Archives of Computational Methods in Engineering (2020) 27:1093–1107
https://doi.org/10.1007/s11831-019-09345-9

ORIGINAL PAPER

An Investigation into Neighbouring Search Techniques in Meshfree Particle Methods: An Evaluation of the Neighbour Lists and the Direct Search

C. A. D. Fraga Filho[1] · L. L. Schuina[1] · B. S. Porto[1]

Received: 24 December 2018 / Accepted: 26 May 2019 / Published online: 1 June 2019

Abstract
Meshfree particle methods are being increasingly employed in solving problems in automotive, aeronautics and oil industries, environmental and geophysical problems, biomechanics and medicine, hydraulic erosion, sediment transport, physics and astronomy, among other areas. Regardless of the application of the particle method, the search for neighbour particles must be done at each numerical iteration (especially in dynamic cases). In 2-D studies, the neighbour lists (linked and Verlet) are techniques commonly used in simulations. This paper presents an investigation of the computational efficiency of the linked list technique through comparison with results of simulations of the direct search (the simplest neighbour search technique). Different numbers of particles and interpolation functions have been used in the tests. By using a simple matrix in the storage of neighbour particles, an improvement in the computational efficiency (in comparison with the direct search's time processing) has not been seen when the linked list algorithm has been utilised. A similar performance between linked list and direct search has been achieved when the neighbour particles have been stored in pairs (even though the cell-linked list has been updated at each numerical iteration). From the analyses of the CPU processing times found in the problems simulated in this work, in which the efficiency of the linked list was only similar to the direct search, it was concluded that is necessary the implementation of a optimisation technique for computational time saving. The Verlet list is a linked list optimisation proposal in which the neighbour list is not update at each numerical iteration. Through an appropriate choice of the cutoff radius, it is ensured that there is no loss in accuracy in the location of neighbouring particles. Optimisation attempts using the Verlet list have been performed but the improvement in the computational efficiency are not satisfactory in all cases.

1 Introduction

Broadly speaking, the meshfree particle methods discretise the continuum domain in a set of particles (or nodal points) aiming to obtain a computational solution of Partial Differential Equations (PDE's) [1]. In this class of methods are included: Smoothed Particle Hydrodynamics (SPH), Moving Particle Semi-Implicit (MPS), Moving Least Square (MLS), Reproducing Kernel Particle (RKPM), Finite Point (FPM), Particle-in-Cell (PI), Particle Finite Element (PFEM) and Molecular Dynamics (MD) methods, among others. Different techniques in the searching of neighbour particles (or nodes) and atoms (in the MD specific case) [1–14] can be employed.

The neighbouring search procedure must be performed at each numerical iteration in dynamic cases, and that has a direct influence on the simulation time. In two-dimensional analyses, neighbour lists (linked and Verlet) are commonly employed in meshfree particle methods [15–18].

The cell-linked or the Verlet list divide the physical domain into a regular grid of congruent rectangles (2-D domain) or rectilinear parallelepipeds (3-D) whose sides measure a cutoff radius. Particles are simply assigned to cells according to their spatial coordinates. Each cell contains a number of particles that can vary during the numerical simulation.

Currently, the octree technique [19, 20] has been applied in conjunction with parallelization, or CUDA GPU processing, in 3-D problems.

In particular, SPH and MPS methods are currently being applied to diverse areas, such as automotive, aeronautics

✉ C. A. D. Fraga Filho
cadff1@gmail.com

1 Federal Institute of Education, Science and Technology of Espírito Santo, Vitória, ES, Brazil

and oil industries, engineering, biomechanics and medicine, geosciences, environmental sciences, oceanography, tribology, applied mathematics, physics and astronomy. The applications of those meshfree particle methods are increasing (some of them are listed below) and the computational efficiency—related to the neighbouring search technique employed—is an aspect that deserves attention.

- Aerospace and aeronautics, automotive and energy production industries, marine and costal engineering, environmental and geophysical problems [21–24].
- Casting [25–27].
- Medicine and biomechanics [28–32].
- Free surface flows [33–35].
- Wave–structure interactions [36, 37].
- Floods and tsunamis [38–40].
- Oil flow in porous rocks [41].
- Multiphase flow and reactive transport in porous media [42].
- Lubrication and tribology [43–45].

Abdelrazek et al. [46] presented a brief comparison between MPS and SPH characteristics.

This paper presents an evaluative investigation of the computational efficiency of the neighbour lists (especially the linked list) in SPH applications. From the literature results obtained in previous studies [47, 48] of our research group, and validated from the comparison with experimental/literature data, it was possible to conclude on the efficiency of the linked list technique in two benchmark engineering problems: lid-driven cavity flow and dam breaking.

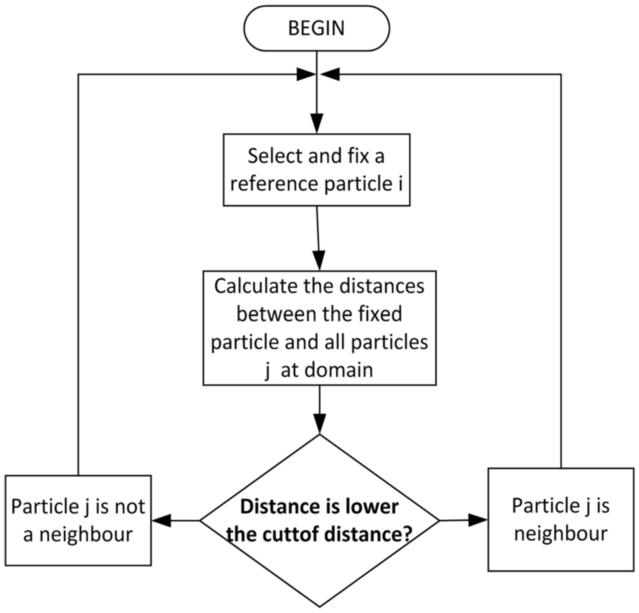
The contributions of this work lie on the presentation of the numerical implementation of a linked list, in providing computational efficiency results using or not the neighbours storage in pairs, and in the discussion on the linked list optimization using the Verlet list.

2 Presentation of the Neighbour Search Techniques

2.1 Direct Search

The direct search is the simplest search technique employed in particle methods. Through the simple comparison between the distance of two particles (a fixed and other in any position of the domain), the neighbour particles are found—if the distance is smaller than the cutoff radius (kh)—and stored in a matrix. Figure 1 presents the flowchart of the direct search algorithm and the neighbourhood of a fixed particle.

The storage of the particles and their neighbours is performed in a matrix, basically in two forms. Figure 2 shows the simplest storage form: a matrix whose number of rows is equal to the number of particles used in the discretization of the domain and number of columns is equal to the largest number of neighbours of each particle that can be found at each numerical iteration. At the beginning of the


BEGIN
Select and fix a reference particle i
Calculate the distances between the fixed particle and all particles j at domain
Distance is lower the cuttof distance?
Particle j is not a neighbour
Particle j is neighbour

