Adaptive Smoothed Particle Hydrodynamics Neighbor Search Algorithm for Large Plastic Deformation Computational Solid Mechanics

Kirk Fraser[1,2]

[1] *University of Quebec at Chicoutimi*

[2] *Predictive Engineering, Quebec Canada*

Abstract

Smoothed Particle Hydrodynamics (SPH) has quickly become one of the most popular mesh-free methods since its introduction in 1977. In the recent years, a great amount of research has been focused on addressing some of the common computational time associated with the SPH method. One of the remaining hurdles is the long computational associated with building the neighbor list. Because of the nature of the original SPH codes (astropyshics), the neighbor search is commonly performed for every element in the domain at each time step. In this work, we develop an optimized neighbor search algorithm that is suitable for deployment on NVidia graphics cards (GPU). The SPH code is written using CUDA Fortran. The algorithm can be used for large plastic deformation computational solid mechanics (CSM) problems. The search uses an adaptive algorithm that updates the neighbor list for individual SPH elements depending whether a plastic strain increment threshold is surpassed. The neighbor list as well as the inter-particle spacing (r_{ij}) is re-used for elements that do not surpass the search update criteria. Although in this work we use a Cell based search, the algorithm can be easily adapted for the Direct Search, the Verlet List or a Tree Sort approach. Monaghan's artificial stress term is added to the momentum equation to suppress the common tensile instability. The XSPH approach is used to update the positions of the SPH elements. The algorithm is shown to reduce the overall computation time by up to 70% without loss of accuracy for CSM simulations when compared with the non-adaptive search method.

Introduction

Smoothed particle hydrodynamics (SPH) is a meshless numerical method that was pioneered by Gingold and Monaghan [1] and at the same time (although independently) by Lucy [2] for astrophysics problems. The method is well adapted to simulation of solid mechanics problems. Because of its meshfree nature, large plastic deformation problems can be treated with far greater ease than by conventional simulation approaches like the finite element method. SPH has become popular for free surface flow simulations such as the impact of waves on offshore structures, the prediction of the flow path of lava (Herault, Billota and Dalrymple [3]), slamming and ditching of space craft on ocean or lake surfaces [4] as well as many other fluid type simulations. SPH is even making its mark in the gaming community for rendering realistic graphics of water, smoke, mud, etc. Recently, there has been a big push in the high performance computing (HPC) industry for increasingly larger scale numerical simulations. In fact, the 1 billion element barrier was recently broken by Dominguez et al. [5]. The authors are the main contributors to the SPHysics (Dual-SPHysics) code. They use a 64 GPU cluster to simulate the multi-billion element simulation.

Because the SPH method is meshfree, spatial derivatives are evaluated by interpolating with neighboring nodes that are within a radius of influence of the concerned particle. Searching for the neighbors of the particles is a laborious task that can account for a significant percentage of the calculation time per time step. Typically the neighbor search is carried out every time step for all the particles in the simulation domain.

In this paper we develop an adaptive neighbor searching approach that is best suited for implementation on NVidia graphics cards using the CUDA Fortan programming language. Ideally, elements in regions in the domain that are not experiencing significant deformation do not need to have their neighbor list updated.

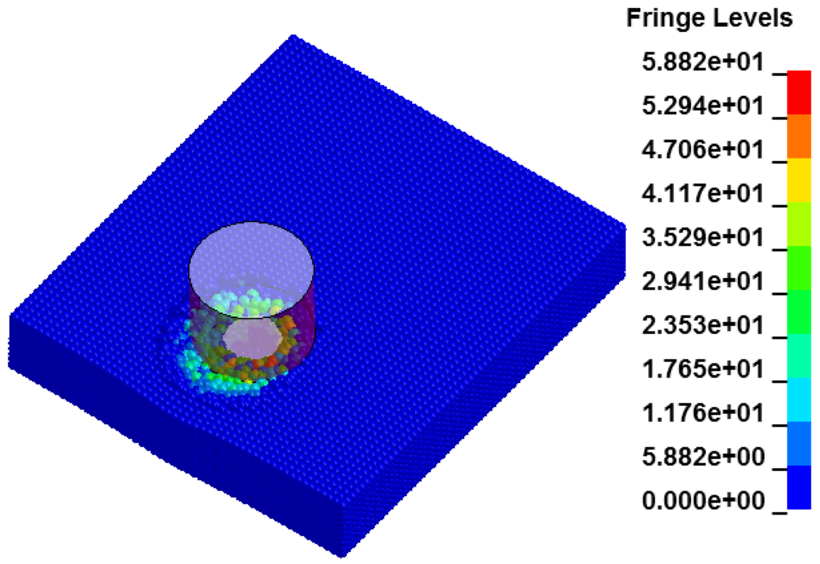


Figure 1 – Typical FSW Simulation

Our method uses an effective plastic strain increment threshold to build a list of elements that will have their neighbor list updated. The approach is especially beneficial for SPH simulations where there is only a small region in the domain that is experiencing large deformation. Metal forming, cutting, machining and friction stir welding are all good examples of simulations that could benefit from an adaptive neighbor search. For example, in friction stir welding (FSW), the plates to be welded are modeled with SPH, only a small area is plastically deforming as shown in Figure 1. Fraser et al. [6] use SPH to evaluate the probability of the development of defects during the FSW process. Pan et al. [7] have also used SPH to evaluate the evolution of the micro structure in a FSW joint.

