INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2008; **58**:625–638
Published online 17 March 2008 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1761

Defining and optimizing algorithms for neighbouring particle identification in SPH fluid simulations

G. Viccione[1,*,†], V. Bovolin[1] and E. Pugliese Carratelli[2]

[1]*Dipartimento di Ingegneria Civile, Università degli Studi di Salerno, Via Ponte don Melillo, 84084 Fisciano, Italy*
[2]*C.U.G.RI.-Piazza Vittorio Emanuele, 84080 Penta di Fisciano, Italy*

SUMMARY

Lagrangian particle methods such as smoothed particle hydrodynamics (SPH) are very demanding in terms of computing time for large domains. Since the numerical integration of the governing equations is only carried out for each particle on a restricted number of neighbouring ones located inside a cut-off radius r_c, a substantial part of the computational burden depends on the actual search procedure; it is therefore vital that efficient methods are adopted for such a search.

The cut-off radius is indeed much lower than the typical domain's size; hence, the number of neighbouring particles is only a little fraction of the total number. Straightforward determination of which particles are inside the interaction range requires the computation of all pair-wise distances, a procedure whose computational time would be unpractical or totally impossible for large problems.

Two main strategies have been developed in the past in order to reduce the unnecessary computation of distances: the first based on dynamically storing each particle's neighbourhood list (Verlet list) and the second based on a framework of fixed cells.

The paper presents the results of a numerical sensitivity study on the efficiency of the two procedures as a function of such parameters as the Verlet size and the cell dimensions. An insight is given into the relative computational burden; a discussion of the relative merits of the different approaches is also given and some suggestions are provided on the computational and data structure of the neighbourhood search part of SPH codes. Copyright © 2008 John Wiley & Sons, Ltd.

Received 19 September 2006; Revised 29 September 2007; Accepted 2 October 2007

KEY WORDS: SPH; Lagrangian methods; Verlet list; cell-linked list

1. INTRODUCTION

Meshless methods such as the—perhaps inappropriately named—smoothed particle hydrodynamics (SPH) have recently become very popular in fluid dynamics.

*Correspondence to: G. Viccione, Dipartimento di Ingegneria Civile, Università degli Studi di Salerno, Via Ponte don Melillo, 84084 Fisciano, Italy.
†E-mail: gviccion@unisa.it

While SPH originated in the context of Astrophysics [1, 2], very soon it found its early applications in fluid mechanics; a recent textbook written by Liu and Liu [3] provides an extensive coverage of its basic principles.

Only recently, however, meshless methods seem to have reached their full maturity in fluid mechanics [4]; the main advantage of such methods arises directly from their Lagrangian nature, in that they do not require a computational grid, thus avoiding burdensome mesh changes when the fluid domain evolves in time; they therefore appear to be particularly well suited to simulate free surface flows [5, 6]. Their applications have been extended to the most diverse situations such as dam break [7–9], wave swash zone hydrodynamics [10–12] and turbulence free surface flows [13–16].

A comprehensive review of SPH theoretical aspects in fluid dynamics was published recently by Monaghan [17], so they need not be discussed here.

Here we are concerned with one particular aspect of SPH and similar approaches, i.e. the handling of data and the neighbour search techniques. In these Lagrangian methods, each particle represents the properties of a small volume of moving fluid and interacts with a number of adjacent particles N_n (local neighbourhood) included within a short range; such a number is naturally much lower than the total number of particles N_{tot} considered in the problem. For instance, for 2D problems Oger *et al.* [18] have shown that a cut-off distance r_{c} encompassing at least 20 neighbouring particles provides acceptable results, that is $r_{\text{c}}/2d_0 = 1.23$, d_0 being the average distance among the particles.

A generic scalar or vector field f_i at a given spatial position $\mathbf{r}_i$ (vectors are in bold) taken at the time t by the fluid particle i is given by a smooth interpolation from the values f_k carried by those particles within the interaction sphere of radius $r_{\text{c}} = 2h$ [4]:

$$f_k = \sum_{k=1}^{N_n} f_k \cdot W(|\mathbf{r}_i - \mathbf{r}_k|, h) \tag{1}$$

where W is the so-called kernel or weighting function and h is known as the smoothing length. We name as *neighbouring particles* or simply as *neighbourhood* those giving effective contribution in Equation (1). Here the cubic spline known as B-spline function devised by Monaghan and Lattanzio [19] is adopted:

$$W(q) = A_n(n_d) \begin{cases} \frac{2}{3} - q^2 + \frac{1}{2}q^2, & 0 \leqslant q < 1 \\ \frac{1}{6}(2-q)^3, & 1 \leqslant q < 2 \\ 0, & q \geqslant 2 \end{cases} \tag{2}$$

where A_n is a normalization coefficient that is a function of the number of dimensions n_d, q is the relative distance between two particles at locations $\mathbf{r}_i$ and $\mathbf{r}_k$, that is $q = |r_i - r_k|/h$.

The total CPU time τ_{tot} in an SPH computation may be considered as the sum of the integration time τ_{int} for the numerical integration in a strict sense (calculating particles' interaction and fluid dynamical parameters) and the search time τ_{s} for the identification of neighbouring particles.

As N_{tot} increases, τ_{s} becomes more and more important, no matter how fast or how highly parallelized the computing system is. It is evident that simulations with a large number of particles are only possible if an efficient neighbourhood search algorithm is employed.

