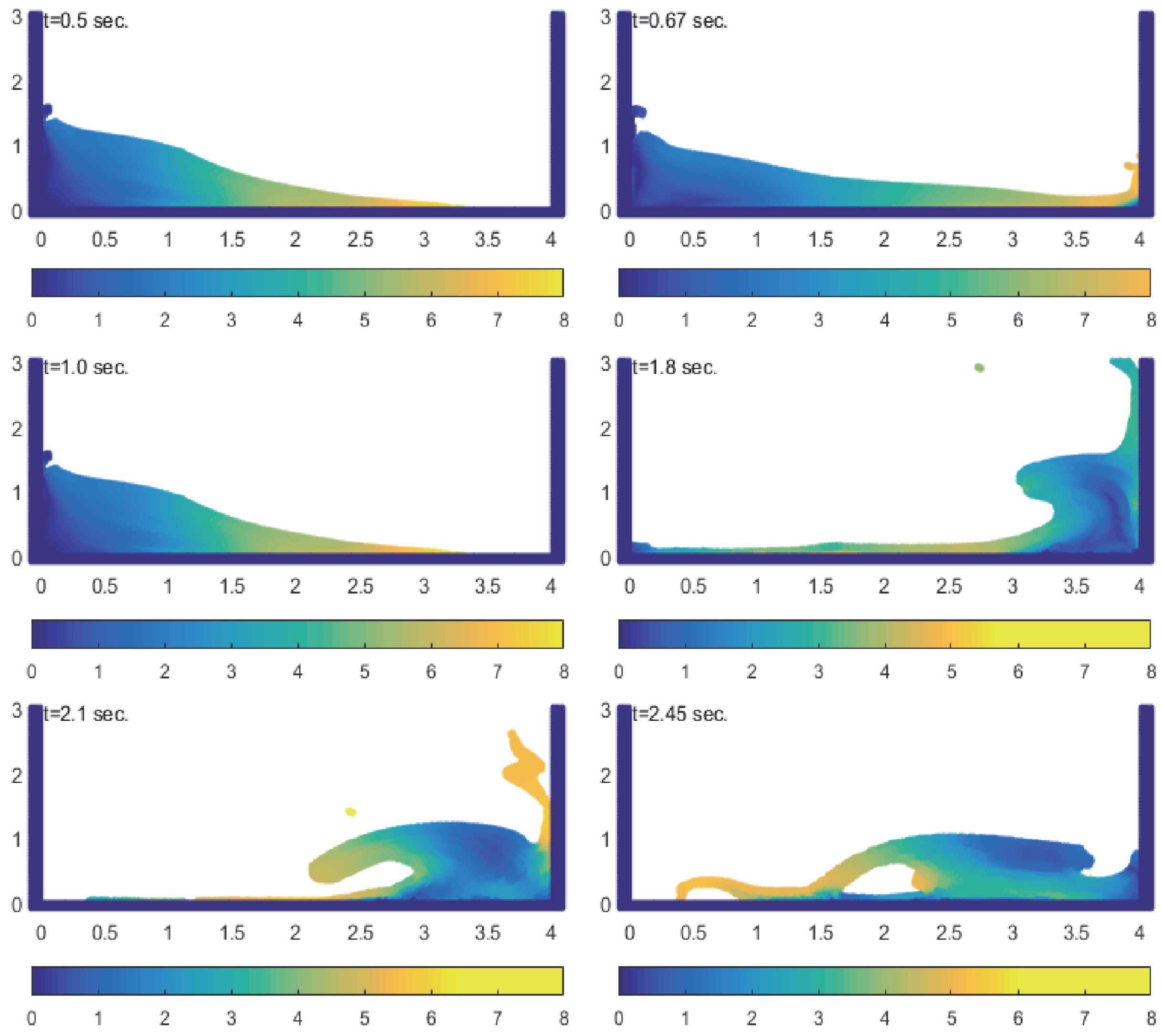


Figure 56: Test f. Snapshots. Colorbar denotes the norm of the velocity

From the snapshots above one may note that our model performs poorly with these boundary conditions. The free surface even before the collision of the wave with the right wall is rippled. Moreover, after the collision we observe that particles have departed from the wave. Also, the overturn and plunge have unnatural forms of free surface. Overall, as expected from the theoretical formulation, this method cannot be used in problems where the driving force is gravity. The rate of the repulsive force it exerts is much higher than the dominating forces of such problems. It may be used when modeling high velocity problems, such as explosions.

4.8. Test g.

In this test case we examined the behavior of the method when using only Lenard-Jones repulsive forces at boundary (equation 49). For the time integration we employed a Verlet scheme as presented in 2.10. Also the kernel function used is the quartic function proposed by Liu MB (2003) in equation (23). The fluid particles are placed in a regular Cartesian grid.

Table 20: Parameter setup for test g.

Smoothing length	1.5 cm
Discretization	1 cm
Number of fluid particles	20000
Boundary Discretization	1 mm
Number of boundary particles	14000
D coefficient of Lenard-Jones forces	1.2
Boundary cut-off distance	1.2 cm
XSPH coefficient ε	0.5
a coefficient of artificial viscosity	0.3
Mass of fluid particles	0.01kg
Time step	0.000025 sec
Initial density	1000 kg/m^3
Artificial speed of sound	120 m/s

The comparison to the experimental data is given below:

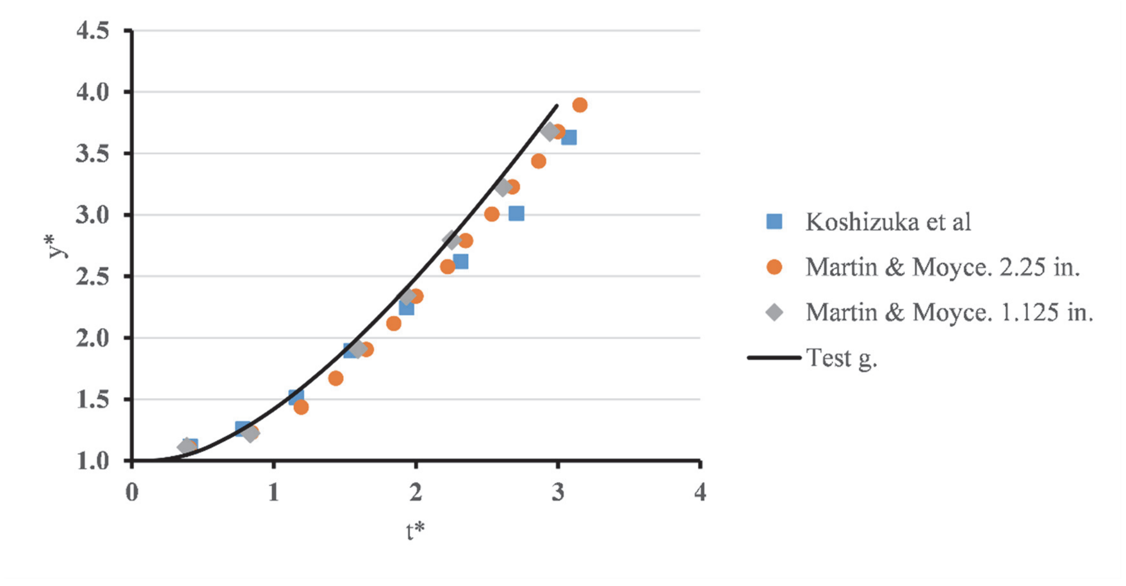


Figure 57: Toe evolution over dry bed for test g (non-dimensional).

We observe that as in test f the simulation results deviate from the experiments, since the model essentially does not provide any friction.

Table 21: Test g errors

	Koshizuka et al	**Martin & Moyce, 2.25 in.**	**Martin & Moyce, 1.125 in.**
Preliminary simulation	0.211	0.244	0.100

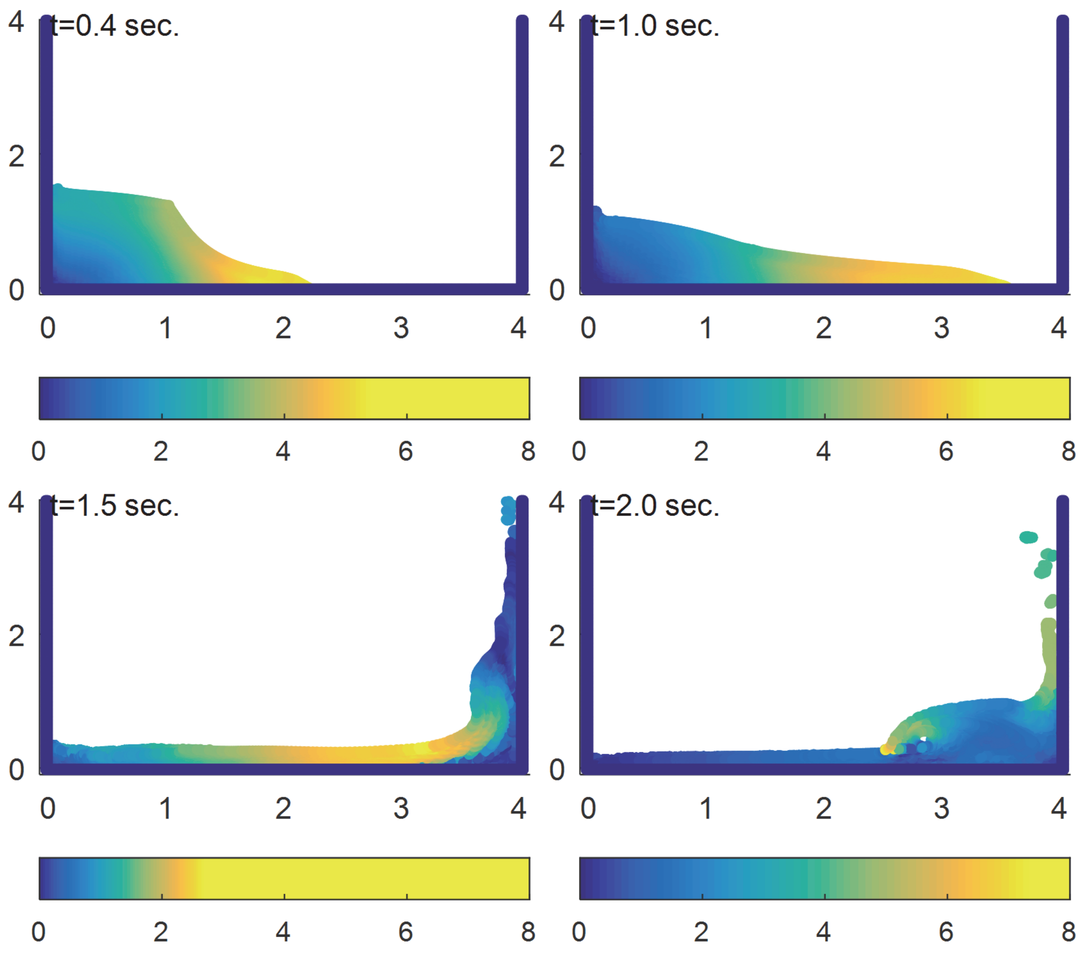


Figure 58: Test g. Snapshots. Colorbar denotes the norm of the velocity

Again we observe particle splashing upon impact on the right wall as well as general disorder on the particles. We will investigate further these disorders in t=0.4 s.

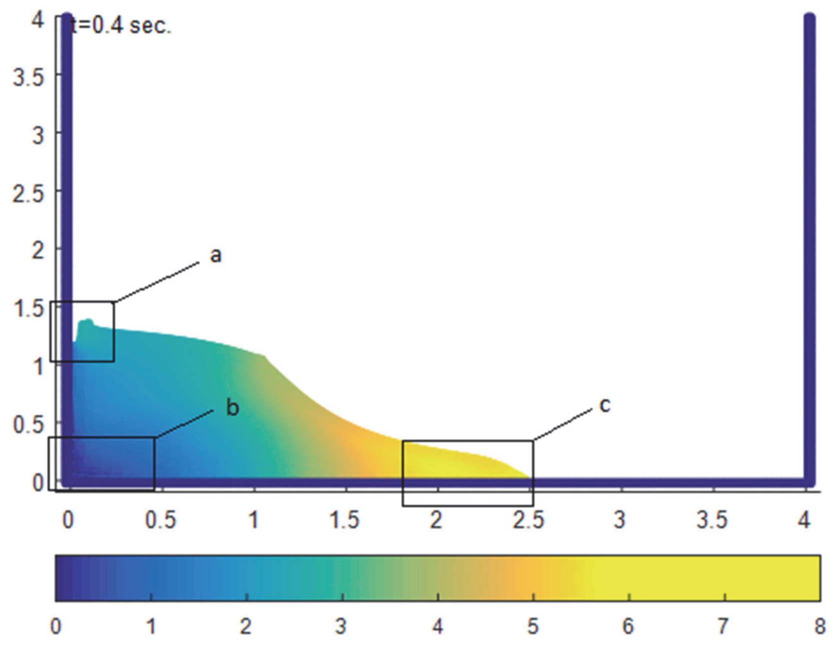


Figure 59: Snapshot of flow at t=0.4 sec.

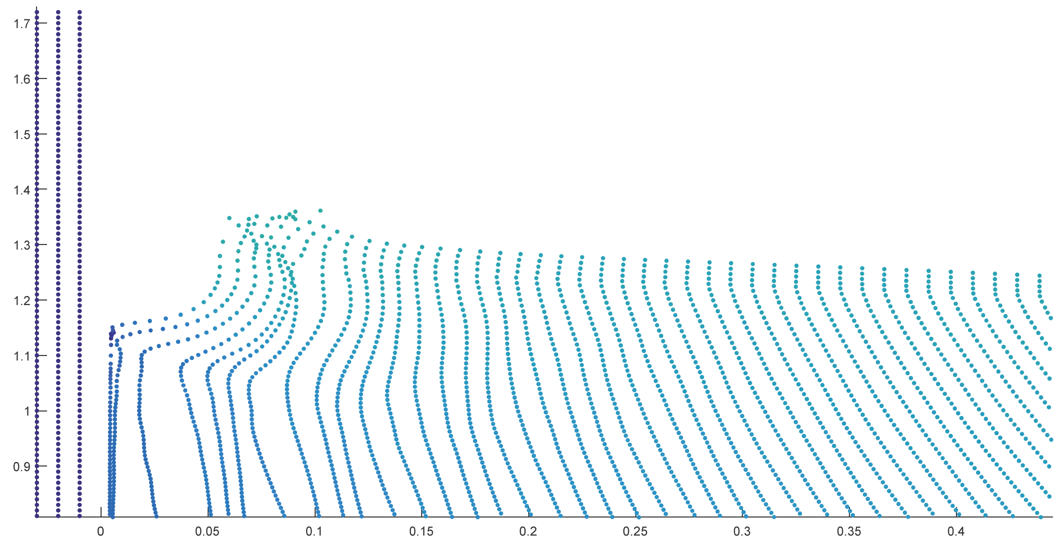


Figure 60: Position a

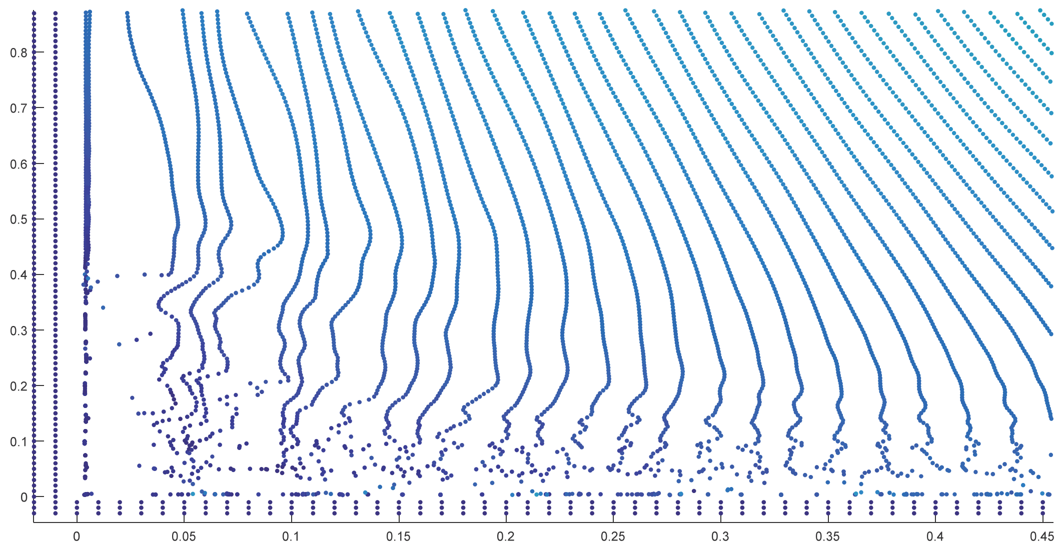


Figure 61: Position b

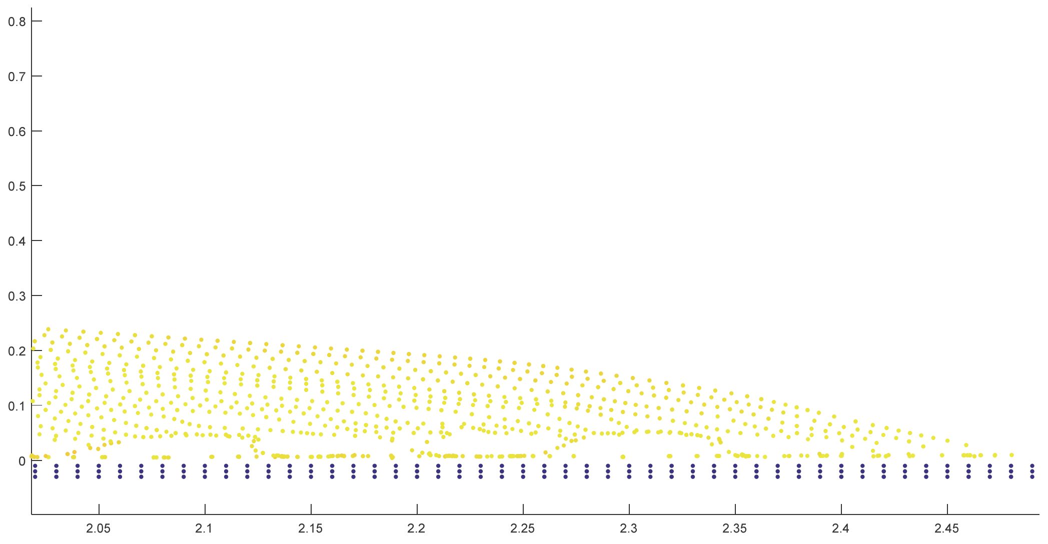


Figure 62: Position c

In all positions we observe large disorder of the fluid particles. Moreover in position c, at the toe, we observe that some of the particles stick to the boundary while others are repelled. In order to gain a better understanding we will present the same positions at t=0.4 s but with the dummy particles employed.