The SPH Method

The basic premise of the SPH method is to reduce a set of partial differential equations (PDE) to a set of ordinary differential equations by an approximate interpolation formulation. A continuous function is approximated by an interpolant (convolution integral):

$$f(\bar{x}) = \int f(\bar{x}')W(\bar{x} - \bar{x}', h)d\bar{x}' \qquad \textbf{Eq 1}$$

$W(\bar{x} - \bar{x}', h)$ is called the kernel function, also commonly referred to as the smoothing function. It is a function of the spatial distance between the point at which the function is to be calculated (calculation point, $\bar{x}$) and the interpolation location ($\bar{x}'$) and h is the smoothing length. The kernel is the key to the SPH method. The continuous SPH interpolation equation must then be written for a set of discrete material points:

$$f(x_i{}^{\alpha}) = \sum_{j=1}^{N_i} \frac{m_j}{\rho_j} f(x_j{}^{\alpha}) W(r, h) \qquad \textbf{Eq 2}$$

x_i is the spatial location vector for particle "i" and x_j for the jth particle, h is the smoothing length that determines the size of the influence domain of the "jth" particles on a particle "i." While m_j, ρ_j are the mass and density of a "jth" particle and $r = |x_i{}^{\alpha} - x_j{}^{\alpha}|$. $W(r,h)$ is again the interpolation kernel, which will be written as W_{ij} in the future. The sum is taken over the total number (N_i) of "j" particles within the influence domain of "i", these are termed the neighbors of the "ith" particle. Figure 2 gives a graphical interpretation of the influence domain.

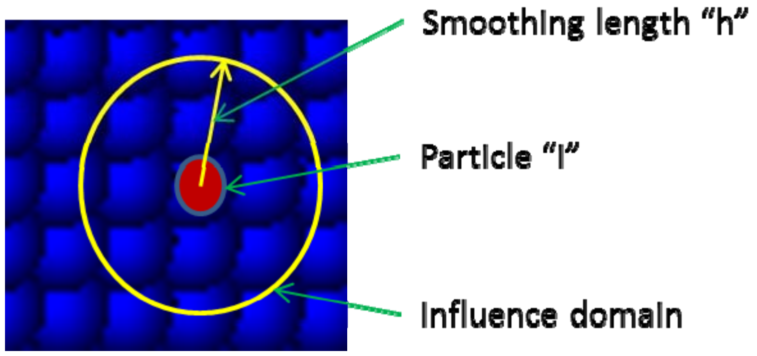


Figure 2 – Influence Domain of Particle "i" on the "jth" Particles

A set of Lagrangian conservation equations must be solved for CSM simulations. The first conservation equation is that of mass:

$$\frac{d\rho}{dt} + \nabla \cdot \rho\bar{v} = 0 \qquad \textbf{Eq 3}$$

Where ρ is the density of the material point, $\bar{v}$ is the velocity and t is time. Using the SPH interpolation, the discrete equation for conservation of mass is then:

$$\frac{d\rho_i}{dt} = \rho_i \sum_{j=1}^{N_i} \frac{m_j}{\rho_j}\left(v_i{}^{\beta} - v_j{}^{\beta}\right)\frac{\partial W_{ij}}{\partial x_i{}^{\beta}} \qquad \textbf{Eq 4}$$

$\frac{\partial W_{ij}}{\partial x_i{}^{\beta}}$ is the first derivative of the smoothing function. Conservation of momentum for a solid continuum body is given by:

$$\frac{d\bar{v}}{dt} = \frac{1}{\rho}\nabla \cdot \bar{\bar{\sigma}} + \bar{g} + \bar{F}_{ext} \qquad \textbf{Eq 5}$$

Where $\bar{\bar{\sigma}}$ is the total stress tensor, $\bar{g}$ is the gravity vector and $\bar{F}_{ext}$ is an external force vector that can be due to boundaries. Again, using the SPH method, this equation can be written as:

$$\frac{dv_i{}^{\alpha}}{dt} = m_j \sum_{j=1}^{N_i}\left(\frac{\sigma_i{}^{\alpha\beta} + \sigma_j{}^{\alpha\beta}}{\rho_i\rho_j} + \Pi_{ij}\right)\frac{\partial W_{ij}}{\partial x_i{}^{\beta}} + g_i{}^{\alpha} + \left(F_{ext}\right)_i{}^{\alpha} \qquad \textbf{Eq 6}$$

Π_{ij} is the artificial viscosity term that is used for problems involving the propagation of shocks. Different forms of conservation of mass and momentum are possible (not discussed in this paper). A smoothing function is needed to evaluate the SPH sums. The cubic B-spline function as proposed by Gingold and Monaghan [8, 9] is by far the most popular smoothing function:

$$W(R,h) = \alpha_d \begin{cases} \frac{2}{3} - R^2 + \frac{1}{2}R^3 \; for\ 0 \le R < 1 \\ \frac{1}{6}(2-R)^3 \quad for\ 1 \le R < 2 \\ \quad 0 \qquad\quad for\ R \ge 2 \end{cases} \qquad \textbf{Eq 7}$$

α_d is a dimension specific constant and $R = \left|x_i{}^{\alpha} - x_j{}^{\alpha}\right|/h$. The popularity of this kernel comes from the fact that it satisfies all the requirements, is relatively easy to implement in common programming languages (although the piecewise nature requires the use of conditional statements) and closely resembles the Gaussian kernel.