The simplest approach to define the neighbourhood is to perform a brute force evaluation of Equation (1), i.e. by checking for each of the N_{f} moving particles and for each time step, the distance with all the other $N_{\mathrm{tot}}-1$ particles, and by selecting only those within a cut-off distance r_{c} as Figure 1 shows. Here $N_{\mathrm{tot}}=N_{\mathrm{f}}+N_{\mathrm{b}}$, N_{b} being the number of boundary particles. The number of operations needed is then $N_{\mathrm{f}}(N_{\mathrm{tot}}-1)/2$ since the distance between each pair needs to be computed just once. Such an algorithm is easy to implement, but its efficiency is so poor that it has never been used except perhaps for very simple problems at the very beginning of the method.

An important step further is the idea to limit the neighbouring particle definition by storing a record of 'probable' neighbouring particles for each moving particle i; such a bookkeeping list, which we will refer to in the following as 'Verlet' list [20] contains the $N_{\mathrm{V},i}$, $i=1,\ldots,N_{\mathrm{f}}$, neighbours within a radius r_{v} slightly greater than the cut-off distance r_{c} (Figure 2). The condition

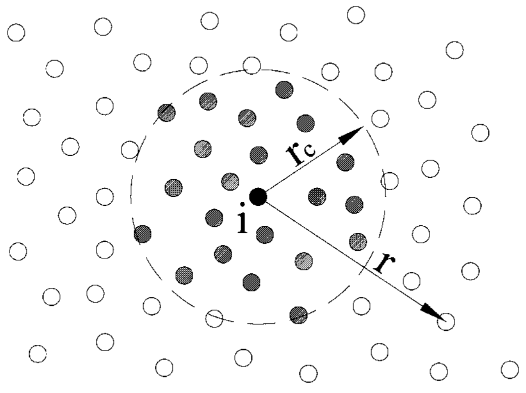


Figure 1. Neighbourhood definition for the fluid particle 'i' by checking the distances with the remaining '$N_{\mathrm{tot}}-1$' particles.

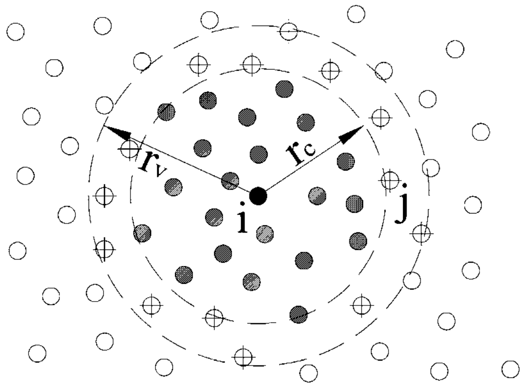


Figure 2. Verlet list definition for the fluid particle 'i'; particles marked with a cross between r_{c} and r_{v} are included in the list but they do not give any contribution to the hydrodynamics properties of the targeted particle.

$N_{\mathrm{V},i} \ll N_{\mathrm{tot}}$, will hold, as long as r_{v} is small enough, even though—obviously—the number of particles stored in the list is greater than those which are required to compute Equation (1). The difference $r_{\mathrm{v}} - r_{\mathrm{c}} = V_{\mathrm{skin}}$ represents a sort of safety 'skin' around the cut-off distance r_{c} and contains some elements which are initially unnecessary for the definition of the ith particle's properties, as shown by the particles marked with a cross in Figure 2. This takes into account the possibility that some or all of them will cross the interaction sphere and thus become part of the neighbourhood in the following time steps.

With this approach the list is kept unchanged for some time steps, until a 'refresh condition' is established; the computational burden of such an operation is of order $O(N_{\mathrm{f}} N_{\mathrm{tot}})$, i.e. the same as a whole domain neighbour search procedure, but it does not need to be performed at each integration time step. If the list refresh were performed too often, there would be no advantage over the simple approach; on the other hand, if too many integration time steps were carried out between successive refresh operations, there would be a chance of errors deriving from a particle external to the list intruding into the cut-off radius r_{c} without being accounted for. There is clearly a balance to reach between the refresh frequency and the r_{v} values (which determine the list length).

A number of criteria have been evolved over the years as a condition to set off the list refresh operation, mostly on the basis of the maximum possible distance s_i travelled by particle i. For instance, Chialvo and Debenedetti [21], within the context of molecular dynamics, proposed the following criteria:

$$s_i = \sum_{j=1}^{m} ds_{i,j} = \sum_{j=1}^{m} |\mathbf{r}_i(t_j) - \mathbf{r}_i(t_{j-1})| \geqslant \alpha \cdot V_{\mathrm{skin}} \tag{3}$$

where $ds_{i,j}$ is the distance travelled in a single time step by the particle i, between the time t_{j-1} and the time t_j, $\mathbf{r}_i(t_j)$ is the actual particle location, α is an empirical parameter taken to be 0.75 and m is the number of time steps needed to overcome a distance greater than $\alpha \cdot V_{\mathrm{skin}}$. Note that t_0 $(j=1)$ is the instant in which the last list refresh operation has occurred. Different empirical criteria were proposed by Blink and Hoover [22].

A different approach, which allows to reduce the number of unnecessary distances, is by introducing a tessellation of the physical space through structured or unstructured grids.[‡]

For a given particle in a given cell, we define 'surrounding particles' as those located within the cell itself and the adjoining cells (Figure 3); the neighbourhood of such a particle is necessarily a subset of the surrounding particles.