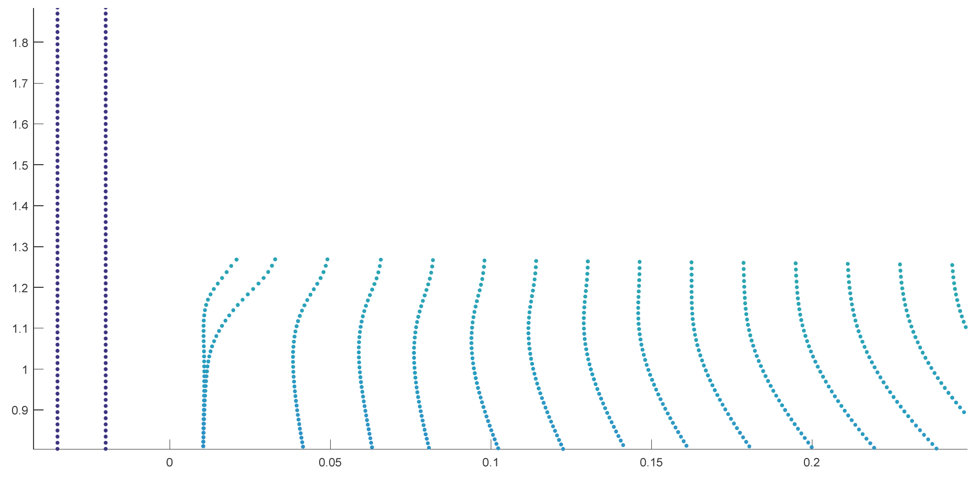


Figure 63: Position a with dummy particles employed

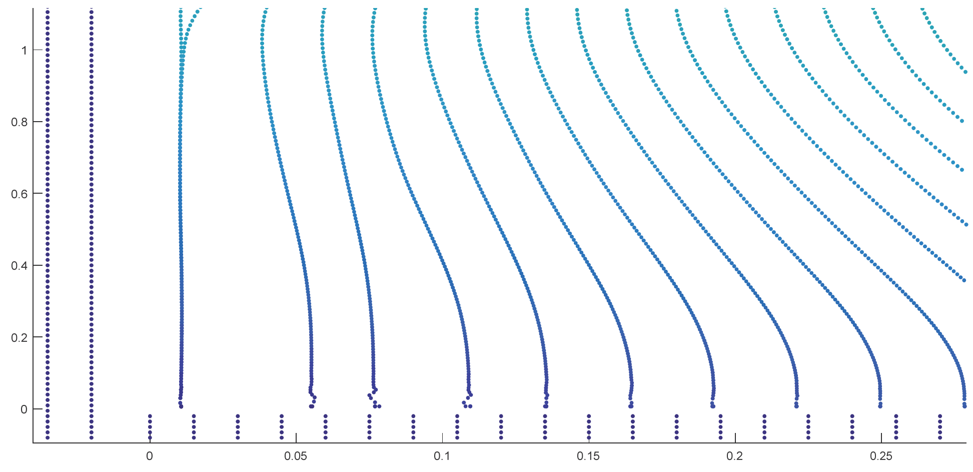


Figure 64: Position b with dummy particles employed

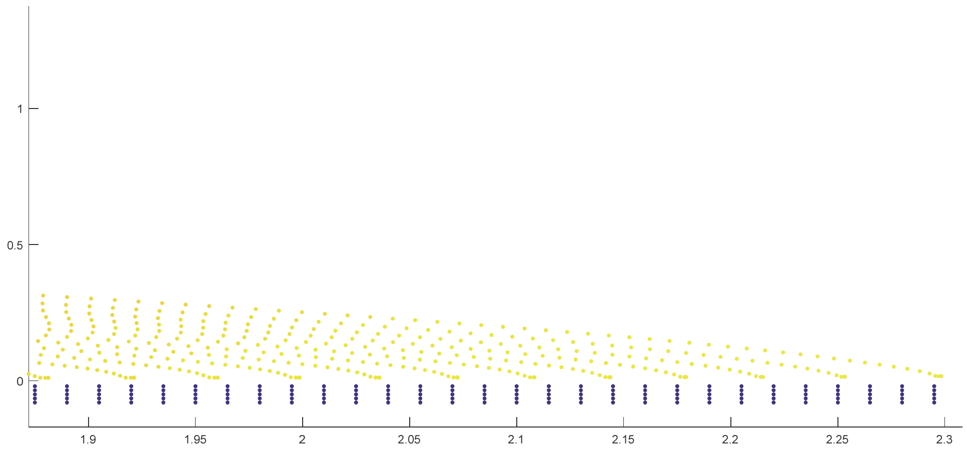


Figure 65: Position c with dummy particles employed

We observe that when the dummy particles are employed along with the repulsive Lenard-Jones forces the distortions are almost alleviated. The reason behind this improvement is the complete integration if the kernel near the boundary as discussed in chapter 2.8

Concluding, we may say that this boundary method indeed is suitable for simulating problems where the driving force is gravity, in contrary with the method used in test f. Although it yields disorders to the particles along the boundary as demonstrated before, it can be used to model sharp objects such as a sharp crested weir or a sluice gate.

4.9. Test h.

During this test case we examine the effect of removing the Shepard density filter. We used dummy boundary particles set with hydrostatic pressure and repulsive Lenard-Jones boundary particles following equation (49). For the time integration we employed a Verlet scheme as presented in 2.10. Also the kernel function used is the quartic function proposed by Liu MB (2003) in equation (23). The fluid particles are placed in regular Cartesian grid.

Table 22: Parameter setup for test h.

Smoothing length	2 cm
Discretization	1 cm
Number of fluid particles	20000
Boundary Discretization	1 mm
Number of boundary particles	14000
D coefficient of Lenard-Jones forces	1.2
Boundary cut-off distance	1.2cm
XSPH coefficient ε	0.5
a coefficient of artificial viscosity	0.3
Mass of fluid particles	0.01kg
Time step	0.000025 sec
Initial density	1000 kg/m^3
Artificial speed of sound	120 m/s

The comparison to experimental results is shown in figure 66.

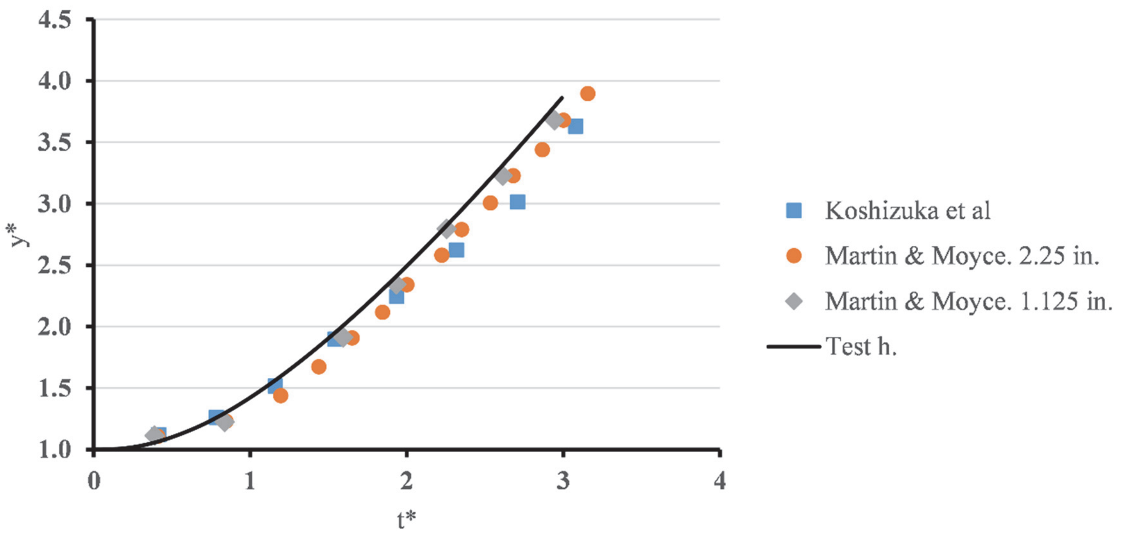


Figure 66: Toe evolution over dry bed for test h (non-dimensional).

Furthermore we compare the results with those obtained during preliminary simulation.

Table 23: Test h errors

	Koshizuka et al	**Martin & Moyce, 2.25 in.**	**Martin & Moyce, 1.125 in.**
Preliminary simulation	0.214	0.260	0.103
Test h.	0.205	0.246	0.096

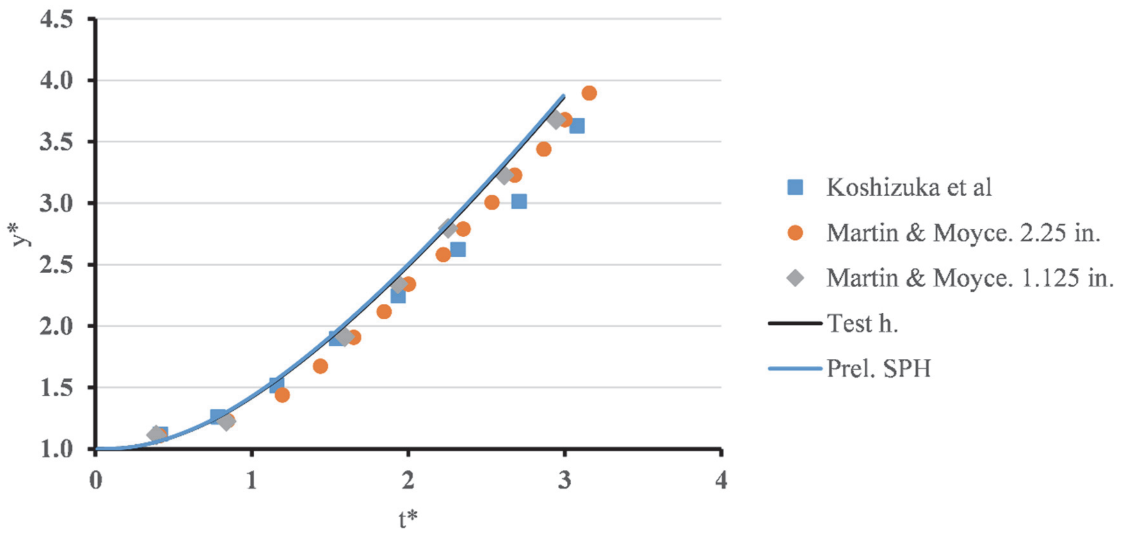


Figure 67: Toe evolution over dry bed for test h compared to Preliminary test (non-dimensional).

It is shown that removing the density filter has negligible effect in the evolution of the flow during the wave propagation. However, we may not detect a difference because up until this point the driving force of the flow is gravity, and pressures have not yet affected the flow significantly. Consequently we must examine the shape of the free surface after the collision with the container wall, in order to come to a conclusion regarding the effect of density filter.

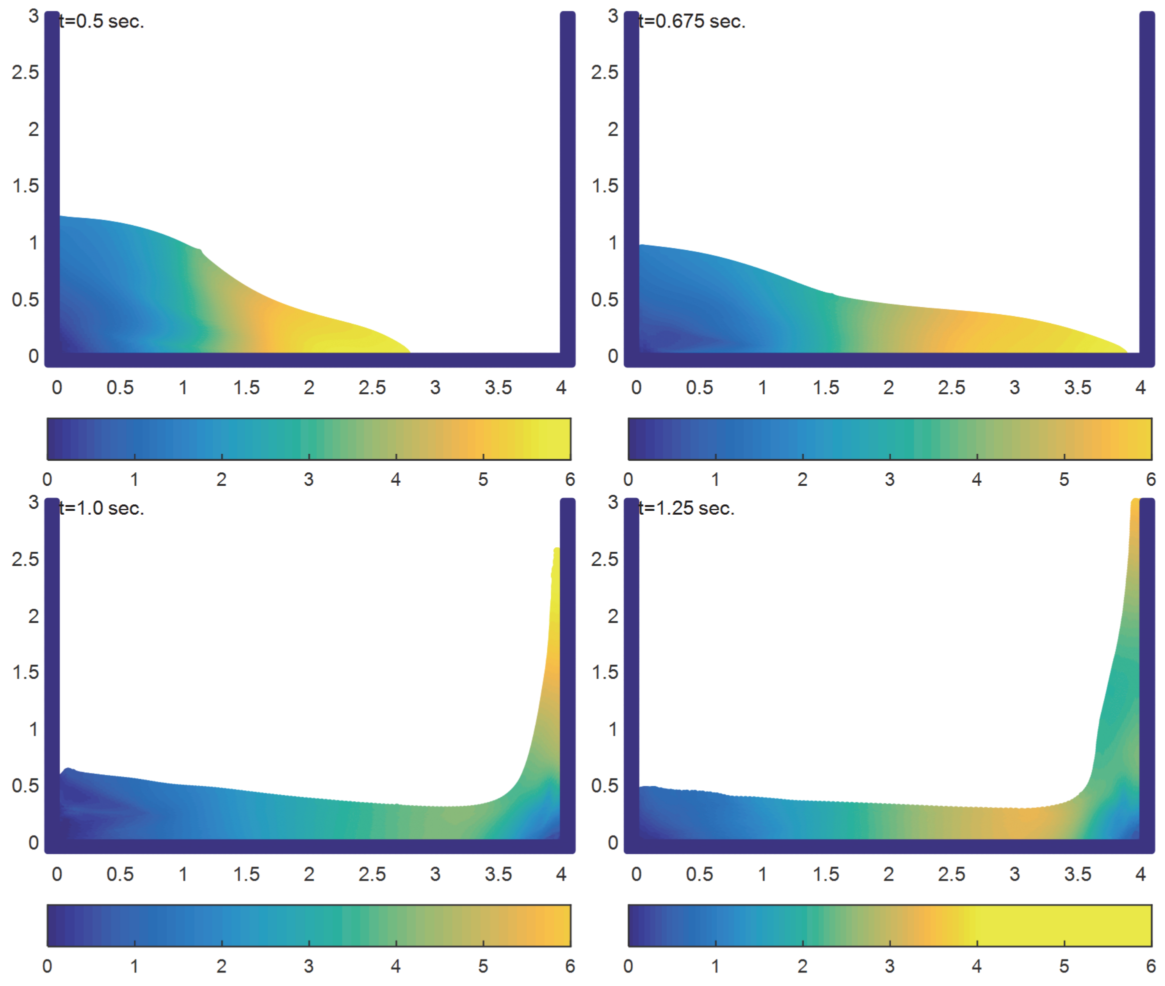


Figure 68: Test i. Snapshots. Colorbar denotes the norm of the velocity

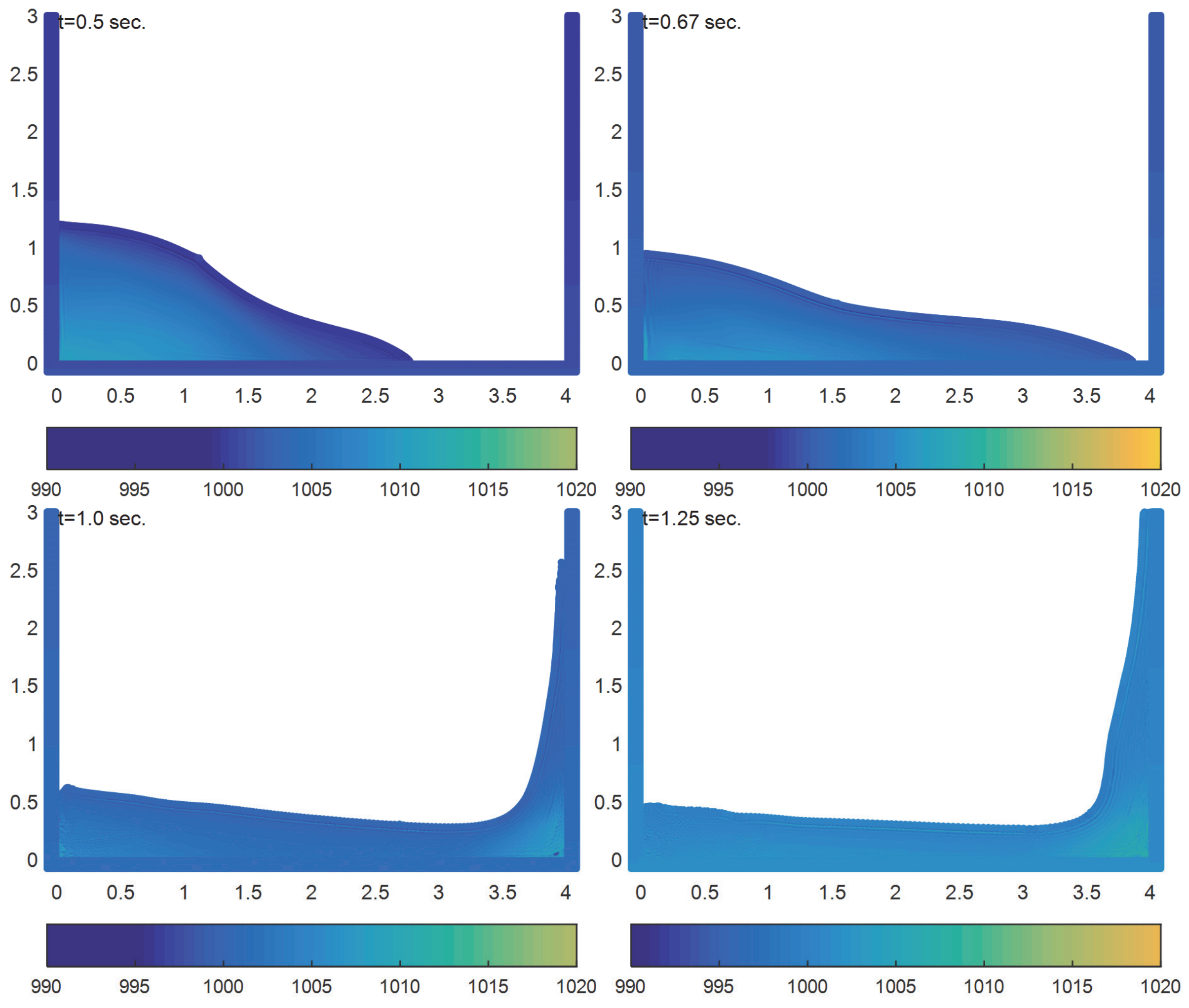


Figure 69: Test d. Snapshots. Colorbar denotes the density distribution

It appears that the free surface shape remains unaffected, however the velocity and density profiles are less smooth. Conclusively, the absence of density filter does not affect the performance of the simulation.

4.10. Test i.

In this chapter we used the Verlet time integration scheme for the simulation and a time step equal to 0.00005 s. We employed dummy boundary particles set with hydrostatic pressure and repulsive Lenard-Jones boundary particles following equation (49). Also, the kernel function used is the quartic function proposed by Liu MB (2003) in equation (23). The fluid particles are placed in a regular Cartesian grid.

Table 24: Parameter setup for test i.

Smoothing length	4 cm
Discretization	1.5cm
Number of fluid particles	10000
Boundary Discretization	1 mm
Number of boundary particles	14000
D coefficient of Lenard-Jones forces	1.2
Boundary cut-off distance	1.8 cm
XSPH coefficient ε	0.5
a coefficient of artificial viscosity	0.3
Mass of fluid particles	0.02kg
Time step	0.00005 sec
Initial density	1000 kg/m^3
Artificial speed of sound	120 m/s

The comparison with the experimental data is given below:

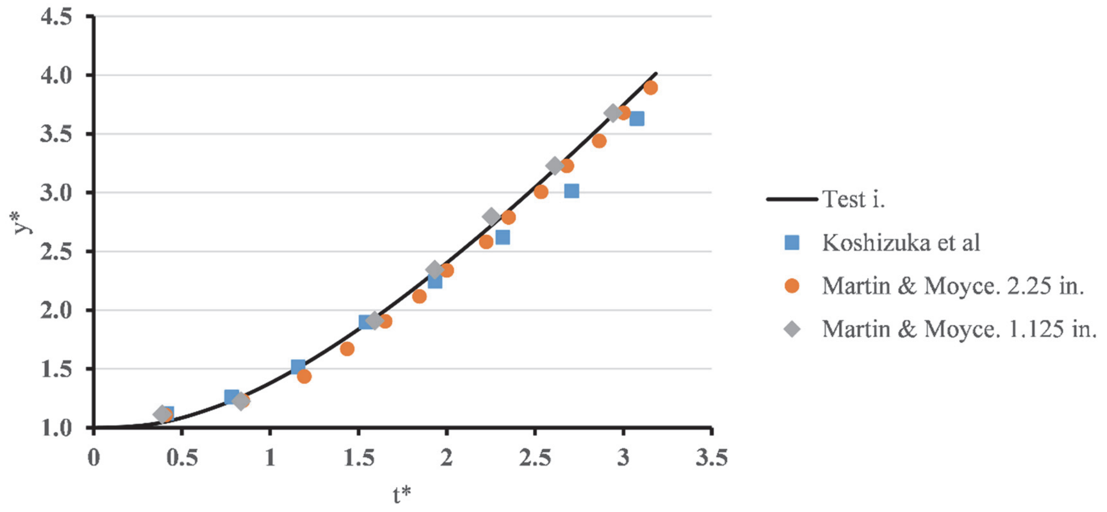


Figure 70 : Toe evolution over dry bed for test i (non-dimensional).

We will also compare our results with a test where a time step equal to 0.000025 s was used. We observe that when the time step is increased the simulation starts to deviate upwards. However, the deviation from experimental results is still not significant.