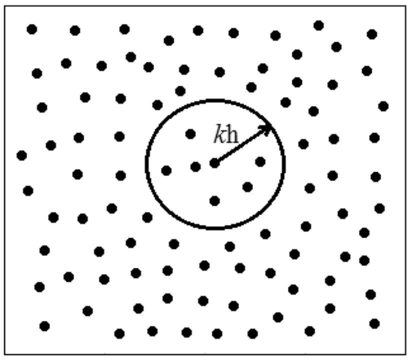
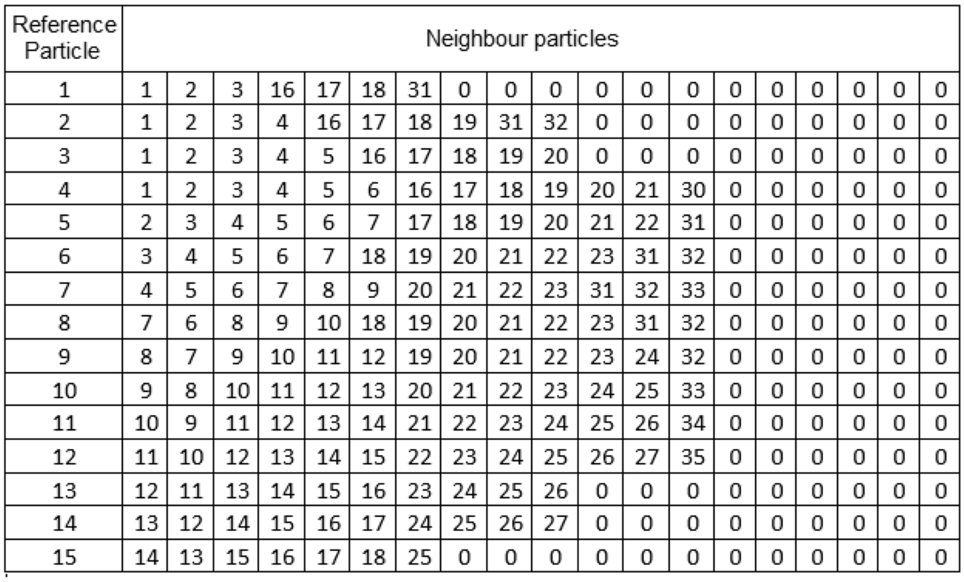
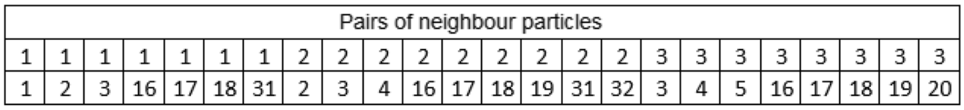


Fig. 1 The direct search algorithm and the neighbourhood of a fixed particle



Reference Particle	Neighbour particles																		
1	1	2	3	16	17	18	31	0	0	0	0	0	0	0	0	0	0	0	0
2	1	2	3	4	16	17	18	19	31	32	0	0	0	0	0	0	0	0	0
3	1	2	3	4	5	16	17	18	19	20	0	0	0	0	0	0	0	0	0
4	1	2	3	4	5	6	16	17	18	19	20	21	30	0	0	0	0	0	0
5	2	3	4	5	6	7	17	18	19	20	21	22	31	0	0	0	0	0	0
6	3	4	5	6	7	18	19	20	21	22	23	31	32	0	0	0	0	0	0
7	4	5	6	7	8	9	20	21	22	23	31	32	33	0	0	0	0	0	0
8	7	6	8	9	10	18	19	20	21	22	23	31	32	0	0	0	0	0	0
9	8	7	9	10	11	12	19	20	21	22	23	24	32	0	0	0	0	0	0
10	9	8	10	11	12	13	20	21	22	23	24	25	33	0	0	0	0	0	0
11	10	9	11	12	13	14	21	22	23	24	25	26	34	0	0	0	0	0	0
12	11	10	12	13	14	15	22	23	24	25	26	27	35	0	0	0	0	0	0
13	12	11	13	14	15	16	23	24	25	26	0	0	0	0	0	0	0	0	0
14	13	12	14	15	16	17	24	25	26	27	0	0	0	0	0	0	0	0	0
15	14	13	15	16	17	18	25	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 2 The simplest form to storage the neighbour particles



Pairs of neighbour particles																							
1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3
1	2	3	16	17	18	31	2	3	4	16	17	18	19	31	32	3	4	5	16	17	18	19	20

Fig. 3 The neighbour particles stored in pairs

simulation all the matrix is initialised with a zero value and as neighbours are found, their numbers are written in matrix positions. In the solution of the conservation physics laws by the meshfree particle method, the location of the first zero in a matrix line informs that there are no longer neighbours for a specific reference particle (identified by the number in the first column of that line).

Figure 3 shows the storage of neighbour particles in pairs [18]. This storage method optimises the interpolation process performed by the meshfree particle method (in the solution of physical conservation equations). The information about a pair of neighbouring particles is used in the interpolations performed for both particles, reducing the number of computational operations.

2.2 Linked List

In the linked list technique, the domain is divided into a grid containing a certain number of cells. Each cell contains a number of particles that can vary during the numerical simulation. The neighbour search is limited to the cells (8 in 2-D and 26 in 3-D) that are in the vicinity of the cell that contains the reference particle. A cell-linked list with informations on the cell that contains the reference particle and those cells in which the neighbouring particles can be located is created. This procedure is performed for each particle at

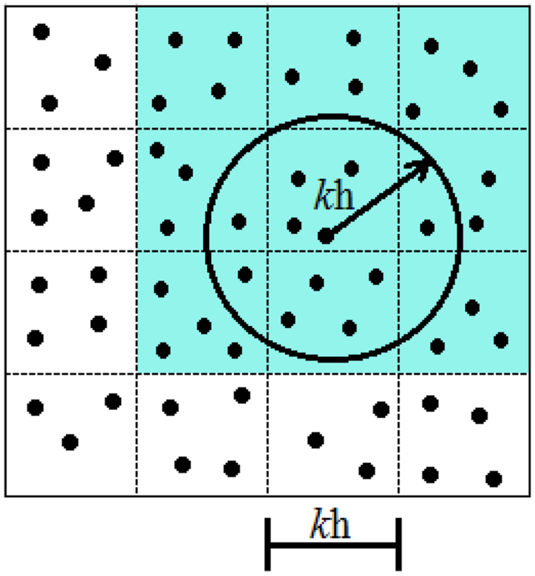
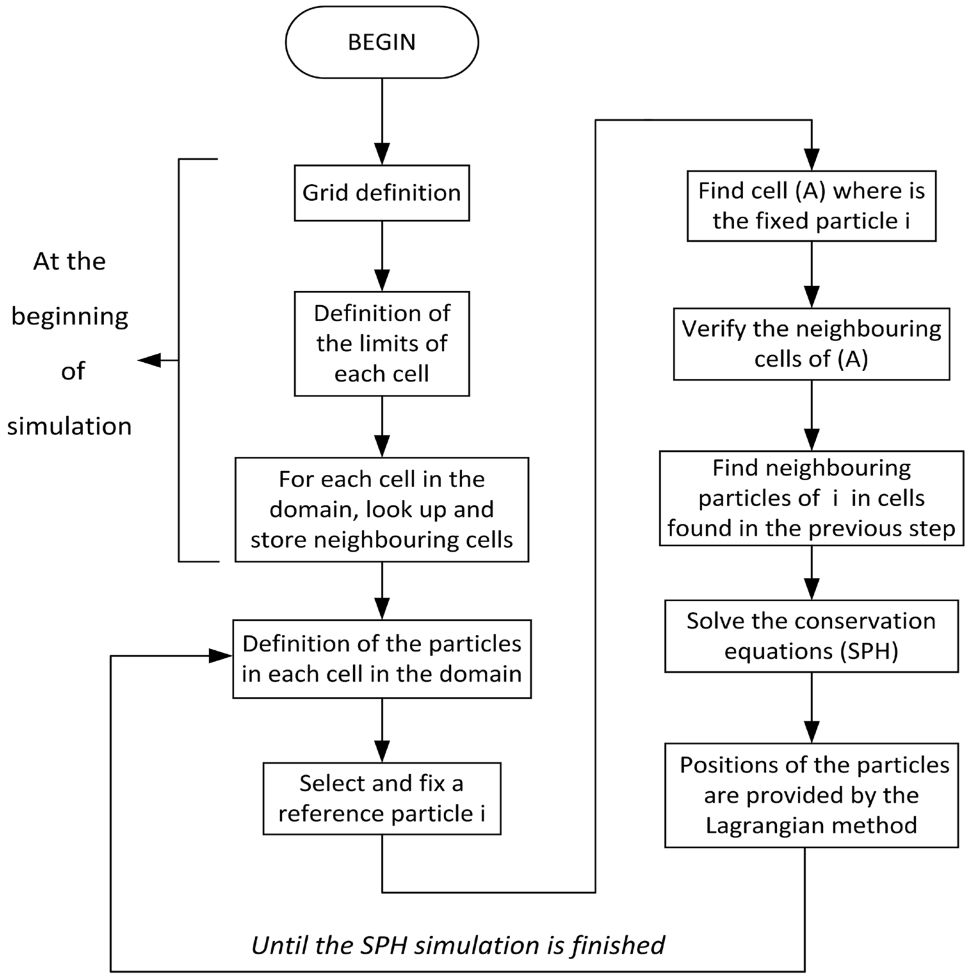


Fig. 4 The domain and the grid cells defined (with cutoff radius equals *k*h). The reduced search region is shaded in light blue

the domain. The calculations of the distances between the reference particle and the others particles at domain are reduced to the region with linked cells (Fig. 4). Figure 5 presents the linked list algorithm implemented for solving

Fig. 5 The linked list algorithm implemented


BEGIN
Grid definition
Definition of the limits of each cell
For each cell in the domain, look up and store neighbouring cells
At the beginning of simulation
Definition of the particles in each cell in the domain
Select and fix a reference particle i
Find cell (A) where is the fixed particle i
Verify the neighbouring cells of (A)
Find neighbouring particles of i in cells found in the previous step
Solve the conservation equations (SPH)
Positions of the particles are provided by the Lagrangian method
Until the SPH simulation is finished

a science/engineering problem. The cell-linked list needs to be updated each numerical iteration, due to migration of the particles to other cells.