This approach requires the construction of a list of particles linked with the cells (Figure 4). The resulting data structure is known as cell-linked list [23] and updates require $O(N_{\mathrm{f}})$ operations of each time step.

For this approach, as for the Verlet list method, the overall efficiency depends very much on the choice of parameters and, in particular, on the balance between the size of the cells against their total number [24, 25]. Large cells imply a longer neighbourhood search procedure, while a greater number of them requires a longer data structure reconstruction.

[‡]Unstructured grids require a list of the connectivity, which specifies the way a given set of vertices make up individual elements.

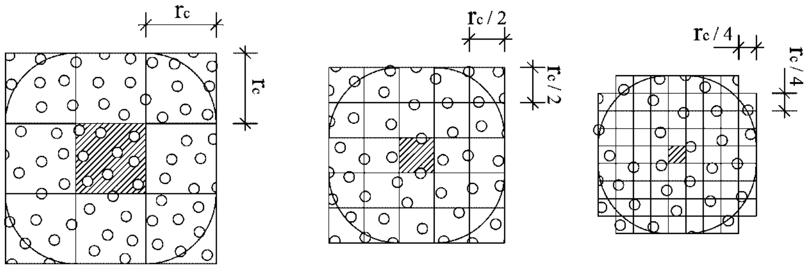


Figure 3. 2D structured rectangular grids. With reference to the hatched cell, *adjoining cells* cover by construction the interaction range (cut-off distance) of any particle within the cell itself. Particles located within such cells are called '*surrounding particles*' (circles in the picture). The neighbourhood of a particle inside the reference cell is then always a subset of the *surrounding particles*.

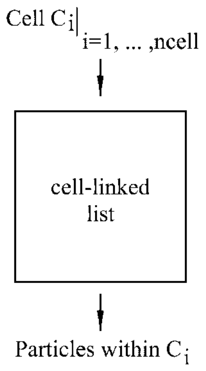


Figure 4. Identification of particles inside the cells though the cell-linked list. In the picture, *n*cell is the total number of cells.

Finally, the Verlet list and the cell-linked list can both coexist in the same algorithm: in this case, the neighbourhood of a moving particle i is built taking into consideration only the particles located inside the adjoining cells of size greater than or equal to r_{v}, since in this case, particles that are further than one cell apart are necessarily beyond the Verlet radius.

As the computational complexity of SPH and similar methods increases, the importance of neighbourhood search is bound to increase as well, up to the point where it become a determinant factor for large-scale computation.

In the following, the efficiency of various search techniques as a function of the computational parameters is numerically evaluated in relation with real-life test cases, by making use of a specially developed SPH software code, which allows different data structures to be tested and the computational time τ_{s} of the search procedure to be evaluated separately from the total τ_{tot} CPU time.

2. NEAREST-NEIGHBOUR SEARCH ALGORITHM DESCRIPTION

2.1. Data structures and startup

Initial and boundary conditions need to be first of all stated at the outset of the code. Time evolution is then described by dynamically allocated arrays, such as:

- position $\mathbf{X}=\mathbf{X}(n_d, N_{\text{tot}})$,
- velocity $\mathbf{V}=\mathbf{V}(n_d, N_{\text{tot}})$,
- density $\rho=\rho(N_{\text{tot}})$,
- pressure $p=p(N_{\text{tot}})$.

According to the definition of N_{tot}, each of the listed data structures comprises two blocks: one referring to the moving particles and the remaining to the boundary ones. Force evaluation is only carried out for particles belonging to the first block.

Once the initial position is set, search data structures can be initialized. The cell-linked list is constructed by partitioning the physical space with a regular grid of congruent rectangles (2D) or rectilinear parallelepipeds (3D). Particles are then simply assigned to cells according to their spatial coordinates.

For plane problems with vertically oriented cells, particle i with coordinates $(x_i(t), y_i(t))$ is easily located inside the cell $C(i)$ by the following equation:

$$C(i)=N_{\text{Cell},y}\cdot\text{INT}\left(\frac{x_i(t)}{D_{\text{Cell},x}}\right)+\text{INT}\left(\frac{y_i(t)}{D_{\text{Cell},y}}\right)+1 \tag{4}$$

where t is the actual computing time, $N_{\text{Cell},y}$ is the number of cells along the vertical direction and $D_{\text{Cell},x}$ and $D_{\text{Cell},y}$ are the cell sizes along the coordinate axes.

Our cell-linked list only holds cells taken by at least one particle. Most SPH problems deal with rapidly varying free surface flows phenomena, where often the number of filled cells is much smaller than the total number; hence building the cell-linked list with 1D data set structures (Figure 5), rather than with a simple matrix, has the advantage of saving a great amount of physical memory. The matrix solution could be adopted instead for problems with a somewhat restricted particle mobility.

As shown in Figure 5, an array must be created containing cell locations inside the cell-linked list; a null entry indicates empty cells. Once the desired pair pointers have been identified, corresponding particles within the pointed cell can be extracted.

Grid cell topology is established building a list of adjoining cells for each cell (Figure 6). Such a table only needs to be constructed once at the beginning, assuming that no dynamic grid refinements are required. For each C_k cell, adjoining cells are then directly defined. Since the cells' sides are here taken to be equal to or greater than the cut-off radius r_{c} (see Figure 3, leftmost picture), each will in general have 8 (2D) or 26 (3D) adjoining ones. The resulting table size is then at most $3\text{rd}\cdot N_{\text{Cell}}$, where N_{Cell} is the number of grid elements discretizing the domain.