The total stress tensor is composed of a deviatoric stress, $\bar{\bar{S}}$, and a hydrostatic stress term, $p\bar{\bar{\delta}}$:

$$\bar{\bar{\sigma}} = \bar{\bar{S}} - p\bar{\bar{\delta}} \qquad \textbf{Eq 8}$$

The deviatoric stress is determined by integrating an objective rate equation such as that of Jaumann:

$$\dot{S} = 2G(\dot{\varepsilon} - tr(\dot{\varepsilon})\delta) + S\Omega^T + \Omega S \qquad \textbf{Eq 9}$$

Where $\dot{\bar{\varepsilon}}$ is the strain rate, $\bar{\Omega}$ is the spin tensor and they are given by:

$$\dot{\varepsilon} = \frac{1}{2}(\nabla \cdot v + \nabla \cdot v^T) \ and \ \Omega = \frac{1}{2}(\nabla \cdot v - \nabla \cdot v^T) \qquad \textbf{Eq 10}$$

The pressure at the material point is found using an appropriate equation of state. Plasticity is considered by using the radial return approach. For an elastic-plastic material with isotropic hardening, a trial stress ($\bar{\sigma}^{trial}$) can be found from the deviatoric stress by initially assuming elastic behavior:

$$\sigma^{trial} = \sqrt{\frac{3}{2}S:S} \qquad \textbf{Eq 11}$$

If the trial stress surpasses the yield stress (σ_y) of the material, the effective plastic strain increment is found from:

$$\Delta\varepsilon^p = \frac{\sigma^{trial} - \sigma_y}{3G + H} \qquad \textbf{Eq 12}$$

G is the shear modulus, $H = \frac{EE^T}{E+E^T}$ is the hardening modulus and E^T is the tangent modulus. Once the plastic strain increment is found, the yield stress is updated:

$$\sigma_y{}^{n+1} = \sigma_y{}^n + H\Delta\varepsilon^p \qquad \textbf{Eq 13}$$

Then the deviatoric stresses are scaled back to the yield surface using:

$$S^{n+1} = \frac{\sigma_y{}^{n+1}}{\sigma^{trial}} S^n \qquad \textbf{Eq 14}$$

Tensile instability in SPH is a known problem in the CSM method. The problem manifests itself as a pairwise clumping of particles that eventually leads to a non-physical numerical fracture of the material. Monaghan [10] and Gray et al. [11] have proposed a simple and effective solution to the tensile instability problem by adding an artificial stress term in the momentum equation. The artificial stress acts as a repulsive force when the particle pair force is due to tension. The momentum equation is modified to include the artificial stress term:

$$\frac{dv_i{}^\alpha}{dt} = \sum_{j=1}^{N_i} m_j \left(\frac{\sigma_i{}^{\alpha\beta}}{\rho_i{}^2} + \frac{\sigma_j{}^{\alpha\beta}}{\rho_j{}^2} + R_{ij}{}^{\alpha\beta} f^n + \Pi_{ij}\right)\frac{\partial W_{ij}}{\partial x_i{}^\beta} + g_i{}^\alpha + (F_{ext})_i{}^\alpha \qquad \textbf{Eq15}$$

f is a function that increases as the separation distance between two particle is decreasing. The exponent on the function, n, is taken as 6.0 in this work. Monaghan proposes the function to be of the form:

$$f = \frac{W(r_{ij})}{W(\Delta p)}$$ **Eq16**

$W(r_{ij})$ is an appropriate smoothing function that does not need to be the same as that used in the evaluation of the conservation equations, r_{ij} is the radial distance between two particles and $W(\Delta p)$ is the value of the chosen smoothing function evaluated at the average particle spacing. The artificial stress term, $R_{ij}^{\alpha\beta}$, is defined as:

$$R_{ij}^{\alpha\beta} = R_i^{\alpha\beta} + R_j^{\alpha\beta}$$ **Eq17**

$$R_i^{\alpha\beta} = \begin{cases} -\epsilon\sigma_i^{\alpha\beta}, & \sigma_i^{\alpha\beta} > 0 \\ 0, & \sigma_i^{\alpha\beta} \leq 0 \end{cases}$$ **Eq18**

ϵ is a constant that depends on the simulation, which is typically between 0.1 and 0.5 (we use 0.3 in this work). One of the drawbacks to this approach is that the stress state for each particle pair must be monitored. A conditional statement is required to assign values to $R_{ij}^{\alpha\beta}$.

Standard Neighbor Search

One of the most common searching methods involves binning the particles into cubes with length of 2*h*. The search is then accomplished by only searching to the neighboring bins of the bin in which the concerned particle resides. Figure 3 shows a schematic of a two dimensional domain that has been set up for a Cartesian grid for the purpose of particle binning. In this situation, the search is only carried out with particles in the eight neighbor bins in 2D or 26 in 3D. This method is a drastic improvement over the direct search without being overly complicated.