2.3 Verlet List

Domínguez et al. [15] and Viccione et al. [16] performed the optimization of the linked list using the Verlet list, whose creation requires the calculations needed to generate the cell-linked list.

In this technique, the neighbour list is built with probable neighbouring particles of each reference particle at the domain, inside a cutoff radius (kh + L) slighter higher than the support radius (kh) used in the linked list.

The definition of the cutoff radius takes into account the maximum possible displacement of the reference particle during a simulation time (related to a certain number of numerical iterations) in which the neighbour list will not be updated [16].

In the Verlet list technique, all particles within the cutoff radius (kh + L), as shown in Fig. 6, will be added to a list of potential neighbour particles, but only those whose distances

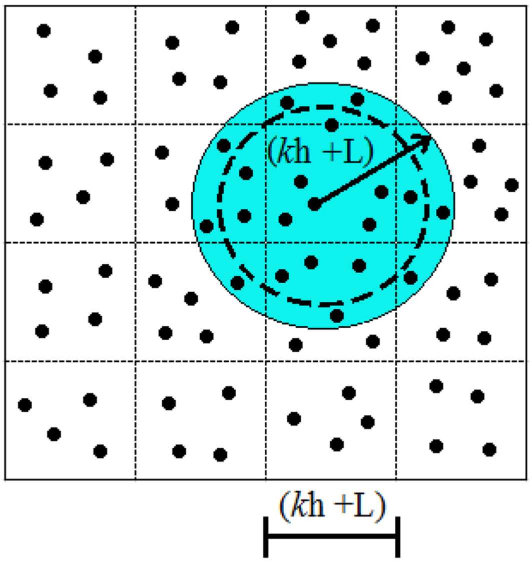


Fig. 6 The cutoff radius used in the Verlet list (kh + L) and the probable neighbours of the reference particle (in the centre of the circle)

for the reference particle are less than or equal to *k*h will be used in the interpolations of the particle method. The neighbour list is not update at the numerical iteration, which results in processing time saving.

In a first analysis, there are no losses in the accuracy in the location of neighbour particles. However, the choice of the cell size must be done carefully in order to guarantee the accuracy and the advantages of the technique.

3 Algorithmic Implementations

3.1 Direct Search

As explained earlier, the direct search is a technique of simple implementation. Through the simple comparison between the distance of two particles (a fixed and other in any position of domain) the neighbour particles are found and stored in a matrix, as shown in Figs. 2 and 3.

3.2 Linked List

In the linked list, there are more complex operations. The following will be presented a concise description of the algorithm implementation, in FORTRAN Programming Language.

Step 1. Firstly, the limits of the spatial region that will receive particles along the simulation must be provided and the coordinates of each cell must be defined.

```
count_cell=0
 DO i=1, Ncellx
  AUX_X = (i-1)*Lxcell
   DO j=1, Ncelly
   count_cell=count_cel+1
   AUX_Y = (j-1)*Lycell
   coord_cell(count_cel,1)= count_cell
   coord_cell(count_cel,2)= AUX_X   !lower abscissa of the cell
   coord_cell(count_cel,3)= AUX_X + Lxcell !higher abscissa of the
   coord_cell (count_cel,4)= AUX_Y  !lower ordinate of the cell
   coord_cell (count_cel,5)= AUX_Y + Lycell !higher ordinate of the
                                            cell
   END DO
 END DO
```

```
where
count_cell is an auxiliar variable for counting the number of cells
I is an auxiliary variable used in the loop
AUX_X is a auxiliar variable for counting the number of cells
in the direction x
AUX_Y is a auxiliar variable for counting the number of cells
in the direction y
Lxcell is the cell length (direction x)
Lycell is the cell height (direction y)
Ncellx is the number of cells in the direction x
Ncelly is the number of cells in the direction y
coord_cell is the matrix that storages the coordinates of the cells
```

Step 2. A routine used in the identification of the cells that contains particles at each numerical iteration needs to be implemented. This procedure reduces the computational processing time, since neighbouring particles search is performed in a reduced domain. In problems where the domain is large, there may be a considerable number of empty cells at each numerical iteration.

```
count_cell = 0
DO i=1, Ncellx*Ncelly
 IF ((coord_cell(i,2).LT.highest_x)).AND.(coord_cell(i,4).LT.
highest_y)) THEN
   count_cell = count_cell + 1
   cell_util(1,count_cell) = i
 END IF
END DO
```

where
cell_util is the vector that storages the identification of the cells with particles
highest_x is the abscissa of the particle furthest from the origin (measured in direction x)
highest_y is the major ordinate of the particle furthest from the origin (measured in direction y)

Step 3. The particles inside each cell must be identified, from the comparison of their coordinates with the coordinates of each cell in the grid.

```
i = 1
DO WHILE (cell_util(1,i).NE.0) !while there are cells with
                                particles
  cont = 1
  DO j=1,ntotal
    IF((x(j,1).GE.coord_cell(cell_util(1,i),2)).AND.
      (x(j,1).LE.coord_cell(cell_util(1,i),3)))THEN
      IF ( x(j,2).GE.coord_cell(cell_util(1,i),4)).AND.
       (x(j,2).LE.coord_cell(cell_util(1,i),5)))THEN
         count = count + 1
          P_cells(cell_util(1,i),count) = j
      END IF
    END IF
  END DO
  i = i+1
END DO
```

where
i and j are auxiliary variables used in the loop
x is the matrix with the coordinates of the particles
P_cells is the matrix that storages the identification of the particles inside each cell at the domain
count is an auxiliary variable used in the definition of the storage position of each particle in the matrix P_cells

Step 4. The neighbours cells of each cell in the grid are identified. This allows that the neighbouring particle search are performed only on the neighbouring cells of a given cell (containing the fixed particle) and not in the whole domain, as in the direct search. During the simulations, the grid remained static. Thus, these operations run only once, at the first iteration.

```
q=1
DO WHILE (cell_util(1,q).NE.0)  !while there are cells with
                                 particles
  count_neighbours = 1
  m=1
  DO WHILE (cell_util(1,m).NE.0)
     IF ((ABS(coord_cell(cell_util(1,m),2) –
       coord_cell(cell_util(1,q),2) ).LE.Lxcell ).OR.
       (ABS(coord_cell (cell_util(1,m),4)-
       coord_cell(cell_util(1,q),4))).LE.Lycell ) THEN
       IF ((ABS(coord_cell(cell_util(1,m),3)-
         coord_cell(cell_util(1,q),3)).LE.Lxcell).OR.
        (ABS(coord_cell (cell_util(1,m),5)-
         coord_cell(cell_util(1,q), 5))).LE.Lycell) THEN
           count_neighbours = count_neighbours + 1
           cell_v(cell_util(1,q),count_neighbours) =
           cell_util(1,m)
       END IF
     END IF
    m=m+1
  END DO
  q=q+1
END DO
```

Step 5. It is necessary to determine the cell in which the reference particle is located. A scan on the non-empty cells is performed until the fixed particle is found, at this moment this procedure finishes. The algorithm below present the operations performed.