Here it is worth remarking that the cell sides must be greater than r_{v}, when the Verlet list is coupled with the cell-linked list, to make sure that the potential neighbouring particles of each moving one inside a cell are within the cell itself and in the adjoining ones.

At this point, if the Verlet list approach is taken, the potential neighbourhood for all moving particles can be quickly identified by making use of the previously built cell-linked list. For each moving particle, the potential neighbours are selected from among the surrounding ones by

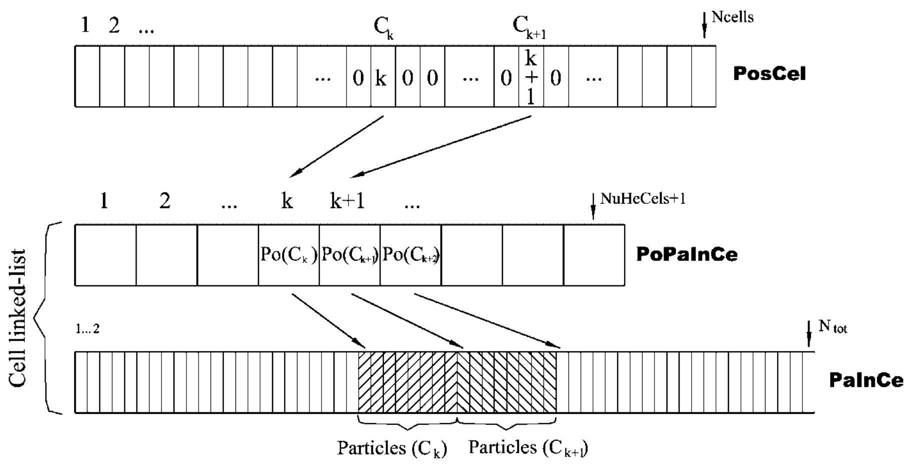


Figure 5. Cell-linked list definition with 1D arrays. Only the filled cells, whose total number is 'NuHeCels', are accommodated. Particles within the generic cell 'C_k' are identified through the pair pointers which indicate the start and end of the subset. Pointer locations to the cells are conveniently stored in the 1D array 'PosCel'. If an entry is null then the corresponding cell is empty.

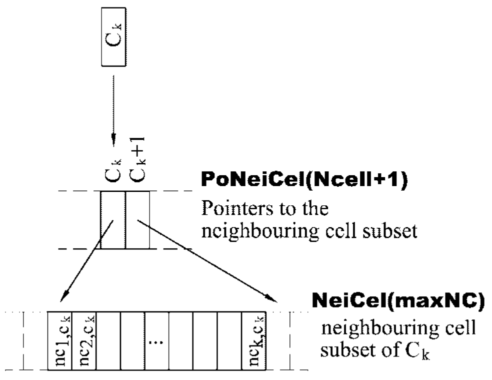

C_k
C_k
C_k+1
PoNeiCel(Ncell+1)
Pointers to the neighbouring cell subset
NeiCel(maxNC)
neighbouring cell subset of C_k
nc1,c_k
nc2,c_k
...
nck,c_k

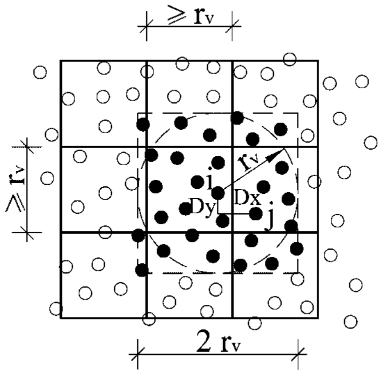
Figure 6. Nearest neighbours cell identification. Adjoining cells of C_k are obtained by querying the list *NeiCel*. The access is done though pair pointers stored into *PoNeiCel*.

choosing only those whose distances along the coordinate axes[§] Dx_j, $j=1,\ldots,n_d$, are less than r_v (Figure 7).

Particles so identified are then progressively placed in the Verlet list which is schematically shown in Figure 8.

The total number of potential neighbouring particles for all moving ones is not strictly proportional to N_f, since the fluid is thought to be slightly compressible. In Figure 9, a flowchart shows the steps featuring the initialization phase.

[§]Making use of such criterion yields some unnecessary particles but at the same time allows to save computing time, since no distances need to be computed.



Dx = x_i-x_j Dy = y_i-y_j

if max(|Dx|, |Dy|) < r_v

then

put "*j*" in the Verlet list of the moving particle "*i*"

Figure 7. Potential neighbour particle extraction for planar problems. Potential neighbours (bold marked) of the moving particle *i* are selected from among the surrounding particles on the basis of the relative distance components.

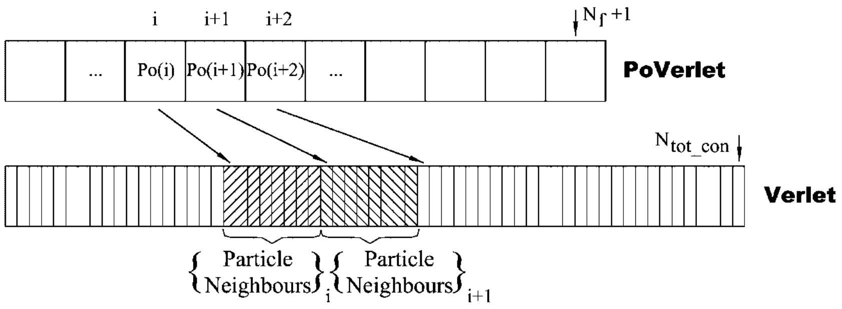


Figure 8. Verlet list construction. Before the simulation begins, potential neighbourhood for each moving particle is stored in a linked list. The access is then granted by means of pointers.