Table 25: Test i. errors

	Koshizuka et al	**Martin & Moyce, 2.25 in.**	**Martin & Moyce, 1.125 in.**
Test i.	0.161	0.160	0.081
Test a.	0.138	0.086	0.147

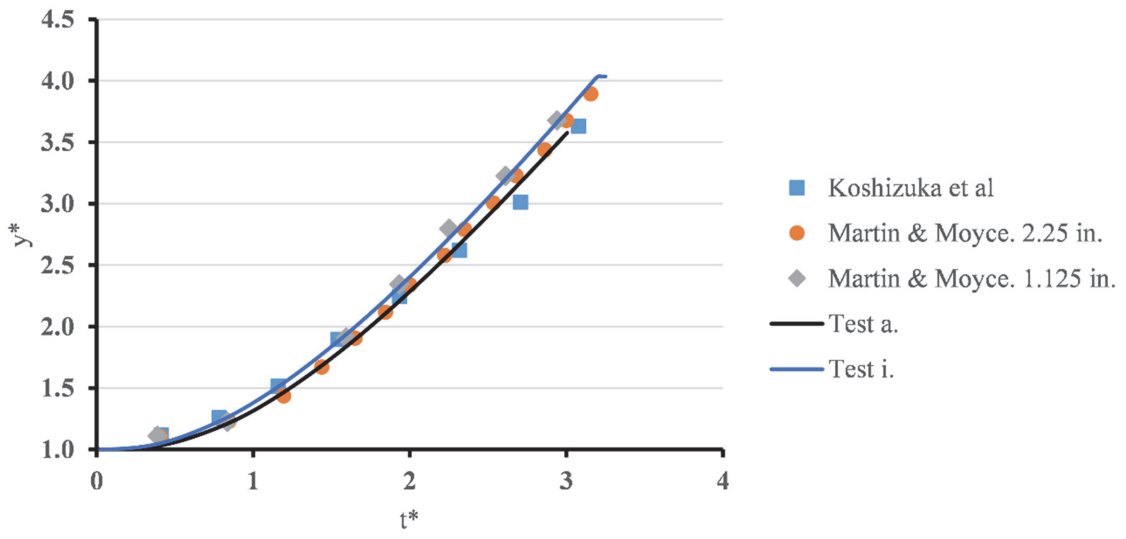


Figure 71: Toe evolution over dry bed for test I compared with test a. (non-dimensional).

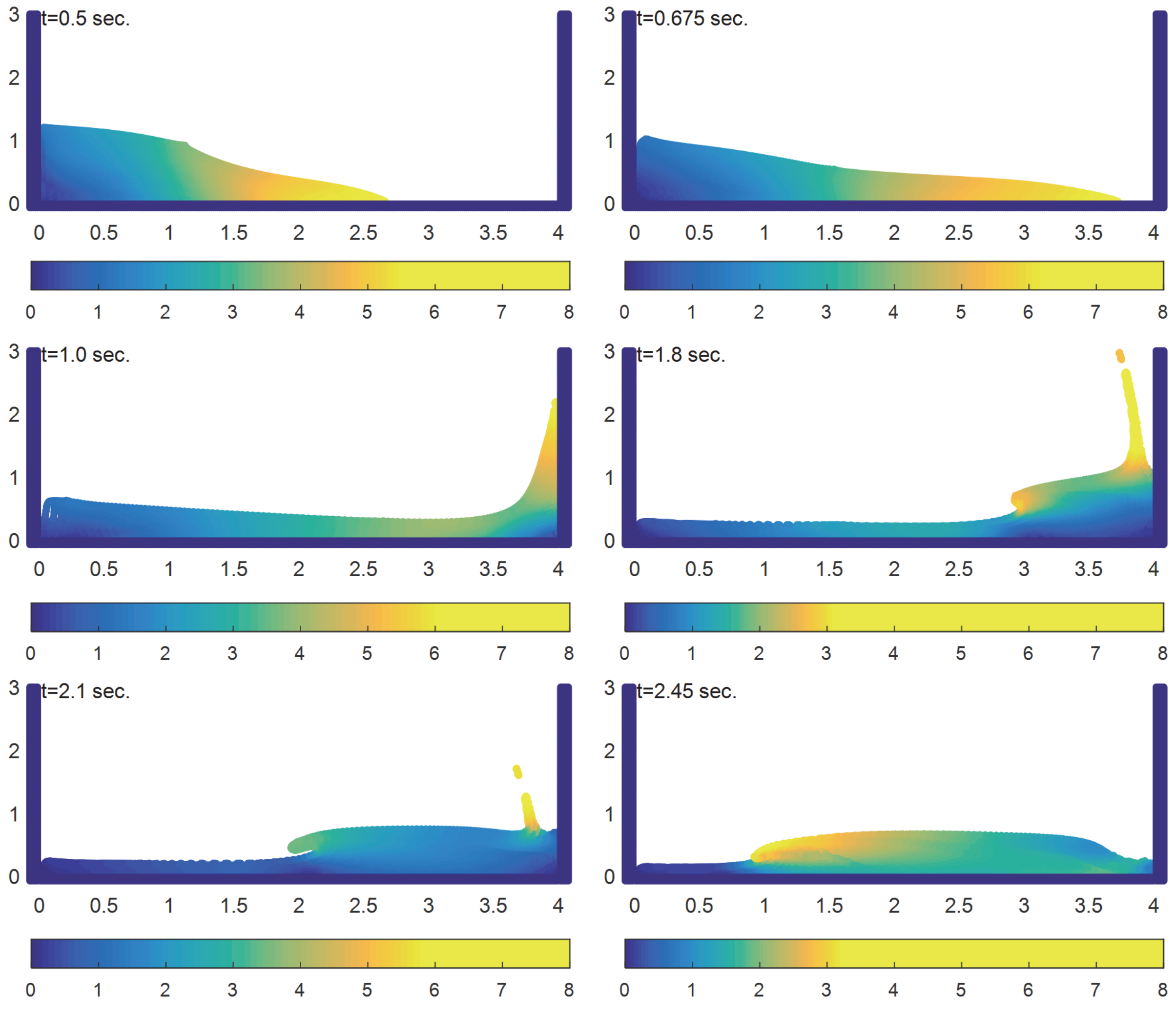


Figure 72: Test i. Snapshots. Colorbar denotes the norm of the velocity

Regarding the behavior of the free surface, we observe that is practically identical with the simulation performed in test a.

4.11. Test j.

In this chapter we used the Predictor - Corrector time integration scheme for the simulation and a time step equal to 0.00005 s. We employed dummy boundary particles set with hydrostatic pressure and repulsive Lenard-Jones boundary particles following equation (49). Also the kernel function used is the quartic function proposed by Liu MB (2003) in equation (23). The fluid particles were placed in a regular Cartesian grid.

Table 26: Parameter setup for test j.

Smoothing length	4 cm
Discretization	1.5cm
Number of fluid particles	10000
Boundary Discretization	1 mm
Number of boundary particles	14000
D coefficient of Lenard-Jones forces	1.2
Boundary cut-off distance	1.8 cm
XSPH coefficient ε	0.5
a coefficient of artificial viscosity	0.3
Mass of fluid particles	0.02kg
Time step	0.00005 sec
Initial density	1000 kg/m^3
Artificial speed of sound	120 m/s

The comparison with the experimental data is given below:

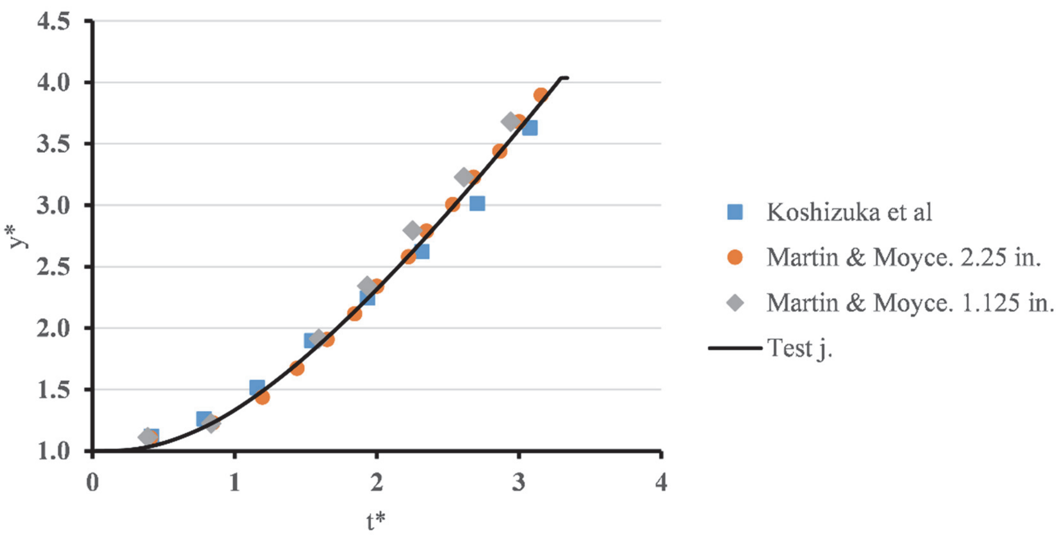


Figure 73: Toe evolution over dry bed for test j (non-dimensional).

We observe that the Predictor-Corrector time integration scheme performs much better for this time step. Essentially we have identical performance with the Verlet scheme when used with time step equal to 0.000025 s. Moreover, comparing the free surface derived in test (j) and in test (a) we find no visible differences. Subsequently we conclude that the performances of Verlet scheme with time step of 0.000025 s and of Predictor – Corrector scheme with time step of 0.00005 s are equivalent.

Table 27: Test j errors

	Koshizuka et al	Martin & Moyce, 2.25 in.	Martin & Moyce, 1.125 in.
Test j.	0.134	0.086	0.123
Test i.	0.161	0.160	0.081
Test a.	0.138	0.086	0.147

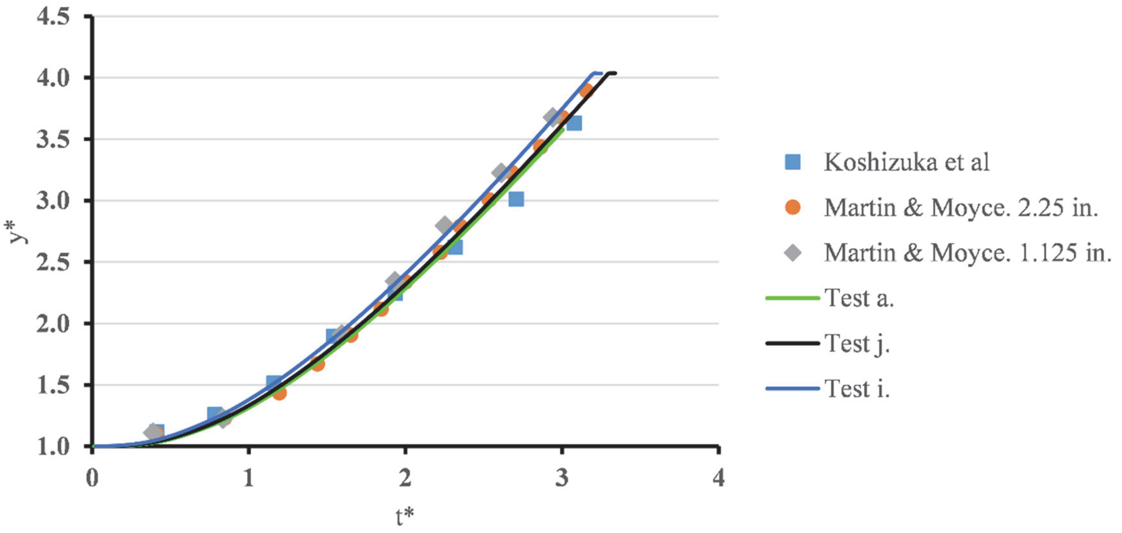


Figure 74: Comparison of toe evolution over dry bed for tests a.,i. and j. (non-dimensional)

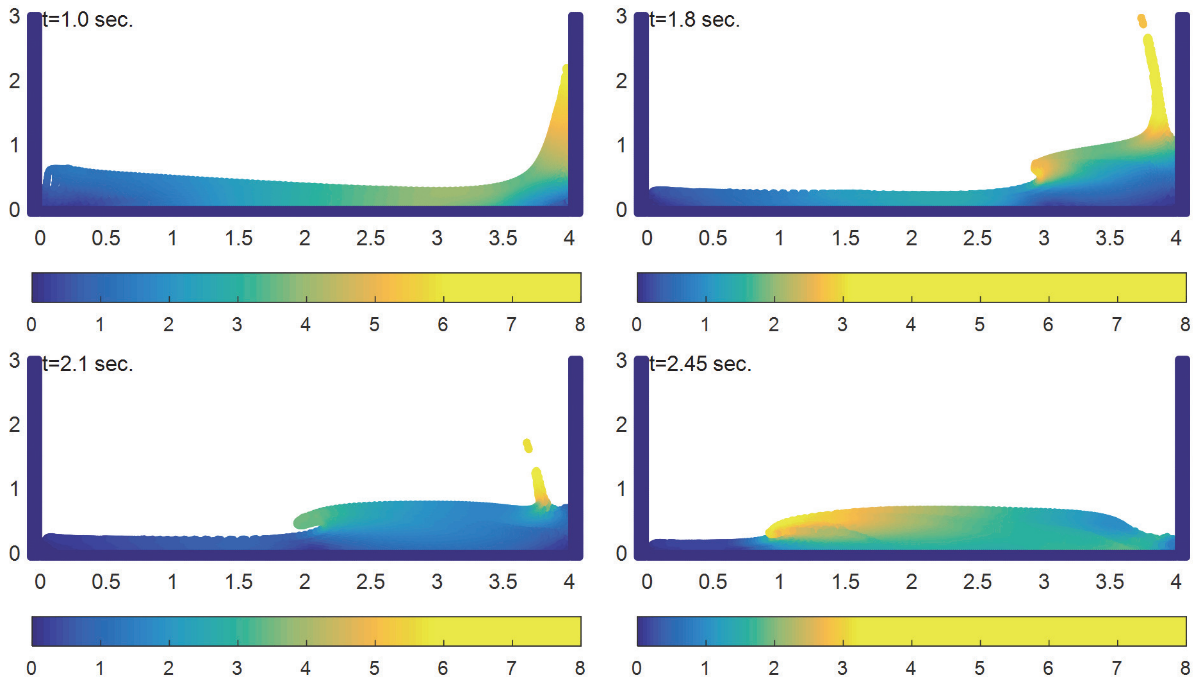


Figure 75: Test j. Snapshots. Colorbar denotes the norm of the velocity

4.12. Test k.

In this chapter we used the Predictor – Corrector time integration scheme for the simulation and a time step equal to 10^{-4} s. We employed dummy boundary particles set with hydrostatic pressure and repulsive Lenard-Jones boundary particles following equation (49). Also the kernel function used was the quartic function proposed by Liu MB (2003) in equation (23). The fluid particles were placed in a regular Cartesian grid.

Table 28: Parameter setup for test k.

Smoothing length	4 cm
Discretization	1.5cm
Number of fluid particles	10000
Boundary Discretization	1 mm
Number of boundary particles	14000
D coefficient of Lenard-Jones forces	1.2
Boundary cut-off distance	1.8 cm
XSPH coefficient ε	0.5
a coefficient of artificial viscosity	0.3
Mass of fluid particles	0.02kg
Time step	0.0001 sec
Initial density	1000 kg/m^3
Artificial speed of sound	120 m/s

Comparison to the experimental data is shown in Figure 74 below:

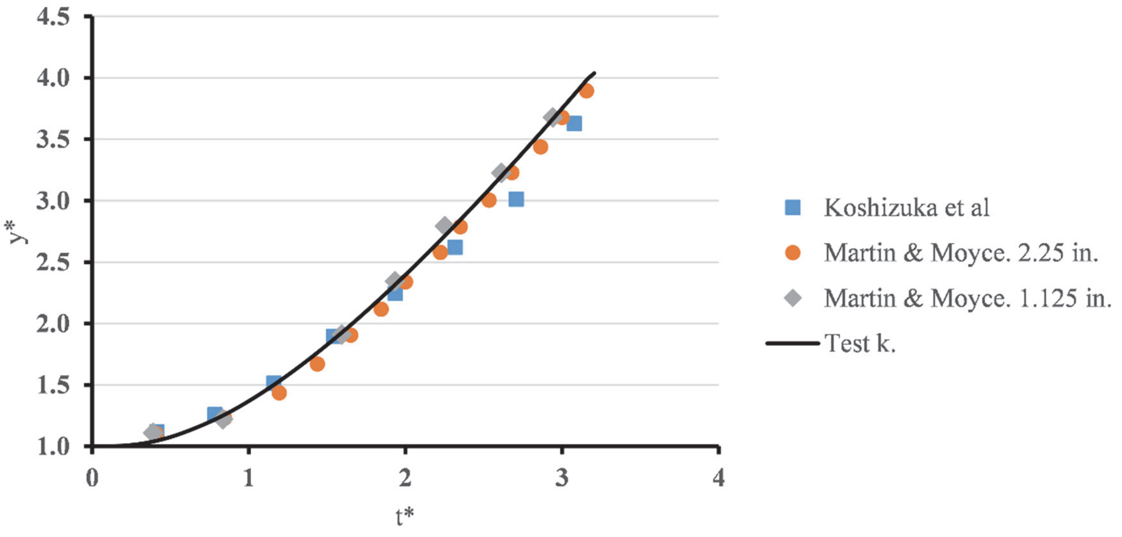


Figure 76: Toe evolution over dry bed for test k (non-dimensional).

Agreement with the experimental data is good but our results deviate slightly upwards from test cases (i) and (j).

	Koshizuka et al	Martin & Moyce, 2.25 in.	Martin & Moyce, 1.125 in.
Test k.	0.149	0.137	0.075

We will now focus on the behavior of the free surface and the distribution of density.