Step 6. After the cell A—in which is located the reference particle—has been found, the neighbours search is finally carried out. The search for neighbours cells of the cell A has been done previously, in the grid generation, and the result is stored in the matrix cell_v. Thus, it is enough to check which particles (inside the neighbours cells of A) are neighbours of the reference particle. This checking is performed by calculating the interparticles distances (which must be lower than *k*h). The excerpt of the algorithm is shown below.

```
DO* i=1,ntotal
 response = 'NO'
 cell = 0
 j= 0
   DO WHILE ((j.LE.(Ncellx*Ncelly)).AND.(response.EQ.'NO'))
   k=2
   j = j+1
     DO WHILE ((P_cells(j,k).NE.0).AND.(response.NE.'YES'))
       IF  (P_cells(j,k).EQ.i) THEN
         cell = j
         response = 'YES'
       END IF
      k = k+1
     END DO
   END DO

*This loop continues in the Step 6
```

```
where
ntotal is the total number of particles at the domain
response is a variable used to indicate the end of the search (when
the cell that storages the reference particle is found)
cell is used to storage the cell that contains the reference particle
j and k are auxiliary variables used in the definition of the storage
position in the matrix P_cells
```

```
*Inside the Loop of Step 5
 num_neighbours = 1
 q = 1
 response_2 = 'NO'
   DO WHILE ((q.LE.n_cells).AND.(response_2.EQ.'NO'))
     q=q+1
     aux_cell = cell_v(cell,q)
     IF (aux_cell.EQ.0) THEN   !Stop criterion: to find a
                                position of the matriz cell_v
                                filled with zero
       response_2  = 'YES'
     END IF


     IF (aux_cell.NE.0)THEN
       DO m=2,n_part
         part_aux = P_cells(aux_cell,m)
         IF (part_aux.EQ.0) THEN !Stop criterion: to find a
                                  position of the matriz
                                  P_cells filled with zero
           EXIT
         END IF

         aux_r =(x(part_aux,1)-x(i,1))**2+(x(part_aux,2)-
         x(i,2))**2
         r = sqrt(aux_r)

       IF (r.LE.(hsml(i)*K_scale) ) THEN
         num_neighbours = num_neighbours + 1
        neighbours_matrix(i,num_neighbours) = part_aux
         r(i,num_neighbours)= aux_r
         dx(i,num_neighbours)=(x(part_aux,1) - x(i,1))
         dy(i,num_neighbours)=(x(part_aux,2) - x(i,2))
       END IF
     END DO
END DO*
```

where

num_neighbours is an auxiliary variable used in the definition of the storage position of each neighbour particle in the neighbours_matrix

response_2 is a variable used to indicate the end of the neighbouring particle search

q is an auxiliary variable used in the identification of neighbour cells with particles

m is an auxiliary variable used in the identification of neighbour particles

n_cells is the maximum number of the neighbours cells

aux_cell is a variable used in the search of cells with neighbour particles

part_aux is used in the identification of each particle (in the calculation of the distance to the reference particle)

n_part is the maximum number of particles inside each cell

aux_r is an auxiliary variable

r is the distance between the reference and a possible neighbour particle

hsml is the smoothing length (SPH method)

K_scale is a scaling factor that depends on the kernel used in the SPH interpolations

neighbours_matrix is a matrix used in the storage of the neighbours of each particle at the domain

dx is the horizontal component of the position difference vector

dy is the vertical component of the position difference vector

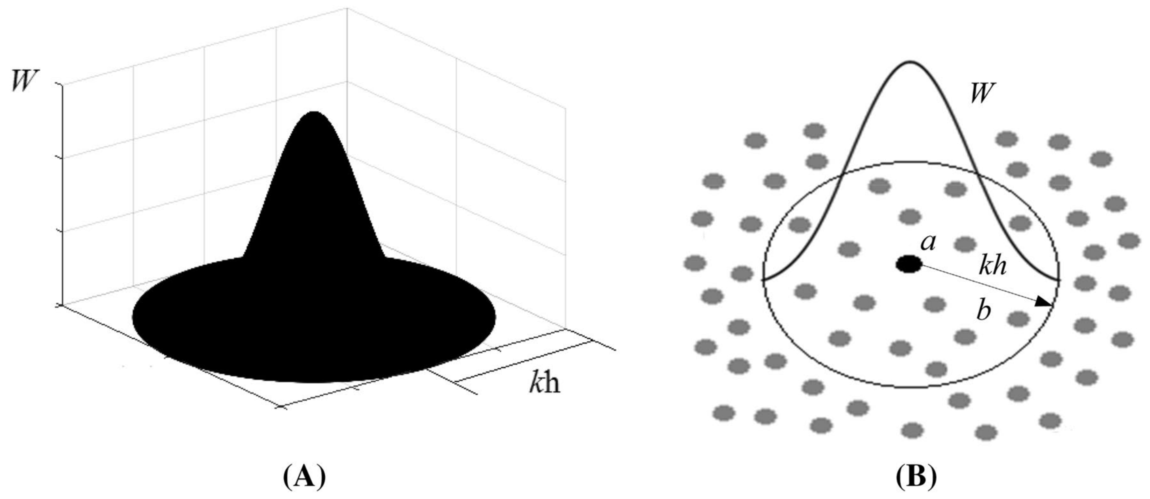


Fig. 7 **a** The continuous smoothing function (W), defined within the domain of influence of radius *k*h. **b** W guarantees the greatest contribution of the nearest neighbouring particles (*b*) to the value of the physical quantity in the reference particle (*a*)

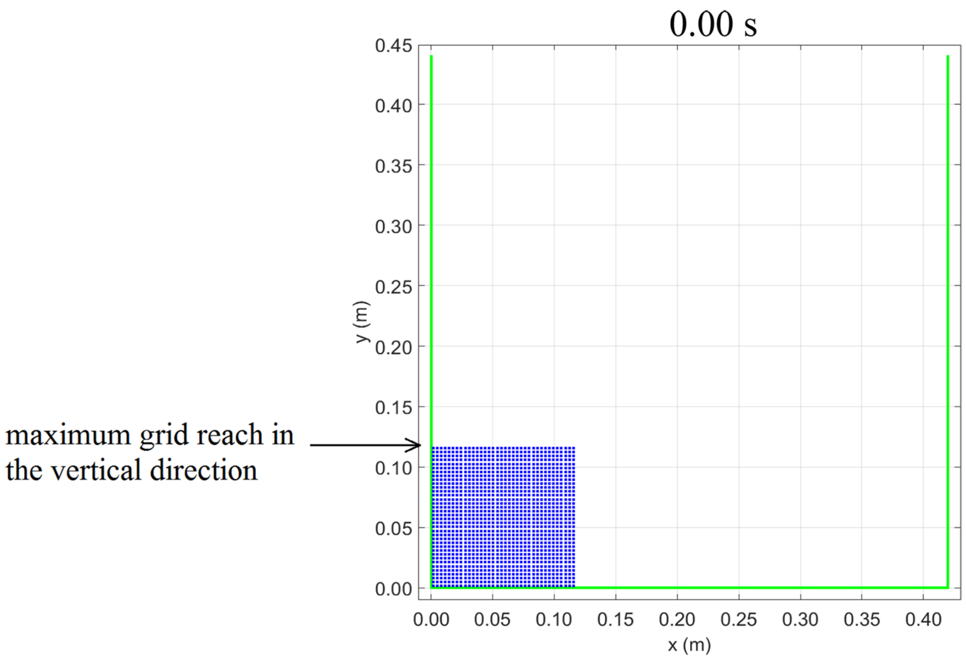


Fig. 8 The simulated geometry and the initial particle setup of the damned water

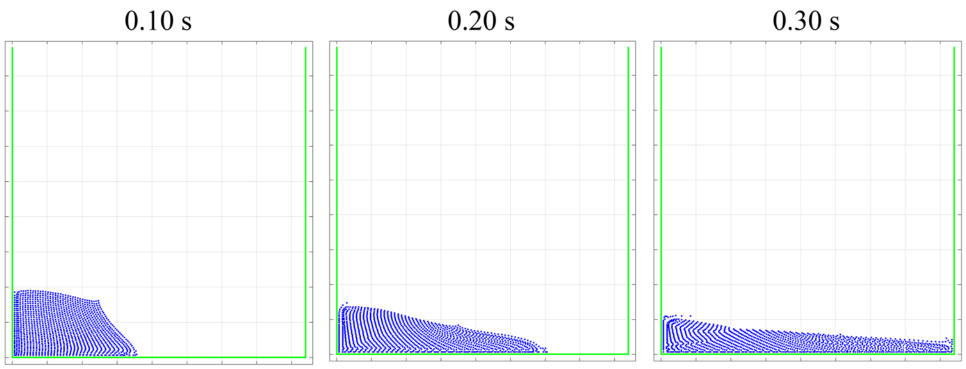


Fig. 9 Evolution of the dam breaking flow until the collision of the wave against the right wall of the reservoir