2.2. Data structure updating

Particles move according to Navier–Stokes equations, the SPH forms of which are given by Monaghan [4]; the cell-linked list needs to be updated each time a particle migrates to a different cell. A finite number of time steps are necessary for a particle to move from one cell to another; hence, a refresh operation would be required only for a portion of N_f. The refresh frequency is a function of several parameters such as the cell edge lengths and the particle velocity.

As the system evolves, if a new cell is occupied, it is consequently added in the cell list. Conversely, if a cell currently present in the list becomes empty, it is then deleted. Migrating particles need to be moved in the list of particles ordered by the cell taken from the particle subset of the old cell to the one referring the new cell.

The Verlet list is refreshed for those moving particles satisfying Equation (3). The updating procedure is carried out by extracting the new particle subset (Figure 7) from the current surrounding particles.

No special treatment is given to isolated particles, i.e. those with no neighbours. The refresh procedure applied for the generic moving particle *i* is sketched in Figure 10.

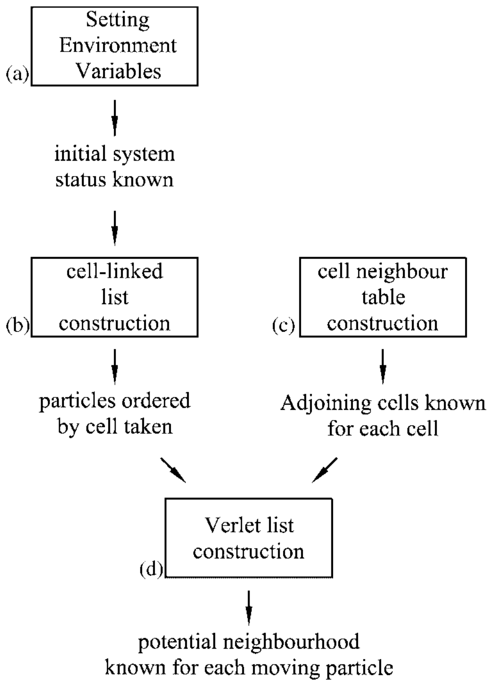

(a) Setting Environment Variables
initial system status known
(b) cell-linked list construction
(c) cell neighbour table construction
particles ordered by cell taken
Adjoining cells known for each cell
(d) Verlet list construction
potential neighbourhood known for each moving particle

Figure 9. Initialization of variables: (a) user defines initial and boundary conditions; (b) initial particle positions are used to construct the cell-linked list employing Equation (4); (c) adjoining cells for each cell are defined on the basis of the grid topology (cell orientation); and (d) surrounding particles for all computing particles are filtered, yielding the potential neighbourhood for each one.

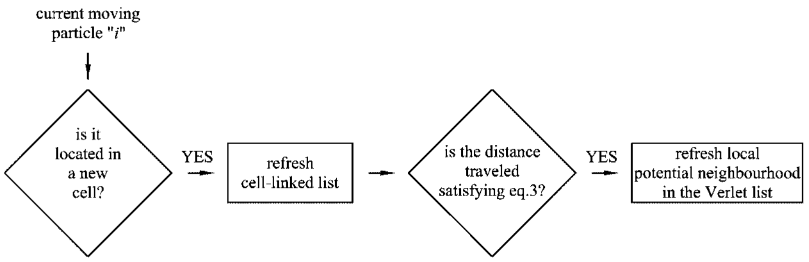

current moving particle "i"
is it located in a new cell?
YES
refresh cell-linked list
is the distance traveled satisfying eq.3?
YES
refresh local potential neighbourhood in the Verlet list

Figure 10. Refresh procedure of data structures. Every time a moving particle moves to a different cell, which is adjacent to the old one, the cell-linked list is updated accordingly. In addition, if the distance travelled satisfies Equation (3), the local potential neighbourhood is refreshed.

3. PERFORMANCE ANALYSIS

With reference to the neighbour search procedure, the cell-linked list updating requires the order of $O(N_f)$ operations to be carried out at each time step. The memory requirement grows linearly with the number of both particles and filled cells. Maintaining the list hence requires some overhead, but on the other hand, the neighbourhood search for each particle has to be carried out only in the cell itself and in the adjoining ones, thus speeding up the search effort of a factor proportional to $N_{\mathrm{tot}}/N_{\mathrm{sp}}$, where the number of surrounding particles might be set as

$$N_{\mathrm{sp}} = \left(\frac{3 \cdot L_{\mathrm{c}}}{d_0}\right)^{n_d} \tag{5}$$

where L_{c} is the average spatial dimension of the cells and d_0 is the initial reference particle spacing.