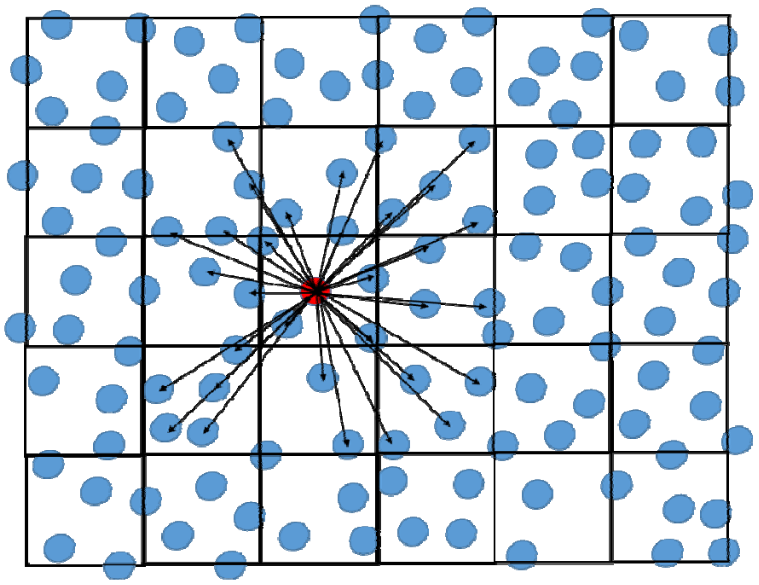


Figure 3 – Schematic of Cell Search Method

The implementation of the neighbor search on the GPU is slightly different because we cannot take advantage of the pairwise interactions. Typically, for a CPU algorithm, the neighbor list will only list the interacting particles once (this is done by requiring that *j* be greater than *i* in the search). And as such only 23 cells must be searched in the cell search method. Conversely the Neighbor search on the GPU is performed for all the pairs; this means that all 26 neighboring cells are searched.

Most commonly, the neighboring pairs are organized into a linked list data structure. The linked list does not waste any memory and is a very efficient data structure.

Adaptive Neighbor Search

Very little work has been undertaken in the research community for adaptive neighbor searching. To our knowledge, the only group that has worked on this is Pelfrey and House [12]. They proposed an adaptive neighbor search for fluid dynamic simulations. Their algorithm is mainly intended to speedup SPH fluid graphics rendering (for video games or cinematography). One of the drawbacks of their method (and most likely why very little work has been done in this field) is that for a typical fluid simulation, not re-evaluating the smoothing function for each particle can introduce significant error. They mention in their publication that the results are "visually" similar with the adaptive search. They do not compare their method quantitatively.

The adaptive neighbor search that we propose is based on the cell search method. Some slight changes are needed to the standard algorithm. In this work, we do not use a linked list data structure. Insertion and deletion from a linked list is somewhat complicated. We use instead a static 2D array with the *i*th columns listing the neighbors for the *i*th particle (see Figure 4). This method is less efficient from a memory standpoint because we must fix the number of memory allocated for the number of rows in the array. That being said, the adaptive algorithm can be adapted to different data structures as well as different search methods (Verlet, Octree…).

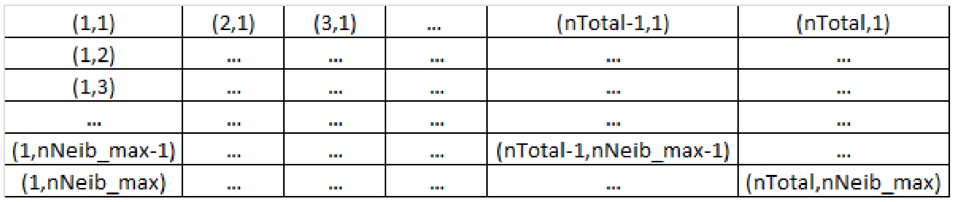
The adaptive search process is as follows:

- Perform a neighbor search on the whole domain for the first cycle. This involves
    - → Finding the bounds of the simulation
    - → Determining the number of cells in each dimension
    - → Binning particles into their cells
    - → Searching for neighbors
- Calculate conservation of mass, momentum, material models, external forces, etc
    - → The effective plastic strain increment is determined based on the material model
- Build a list of elements with an effective plastic strain ($\Delta\varepsilon^p$) that surpasses a threshold, *adapt_thresh*. This is done by:
    - → Cycle through the elements, if an elements $\Delta\varepsilon^p$ is greater than *adapt_thresh* place the element id in a list called *AdaptSearchList*
    - → Mark the element in another array called *AdaptNode*, this array can be used to show which elements are adapted in the post-processor
    - → Increment a (*nNodeAdaptSearch*) counter to keep track of how many elements are in the *AdaptSearchList*. On the GPU, we use an atomicinc() function to prevent incorrect incrementation of *nNodeAdaptSearch*.
- Pass *nNodeAdaptSearch* and *AdaptSearchList* to the cell search subroutine. The search is performed only for the nodes in the *AdaptSearchList*
    - → The neighbors are updated

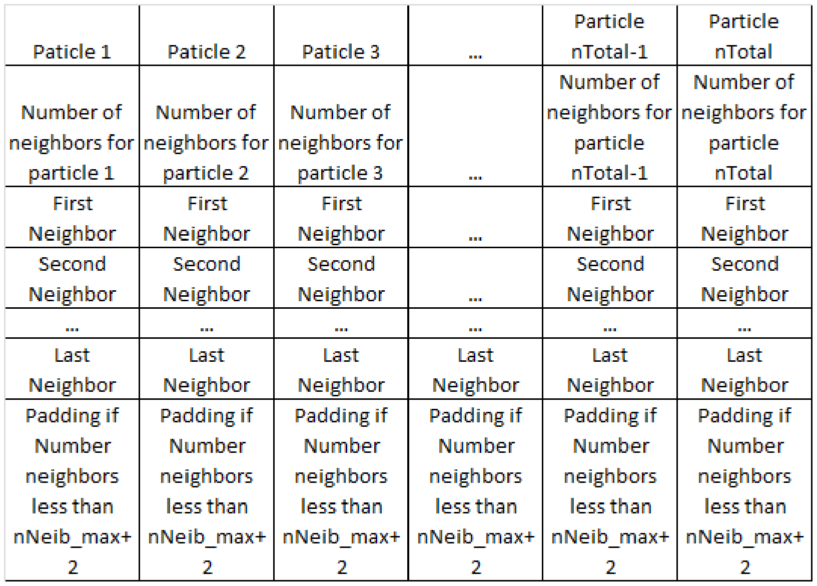
→ The smoothing function values are updated

- The neighbor list and smoothing function values are kept and re-used for the SPH elements that are not on the *AdaptSearchList*

The *AdaptNode* array is used to visualize which elements are processed in the adaptive search. This is an important consideration to ensure that the field of elements adapted is not overly discontinuous.