Table 1 Processing times for different combinations of kernel/number of particles in the dam breaking simulations

Interpolation function	Number of particles	CPU time (s)[a]	
		Direct search	Linked List
Cubic spline	1296	56 min 55 s	2 h 50 min 54 s
New quartic	1296	34 min 47 s	2 h 42 m 55 s
Quintic spline	1296	1 h 04 m 08 s	2 h 25 m 24 s

[a]Serial simulations performed in a computer with 16 GB of RAM memory, Intel Core i7-7700HQ, and Linux operational system

As final result, the neighbours of each particle of the domain are stored in the neighbours_matrix.

4 Lagrangian Particle Modelling

The essence of the particle methods is the discretization of the continuum domain in a finite number of particles and, from interpolations, obtain approximations for physical quantities for each particle of the domain. Only the particles in the vicinity of each particle fixed for study (which are within the domain of influence of the first, that is, within the space region with radius equals to *k*h, in which *k* is a scaling factor that depends on the kernel used), will contribute to the prediction of the physical properties of the reference particle. The contribution given by each neighbouring particle to the value of the physical quantity is inversely proportional to its distance to the reference particle. Smoothing functions (kernels) are used in the interpolations of the particle method. The kernels used in this work can be found in [18], and are presented below.

Cubic spline:

$$W(q,h) = \frac{15}{7\pi h^2}\begin{cases} \left(\frac{2}{3} - q^2 + \frac{1}{2}q^3\right), & 0 \le q \le 1 \\ \left(\frac{1}{6}(2-q)^3\right), & 1 < q \le 2 \\ 0, & \text{in the other case.} \end{cases} \tag{1}$$

Gaussian:

$$W(q,h) = \frac{1}{\pi h^2}\begin{cases} e^{-q^2}, & 0 \le q \le 3 \\ 0, & \text{in the other case.} \end{cases} \tag{2}$$

New quartic:

$$W(q,h) = \frac{15}{7\pi h^2}\begin{cases} \left(\frac{2}{3} - \frac{9}{8}q^2 + \frac{19}{24}q^3 - \frac{5}{32}q^4\right), & 0 \le q \le 2 \\ 0, & \text{in the other case.} \end{cases} \tag{3}$$

Quintic spline:

$$W(q,h) = \frac{7}{478\pi h^2}\begin{cases} (3-q)^5 - 6(2-q)^5 + 15(1-q)^5, & 0 \le q \le 1 \\ (3-q)^5 - 6(2-q)^5, & 1 < q \le 2 \\ (3-q)^5, & 2 < q \le 3 \\ 0, & \text{in the other case.} \end{cases} \tag{4}$$

where

W is the interpolation function (kernel).
r is the position occupied by the particle at the domain.
h is the smoothing length.
a and *b* are subscripts referring to the reference particle and the neighbouring particle respectively.

$q = \left|r_a - r_b\right| / h.$

Figure 7 shows a continuous smoothing function and its utilisation in the interpolations at the domain discretised by particles.

According to a general analysis, the modelling of fluid flow and energy transport is carried out by the physical conservations laws of mass, momentum and energy. In this work, two hydrodynamic cases were studied in a 2-D domain: dam breaking and lid-driven cavity flow. The fluid has been considered as Newtonian, incompressible and isothermal. The mass conservation and Navier–Stokes equations have been discretised and solved by the Smoothed Particle Hydrodynamics method in both problems (Eqs. (5)–(6)):

$$\frac{d\rho_a}{dt} = \sum_{b=1}^{n} m_b\left(\mathbf{v}_a - \mathbf{v}_b\right) \cdot \nabla W(\boldsymbol{X}_a - \boldsymbol{X}_b, h) \tag{5}$$

$$\frac{d\mathbf{v}_a}{dt} = -\sum_{b=1}^{n} m_b\left(\frac{P_a}{\rho_a^2} + \frac{P_b}{\rho_b^2}\right)\nabla W(\boldsymbol{X}_a - \boldsymbol{X}_b, h) + 2\upsilon_a \sum_{b=1}^{n} \frac{m_b}{\rho_b}(\mathbf{v}_a - \mathbf{v}_b)\frac{\boldsymbol{X}_a - \boldsymbol{X}_b}{\left|\boldsymbol{X}_a - \boldsymbol{X}_b\right|^2} \cdot \nabla W(\boldsymbol{X}_a - \boldsymbol{X}_b, h) + \mathbf{g} \tag{6}$$

where d/dt is the material derivative, *m* is the mass, *n* is the number of neighbouring particles within the influence domain, **v** is the velocity, t is the time, ∇ is the mathematical nabla operator, *W* is the kernel, ***X*** is the particle position, *h* is the smoothing length, *P* is the absolute pressure, *υ* is the kinematic viscosity, *ρ* is the density, **g** is the gravity, the subscripts *a* and *b* refer to the reference particle and the neighbouring particle, respectively.