In the limiting case in which $L_{\mathrm{c}} = r_{\mathrm{c}}$, the effective interaction area of a moving particle is only a fraction of the search region, i.e. $\pi/9 \sim \frac{1}{3}$, $(4\pi/81 \sim \frac{1}{6})$ for $n_d = 2$, (3). Reducing the cell size to a quantity less than r_{c}, allows better approximating of the interaction sphere. On the other hand, the total number of cells increases significantly and, at the same time, a greater number of empty cells unavoidably appear. For three-dimension problems, taking into account a $5 \times 5 \times 5$ elementary box, whose cell size is now $\frac{1}{2} r_{\mathrm{c}}$, the fraction of unutilized volume is reduced by half, that is $4\pi/(3 * 2.25^3) \sim \frac{1}{3}$. Nevertheless, the typical number of neighbouring cells changes from 26 to 124, thus making the vector of neighbouring cells (Figure 6) five times larger for a 3D problem [26].

Therefore, large values of L_{c} imply a great number of surrounding particles and hence, a great number of unnecessary distances have to be computed. Short values imply cell data structures with a great number of entries, thus making cumbersome the refresh process of the cell-linked list.

When the cell-linked list is coupled with the Verlet list, the search effort is further enhanced. The number of potential neighbouring particles N_{np} is indeed lower than N_{sp}, as can be seen in Equation (5), replacing 2 r_{v} with 3 L_{c}, where 2 $r_{\mathrm{v}} < 3L_{\mathrm{c}}$. Moreover, the neighbour list for each moving particle does not have to be updated each time step, according to Equation (3). The effort to maintain the list of neighbouring particles scales therefore as $O(\beta N_{\mathrm{f}} N_{\mathrm{np}})$, β being less than unity.

There are again two limiting cases: for $r_{\mathrm{v}} \to r_{\mathrm{c}}$ ($V_{\mathrm{skin}} \to 0$) there is no safety shell around the interaction cut-off distance; hence, the list has to be updated at every time step, that is $\beta = 1$. If the cell-linked list algorithm is not taken into account, building the Verlet table is of the order $O(N_{\mathrm{f}} \cdot N_{\mathrm{tot}})$. For $r_{\mathrm{v}} \to \infty$, the number of list refresh decreases but the resulting Verlet list becomes too large.

The computational search time τ_{s} is thus strongly affected by both the typical spatial scale L_{c} and the Verlet radius r_{v} (or equivalently by the Verlet skin V_{skin}). The function $\tau_{\mathrm{s}}(L_{\mathrm{c}}, V_{\mathrm{skin}})$ is then characterized by two local minima as will be shown by considering a dam break simulation.

4. RESULTS AND CONCLUSIONS

In order to analyse the best performance as a function of the cell size L_{c} and the Verlet skin V_{skin}, a study was carried out for a typical test case, i.e. a dam break type event over a rectangular channel. The volume of fluid, 10 m long and 7 m high, is discretized using 50 000 particles whose initial positions are shown in detail in Figure 11. Boundaries are modelled with fixed particles forming a staggered grid [9] and exerting a repulsive force to prevent particle penetration [4]. The following parameters were considered:

τ_{s}, neighbourhood particle identification running time;

$\tau_{s,b}$, 'brute force' neighbourhood particle identification running time, i.e. with no Verlet table or fixed cell;
τ_{tot}, total CPU time;
$\tau_{tot,b}$, 'brute force' total CPU time.

Over time, the fluid flows over the downstream part of the channel, varying its shape and occupying previously empty cells (Figure 12).

In order to evaluate the computational time, the various neighbourhood search parameters were varied one at a time, while the integration parameter was kept constant. The computational time step was fixed at 10^{-4} s and cut-off distance r_c was taken to be 2.6 $d_0, d_0 = 0.04$ m being the reference distance between particles (Figure 11).

Several cell sizes were also taken into account. In each case, different values were considered for the ratio $\alpha = r_v/r_c$ between the Verlet radius r_v and the cut-off r_c. $\alpha = 1$ means that no Verlet table is taken into account and the search for neighbouring particles is established at each time step.

The most relevant results are shown in Figure 13.

Figure 13(a) refers to the case with no fixed cell structure. Here obviously the 'Verlet list' procedure is highly beneficial, even though it appears that the size of the list must be carefully chosen, in order to make full use of its advantage. The effect is still present as the grid gets finer (Figure 13(b)–(e)).

Figure 13(f) shows that too fine a grid can increase the execution speed, because of the burden managing of the data structures related to the search procedure.

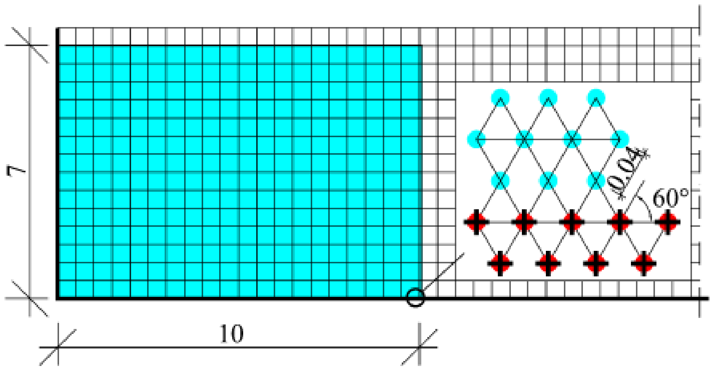


Figure 11. Domain discretization with particles placed on a staggered grid.