(1,1)	(2,1)	(3,1)	...	(nTotal-1,1)	(nTotal,1)
(1,2)	...	...	...	...	...
(1,3)	...	...	...	...	...
...	...	...	...	...	...
(1,nNeib_max-1)	...	...	...	(nTotal-1,nNeib_max-1)	...
(1,nNeib_max)	...	...	...	...	(nTotal,nNeib_max)



Paticle 1	Paticle 2	Paticle 3	...	Particle nTotal-1	Particle nTotal
Number of neighbors for particle 1	Number of neighbors for particle 2	Number of neighbors for particle 3	...	Number of neighbors for particle nTotal-1	Number of neighbors for particle nTotal
First Neighbor	First Neighbor	First Neighbor	...	First Neighbor	First Neighbor
Second Neighbor	Second Neighbor	Second Neighbor	...	Second Neighbor	Second Neighbor
...	...	...	...	...	...
Last Neighbor	Last Neighbor	Last Neighbor	Last Neighbor	Last Neighbor	Last Neighbor
Padding if Number neighbors less than nNeib_max+2	Padding if Number neighbors less than nNeib_max+2	Padding if Number neighbors less than nNeib_max+2	Padding if Number neighbors less than nNeib_max+2	Padding if Number neighbors less than nNeib_max+2	Padding if Number neighbors less than nNeib_max+2

Figure 4 – Neighbor Array Layout, *Neib(nTotal,nNeib_max)*

The first important notion that will be mentioned is that often we must include a large simulation domain in CSM problems. More often than not, there are large portions of the domain that are responding elastically. This is a stark contrast to a fluid simulation where the whole domain is "active".

The next concept is that for infinitesimal strains the Piola-Kirchhoff stress tensor is equivalent to the Cauchy stress tensor. This is important because the total Lagrangian formulation in SPH classically uses the Piola-Kirchhoff stress tensor. This concept leads us to the understanding that for infinitesimal elastic strains, the total Lagrangian formulation can be written using the Cauchy stress tensor and the deviatoric stresses can be found from integrating the Jaumman rate equation.

The region of the domain that is deforming elastically does not get their neighbor list or smoothing function updated, this is equivalent to using the total Lagrangian formulation in the elastic zones. The elements that are undergoing plastic deformation will have their neighbors list and smoothing function re-evaluated. This is akin to using the Eulerian formulation in the plastic areas. As we know, the Eulerian formulation suffers from tensile instability, so we include the artificial stress terms to stabilize the material under tension.

SPH Code Verification

Before testing the performance of the adaptive search algorithm, we will first show that the SPH code is able to reproduce meaningful results. A tensile test of a cylindrical specimen will be used to evaluate the SPH code. A baseline case is setup and run in LS-DYNA® using finite elements, the geometry and properties of the steel cylinder is shown in Figure 5.

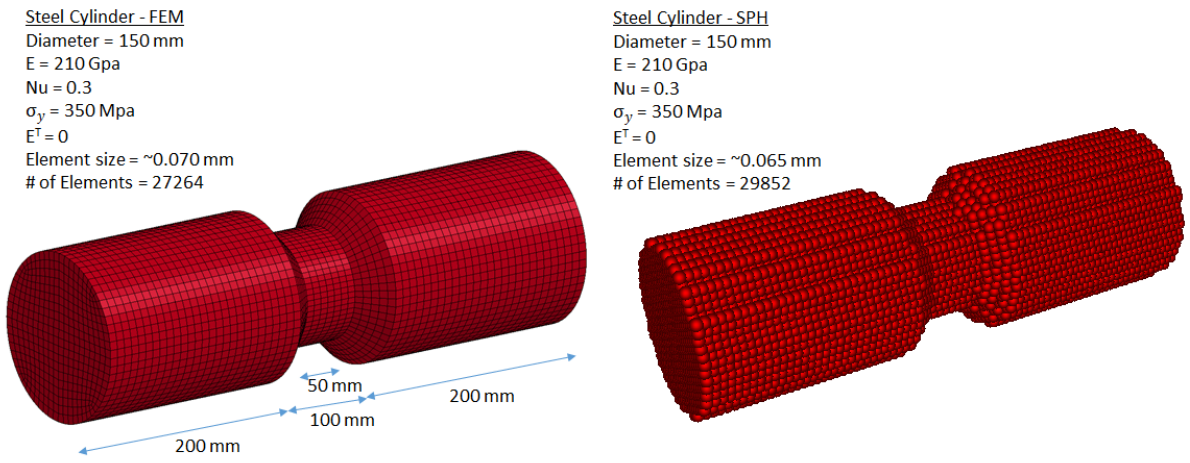


Figure 5 – FEM and SPH Steel Cylinder Dimensions and Properties

One end of the cylinder is held fixed, while the other end of the cylinder is given a prescribed velocity that increases from zero to 1.0 m/s at *t*=0.005s, the velocity is held constant at 1 m/s from 0.005s to 0.01s (simulation end time). The results from the FEM model at *t*=0.01s are shown in Figure 6, we can see that the cylinder yields at the reduced area. The maximum effective plastic strain found in the model is 0.281. Figure 7 shows the results for the SPH code, the max effective plastic strain was found to be 0.286. Results of the LS-DYNA simulations is shown using LS-PrePost® 4.1, the results from the SPH code are shown using Paraview v3.12.0. Figure 8 gives a comparison of the effective plastic strain for an element located at the center of the steel bar (location of max plastic strain). We can see a very good correlation between the FEM model and the SPH code.