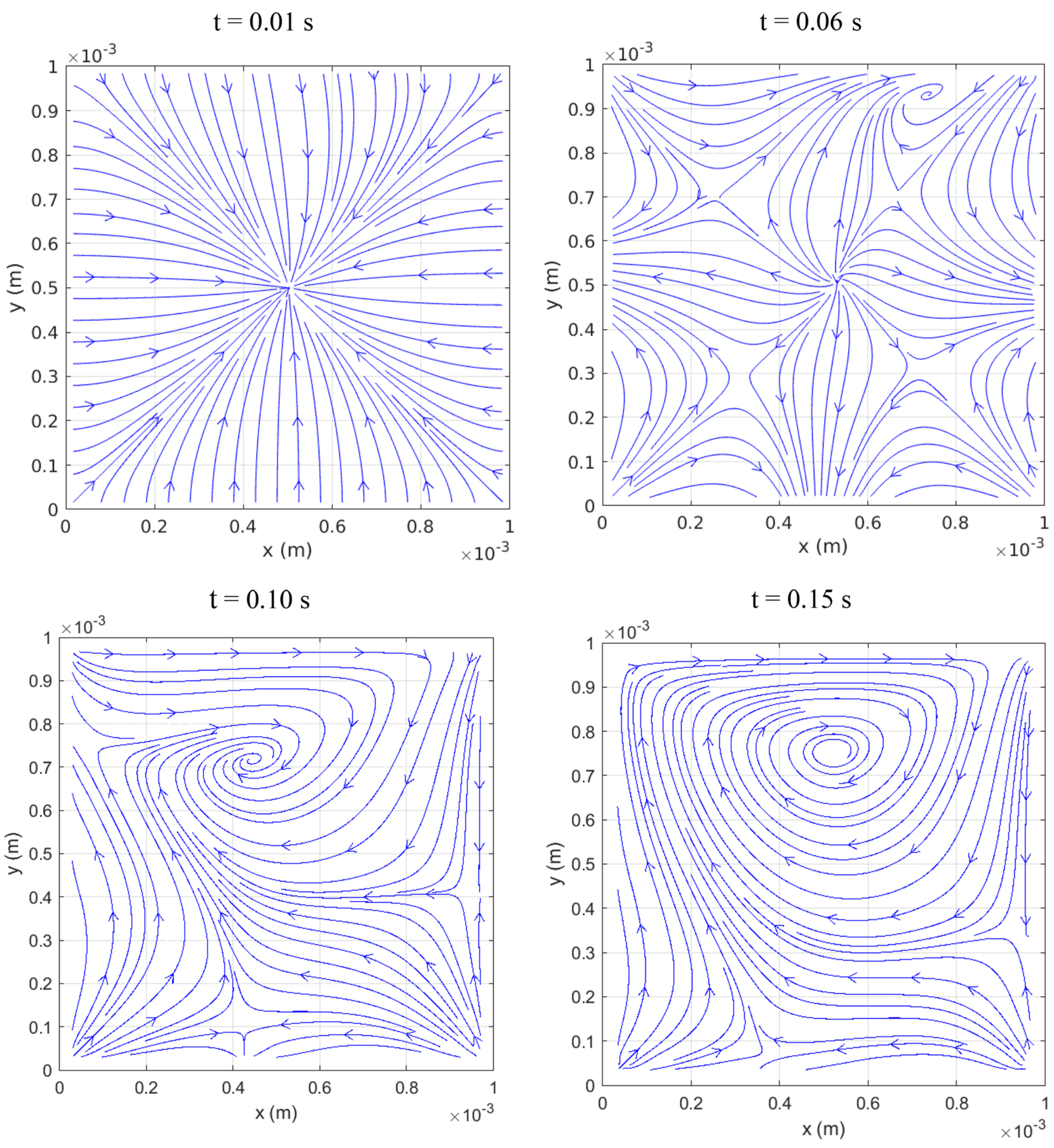


Fig. 10 Streamlines in certain instants of time and the formation of the primary vortex at the steady state (the quintic spline kernel and 40×40 particles per side of cavity were employed in the simulation)

For an incompressible and isothermal flow, the rate of change of the internal energy in time is null. Due to this, the energy conservation equation was not solved in the cases presented in this work.

4.1 Dam Breaking

The studies of dam breaks are of great importance in the prevention of accidents, which can cause serious environmental consequences, besides being a risk to the resident populations in the vicinity of the dams.

The 2-D geometry simulated was a tank with a height a length of 0.420 m and a height of 0.440 m, as shows Fig. 8. The damned water, located at the right side of the reservoir, had a height of 0.114 m height and a width of 0.114 m (aspect ratio equals 1).

A static grid of 52 cells along the length of the tank and 15 cells arranged in the y direction (up to the dammed water line) was defined. At each numerical iteration, the cells containing particles were identified. Thus, the neighbour search operations were performed in a reduced domain (see Step 2 in Sect. 3.2). Cubic spline, new quartic and quintic spline

Table 2 Processing times for different combinations of kernel/number of particles in the lid-driven cavity flow simulations

Interpolation function	Number of particles (per side of the cavity)	CPU time (s)[a]	
		Direct search	Linked List
	30×30	153	159
Cubic spline	40×40	291	294
	50×50	516	501
Gaussian	30×30	168	176
	40×40	323	318
	50×50	545	548
Quintic spline	30×30	164	167
	40×40	314	327
	50×50	540	551

[a]Serial simulations performed in a computer with 16 GB of RAM memory, Intel Core i7-7700HQ, and Linux operational system

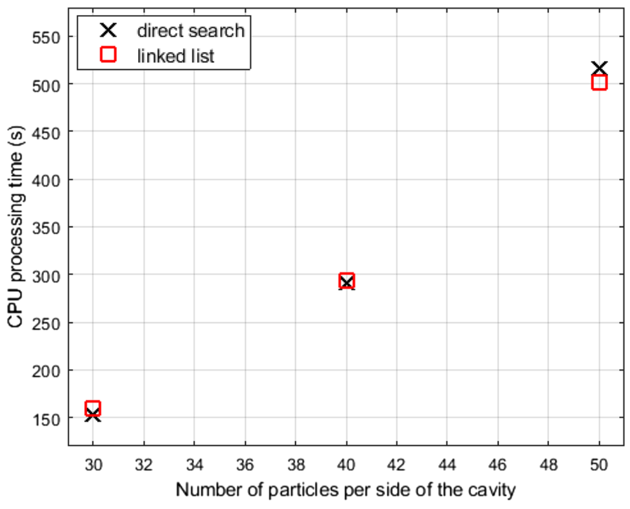


Fig. 11 CPU processing times for simulations using the cubic spline kernel

kernels [16] have been used in the SPH interpolations. The timestep was 1.0×10^{-5} s. 30,000 numerical iterations were performed in the simulations. Figure 8 shows the initial setup of the particles used in the discretization of the damned water.

Figure 9 presents the evolution of the dam breaking flow in different instants of time. Complete simulation results and validation of the collapse of dammed water can be found in [47].

The simulations (using the direct search and the linked list techniques) finished when the wave achieved the left wall of the reservoir (t = 0.30 s) The storage of the neighbouring particles was carried out in the simple form possible, using a matrix as that shown in Fig. 2. The simulation processing

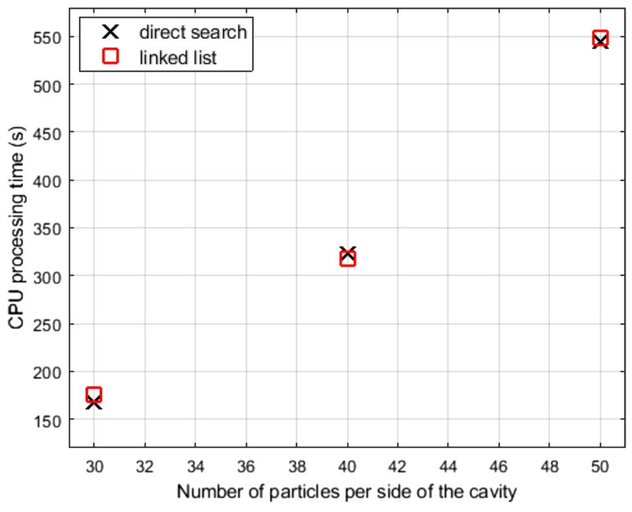


Fig. 12 CPU processing times for simulations using the Gaussian kernel

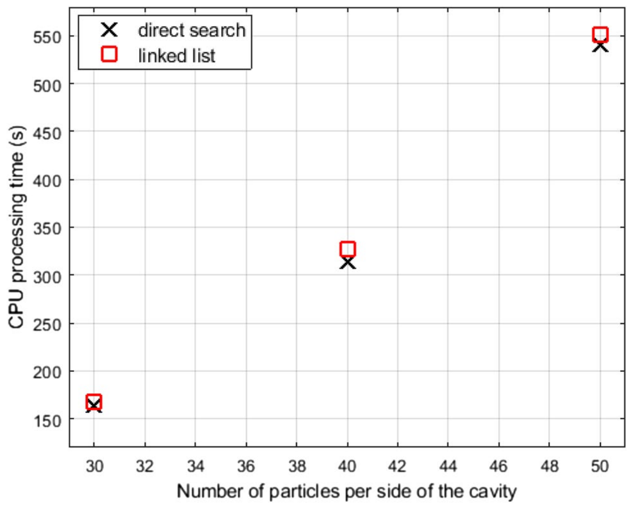


Fig. 13 CPU processing times for simulations using the quintic spline kernel

times for different combinations of interpolation function/ number of particles are shown in Table 1.

Considering the surprising results found in the first simulations, with lower performance of the linked list, more studies were carried out aiming to optimise the first algorithm implemented.

Literature [18] presented another form of neighbouring particles storage—(in pairs)—as shows Fig. 3. By using this technique, fewer computational operations are performed and it is expected that there will be a decrease in CPU time. New simulations have been performed for the 2-D lid-driven cavity flow, as presented in the next subsection.

4.2 2-D Lid-Driven Cavity Flow

Cavities are seen in engineering problems related to environmental and atmosphere. Depressions and valleys, dynamics of lakes, fuselage and wings of airplanes, boat hulls and car bodies, sports stadiums, systems for the continuous deposition of photosensitive films on substrates and photographic papers, material processing, metal casting and galvanizing are situations in which fluid flows in cavities are found [48].

The sides of the square cavity simulated were 1.0×10^{-3} m. Grids of 12×12 cells, 16×16 cells and 20×20 cells, disposed in x and y directions respectively, were used in simulations (when employed 30×30, 40×40 and 50×50 particles per side of the cavity, respectively). The constant velocity of the box cover was 1.0×10^{-4} m. Cubic and quintic spline kernels have been used in the SPH interpolations. The timestep was 5.0×10^{-5} s and 30,000 numerical iterations were performed.