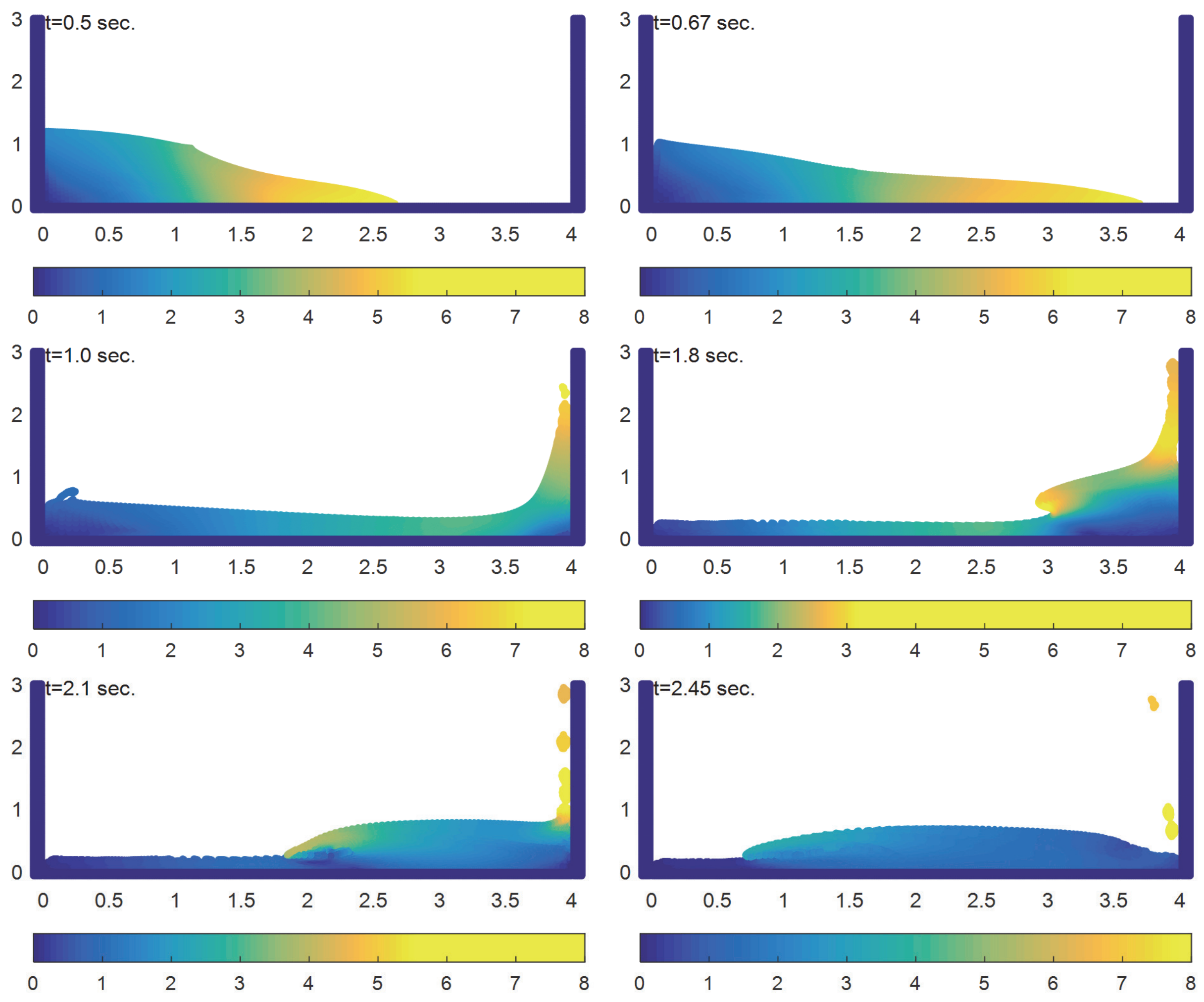
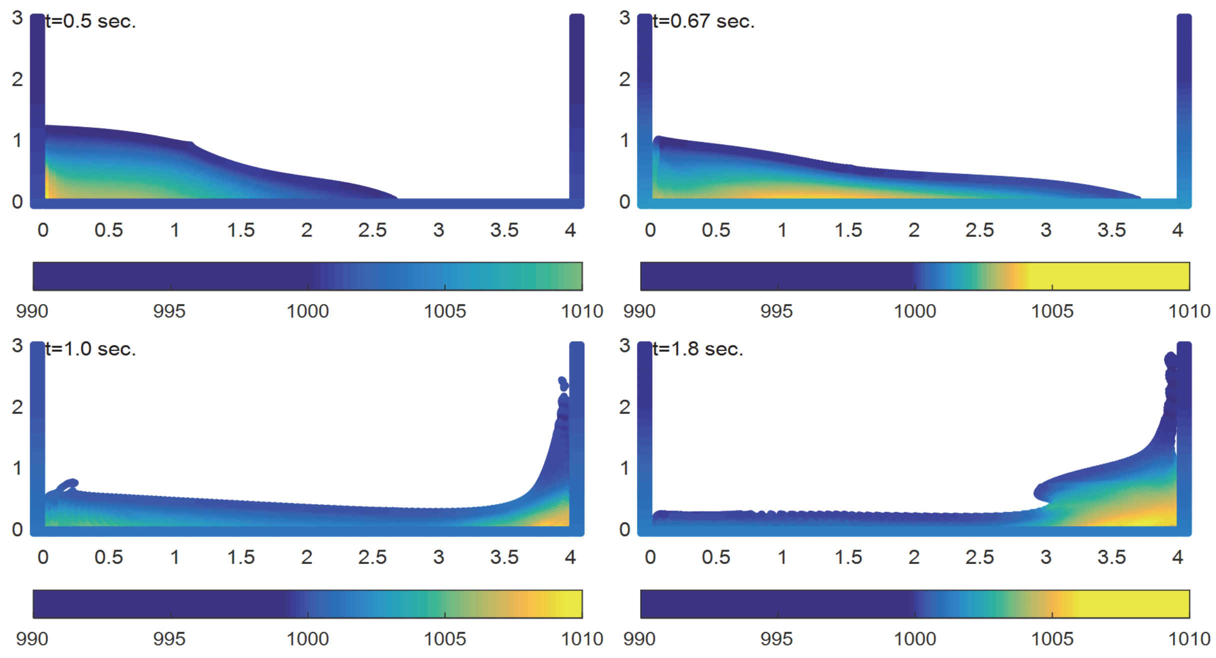


Figure 77: Test k. Snapshots. Colorbar denotes the norm of the velocity

We have obtained a smooth free surface profile from this simulation, which is in agreement with the one obtained during test (a).



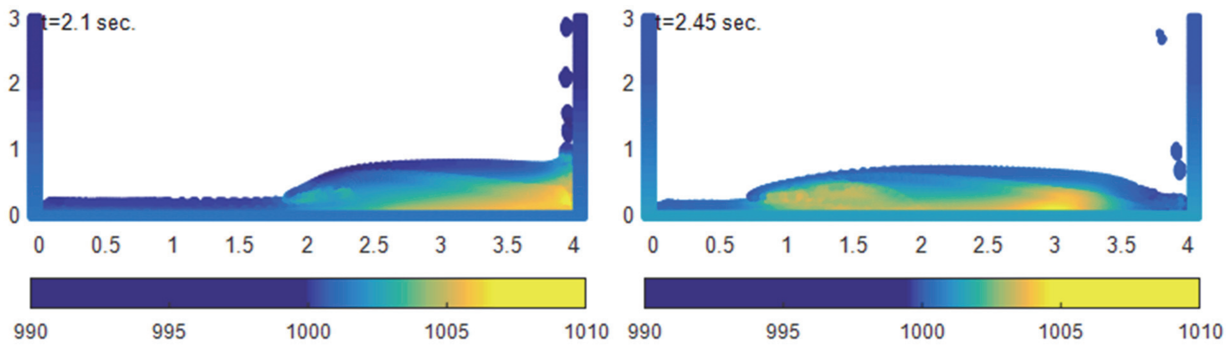


Figure 78: Test k. Snapshots. Colorbar denotes the density distribution

From figure 78 we see that the distribution of density differs from the one obtained in test (a) as it shows areas with higher densities near the boundaries. We have kept the same problem depended parameters in the two tests, except for the time step and the time integration scheme, and we conclude that this behavior is the result of the increment of time step. As it was demonstrated in figures 17 and 18 large time steps results in non-progressive interaction between fluid particles themselves, as well as boundary particles with fluid particles, resulting in pressure rise as particles approach each other.

4.13. Test l.

In this chapter we used the Verlet time integration scheme for the simulation and a time step equal to 10^{-4} s. We employed dummy boundary particles set with hydrostatic pressure and repulsive Lenard-Jones boundary particles following equation (49). Also the kernel function used was the quartic function proposed by Liu MB (2003) in equation (23). The fluid particles were placed in a regular Cartesian grid.

Table 29: Parameter setup for test l.

Smoothing length	2.2 cm
Discretization	1.5cm
Number of fluid particles	10000
Boundary Discretization	1 mm
Number of boundary particles	14000
D coefficient of Lenard-Jones forces	1.2
Boundary cut-off distance	1.8 cm
XSPH coefficient ε	0.5
a coefficient of artificial viscosity	0.3
Mass of fluid particles	0.02kg
Time step	0.0001 sec
Initial density	1000 kg/m^3
Artificial speed of sound	120 m/s

The comparison to the experimental data is given below:

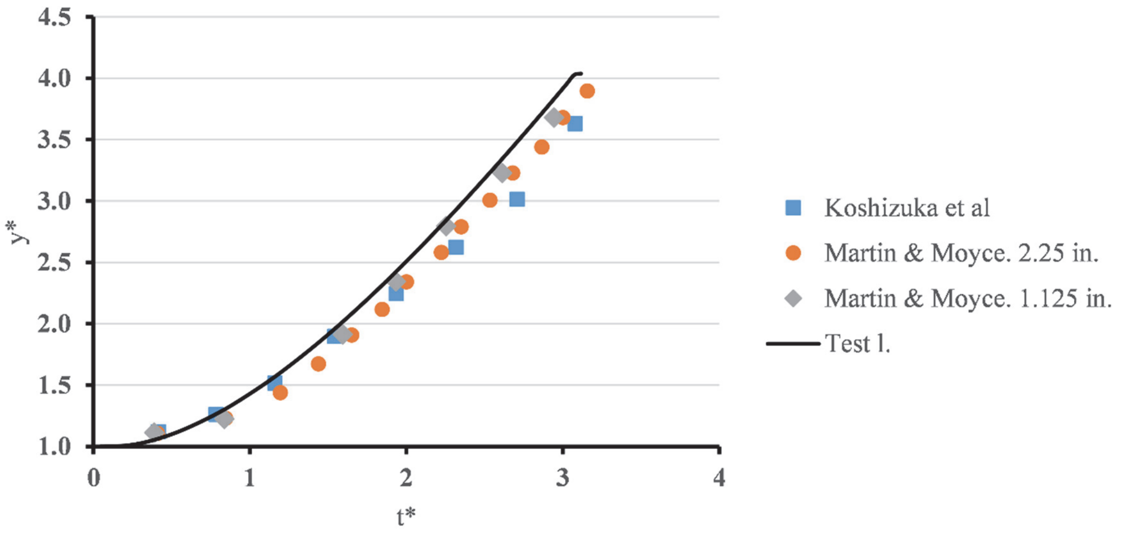


Figure 79: Toe evolution over dry bed for test l (non-dimensional).

We observe that the curve is deviating as the time increases. This is also shown in the errors from the experimental data:

Table 30: Test l. errors

	Koshizuka et al	Martin & Moyce, 2.25 in.	Martin & Moyce, 1.125 in.
Test l.	0.224	0.285	0.117

We will now focus on the free surface behavior. Snapshots of the simulation at certain time steps are presented, and we focus on the plunging of the overturning wave.

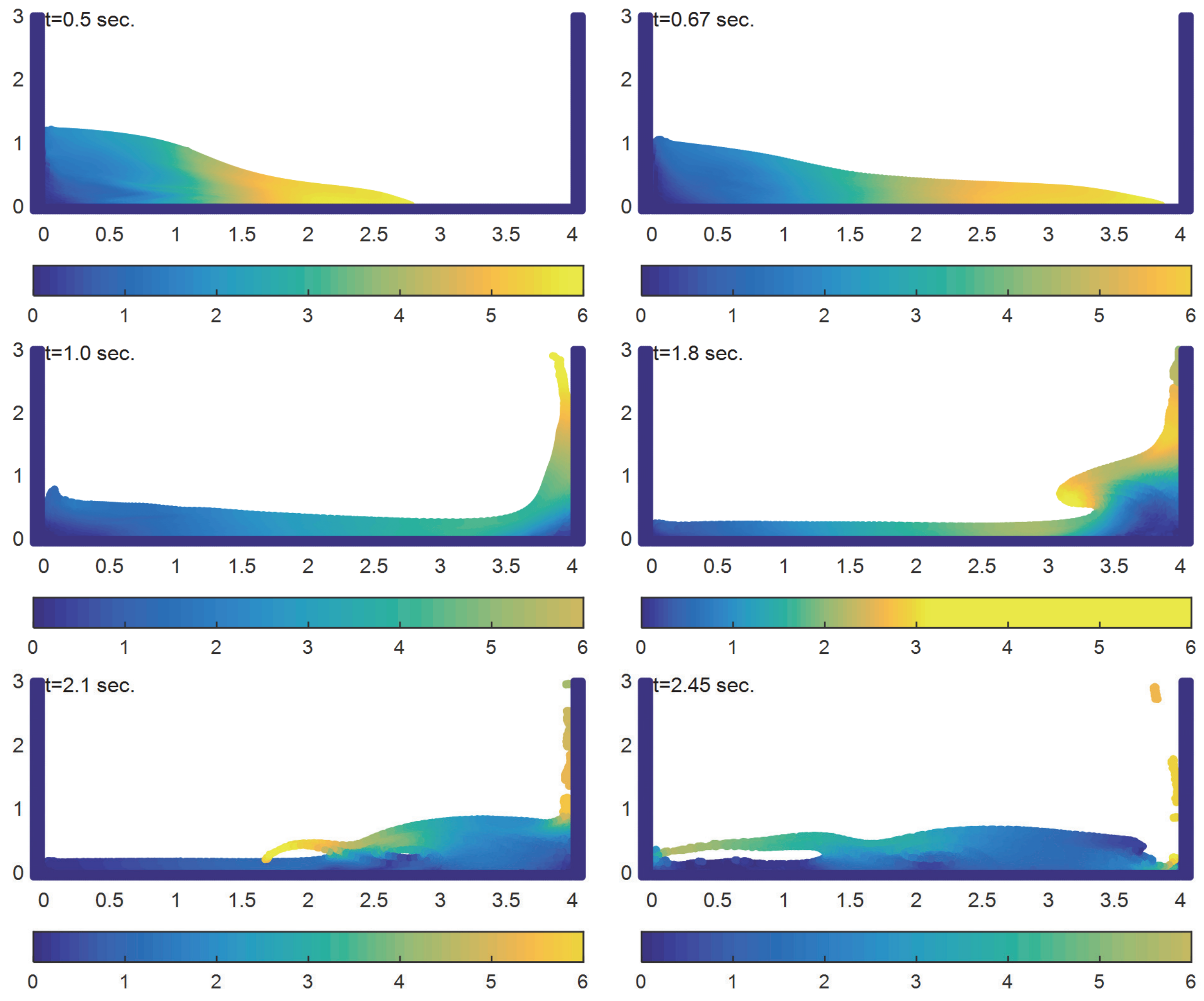


Figure 80: Test I. Snapshots. Colorbar denotes the norm of the velocity

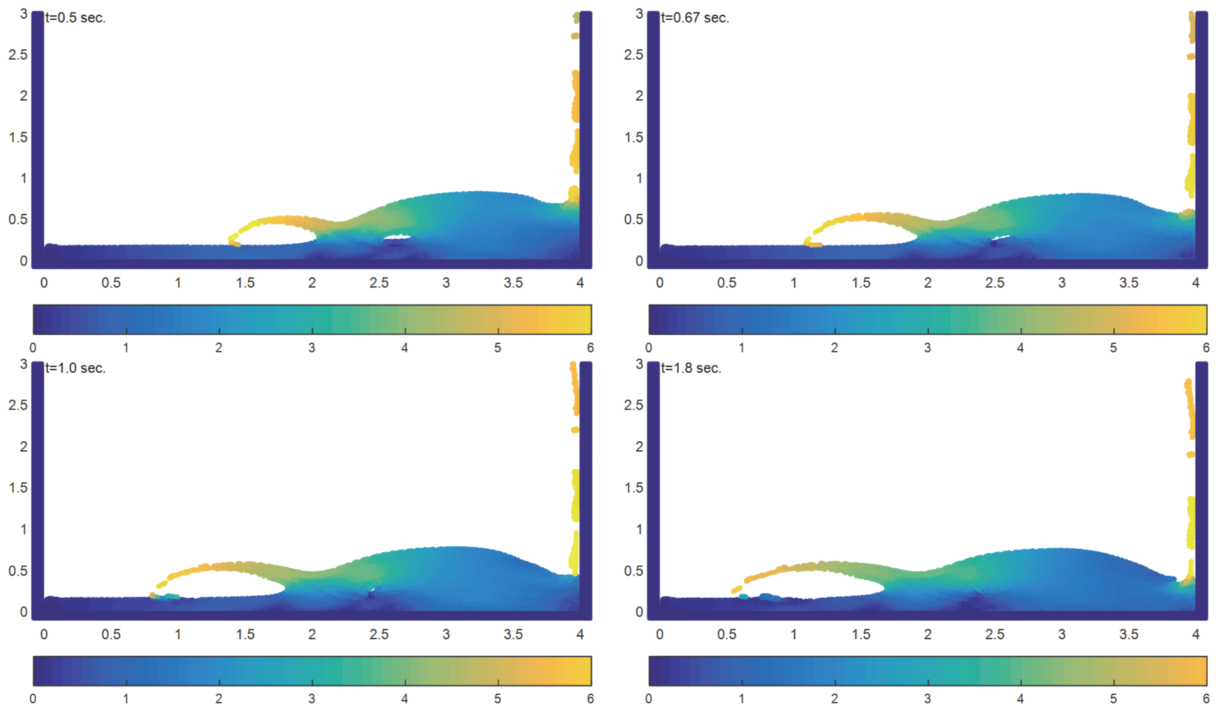


Figure 81: Snapshots of plunging wave in test I. Colorbar denotes the norm of the velocity

We can observe that the behavior of the plunging wave differs from the preliminary case. Specifically the air socket of the plunging wave is slightly larger and appears more natural. However this is not the case as test (a). where the XSPH correction was responsible. As it can be seen in figure 82.

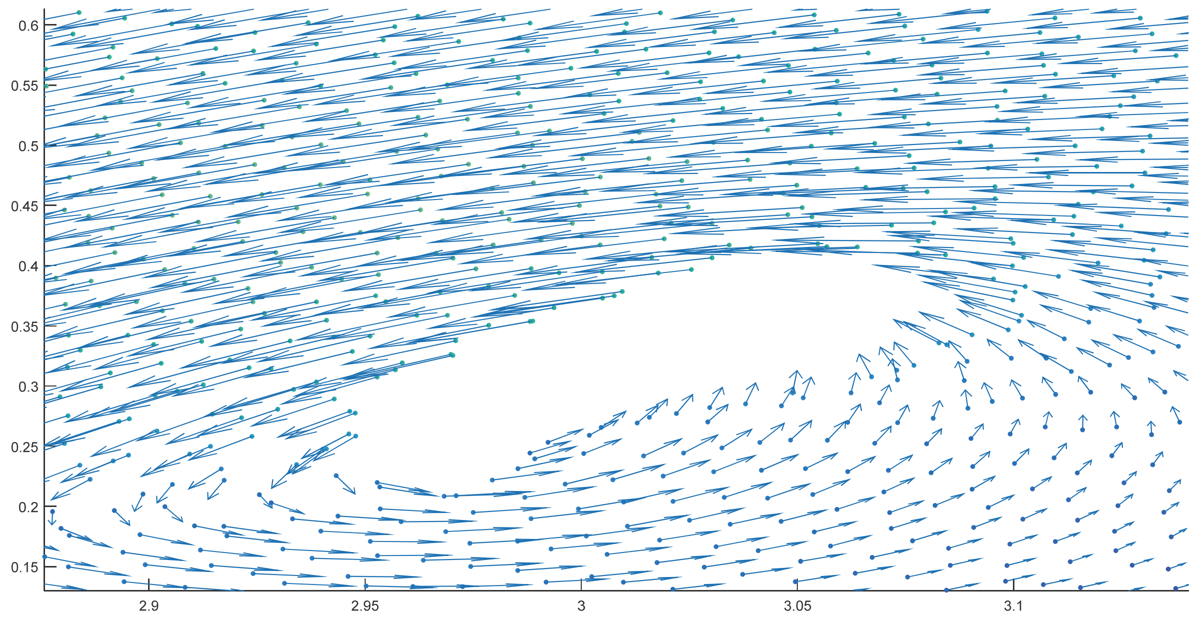


Figure 82: Snapshot of the plunging wave at t=1.95 sec. with velocity arrows

From the velocity arrows of the particles below the wave tongue it is evident that the XSPH correction does not change their speed unnaturally. Therefore the change in the free surface behavior is attributed to the change of the discretization along with the smoothing length.

4.14. Test m.

In this chapter we used the Verlet time integration scheme for the simulation and a time step equal to 10^{-4} s. However in this test the smoothing length is increased to 4 cm in order to compare the results with previous tests and examine the impact of it in high time steps. Also in this simulation we have employed dummy boundary particles set with hydrostatic pressure and repulsive Lenard-Jones boundary particles following equation (49). The kernel function used was the quartic function proposed by Liu MB (2003) in equation (23). The fluid particles were placed in a regular Cartesian grid.

Table 31: Parameter setup for test m.