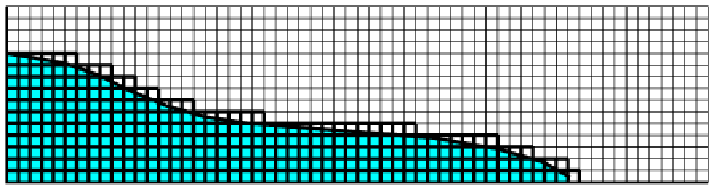


Figure 12. Dam break simulation with the SPH technique. The amount of filled cells (shaded) determine the cell-linked list size.

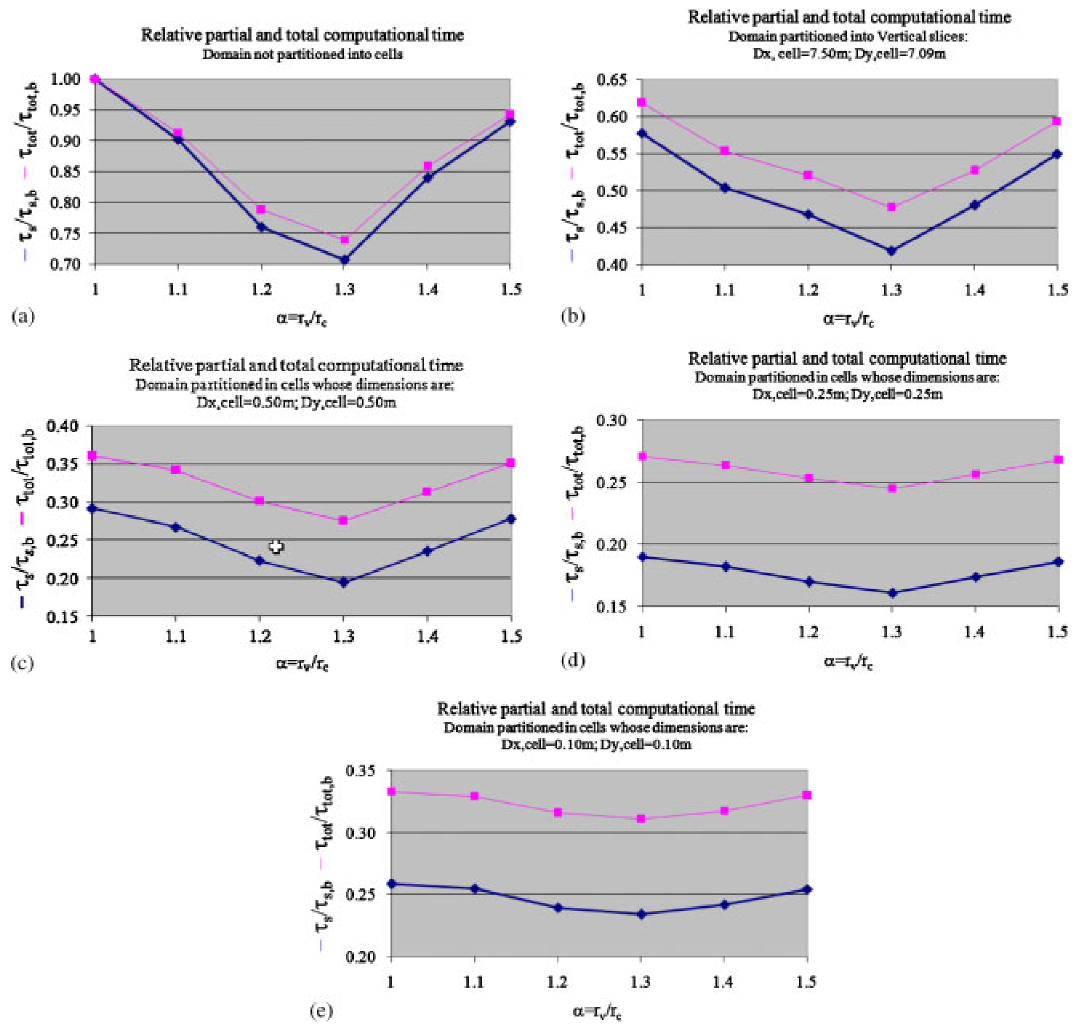


Figure 13. CPU time for different SPH runs. Overheads are scaled with the computation time needed to carry out the same simulation with the brute force evaluation approach.

With reference to the relative running time $\tau_s/\tau_{s,b}$ taken to perform the neighbour search procedure, Figure 14 summarizes the previous results, showing how both the cell dimension and the Verlet skin affect the speed execution.

Compared with the brute force approach a properly tuned algorithm leads to sixfold saving in time; optimal parameters seem to be given by ($L_c = 0.25\,\text{m}$; $V_{\text{skin}} = 0.3r_c$).

There is an obvious advantage in decreasing the cell size; however, if the size is further decreased below an optimum value ($L_c = 0.25\,\text{m}$ in the case considered), the computational time increases again. The division of the domain may be all-important in order to keep the computation time within reasonable limits.

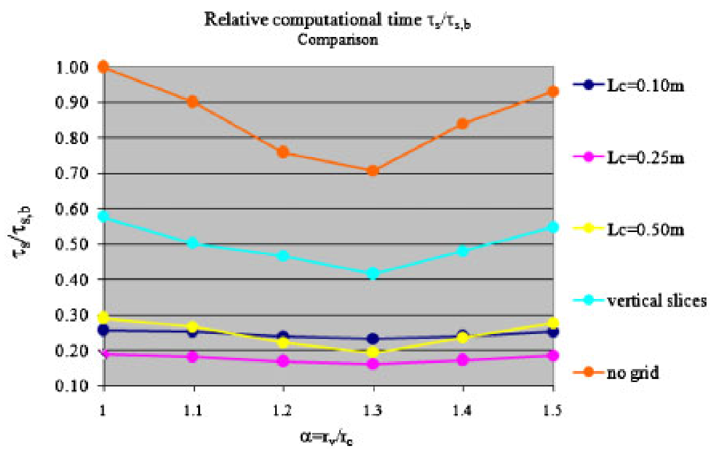


Figure 14. CPU time comparison with reference to the searching procedure.