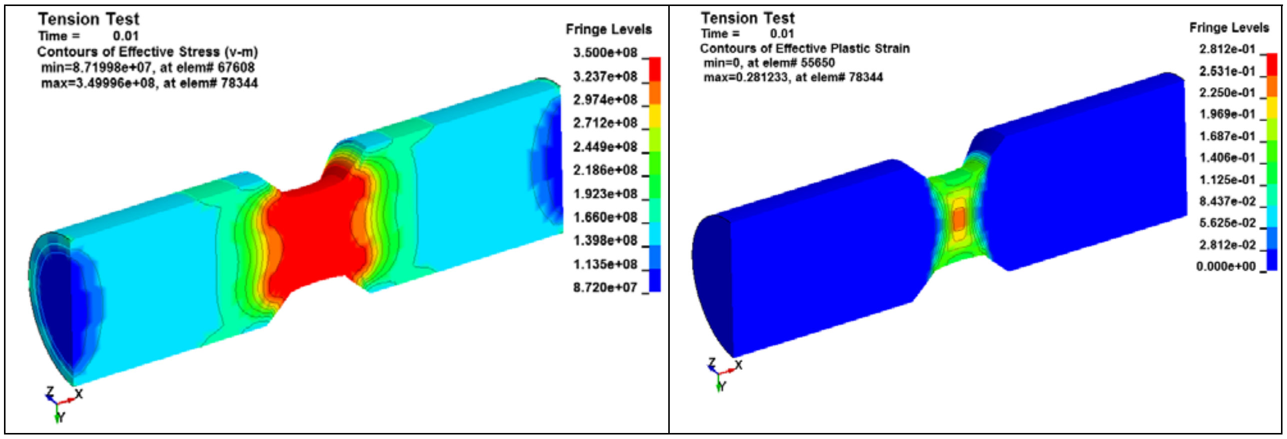


Figure 6 – Tension Test in LS-DYNA – von Mises Stress and Effective Plastic Strain

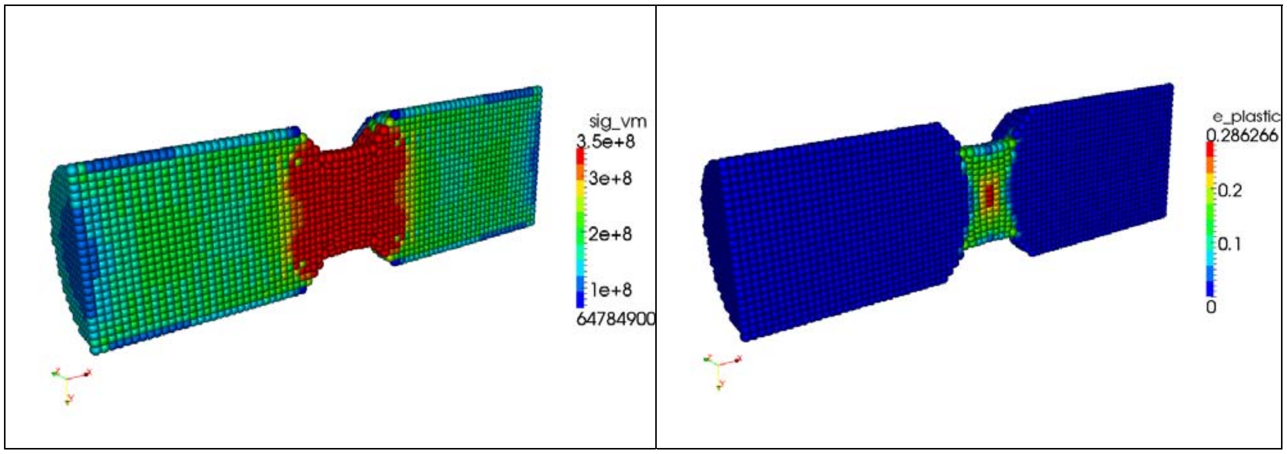


Figure 7 – Tensile Test in SPH Code – von Mises Stress and Effective Plastic Strain