Smoothing length	4 cm
Discretization	1.5cm
Number of fluid particles	10000
Boundary Discretization	1 mm
Number of boundary particles	14000
D coefficient of Lenard-Jones forces	1.2
Boundary cut-off distance	1.8 cm
XSPH coefficient ε	0.5
a coefficient of artificial viscosity	0.3
Mass of fluid particles	0.02kg
Time step	0.0001 sec
Initial density	1000 kg/m^3
Artificial speed of sound	120 m/s

The comparison to the experimental results is given below:

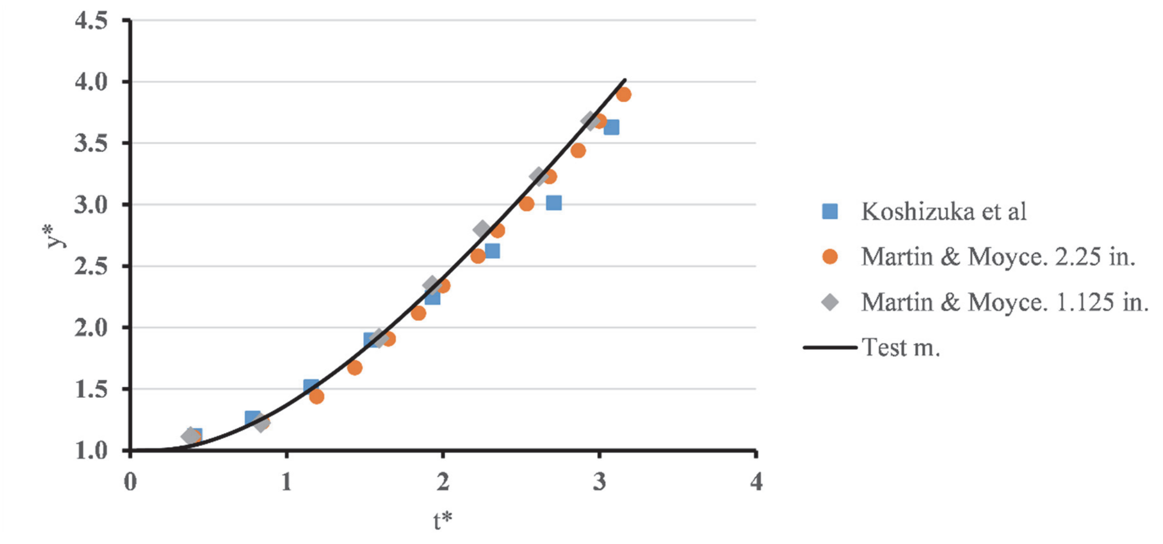


Figure 83: Toe evolution over dry bed for test m (non-dimensional).

Good agreement with the experimental data is observed. From comparison to the results from test (i) we see that raising the smoothing length improves the performance.

Table 32: Test l errors

	Koshizuka et al	**Martin & Moyce, 2.25 in.**	**Martin & Moyce, 1.125 in.**

Test l.	0.224	0.285	0.117
Test m.	0.158	0.146	0.076

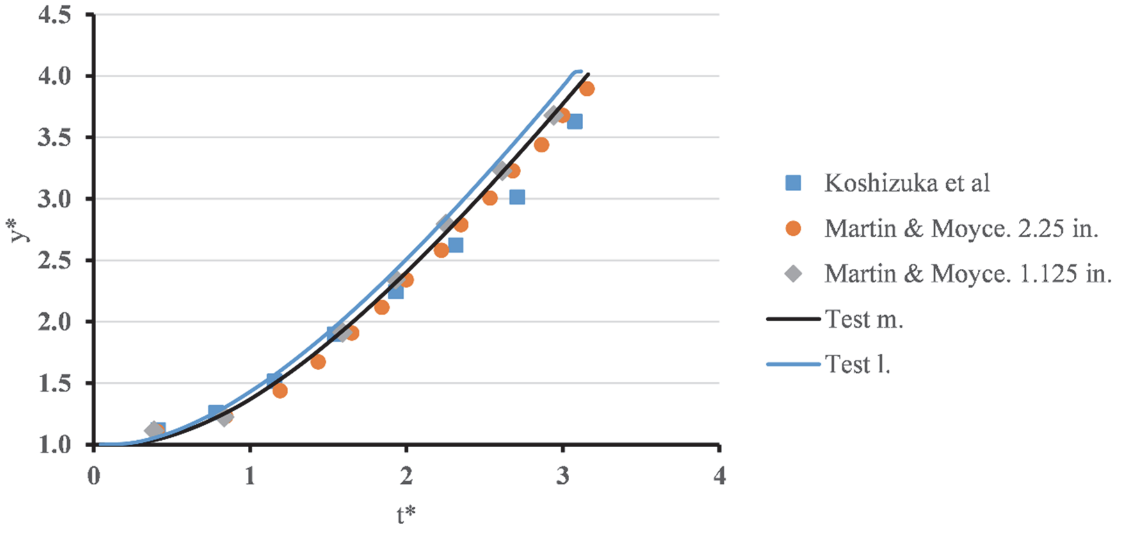


Figure 84: Toe evolution over dry bed for test m compared to test l (non-dimensional).

In particular we notice that the results acquired from test (i) deviate upwards if compared to those from test (m). We continue with the evaluation of the free surface behavior and density distribution.

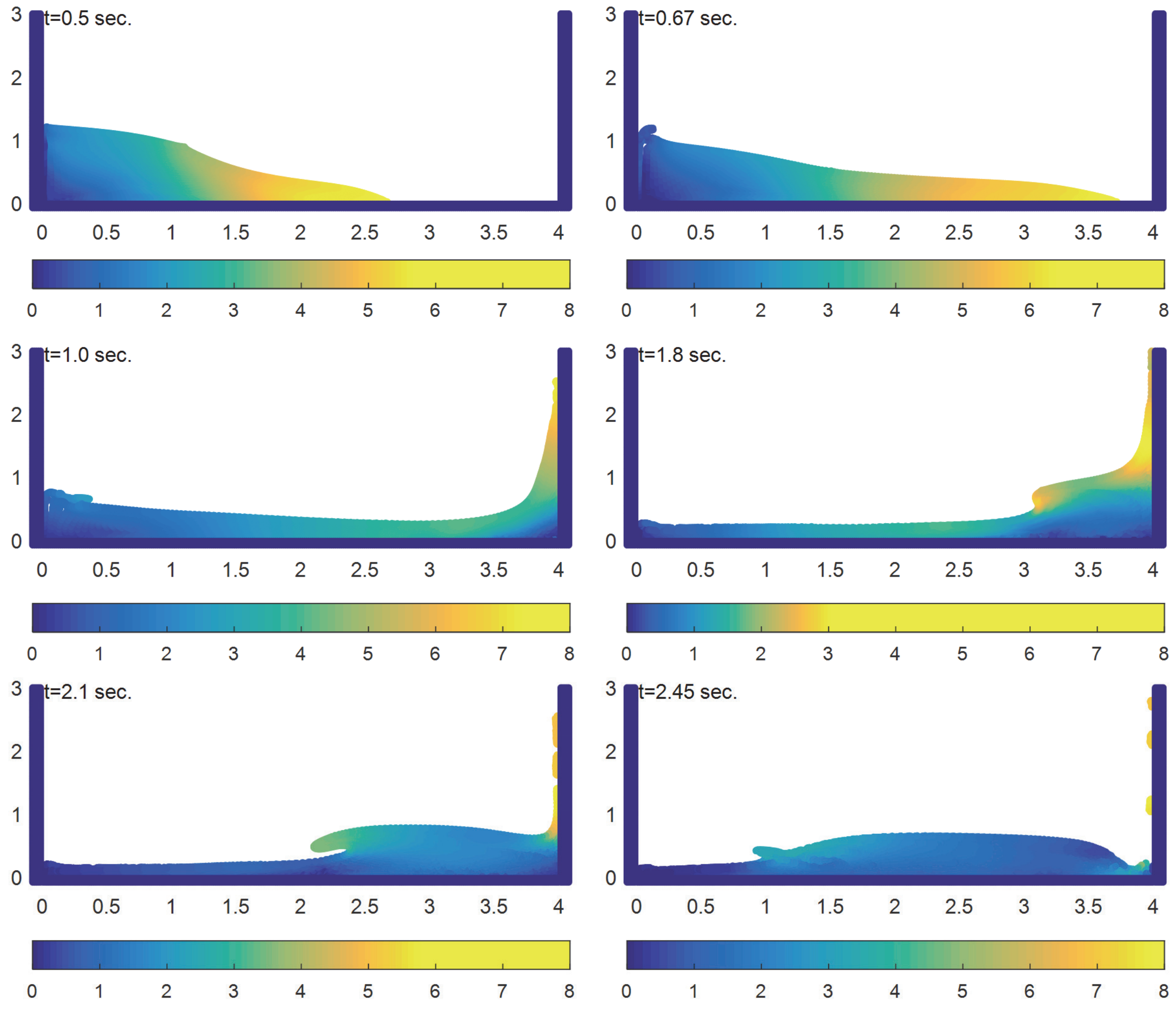


Figure 85: Test m. Snapshots. Colorbar denotes the norm of the velocity

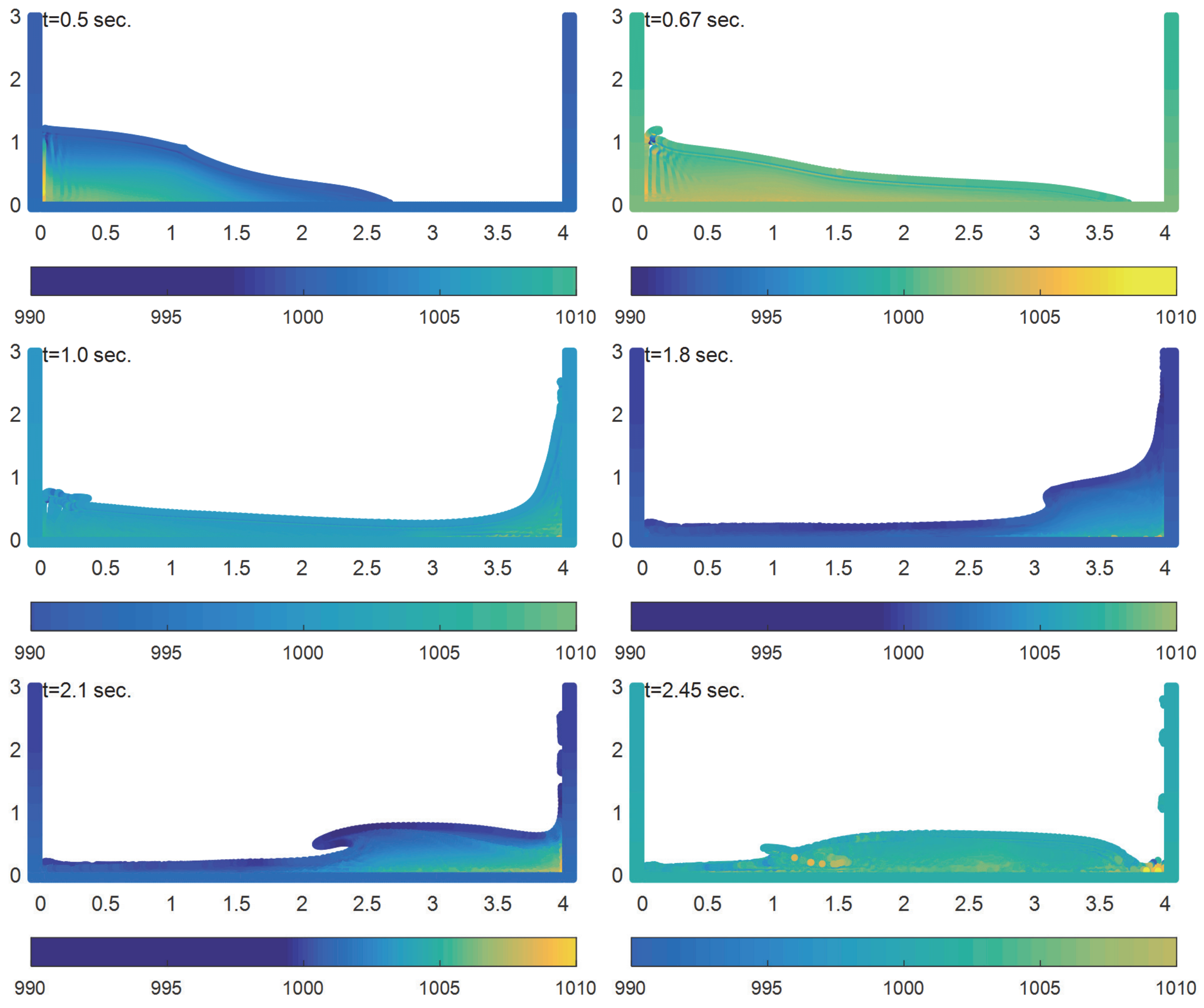


Figure 86: Test m. Snapshots. Colorbar denotes the density distribution

As expected the free surface generated from test (m) behaves in a similar manner with that from test (a), because it is affected by the XSPH velocity correction as it was discussed earlier. Consequently the plunge of the overturning wave “drowns”. However it seems to be similar with that from test (k) but not with test (i) for the same reason. Moreover the density distribution appears less smooth and in order than test (k).

4.15. Hydrostatic Tank

We will now discuss the problem of the hydrostatic tank. This problem might seem as trivial however it is of great importance. Many problems such as flow over a spillway, steady state flow in open channel, wave propagation etc, require a hydrostatic state in pressure as initial condition. However in SPH such condition cannot be set explicitly. Because of the weakly compressible nature of the method, the pressure is a function of density, where density is a result of the particle positions. Consequently, in order to set the initial hydrostatic pressure one must place the particles differently. However this cannot be predetermined. A method to overcome this problem is to start from a regular Cartesian grid (zero pressure) and let the fluid reach the desired state before starting simulating the actual problem. Note that computationally this is an expensive method. Violeau (2012) states, in the attempt to simulate flow in an open channel, the steady state from zero initial pressure, was achieved after 10^5 time steps.

We will test the method in the problem of the hydrostatic tank. The tank width is 1 meter and height 0.6 meters. The depth of the fluid is 0.4 meters. The parameter setup for the run is the following.

Table 33: Parameter setup in first run of hydrostatic tank

Smoothing length	4 cm
Discretization	1 cm
Number of fluid particles	5000
Boundary Discretization	1 mm
Number of boundary particles	2200
D coefficient of Lenard-Jones forces	0.4
Boundary cut-off distance	1.2cm
XSPH coefficient ε	0.5
a coefficient of artificial viscosity	0.3
Mass of fluid particles	0.01kg
Time step	0.0001 sec
Initial density	1000 kg/m^3
Artificial speed of sound	20 m/s

The boundary was simulated using dummy particles with hydrostatic pressure and Lenard-Jones repulsive forces according to equation (49). For the time integration we employed a Verlet scheme as presented in 2.10. Also the kernel function used is the quartic function proposed by Liu MB (2003) in equation (23). The fluid particles were placed in a regular Cartesian grid.

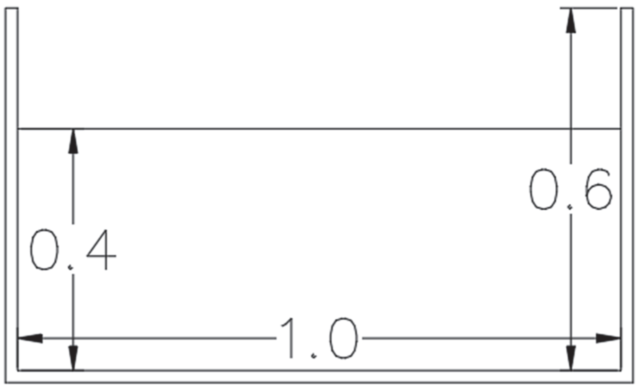


Figure 87: Tank dimensions for hydrostatic problem

Timeshots of the simulation are presented below:

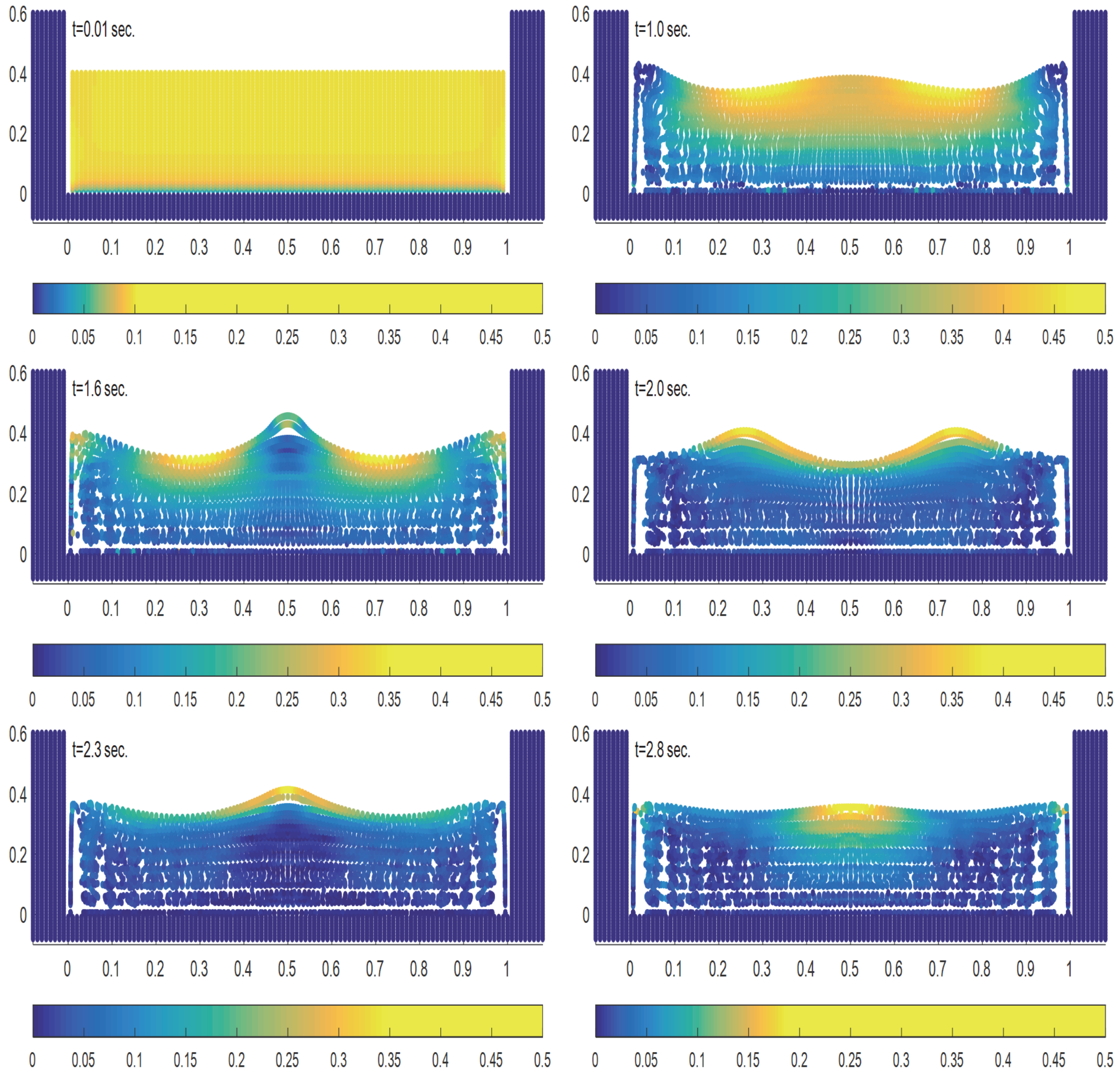


Figure 88: Snapshots of the simulation of settling hydrostatic tank. Colorbar denotes the norm of the velocity

When the simulation starts the gravity force instantaneously acts upon the fluid. When the fluid hits the bottom of the tank a standing wave is created that moves the surface of the water. We observe that even after 2.8 seconds of simulation time the fluid has not yet settled. The fluid settles after 5.4 seconds (not shown here) which is a very significant computational effort. For this demonstration problem the total run time was 9.5 hours. Usual SPH simulations involve one order of magnitude more fluid particles and smaller time steps. Therefore this approach is not adequate. To this aim Adami S. (2012) presented a damping term to progressively evolve the effect of gravity.

$$\zeta(t) = \begin{cases} 0.5\left[sin\left(\left(-0.5+\frac{t}{t_{damp}}\right)\pi\right)+1\right], & t \leq t_{damp} \\ 1, & t > t_{damp} \end{cases} \tag{68}$$

Where t_{damp} is the total damping simulation time.