Figure 10 shows streamlines inside the square cavity and the formation the primary vortex when the steady state has been reached. A more complete analysis of this problem can be found in [48]. The simulation processing times are shown in Table 2 and Figs. 11, 12, and 13.

The storage in pairs of the neighbouring particles promoted an improvement in the computational efficiency of the linked list. The differences between the simulation times provided by this technique and the direct search decresead significantly (which is in accordance with the conclusions presented in [18]). Even so, after the CPU times analysis it was verified that the computational efficiency of the linked list technique was only similar to that of the direct search.

5 Concluding Remarks

From the analysis of the results it is possible to conclude that the simple use of the linked list technique is not sufficient for the best efficiency of the neighbouring particle searching. The use of the storage of neighbouring particles in pairs was very important for the reduction of computational time, but it was not enough for a great improvement in the linked list simulation time. It was verified that a linked list optimisation method could improve performance.

The Verlet list is an optimisation technique proposed, that uses calculations required in the generation of the cell-linked list. Despite the advantages of not updating the list of neighbours at each numerical iteration can provide, the choice of an appropriate cutoff radius is fundamental to ensure the good performance of this technique.

Literature shows, however, that the choice of the appropriate neighbour list technique (linked or Verlet) is not so simple. The comparative advantage of the cell-linked list increases with the reduction of the cells size, to the point which the Verlet list approach is practically useless [16].

The CPU simulations of a 3-D collapse of a dam break and its interaction with a rectangular structure, using the SPH particle method, are presented in [15]. Analysing the memory requirements, Verlet was less efficient than linked list. The performance of the Verlet list dependent on the number of steps (ns) that the list kept unaltered. For low and high values of ns, the linked list method was faster than the Verlet list. Only in an intermediate region, the Verlet list was faster, with a maximum improvement close to 6%, for ns equal to 7 timesteps. The runtime improvement is rather moderate, especially when considering the memory requirements of the Verlet list: higher than the requirements of the linked list. An improvement in runtime higher than 8% was obtained when a change in the formulation of the SPH method (using a kernel gradient correction) was used.

In summary, the Verlet list is a linked list optimisation attempt which results are not the best in all cases. From the literature results [15], it is conclude that a wise choice of some parameters, such as cell size and number of timesteps in which the neighbours list will remain unchanged, is necessary so that a reasonable improvement in the processing time (below 10%) is reached. In addition, the memory requirement in the Verlet list was higher than in the linked list in CPU simulations.

Recently, the literature [49] presented a parallel fast neighbour search method and communication strategy for meshless particle methods. An algorithm with a multi-resolution-based hierarchical data structure is employed to construct ghost buffer particles (neighbour particles on remote processor nodes). Cell-linked and Verlet lists were applied in this search technique. Two applications of the parallel algorithm in fluid dynamics simulations using SPH method were performed: (a) with an adaptive smoothing-length and (b) employing a new physics-driven partitioning method [50]. The results achieved demonstrated accuracy and efficiency in the process of construction of the buffer ghost particles and that the new partitioning method is promising.

Compliance with Ethical Standards

Conflict of interest The authors certify that they have no affiliation with or involvement in any organization or entity with any financial or non-financial interest in the subject matter or materials discussed in this manuscript.

References

1. Li S, Liu WK (2002) Meshfree and particle methods and their applications. Appl Mech Rev 55(1):1–34. https://doi.org/10.1115/1.1431547
2. Huerta A, Belytschko T, Fernandez-Méndez S, Rabczuk T (2004) Meshfree methods. Encyclopedia of computational mechanics. Wiley, New York
3. Idelsohn SR, Oñate E, Becker P (2018) Particle methods in computational fluid dynamics. In: Stein E, de Borst R, Hughes TJR (eds) Encyclopedia of computational mechanics, 2nd edn. Wiley, Chichester, pp 1–41
4. Onderik J, Ďurikovič R (2007) Efficient neighbor search for particle-based fluids. J Appl Math Stat Inform (JAMSI), 2(3). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.105.6732&rep=rep1&type=pdf. Accessed 19 April 2019
5. Fraga Filho CAD (2019) Smoothed particle hydrodynamics: fundamentals and basic applications in continuum mechanics. Springer Nature, Basel
6. Olliff J, Alford B, Simkins DC Jr (2018) Efficient searching in meshfree methods. Comput Mech 62(6):1461–1483. https://doi.org/10.1007/s00466-018-1574-9
7. Koshizuka S, Nobe A, Oka Y (1998) Numerical analysis of breaking waves using the moving particle semi-implicit method. Int J Numer Methods Fluids 26(7):751–769. https://doi.org/10.1002/(SICI)1097-0363(19980415)26:7%3c751:AID-FLD671%3e3.0.CO;2-C
8. Slattery SR, Hamilton SP, Evans TM (2015) A modified moving least square algorithm for solution transfer on a spacer grid surface. In: Proceedings of ANS AMC2015—join international conference in mathematics and computation, Nashville, Tenessee, 2015. https://www.casl.gov/sites/default/files/docs/CASL-U-2015-0177-000.pdf. Accessed 19 April 2019
9. Jun S, Liu WK, Belytschko T (1998) Explicit reproducing kernel particle methods for large deformation problems. Int J Numer Methods Eng 41(1):137–166. https://doi.org/10.1002/(SICI)1097-0207(19980115)41:1%3c137:AID-NME280%3e3.0.CO;2-A
10. Wu N, Chen BS, Tsay T (2014) A review on the modified finite point method. Math Probl Eng 1–29:2014. https://doi.org/10.1155/2014/350364
11. Samuel H (2018) A deformable particle-in-cell method for advective transport in geodynamic modelling. Geophys J Int 214(3):1744–1773. https://doi.org/10.1093/gji/ggy231
12. Wang D, Hsiao F, Chuang C, Lee Y (2007) Algorithm optimization in molecular dynamics simulation. Comput Phys Commun 177(7):551–559. https://doi.org/10.1016/j.cpc.2007.05.009
13. Howard MP, Anderson JA, Nikoubashmana A, Glotzerb SC, Panagiotopoulos AZ (2016) Efficient neighbor list calculation for molecular simulation of colloidal systems using graphics processing units. Comput Phys Commun 203:45–52. https://doi.org/10.1016/j.cpc.2016.02.003
14. Oñate E, Idelsohn SR, Del Pin F, Aubry R (2004) The particle finite element method. An overview. Int J Computat Methods 1(2):267–307. https://doi.org/10.1142/s0219876204000204
15. Domínguez JM, Crespo AJC, Gómez-Gesteira M, Marongiu JC (2011) Neighbour lists in smoothed particle hydrodynamics. Int J Numer Methods Fluids 67(12):2026–2042. https://doi.org/10.1002/fld.2481
16. Viccione G, Bovolin V, Carratelli EP (2008) Defining and optimizing algorithms for neighbouring particle identification in SPH fluid simulations. Int J Numer Methods Fluids 58(6):625–638. https://doi.org/10.1002/fld.1761
17. Winchenbach R, Hochstetter H, Kolb A (2016) Constrained neighbor lists for SPH-based fluid simulations. In: Proceedings of eurographics/ACM SIGGRAPH symposium on computer animation, pp 49–56, Zurich, Switzerland, 2016. https://pdfs.semanticscholar.org/a7ef/0a369943a4cf616d96ccc85480de07606e69.pdf. Accessed 19 April 2019
18. Liu GR, Liu MB (2003) Smoothed particle hydrodynamics: a meshfree particle method. World Scientific, Singapore
19. Rajasekaran S, Reif J (2007) Handbook of parallel computing: models, algorithms and applications. Chapmann and Hall/CRC, Boca Raton
20. Jahanbakhsh E, Pacot O, Avellan F (2012) Implementation of a parallel SPH-FPM solver for fluid flows. Zetta Numer Simul Sci Technol 1:16–20
21. Shadloo MS, Oger G, Le Touzé D (2016) Smoothed particle hydrodynamics method for fluid flows, towards industrial applications: motivations, current state, and challenges. Comput Fluids 136(10):11–34. https://doi.org/10.1016/j.compfluid.2016.05.029
22. Leroy A (2014) Un nouveau modèle SPH incompressible: vers l'application à des cas industriels. Ph.D. Thesis, Université Paris-Est, France, 2014. http://www.theses.fr/2014PEST1065. Accessed 20 July 2018
23. Smojvera I, Ivančevića D (2017) Application of numerical methods in the improvement of safety of aeronautical structures. Transport Res Procedia 28:164–172. https://doi.org/10.1016/j.trpro.2017.12.182
24. Koshizuka S (2012) Current achievements and future perspectives on particle simulation technologies for fluid dynamics and heat transfer. J Nucl Sci Technol 48(2):155–168. https://doi.org/10.1080/18811248.2011.9711690
25. Cleary P, Joseph H, Vladimir A, Nguyen T (2002) Flow modelling in casting processes. Appl Math Model 26(2):171–190. https://doi.org/10.1016/S0307-904X(01)00054-3
26. Cleary PW, Savage G, Ha J, Prakash M (2014) Flow analysis and validation of numerical modelling for a thin walled high pressure die casting using SPH. Computat Particle Mech 1(3):229–243. https://doi.org/10.1007/s40571-014-0025-4
27. He Y, Zhou Z, Cao W, Chen W (2011) Simulation of mould filling process using smoothed particle hydrodynamics. Trans Non-Ferrous Met Soc China 21:2684–2692. https://doi.org/10.1016/S1003-6326(11)61111-4
28. Lluch E., Doste R., Giffard-Roisin S., This A., Sermesant M., Camara O., De Craene M., Morales H. Smoothed particle hydrodynamics for electrophysiological modeling: an alternative to finite element methods. In: Proceedings of the 9th international conference on functional imaging and modelling of the heart—FIMH 2017, Toronto, Canada, vol 141, pp 333–343. Springer, Berlin
29. Toma M (2018) The emerging use of SPH in biomedical applications. Signif Bioeng Biosci 1(1). https://pdfs.semanticscholar.org/bb44/ee090961aff9f80787ffdab66ae2558e6552.pdf. Accessed 20 July 2018
30. Shahriari S, Kadem L, Rogers BD, Hassan I (2012) Smoothed particle hydrodynamics method applied to pulsatile flow inside a rigid two-dimensional model of left heart cavity. Int J Numer Methods Biomed Eng 28(11):1121–1143. https://doi.org/10.1002/cnm.2482
31. Al-Saad M, Kulasegaram S, Bordas S (2018) Blood flow simulation using smoothed particle hydrodynamics. In: Proceedings of the VII European congress on computational methods in applied sciences and engineering—ECCOMAS Congress 2016, Vol 4, pp 8241–8246, Crete Island, Greece, 2016. https://www.eccomas2016.org. Accessed 20 July 2018
32. Hieber SE, Walther JH, Koumoutsakos P (2004) Remeshed smoothed particle hydrodynamics simulation of the mechanical behavior of human organs. Technol Health Care 12(4):305–314. https://content.iospress.com/articles/technology-and-health-care/thc00339. Accessed 19 April 2019