It also appears that while both the cell-linked list and the Verlet list do improve the computational time, the comparative advantage of the linked cell increases with the decreased size of cells, to the point where the Verlet list approach is practically of no use whatsoever. However, for non-optimal values of cell size, the Verlet list algorithm can provide a definitive advantage in reducing computational times.

ACKNOWLEDGEMENTS

This work was equally shared between the authors. SPH computations were carried out on the Opteron quad processor machine owned and maintained by the C.U.GR.I. centre. The authors are grateful to Prof. R. A. Darlymple (the Johns Hopkins University) for his invaluable help and advice on the developed searching procedure.

REFERENCES

1. Gingold RA, Monaghan JJ. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society* 1977; **181**(2):375–389.
2. Lucy LB. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal* 1977; **82**:1013–1020.
3. Liu GR, Liu MB. *Smooth Particle Hydrodynamics*, *A Meshfree Particle Method*. World Scientific: Singapore, 2003.
4. Monaghan JJ. Simulating free surface flows with SPH. *Journal of Computational Physics* 1994; **110**:399–406.
5. Roubtsova V, Kahawita R. The SPH technique applied to free surface flows. *Computers and Fluids* 2006; **35**(10):1359–1371.
6. Ata R, Soulaïmani A. A stabilized SPH method for inviscid shallow water flows. *International Journal for Numerical Methods in Fluids* 2005; **47**:139–159.
7. Gómez-Gesteira M, Dalrymple RA. Using a 3D SPH method for wave impact on a tall structure. *Journal of Waterways*, *Port*, *Coastal*, *and Ocean Engineering* 2004; **130**(2):63–69.
8. Löhner R, Yang C, Oñate E. Simulation of flows with violent free surface motion and moving objects using unstructured grids. *International Journal for Numerical Methods in Fluids* 2007; **53**:1315–1338.
9. Gómez-Gesteira M, Cerqueiro D, Crespo C, Dalrymple RA. Green water overtopping analyzed with a SPH model. *Ocean Engineering* 2005; **32**:223–238.
10. Dalrymple RA, Rogers BD. Numerical modeling of water waves with the SPH method. *Coastal Engineering* 2006; **53**:141–147.

11. Lo EYM, Shao SD. Simulation of near-shore solitary wave mechanics by an incompressible SPH method. *Applied Ocean Research* 2002; **24**:275–286.
12. Gotoh H, Shao SD. SPH—LES model for numerical investigation of wave interaction with partially immersed breakwater. *Coastal Engineering Journal* 2004; **46**:39–63.
13. Shao SD. Incompressible SPH simulation of wave breaking and overtopping with turbulence modelling. *International Journal for Numerical Methods in Fluids* 2006; **50**:597–621.
14. Shao SD. Simulation of breaking wave by SPH method coupled with k–ε model. *Journal of Hydraulic Research* 2005; **44**(3):338–349.
15. Shao SD, Ji C. SPH. Computation of plunging waves using a 2-D sub-particle scale (SPS) turbulence model. *International Journal for Numerical Methods in Fluids* 2006; **51**:913–936.
16. Violeau D, Issa R. Numerical modelling of complex turbulent free-surface flows with the SPH method: an overview. *International Journal for Numerical Methods in Fluids* 2007; **53**:277–304.
17. Monaghan JJ. Smoothed particle hydrodynamics. *Reports on Progress in Physics* 2005; **68**:1703–1759.
18. Oger G, Doring M, Alessandrini B, Ferrant P. Two-dimensional SPH simulations of wedge water entries. *Journal of Computational Physics* 2006; **213**:803–822.
19. Monaghan JJ, Lattanzio JC. A refined particle method for astrophysical problems. *Astronomy and Astrophysics* 1985; **149**:135–143.
20. Verlet L. Computer experiments on classical fluids. *Physical Review* 1967; **159**(1):98–103.
21. Chialvo AA, Debenedetti PG. On the use of the Verlet neighbor list in molecular dynamics. *Computer Physics Communications* 1983; **60**:215–224.
22. Blink JA, Hoover WG. Fragmentation of suddenly heated liquids. *Physical Review A* 1985; **32**(2):1027–1035.
23. Allen MP, Tildesley DJ. *Computer Simulation of Liquids*. Clarendon Press: Oxford, 1987.
24. Viccione G, Bovolin V, Carratelli EP. A fast neighbour-search algorithm for free surface flow simulations using SPH. Paper accepted at *ECCOMAS Thematic Conference on Computational Methods in Structural Mechanics and Earthquake Engineering*, Rethymno, Crete, Greece, 13–15 June 2007.
25. Bovolin V, Viccione G. An optimized SPH algorithm for neighboring particles identification in free surface flows. *First SPHERIC Workshop*, Rome, Italy, 10–12 May 2006.
26. Mattson W, Betsy MR. Near-neighbor calculations using modified cell-linked list method. *Computer Physics Communications* 1999; **119**:135–148.