Also a higher artificial viscosity factor α can be used during the damping time in order to diffuse the wave faster. Plots for different t_{damp} are presented below:

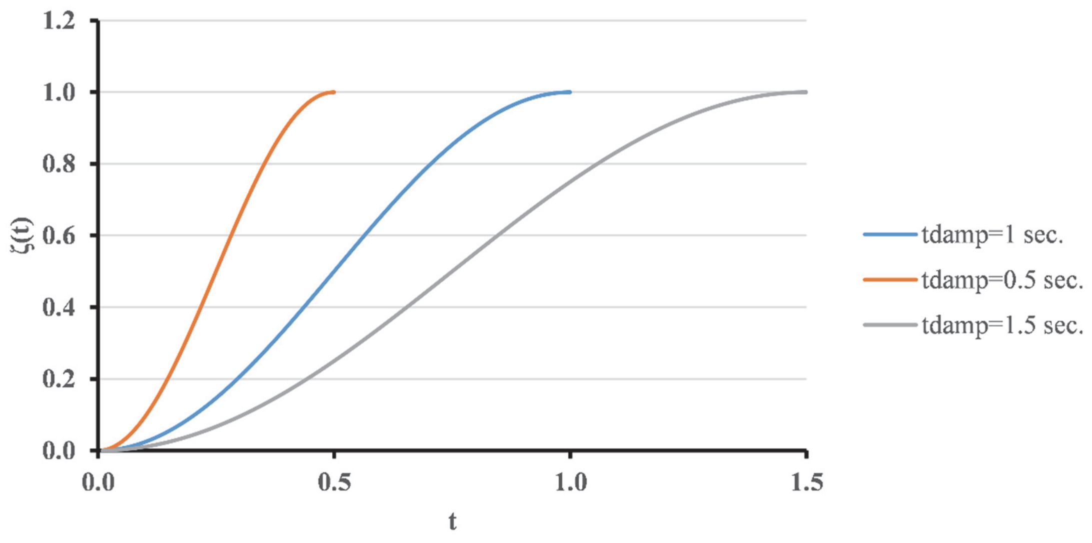


Figure 89: ζ dampening factor for different t_{damp} parameters

For the second run of the simulation we have chosen $tdamp$=1 s and α=0.5. The results are presented in the figure below. We observe immediately that the height of the stationary wave created is significantly smaller than that of the previous simulation. More importantly the fluid settles in around 3 seconds of simulation time saving precious computational time. It is important to note that in a problem with a wider tank or container the energy dissipation would be bigger, thus resulting in even smaller computational time to achieve hydrostatic initial conditions.

Another observation that we made is the fluid deformation. It can be seen that in both simulations the fluid settles for almost 30 cm, a 25% depth reduction. This is caused by the weak compressible assumption and is a function of the reference pressure $B = \frac{\rho_o c^2}{\gamma}$. The higher the speed of sound is set the smaller the final compression of the fluid will be. Consequently in the problems involving low speed of sound parameter, such as the problem we examined here, if we want a specific final height of the fluid we must take into account the fluid compressibility and the set the initial depth accordingly high.

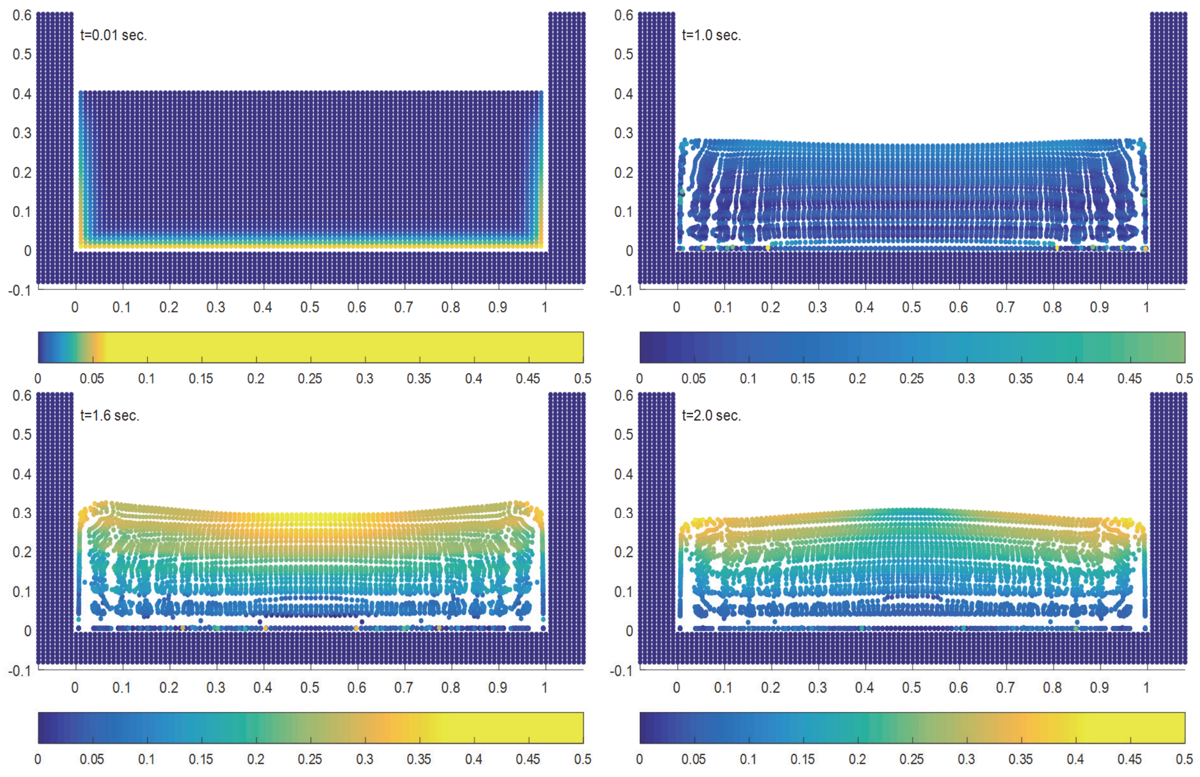


Figure 90: Snapshots of the simulation of settling hydrostatic tank using the dampening term. Colorbar denotes the norm of the velocity

5. Summary, Results and Discussion

5.1. Summary

In this thesis we have presented the Smoothed Particle Hydrodynamics computational method and implemented it on an algorithm developed in Matlab environment. This method is a purely lagrangian approach to the simulation of fluids and bears strong conservation properties. It utilizes the integral approximation of a function in order to discretize the governing partial differential equations i.e. the Euler equations of motion and continuity equation. Several implementations of this method were developed and compared with respect to the speed of calculations, keeping in mind that the interpreted nature of the Matlab environment reduces the performance significantly. In order to validate our algorithm the problem of a collapsing column inside a dry tank was adopted, representing the abstract model of a dam break. This problem is proposed by many researchers for code validation. Testing of our algorithm yielded satisfying results compared with experimental data and computational results from other researchers.

The main scope of the thesis was to analyze thoroughly the effect of various parameters of the method in the performance of the simulations. To this aim we created thirteen parameter sets and implemented them in the same validation problem. More specifically we tested three different discretizations, varying smoothing lengths, time steps, different time integration schemes, different boundary conditions and the effect of numerical treatment such as XSPH velocity correction and density filter. Thus we have covered the investigation of the main parameters of the method and deduced the optimal setup for each problem.

Furthermore we have implemented our algorithm with the knowledge gained from the previous simulations, in the problem of the hydrostatic tank. This might seem trivial, but comes across one of the main drawbacks of SPH, the difficulty to impose initial conditions. In this problem we have shown how one might impose hydrostatic pressure distribution to the particles.

5.2. Results

The performance of the method was quite satisfactory overall. We have demonstrated that using SPH one can simulate adequately the behavior of free surface flows. Regarding the problem of water column almost all of our parameter settings resulted in good agreement with the experimental data, as seen in figure 89.

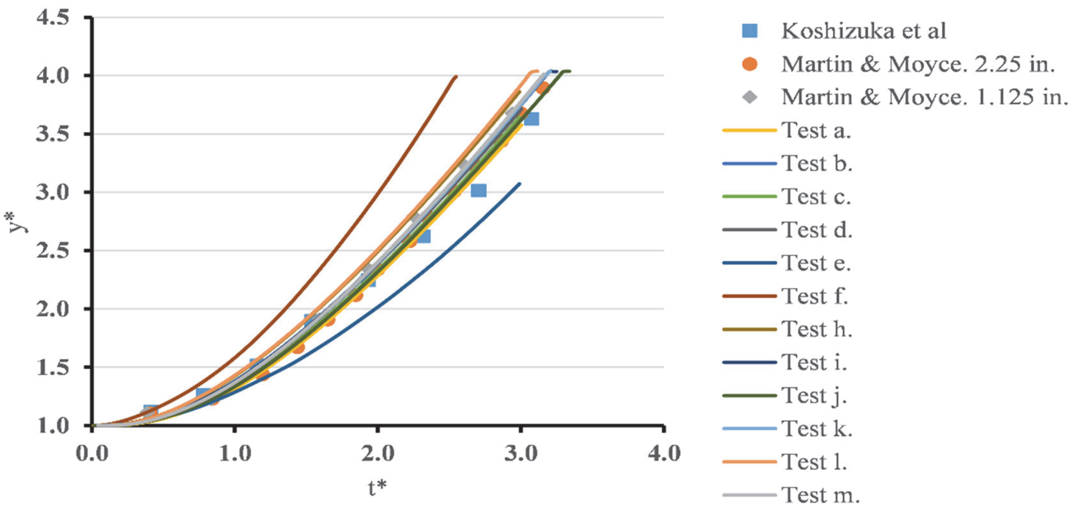


Figure 91: Toe evolution over dry bed for all tests (non-dimensional).

The only exceptions of adequate performance are the ones of tests € and (f), were the XSPH correction took into account the dummy boundary particles and the Lenard-Jones forces of equation 48 were used respectively.

Moreover we have demonstrated that even with a discretization of 2 cm we can obtain satisfactory results. However, we observed that the results are affected from the discretization as well as from the smoothing length. If we consider the foundation of SPH method, i.e. the integral approximation, the reason becomes clear. In the integral approximation the function is represented by an integral, which in our case is represented by a summation over the fluid particles. Therefore the accuracy of the method is a function of the number of particles inside the influence domain. The greater the number the better is the approximation. Consequently, the discretization cannot define the number of particles alone, therefore it is not necessarily an index of accuracy. The smoothing length and discretization are both necessary to define the accuracy of the method.

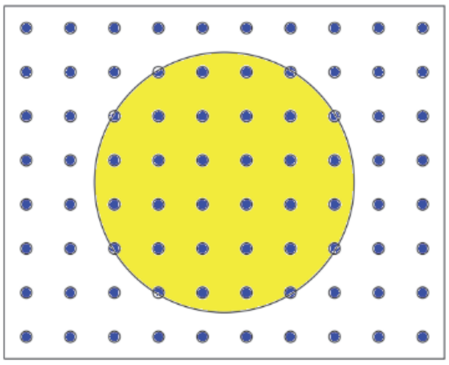


Figure 92: Influence domain of a kernel

The particle interpolation nature of SPH method has created some confusion among researchers who considered it as a Monte Carlo method. However, this is far from true and creates some serious implications. As described in the original paper by Metropolis N. (1949) the Monte Carlo method assumes a uniformly distributed random set of the variables taken into account, which in our case is the position of particles in space. In contrary the position of the particles in the SPH method is fully described deterministically by the discrete governing equations. Consequently we cannot claim that the SPH method belongs to the class of Monte Carlo methods. However, this deprives us from the theoretical tools to analyze the accuracy and errors of our discrete interpolation, which still remains an open problem Violeau (2012).

According to the above, we define the number of particles inside the influence domain as resolution. Then we calculate the corresponding measure for each discretization and smoothing length.

Table 34: Resolution

Discretization	Smoothing Length	Resolution
1 cm	2 cm	48
1.5 cm	4 cm	84
2 cm	2 cm	12
2 cm	3 cm	28

According to table 34 the resolution of test case (a) is almost double than the preliminary test, which explains the better performance of this setup and better agreement with the experimental data, as shown by the error estimates in table 35.

Table 35: Test a and preliminary test error comparison

	Resolution	Koshizuka et al	Martin & Moyce, 2.25 in.	Martin & Moyce, 1.125 in.
Test a.	84	0.138	0.086	0.147
Preliminary simulation	48	0.214	0.260	0.103
Test b.	12	0.194	0.226	0.091

Test c.	28	0.144	0.147	0.084

Next, we have demonstrated the behavior of the method when the time step is increased. In figures 91 and 92 we present a comparison of the results for different time steps, using the Verlet and the Predictor-Corrector time integration schemes.

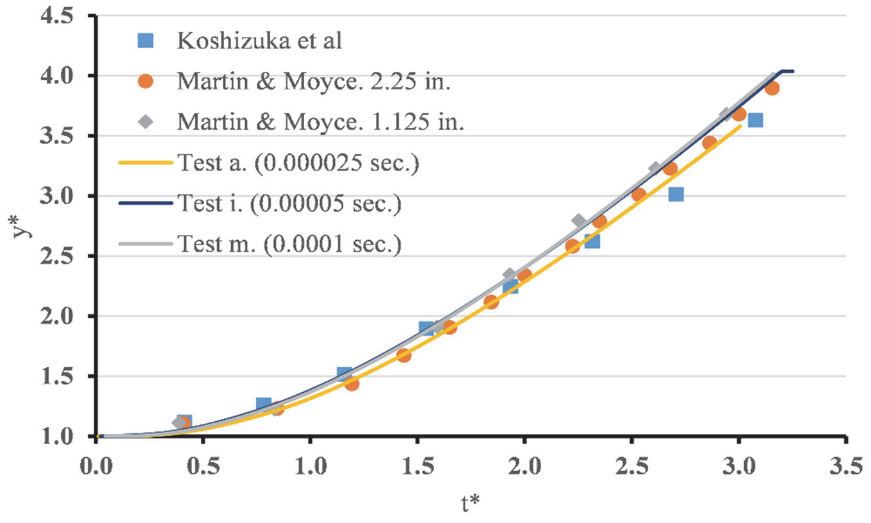


Figure 93: Toe evolution over dry bed for different time steps using Verlet scheme (non-dimensional).

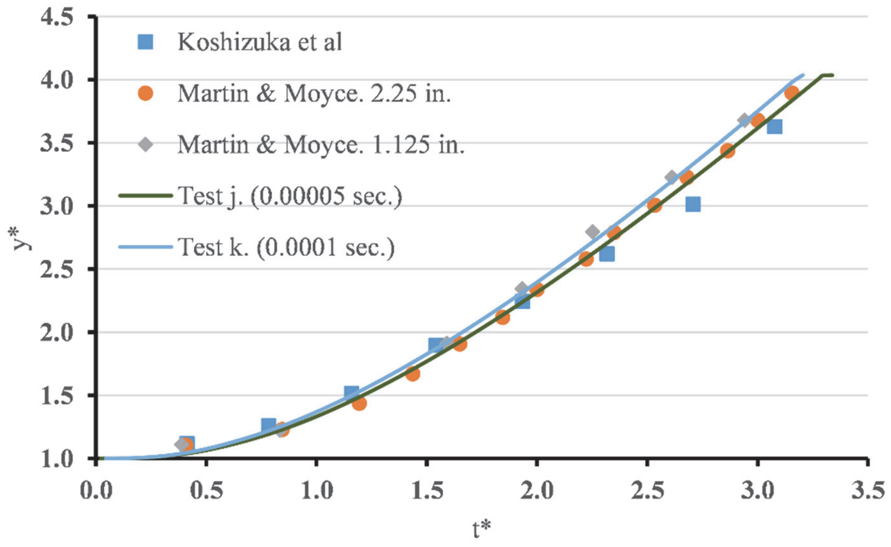


Figure 94: Toe evolution over dry bed for different time steps using Predictor – Corrector scheme (non-dimensional).

It is common ground for both time integration methods that when the time step increases, the results start to deviate upwards. However, if we compare the Predictor – Corrector scheme with time step of 0.00005 s and the Verlet scheme with one half of the previous time step, differences in performance are negligible. This performance though comes at a cost. As explained earlier, two calculations of the derivatives are needed for the Predictor–Corrector scheme, resulting in computational time equal to the Verlet scheme with one half time step. Consequently, use of more information for the time integration that gives an advantage to the method, is simultaneously a drawback for the computation.

As far as toe evolution is concerned for time step of 0.0001 s both methods perform equally. However that is not the case when focusing in the density distribution over the particles.

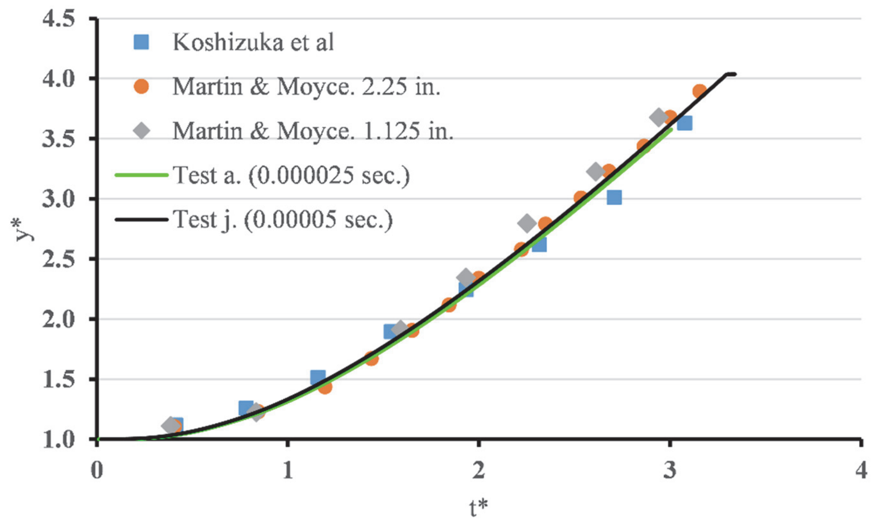


Figure 95: Toe evolution over dry bed for test a and test j (non-dimensional).

From the figure 94 it is evident that the time integration scheme affects the density distribution as well as the free surface. It is observed that the Predictor – Corrector (left column) yields densities closer to the preliminary case. Concluding, we may say that this time integration scheme is more accurate than the Verlet one, however it bears the serious disadvantage of high computational effort. Finally we recommend that this method may be used when it is possible.

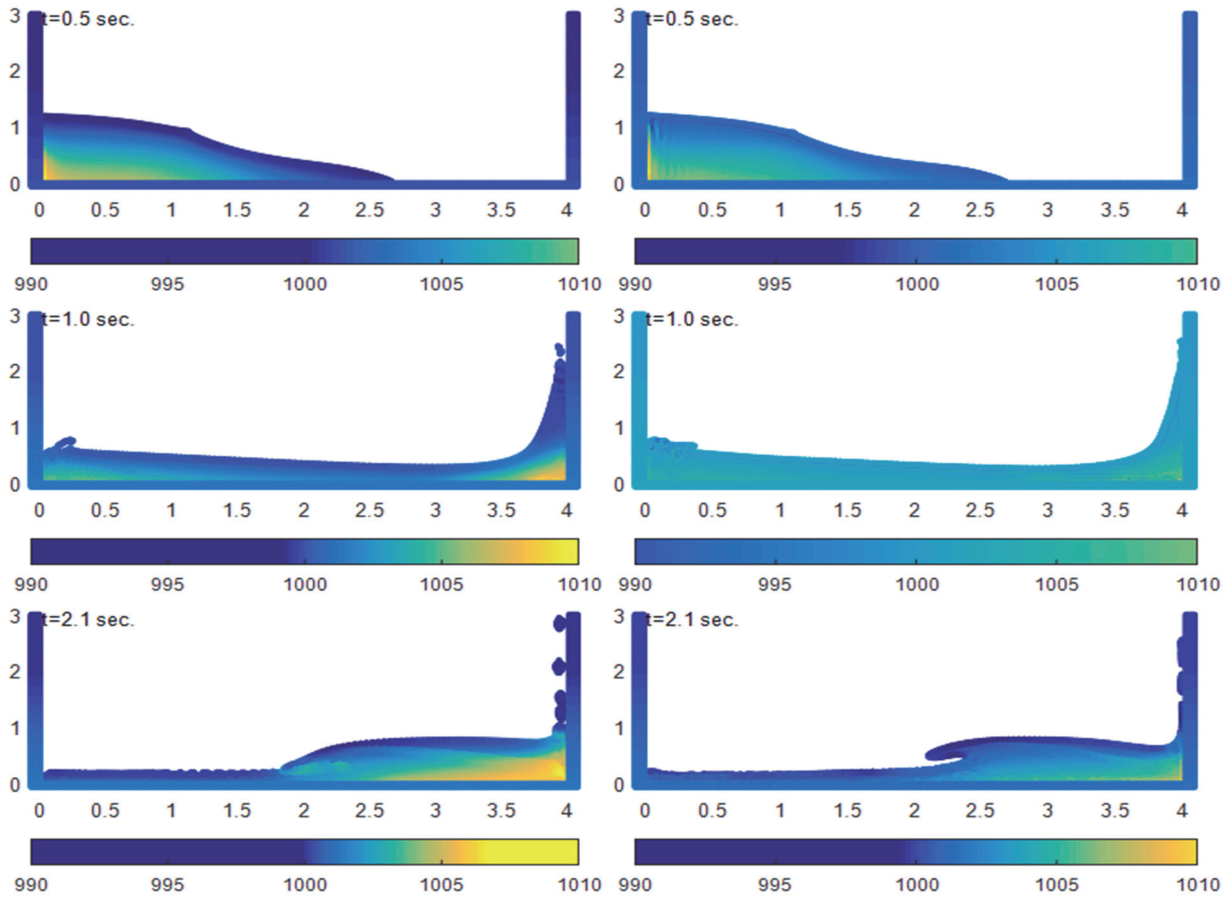


Figure 96: Comparison of density distribution for tests k (left column) and m (right column). Colorbar denotes density

During test (a) the effect of the XSPH velocity correction was thoroughly examined. As stated by Monaghan (1992b) this correction is used to keep the particles in order in the absence of viscosity and moves the particles with a velocity that is closer to the average velocity of the neighborhood. However, we have demonstrated that this correction can yield unnatural results in some flow phenomena.