33. Gómez-Gesteira M, Rogers BD, Dalrymple RA, Crespo AJC (2010) State-of-the-art of classical SPH for free-surface flows. J Hydraul Res 48:6–27. https://doi.org/10.1080/00221686.2010.9641242
34. Xu X, Ouyang J, Yang B, Liu Z (2013) SPH simulations of three-dimensional non-Newtonian free surface flows. Comput Methods Appl Mech Eng 256:101–116. https://doi.org/10.1016/j.cma.2012.12.017
35. Ataie-Ashtiani B, Farhadi L (2006) A stable moving-particle semi-implicit method for free surface flows. Fluid Dyn Res 38(4):241–256. https://doi.org/10.1016/j.fluiddyn.2005.12.002
36. Gómez-Gesteira M, Dalrymple RA (2003) Using a three-dimensional smoothed particle hydrodynamics method for wave impact on a tall structure. J Waterw Port Coast Ocean Eng 130(2):63–69. https://doi.org/10.1061/(ASCE)0733-950X(2004)130:2(63)
37. Didier E, Neves DRCB, Martins R, Neves MG (2014) Wave interaction with a vertical wall: SPH numerical and experimental modelling. Ocean Eng 88:330–341. https://doi.org/10.1016/j.oceaneng.2014.06.029
38. Vacondio R, Rogers BD, Stansby PK, Mignosa P (2013) Shallow water SPH for flooding with dynamic particle coalescing and splitting. Adv Water Resour 58:10–23. https://doi.org/10.1016/j.advwatres.2013.04.007
39. Liang Q, Xia X, Hou J (2015) Efficient urban flood simulation using a GPU-accelerated SPH model. Environ Earth Sci 74(11):7285–7294. https://doi.org/10.1007/s12665-015-4753-4
40. Murotani K, Koshizuka S, Tasuku T, Shibata K, Mitsume N, Yoshimura S, Tanaka S, Hasegawa K, Nagai E, Fujisawa T (2014) Development of hierarchical domain decomposition explicit MPS method and application to large-scale tsunami analysis with floating objects. J Adv Simul Sci Eng 1(1):16–35. https://doi.org/10.15748/jasse.1.16
41. Tilke PG, Holmes DW, Williams JR (2010) Characterizing flow in oil reservoir rock using smooth particle hydrodynamics. In: AIP conference proceedings, vol 1254, p 278. https://doi.org/10.1063/1.3453824
42. Tartakovsky AM, Trask N, Pan K, Jones B, Pan W, Williams JR (2016) Smoothed particle hydrodynamics and its applications for multiphase flow and reactive transport in porous media. Comput Geosci 20(4):807–834. https://doi.org/10.1007/s10596-015-9468-9
43. Schnabel D, Özkaya E, Biermann D, Eberhard P (2018) Modeling the motion of the cooling lubricant in drilling processes using the finite volume and the smoothed particle hydrodynamics methods. Comput Methods Appl Mech Eng 329:369–395. https://doi.org/10.1016/j.cma.2017.09.015
44. Liu H, Arfaoui G, Stanic M, Montigny L, Jurkschat T, Lohner T, Stahl K (2018) Numerical modelling of oil distribution and churning gear power losses of gearboxes by smoothed particle hydrodynamics. In: Proceedings of the institution of mechanical engineers, part J: journal of engineering tribology. https://doi.org/10.1177/1350650118760626
45. Dong XW, Liu GR, Li Z, Zeng W (2016) A smoothed particle hydrodynamics (SPH) model for simulating surface erosion by impacts of foreign particles. Tribol Int 95:267–278. https://doi.org/10.1016/j.triboint.2015.11.038
46. Abdelrazek AM, Kimura I, Shimizu Y (2014) Comparison between SPH and MPS methods for numerical simulations of free surface flow problems. J Jpn Soc Civ Eng Ser. B1 (Hydraul Eng) 70(4):I_67–I_72. https://doi.org/10.2208/jscejhe.70.i_67
47. Fraga Filho CAD, Chacaltana JTA (2015) Study of fluid flows using Smoothed Particle Hydrodynamics: the modified pressure concept applied to quiescent fluid and dam breaking. In: Proceedings of the XXXVI Iberian Latin American congress on computational methods in engineering—CILAMCE 2015, Rio de Janeiro, Brazil. https://doi.org/10.20906/cps/cilamce2015-0071
48. Fraga Filho C, Chacaltana JTA, Pinto WJN (2018) Meshless Lagrangian SPH method applied to isothermal lid-driven cavity flow at low-*Re* numbers. Comput Part Mech. https://doi.org/10.1007/s40571-018-0183-x
49. Fu L, Ji Z, Hu XY, Adams NA (2019) Parallel fast-neighbor-searching and communication strategy for particle-based methods. Eng Comput. https://doi.org/10.1108/ec-05-2018-0226
50. Fu L, Litvinov S, Hu XY, Adams NA (2017) A novel partitioning method for block-structured adaptive meshes. J Comput Phys 341:447–473. https://doi.org/10.1016/j.jcp.2016.11.016