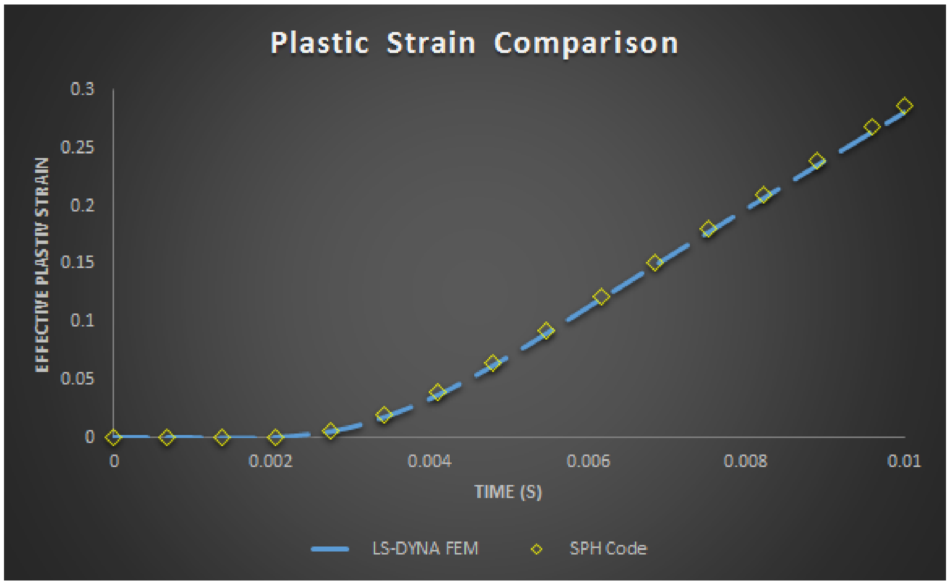


Figure 8 – Comparison of Effective Plastic Strain at Center of Specimen

Adaptive Search Performance Test

We have shown that the SPH code is able to produce viable results for elastic-plastic materials under tension. Now we will run a larger model and test the performance improvement with the adaptive neighbor search.

For the adaptive search to be beneficial, we want to have a problem domain that is subject to elastic deformation over a significant region and a small region undergoing plastic deformation. We will use a model of a steel bullet penetrating a large aluminum plate. The SPH model is shown in Figure 9. The bullet is given an initial velocity of 400 m/s, the simulation is run for 0.001s, this is long enough for the bullet to penetrate the plate.

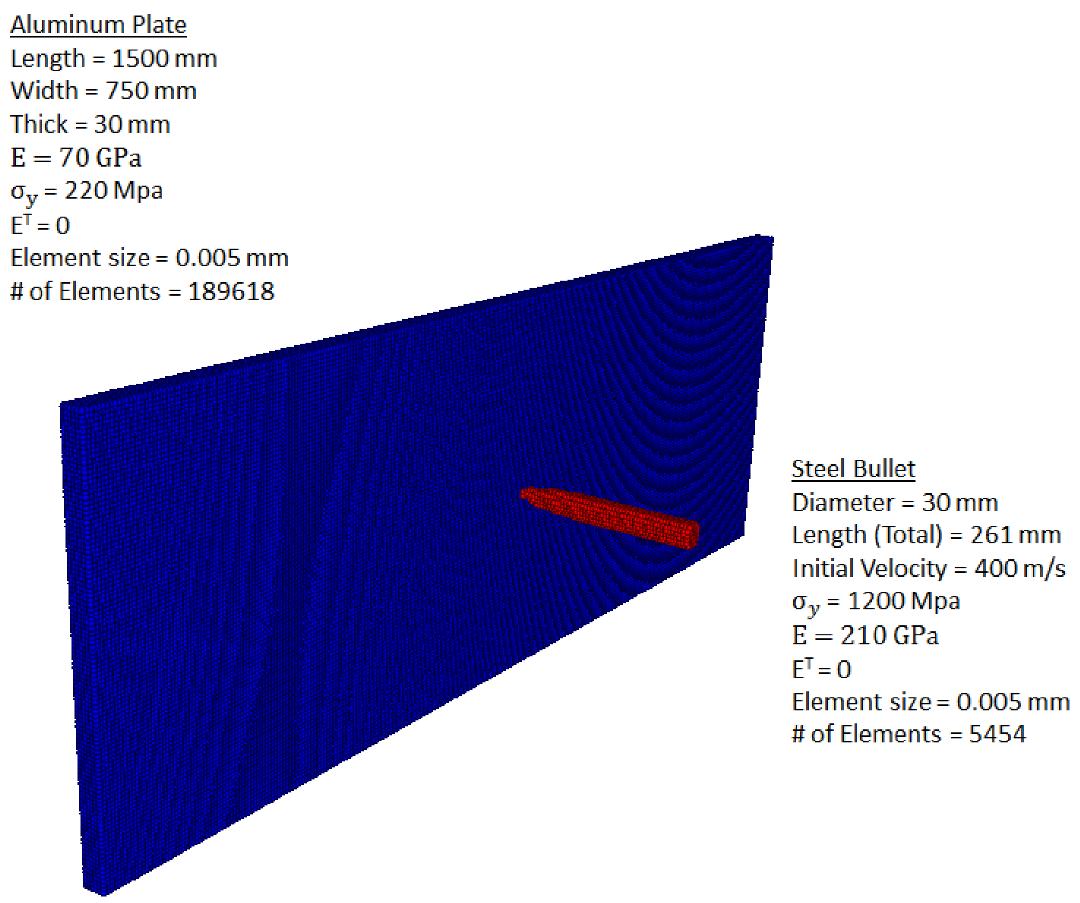


Figure 9 – Penetration Model Used to Test the Adaptive Search Performance

A total of five versions of the penetration problem are run:

1- Standard Eulerian kernel with neighbor search and smoothing evaluated every cycle
2- Total Lagrangian kernel
3- Adaptive search with $\Delta\varepsilon^p$ threshold (*adapt_thresh)* set to 1.0E-4
4- Adaptive search with $\Delta\varepsilon^p$ threshold (*adapt_thresh)* set to 1.0E-5
5- Adaptive search with $\Delta\varepsilon^p$ threshold (*adapt_thresh)* set to 1.0E-6

The longest simulation time is as expected for the standard search every cycle. The fastest is when the search is only performed at the first time step (Total Lagrangian). In this study, we found a speedup of 2.1x between these two. Note that the total Lagrangian case is only used to give us a maximum possible speedup, the results are not meaningful and are as such not further discussed. The three remaining versions use the adaptive search algorithm with different threshold criteria levels to trigger an SPH element to be included in the search. For the case of the threshold set at 1.0E-4, the residual velocity of the bullet does not match up (123.7 m/s compared to 141.0 m/s). Both the 1.0E-5 and 1.0E-6 threshold settings result in a residual velocity that matches well with the standard search method. A comparison of the simulation times and the residual velocities are shown in Table 1. Figure 10 gives a comparison of the relative speedup compared to the standard Eulerian kernel. A speedup of 70% and 40% for *adapt_thresh* 1.0E-5 and 1.0E-6 respectively were obtained.