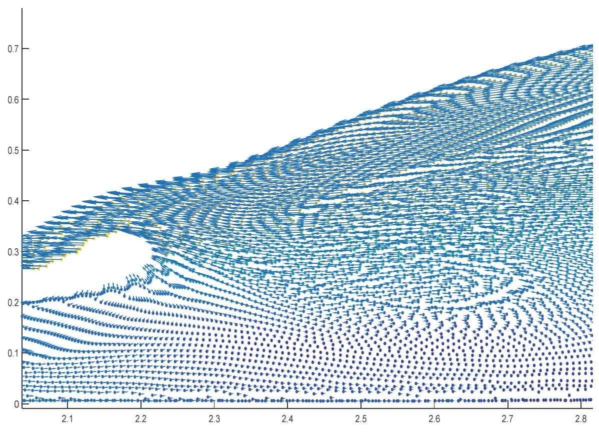


Figure 98: Plunging wave using XSPH

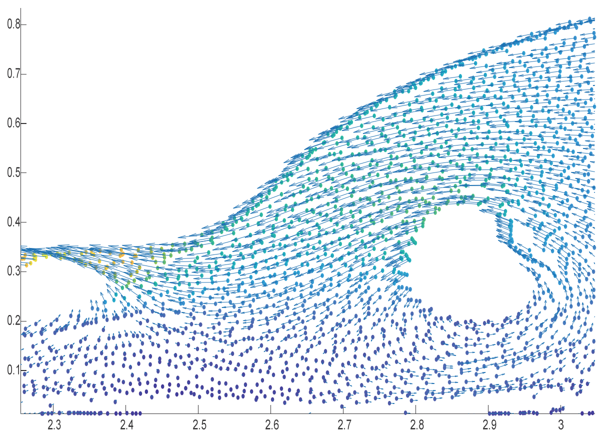


Figure 97: Plunging wave without XSPH

Consequently, from our point of view, these results indicate the need to incorporate a turbulence model in the method to incorporate the viscosity effect and eliminate the need for the XSPH velocity correction.

We have also examined the effect of different boundary conditions in the simulation. More specifically we have employed two versions of Lenard – Jones repulsive forces, the dummy boundary particles combined with Lenard – Jones, and the dynamic boundary particles method. We have concluded that use of the repulsive forces described by equation (48) is not sufficient to represent the boundary in problems were the driving force is gravity. Furthermore we have shown the advantages of using together the Lenard – Jones repulsive boundary and the dummy particles. From

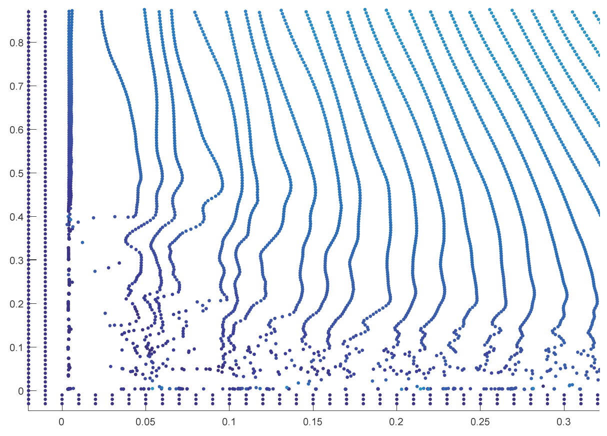


Figure 100: Use of Lenard – Jones repulsive forces

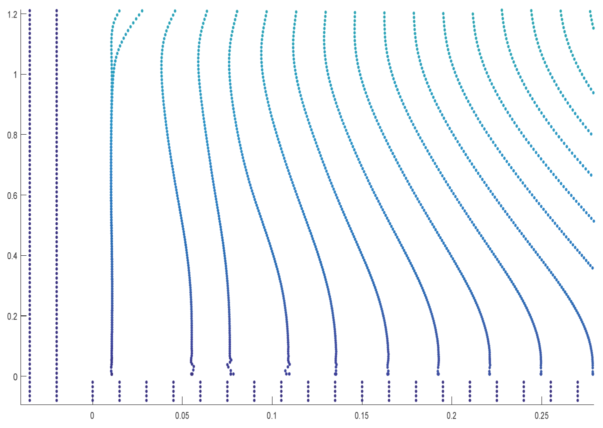


Figure 99: Use of Lenard – Jones and dummy particles

figures 97 and 98 it is evident that the complete kernel integration reduces effectively the disorders of the particle movement. The same results apply to the dynamic boundary particles. Moreover using this method we have achieved a better density distribution. It should be noted that our algorithm performed 40% slower than the former boundary treatment, when employing the dynamic boundary particles.

So far we have regarded no energy conservation. The energy of a system of particles is the sum of kinetic, dynamic and thermal energies.

$$E = \sum m_i\left(u_i^2 + v_i^2 + w_i^2\right) \tag{69}$$

$$D = \sum m_i g z_i \tag{70}$$

$$T = \sum m_i e_i \tag{71}$$

In figure 99 we plot these quantities normalized by the initial energy of the system for the preliminary test.

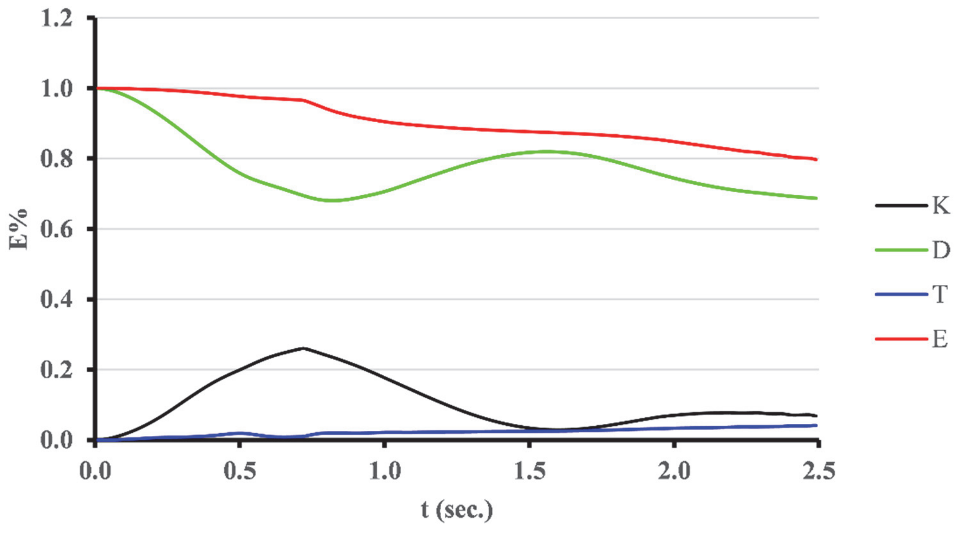


Figure 101: Figure evolution for test a

It is observed that the total energy of the system is reduced by 20% which may be the result of use of the Lenard – Jones forces. We can compare this result with test (d) where dynamic boundary particles were used.

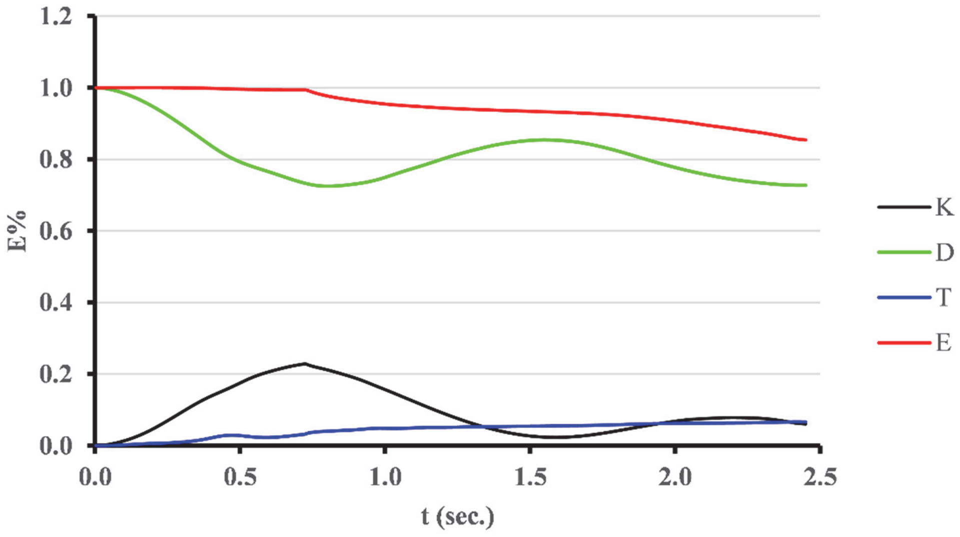


Figure 102: Energy evolution for test d

It appears that the boundary method affects the energy of the system. The energy is conserved up to 0.7 s in the simulation, in contrast with the preliminary case. The cause of this lies in the nature of the Lenard – Jones forces. These forces do not represent a natural force, but constitute a method to keep the particles inside the computational domain. Therefore the work they impose in the particles cannot be transformed in a different form of energy, thus a portion of the total energy is lost from the system. This can be demonstrated by the loss of dynamic energy in the two cases. If we add the loss of the total energy to the dynamic energy in the first case, we approximate efficiently the evolution of the dynamic energy in the second case, as it can be observed in figure 101.

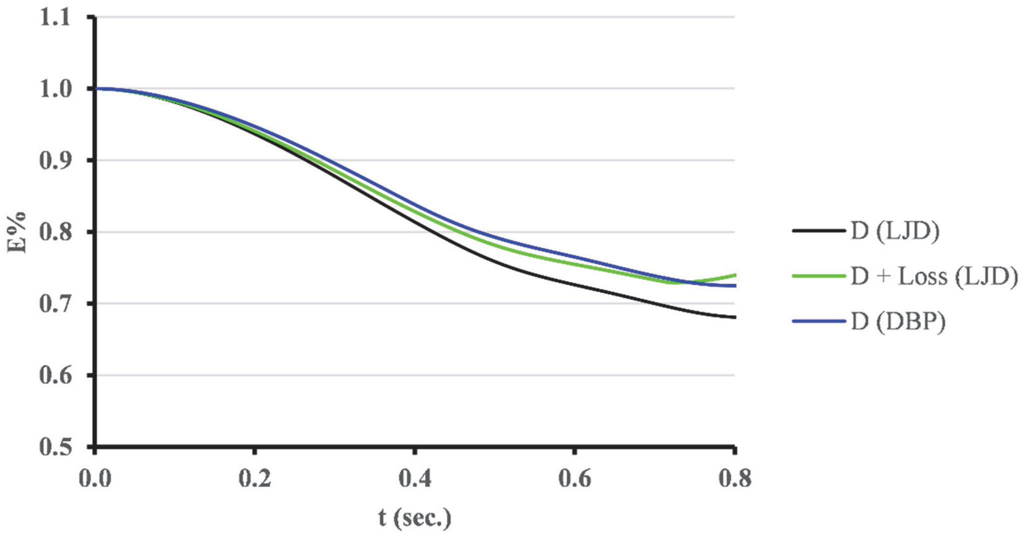


Figure 103: Dynamic energy evolution for Lenard – Jones and Dummy particles boundary (LJD), LJD and losses and dynamic boundary particles (DBP).

Despite the improvement in energy conservation in the first part of the simulation, at the end we end up with 15% total energy loss. It can be seen that after approximately 0.7 s of simulation time, there is a distinct change in the slope of the total energy curve vs time. This time is the time collision of fluid with the container wall takes place and the kinetic energy is higher. During the collision the flow should develop high turbulent stresses which would result in an increase of internal energy that does not happen in our simulation. It seems that the artificial viscosity we have used cannot reproduce this phenomenon. In a similar investigation Gomez-Gesteira (2010) have adopted a turbulence model to reproduce the effect of viscosity. Their results show a maximum 0.9 % change in the total energy of the system. Consequently we will attribute the loss of energy to the improper modeling of viscosity however, this assumption needs thorough testing.

5.3. Conclusions

Through our experience from this thesis we have derived some useful conclusions about the SPH method. In particular:

- We have shown that the SPH method can adequately simulate open boundary unsteady flows
- Large number of particles must be used in order to achieve a good level of accuracy
- The use of XSPH velocity correction must be reconsidered
- The compressibility must be considered when designing a problem with a desired flow depth
- The development of an SPH based algorithm is a fairly easy concept when compared to the development of grid-based algorithms
- It is a very expensive computational method as far as execution time is concerned

- The use of Matlab environment, although it offers great development advantages, is very slow to the extent of rendering the code useless in large scale problems.

5.4. Suggestions for future work

The driving force of this thesis was to gain experience in the method of Smoothed Particle Hydrodynamics and develop a code that will be the basis of future research in the field. It is the author's belief that such methods will be the future tools for free surface and multi-phase flows. With this context in mind and from the experience gained through our investigations we propose below some ideas about future work:

- Further optimization of the code in order to improve its speed and robustness. Translation to a compiled language is advised.
- Expansion of the model and code into 3D
- Implementation of a turbulence model and investigation of it's behavior along with XSPH velocity correction.
- Investigation of the connection between turbulence model and energy conservation
- Further investigation on the application of hydrostatic initial conditions
- Validation test of the method for problems where pressure is significant such us breakwater structures
- Validation and sensitivity analysis of the velocity flow field
- Expansion of the method to treat diffusion problems such as buoyant jets

Bibliography

ADAMI S. , H. X. Y., ADAMS N.A. 2012. A generalized wall boundary condition for smoothed particle hydrodynamics. *Journal of Computational Physics*

BENNETT, A. 2006. *Lagrangian Fluid Dynamics*, Cambridge University Press.

COLAGROSSI ANDREA, L. M. 2003. Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *Journal of Computational Physics*.

COLE, R. H. 1948. *Underwater Explosions,* Princeton Princeton University Press.

CRESPO A. J. C., G.-G. M., DALRYMPLE R. A. 2007. Boundary Conditions Generated by Dynamic Particles in SPH Methods. *Computers,Materials and Continua*.

DALRYMPLE R. A. , K. O. 2001. SPH Modelling of Water Waves *Coastal Dynamics*.

GINGOLD RA, M. J. 1977. Smoothed particle hydrodynamics- theory and aplication to non spherical stars. *Monthly Notices of the Royal Astronomical Society*.

GINGOLD RA , M. J. 1982. Kernel Estimates as a Basis for General Particle Methods in Hydrodynamics *Journal of Computational Physics*

GOMEZ-GESTEIRA, M. 2010. State-of-the-art of classical SPH for free-surface flows. *Journal of Hydraulic Research*.

HOPKINS, P. F. 2014. GIZMO: A New Class of Accurate, Mesh-Free Hydrodynamic Simulation Methods. *Monthly Notices of the Royal Astronomical Society*.

J.P., M. 1996. Analysis of Smoothed Particle Hydrodynamics with Aplicatons. Monash University.

KOSHIZUKA S. , O. Y. 1996. Moving-particle semi-implicit method for fragmentation of compressible fluid. *Nuclear science engineering*

LATTANZIO J. C. , M. J. J., PONGRACIC H., SCHWARZ M. P. 1986. Controlling Penetration *SIAM Journal of Scientific and Statistical Computations*.

LEIMKUHLER BENEDICT J., R. S., SKEEL ROBERT D. 1996. Integration Methods for Molecular Dynamics. *Mathematical Approaches to Biomolecular Structure and Dynamics.* Springer.

LIBERSKY L. D., P. A. G., CARNEY T. C, HIPP J. R. ALLAHDADI F. A 1993. High strain Lagrangian hydrodynamics-a three-dimensional SPH code for dynamic material response. *Journal of Computational Physics*.

LIBERSKY L. D., R. P. W. C. T. C. 1995. SPH calculations of fragmentation. *Proceedings of the 3rd U. S. Congress on Computational Mechanics*.

LIU G.R. , L. M. B. 2003. *Smoothed Particle Hydrodynamics, a Meshfree Particle Method*, World Scientific

LIU MB, L. G., LAM KY 2003. Constructing smoothing functions in smoothed particle hydrodynamics with aplications. *Journal of Computational Applied Mathematics*.

LUCY 1977. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*.

MARONE, S. Enhanced Boundary Treatment in 2D Smoothed Particle Hydrodynamic models.

MARTIN J.C. , M. W. J. 1952. An Experimental Study of the Collapse of Liquid Water Columns on a Rigid Horizontal Plane. *Philosophical Transactions of The Royal Society of London*.

METROPOLIS N. , U. S. 1949. The Monte Carlo method. *Journal of the American Statistical Association*.

MONAGHAN, J. 1989. On the Problem of Penetration in Particle Methods. *Journal of Computational Physics*.

MONAGHAN, J. 1992a. Simulating Free Surface Flows with SPH *Journal of Computational Physics*.

MONAGHAN, J. 1992b. Smoothed Particle Hydrodynamics. *Annual Review of Astronomy and Astrophysics*

MONAGHAN, J. J. 2005. Smoothed Particle Hydrodynamics. *Reports on Progress in Physics*

MONAGHAN JJ, L. J. 1985. A refined particle methd for astrophysical problems *Astronomy and Astrophysics*

MONAGHAN JJ. , K. A. 1999. Solitary waves on a Cretan beach. *Journal of Waterway, Port, Coastal, and Ocean Engineering*.

VERLET, L. 1967. Comyuter "Exyeriments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. *Physical Review*.

VIOLEAU, D. 2012. *Fluid Mechanics and the SPH Method*, Oxford Univrersity Press.

VIOLEAU D. , I. R. 2007. Numerical modelling of complex turbulent free-surface flows with the SPH method: an overview *International Journal for Numerical Methods in Fluids*.

WENDLAND, H. 1995. Piecewise polynomial, positive definite and compactly supported. *Advances in Computational Mathematics*.

Appendix A: List of Symbols

Variable	Description
h	Smoothing length
R	Relative non-dimensional distance
x	Position along x axis
y	Position along y axis
z	Position along z axis
u	Velocity along x axis
v	Velocity along y axis
w	Velocity along z axis
du/dt	Acceleration along x axis
dv/dt	Acceleration along y axis
dw/dt	Acceleration along z axis
g	Acceleration of gravity
ρ	Density
$d\rho/dt$	Density derivative
e	Thermal energy
de/dt	Thermal energy derivative
P	Pressure
m	Mass
W	Kernel function
L	Lagrangian of particle system
Π	Artificial viscosity
c	Artificial speed of sound
Δt	Time step

Appendix B: List of variables used in code

Variable	Explanation
po	Initial fluid density
co	Artificial speed of sound
mo	Particle mass
ho	Initial radious of influence
g	Gravitational acceleration
CF	Fluid particle coordinates
NF	Number of fluid particles
CB	Boundary particle coordinates
NB	Number of boundary particles
YB	y values of boundary particles
ZB	z values of boundary particles
N	Number of total fluid particles
YF	y values of fluid particles
ZF	z values of fluid particles
po	Initial particle density
p	Particles density
m	Particles mass
vo	Initial velocity of particles in y axis
v	Particles velocity in y axis
dvdt	Acceleration in y axis
dpdt	Density derivatives
e	Thermal energy
dedt	Derivative of thermal energy
P	Particles pressure
h	Smoothing length
DT	Time step
t	Cumulative time
tend	End of simulation time
visc	Dynamic viscosity
dr	Particle discretization
up	Particles velocity in x axis for previous step
vh	Particles velocity in y axis for previous step
pp	Particles density for previous step
ep	Thermal energy for previous step
psh	Corrected densities from Shepard filter
Gmaxy	Maximum grid coordinate along y axis
Gmaxz	Maximum grid coordinate along z axis
GY	y coordinates of search grid
GZ	z coordinates of search grid
GP	Grid position for each particle
s	Structure containing the particle ID's contained on each cell
uGP	Unique grid cells

pc	Adjacent cells
r	Distance between particles
k	Dimensionless distance between particles
yk	y coordinate of neighboring particles
zk	z coordinate of neighboring particles
rk	Distance between neighboring particles
hk	Smoothing length of neighboring particles
pk	Densities of neigboring particles
Pk	Pressure of neighboring particles
mk	Mass of neighboring particles
vk	Velocities along y axis of neighboring particles
wk	Velocities along z axis of neighboring particles
dWy	Kernel derivative along y axis
dWz	Kernel derivative along z axis
vdot	Dot product for velocities and positions
xdot	Dot product for positions
pbar	Average of pair particles
ArV	Artificial viscosity
YFn	New coordinate along y axis
ZFn	New coordinate along z axis
vn	New velocity along y axis
wn	New velocity along z axis
pn	New density
en	New thermal energy
XSPHY	XSPH velocity correction along y axis
XSPHZ	XSPH velocity correction along z axis
LJ	Lenard - Jones forces along y and z axis

Appendix C: Code

Main script of the SPH code:

```
clear;

%Initial values
po=1000;
visc=10^-3;
co=120;
ho=0.02;
g=9.81;
%Setting geometry
dr=0.02;
z=[dr:dr:2];
y=[dr:dr:1];
CF=combvec(y,z);
CF=transpose(CF);
IZF=CF(:,2);
NF=length(CF);
z1=[-0.08:dr:6];
y1=[-0.08:dr:-dr];
CF1=combvec(y1,z1);
CF1=transpose(CF1);
z2=[-0.08:dr:-dr];
y2=[0:dr:4];
CF2=combvec(y2,z2);
CF2=transpose(CF2);
y3=[4.01:dr:4.08];
CF3=combvec(y3,z1);
CF3=transpose(CF3);
CF=[CF;CF1;CF2;CF3];
mo=0.04;
b=transpose([0:0.001:4]);
c=0.*b;
backwz=transpose([0:0.001:6]);
backwy=-0.0*ones(size(backwz,1),1);
frontwz=transpose([0:0.001:6]);
frontwy=4.*ones(size(frontwz,1),1);
YB=[b;backwy;frontwy];
ZB=[c;backwz;frontwz];
NB=length(YB);
YF=CF(:,1);
ZF=CF(:,2);
N=length(YF);
m=ones(N,1)*mo;
vo=0;
v=ones(N,1)*vo;
w=zeros(N,1);
e=zeros(N,1);
dvdt=zeros(N,1);
dwdt=zeros(N,1);
dpdt=zeros(N,1);
dedt=zeros(N,1);
%p states derivatives of the previous time step
vp=zeros(N,1);
wp=zeros(N,1);
%pp=ones(N,1).*po;
```

```
ep=zeros(N,1);
P=[zeros(NF,1);po.*g.*(2-ZF(NF+1:N))];
P=(ZF<=2).*P;
p=[ones(NF,1)*po;po.*(7*P(NF+1:N)./co^2./po+1).^(1/7)];
pp=[ones(NF,1)*po;po.*(7*P(NF+1:N)./co^2./po+1).^(1/7)];
psh=zeros(N,1); %Corrected from Shepard Filter
h=ones(N,1)*ho;
t=zeros(10^8,1);
DT=0.000025;
t(1)=DT;
tend=2.5;
it=1;

while t(it)<=tend

    Gmaxz=round(max(ZF(1:NF))+10*ho,1);
    Gmaxy=round(max(YF(1:NF))+10*ho,1);
    [GY,GZ]=meshgrid(-0.09:2*ho+0.01:Gmaxy,-0.09:2*ho+0.01:Gmaxz);
    GP=zeros(N,1);
    l2=size(GY,2); %l
    l1=size(GY,1); %h
    s=struct('a',zeros(l1*l2,1));
    npic=zeros(l1*l2,1); %number of particles in cell

    for i=1:l1-1
        for j=1:l2-1
            I=find(and(YF>GY(i+1,j) & YF<=GY(i+1,j+1),ZF>GZ(i,j+1) &
ZF<=GZ(i+1,j+1)));
            GP(I)=(i-1)*(l2-1)+j;
            s((i-1)*(l2-1)+j).a=struct('b',I);
            %npic((i-1)*(l2-1)+j)=size(I,1);
        end
    end
    s((i-1)*(l2-1)+j+1).a=struct('b',[]);
    ii=(i-1)*(l2-1)+j+1;
    uGP=unique(GP);
    if uGP(1)==0
        uGP(1)=[];
    end
    pc=zeros(size(uGP,1),9);
    ipc=zeros(max(GP),1);
    for i=1:size(uGP,1)
        gp=uGP(i);
        ipc(gp)=i;
        pc(i,1)=gp;
        gate=gp/(size(GY,2)-1)==int32(gp/(size(GY,2)-1));
        pc(i,2)=min((1-gate)*(gp+1),(l1-1)*(l2-1));
        pc(i,2)=(pc(i,2)>gp)*pc(i,2);
        gate=(gp-1)/(size(GY,2)-1)==int32((gp-1)/(size(GY,2)-1));
        pc(i,3)=(1-gate)*(gp-1);
        pc(i,4)=((gp+l2-1)/(l2-1)<(l1-1))*(gp+l2-1);
        pc(i,5)=((gp-(l2-1))/(l2-1)>0)*(gp-(l2-1));
        pc(i,6)=(pc(i,2)>0)*(pc(i,4)>0)*(gp+1+l2-1);
        pc(i,7)=(pc(i,2)>0)*(pc(i,5)>0)*(gp+1-(l2-1));
        pc(i,8)=(pc(i,3)>0)*(pc(i,4)>0)*(gp-1+l2-1);
        pc(i,9)=(pc(i,3)>0)*(pc(i,5)>0)*(gp-1-(l2-1));
    end
    jj=find(pc==0);
    pc(jj)=ii;
    % Type I Boundary Particles
    LJ=TypeIBoundary2D(YF,ZF,YB,ZB);
    maxfij=zeros(N,1);
    parfor i=1:NF
        gp=GP(i);
        I0=pc(ipc(gp),:);

I1=[s(I0(1)).a.b;s(I0(2)).a.b;s(I0(3)).a.b;s(I0(4)).a.b;s(I0(5)).a.b;s(I0(6
)).a.b;s(I0(7)).a.b;s(I0(8)).a.b;s(I0(9)).a.b];
        r=sqrt((YF(I1)-YF(i)).^2+(ZF(I1)-ZF(i)).^2);
        k=r./h(i);
        a=find(k<=2);
        k=k(a);
        I=I1(a);
        ii=find(k==0);
        yk=YF(I);
        zk=ZF(I);
        rk=r(a);
        hk=h(I);
        pk=p(I);
        Pk=P(I);
        mk=m(I);
        vk=v(I);
        wk=w(I);
        dWy=15/(56*pi()*h(i)^3)*(-18*k+19*k.^2-5*k.^3).*(-
yk+YF(i))./(rk+0.0001);
        dWz=15/(56*pi()*h(i)^3)*(-18*k+19*k.^2-5*k.^3).*(-
zk+ZF(i))./(rk+0.0001);
        %Artificial viscosity
        vdot=((v(i)-vk).*(YF(i)-yk)+(w(i)-wk).*(ZF(i)-zk));
        xdot=(YF(i)-yk).^2+(ZF(i)-zk).^2;
        pbar=(p(i)+pk)./2;
        ArV=(-0.3*co*h(i).*(vdot)./(xdot+0.01.*h(i).^2))./(pbar+0.0001);
        ArV=(I<=NF).*(vdot<0).*ArV;
        maxfij(i)=max(abs(h(i).*(vdot)./(xdot)));
        partsum=(P(i)/p(i)^2+Pk./pk.^2).*dWy+ArV.*dWy;%
        partsum=mk.*partsum;
        dvdt(i)=-sum(partsum)+LJ(i,1);

        partsum=(P(i)/p(i)^2+Pk./pk.^2).*dWz+ArV.*dWz;%
        partsum=mk.*partsum;
        dwdt(i)=-sum(partsum)-g+LJ(i,2);

        vwdot=(v(i)-vk).*dWy+(w(i)-wk).*dWz;

        partsum=(P(i)/p(i)^2+Pk./pk.^2).*dWz+ArV.*dWz;%
        partsum=mk.*partsum.*vwdot;
        dedt(i)=0.5.*sum(partsum);

        partsum=mk.*vwdot;
        dpdt(i)=sum(partsum);

    end

    %Verlet Solver
    if it/50==int32(it/50)
        YFn=YF+v*DT+0.5*dvdt*DT^2;
        ZFn=ZF+w*DT+0.5*dwdt*DT^2;
        vn=v+dvdt*DT;
        wn=w+dwdt*DT;
        pn=p+dpdt*DT;
        en=e+dedt*DT;
    else
        YFn=YF+v*DT+0.5*dvdt*DT^2;
        ZFn=ZF+w*DT+0.5*dwdt*DT^2;
        vn=vp+2*dvdt*DT;
        wn=wp+2*dwdt*DT;
        pn=pp+2*dpdt*DT;
        en=ep+2*dedt*DT;
    end

    %XSPH
    XSPHY=zeros(N,1);
    XSPHZ=zeros(N,1);

    parfor i=1:NF
        gp=GP(i);
        I0=pc(ipc(gp),:);

I1=[s(I0(1)).a.b;s(I0(2)).a.b;s(I0(3)).a.b;s(I0(4)).a.b;s(I0(5)).a.b;s(I0(6
)).a.b;s(I0(7)).a.b;s(I0(8)).a.b;s(I0(9)).a.b];
        I1(I1>NF)=[];
        r=sqrt((YF(I1)-YF(i)).^2+(ZF(I1)-ZF(i)).^2);
        k=r./h(i);
        a=find(k<=2);
        k=k(a);
        I=I1(a);
        pk=pn(I);
        mk=m(I);
        vk=vn(I);
        wk=wn(I);
        W=15/7/(pi()*h(i)^2)*(2/3-9/8*k.^2+19/24*k.^3-5/32*k.^4);
        vji=vk-vn(i);
        wji=wk-wn(i);
        pbar=(pn(i)+pk)./2;
        partsum=mk.*vji.*W./(pbar+0.0001);
        XSPHY(i)=sum(partsum);
        partsum=mk.*wji.*W./(pbar+0.0001);
        XSPHZ(i)=sum(partsum);
    end

    vn=vn+0.5*XSPHY;
    wn=wn+0.5*XSPHZ;

    vp=v;
    wp=w;
    pp=p;
    ep=e;

    p=pn;
    v=vn;
    w=wn;
    e=en;
    YF=YFn;
    ZF=ZFn;

    if it/30==int32(it/30)
        parfor i=1:NF
            r=sqrt((YF(1:NF)-YF(i)).^2+(ZF(1:NF)-ZF(i)).^2);
            k=r./h(i);
            I=find(k<=2);
            k=k(I);
            ii=find(k==0);
            pk=pn(I);
            mk=m(I);
            vk=vn(I);
            wk=wn(I);
            W=15/7/(pi()*h(i)^2)*(2/3-9/8*k.^2+19/24*k.^3-5/32*k.^4);
            %Shepard filter
            Wbar=W./sum(W.*mk./pk);
            p(i)=sum(mk.*Wbar);
        end
    end
    P=co^2*po/7*((p./po).^7-1);


    if it/100==int32(it/100)

        figure(it)
        scatter(YF,ZF,100,v,'.')
        hold on
        scatter(YB,ZB,'.')
        hold off

        figure(it+1)
        scatter(YF,ZF,100,P,'.')
        hold on
        scatter(YB,ZB,'.')

        %arxeia eksodou
        out=[YF,ZF,v,w,dvdt,dwdt,p,P,h];
        filename=sprintf('%d.mat',it);

save(filename,'YF','ZF','v','w','e','dvdt','dwdt','dedt','p','P','h')
        it
    end

    it=it+1;
    t(it)=t(it-1)+DT;

end
```

Script computing the Lenard – Jones Boundary forces

```
function LJ=TypeIBoundary2D(YF,ZF,YB,ZB,D,ro)
NF=length(YF);
NB=length(YB);
P1=2;
P2=4;
LJ=zeros(NF,2); %Lennard-Jones repulsive force
parfor i=1:NF
    r=sqrt((YB-YF(i)).^2+(ZB-ZF(i)).^2);
    c=bsxfun(@le,r,ro);
    I=find(c==1);
    npb=size(I,1); %number of proximate boundary particles for L-J
    if npb~=0
        cpb=zeros(npb,2);
        cpb(:,1)=YB(I);
        cpb(:,2)=ZB(I);
        XIJ=[cpb(:,1)-YF(i),cpb(:,2)-ZF(i)];
        rij=sqrt((cpb(:,1)-YF(i)).^2+(cpb(:,2)-ZF(i)).^2);
        ljtemp=D*((ro./rij).^P1-(ro./rij).^P2)./rij;
        lj=[ljtemp,ljtemp].*XIJ;
        LJ(i,:)=sum(lj);
    end
end
end
```

Script creating the subplots

```
ymin=-0.08;
ymax=4.08;
zmin=-0.1;
zmax=3;
ytext=0.01;
ztext=2.95;
cmax=6;
load('20000.mat')
subaxis(3,2,1,'SpacingVert',0,'SpacingHoriz',0.04,'MarginTop',0.01,'MarginB
ottom',0);
scatter(YF,ZF,100,sqrt(v.^2+w.^2),'.')
grid off
axis([ymin ymax zmin zmax])
text(ytext,ztext,'t=0.5 sec.')
c=colorbar('southoutside','Limits',[0,cmax]);
load('27000.mat')
subaxis(3,2,2,'SpacingVert',0,'SpacingHoriz',0.04,'MarginTop',0.01,'MarginB
ottom',0);
scatter(YF,ZF,100,sqrt(v.^2+w.^2),'.')
grid off
axis([ymin ymax zmin zmax])
text(ytext,ztext,'t=0.675 sec.')
c=colorbar('southoutside','Limits',[0,cmax]);
load('40000.mat')
subaxis(3,2,3,'SpacingVert',0,'SpacingHoriz',0.04,'MarginTop',0.01,'MarginB
ottom',0);
scatter(YF,ZF,100,sqrt(v.^2+w.^2),'.')
grid off
axis([ymin ymax zmin zmax])
text(ytext,ztext,'t=1.0 sec.')
c=colorbar('southoutside','Limits',[0,cmax]);
load('72000.mat')
subaxis(3,2,4,'SpacingVert',0,'SpacingHoriz',0.04,'MarginTop',0.01,'MarginB
ottom',0);
scatter(YF,ZF,100,sqrt(v.^2+w.^2),'.')
grid off
axis([ymin ymax zmin zmax])
text(ytext,ztext,'t=1.8 sec.')
c=colorbar('southoutside','Limits',[0,cmax]);
load('84000.mat')
subaxis(3,2,5,'SpacingVert',0,'SpacingHoriz',0.04,'MarginTop',0.01,'MarginB
ottom',0);
scatter(YF,ZF,100,sqrt(v.^2+w.^2),'.')
grid off
axis([ymin ymax zmin zmax])
text(ytext,ztext,'t=2.1 sec.')
c=colorbar('southoutside','Limits',[0,cmax]);
load('98000.mat')
subaxis(3,2,6,'SpacingVert',0,'SpacingHoriz',0.04,'MarginTop',0.01,'MarginB
ottom',0);
```

```
scatter(YF,ZF,100,sqrt(v.^2+w.^2),'.')
grid off
axis([ymin ymax zmin zmax])
text(ytext,ztext,'t=2.45 sec.')
c=colorbar('southoutside','Limits',[0,cmax]);
```

Subaxis function is created by Aslak Grinsted

Script creating the density subplots

```
ymin=-0.08;
ymax=4.08;
zmin=-0.1;
zmax=3;
ytext=0.01;
ztext=2.95;
cmin=990;
cmax=1010;
load('20000.mat')
subaxis(3,2,1,'SpacingVert',0,'SpacingHoriz',0.04,'MarginTop',0.01,'MarginB
ottom',0);
scatter(YF,ZF,100,p,'.')
grid off
axis([ymin ymax zmin zmax])
text(ytext,ztext,'t=0.5 sec.')
c=colorbar('southoutside','Limits',[cmin,cmax]);
load('27000.mat')
subaxis(3,2,2,'SpacingVert',0,'SpacingHoriz',0.04,'MarginTop',0.01,'MarginB
ottom',0);
scatter(YF,ZF,100,p,'.')
grid off
axis([ymin ymax zmin zmax])
text(ytext,ztext,'t=0.67 sec.')
c=colorbar('southoutside','Limits',[cmin,cmax]);
load('40000.mat')
subaxis(3,2,3,'SpacingVert',0,'SpacingHoriz',0.04,'MarginTop',0.01,'MarginB
ottom',0);
scatter(YF,ZF,100,p,'.')
grid off
axis([ymin ymax zmin zmax])
text(ytext,ztext,'t=1.0 sec.')
c=colorbar('southoutside','Limits',[cmin,cmax]);
load('72000.mat')
subaxis(3,2,4,'SpacingVert',0,'SpacingHoriz',0.04,'MarginTop',0.01,'MarginB
ottom',0);
scatter(YF,ZF,100,p,'.')
grid off
axis([ymin ymax zmin zmax])
text(ytext,ztext,'t=1.8 sec.')
c=colorbar('southoutside','Limits',[cmin,cmax]);
load('84000.mat')
subaxis(3,2,5,'SpacingVert',0,'SpacingHoriz',0.04,'MarginTop',0.01,'MarginB
ottom',0);
scatter(YF,ZF,100,p,'.')
grid off
axis([ymin ymax zmin zmax])
text(ytext,ztext,'t=2.1 sec.')
c=colorbar('southoutside','Limits',[cmin,cmax]);
load('98000.mat')
```

```
subaxis(3,2,6,'SpacingVert',0,'SpacingHoriz',0.04,'MarginTop',0.01,'MarginB
ottom',0);
scatter(YF,ZF,100,p,'.')
grid off
axis([ymin ymax zmin zmax])
text(ytext,ztext,'t=2.45 sec.')
c=colorbar('southoutside','Limits',[cmin,cmax]);
```

Script that creates the gif pictures

```
filename='ColumnCollapse_4.gif';
figure
for n = 1300:100:47000
    load(sprintf('%d.mat',n));

scatter(YF(1:20000),ZF(1:20000),100,sqrt(v(1:20000).^2+w(1:20000).^2),'.')
    c=colorbar('southoutside','Limits',[0,12]);
    c.Label.String = 'v (m/s)';
    hold on
    scatter(YB,ZB,'.')
    hold off
    drawnow
    frame = getframe(1);
    im = frame2im(frame);
    %[A,map] = rgb2ind(im,256);
    [A(:,:,1,n),map]=rgb2ind(im,256);
    %{
    if n == 100;
        imwrite(A,map,filename,'gif','LoopCount',Inf,'DelayTime',0.0025);
    else

imwrite(A,map,filename,'gif','WriteMode','append','DelayTime',0.0025);
    end
    %}
end
imwrite(A,map,filename,'gif','LoopCount',Inf,'DelayTime',0.0025);
```