Table 1 – Comparison of Results for the Different Cases

Method	Time (seconds)	Speedup from Eulerian Case	Residual Velocity (m/s)
Eulerian (No Adapt)	372	1.0x	141.0
Total Lagrangian	175	2.1x	N/A
Adapt – $\Delta\varepsilon^p$ Threshold = 1.0E-4	195	1.9x	123.7
Adapt – $\Delta\varepsilon^p$ Threshold = 1.0E-5	221	1.7x	141.0
Adapt – $\Delta\varepsilon^p$ Threshold = 1.0E-6	274	1.4x	141.0

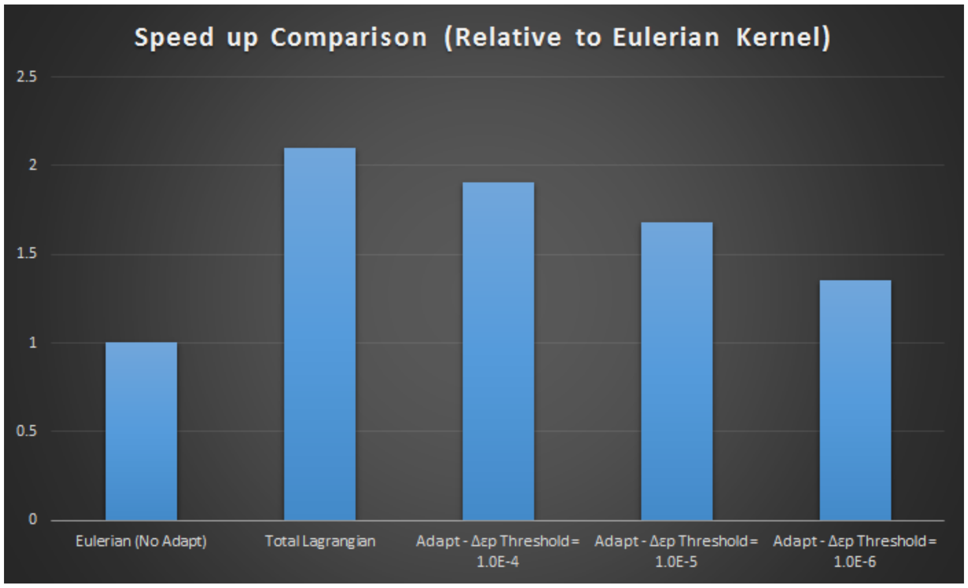


Figure 10 – Speedup Comparison with Adaptive Search Method

To fully test the method, a point on the center of the tailing end of the bullet is used to compared the velocity of the bullet throughout the simulation for the standard search method (solid line) and the adaptive method with *adapt_thresh = 1.0E-6* (hollow square markers), the comparison is shown in Figure 11. The residual velocity of the bullet for the two cases is almost identical.

The power of the algorithm comes from the fact that only SPH elements that truly need their neighbors and smoothing function values updated are treated as the simulation progresses. Figure

12 shows the elements that are included in the search for four different times in the simulation (showing *AdaptNode* fringe). We can plainly see the region where the search is to be carried out is progressively changed as the simulation advances. Figure 13 shows a comparison of the vonMises stresses for the standard search case and for the adaptive search with *adapt_thresh* = 1.0E-06.

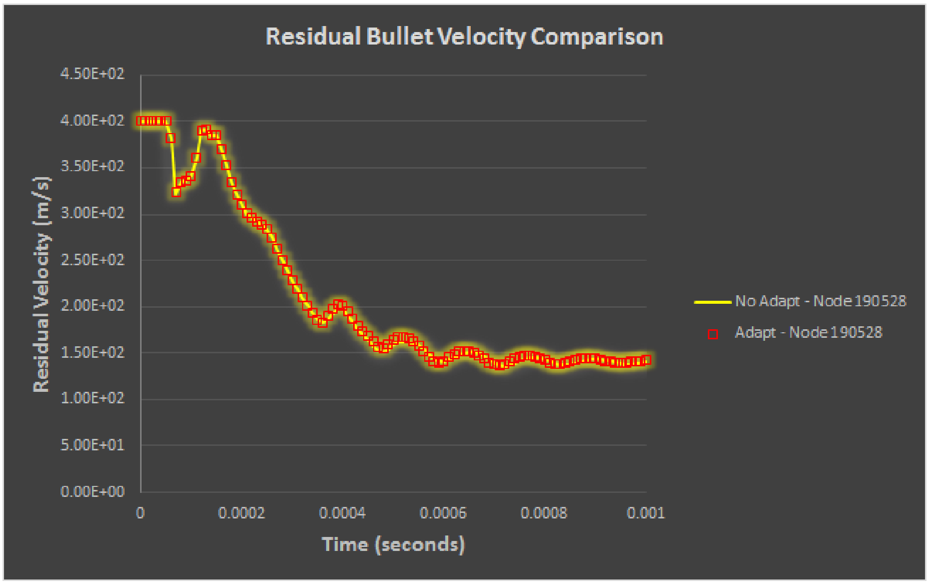


Figure 11 – Residual Bullet Velocity Comparison

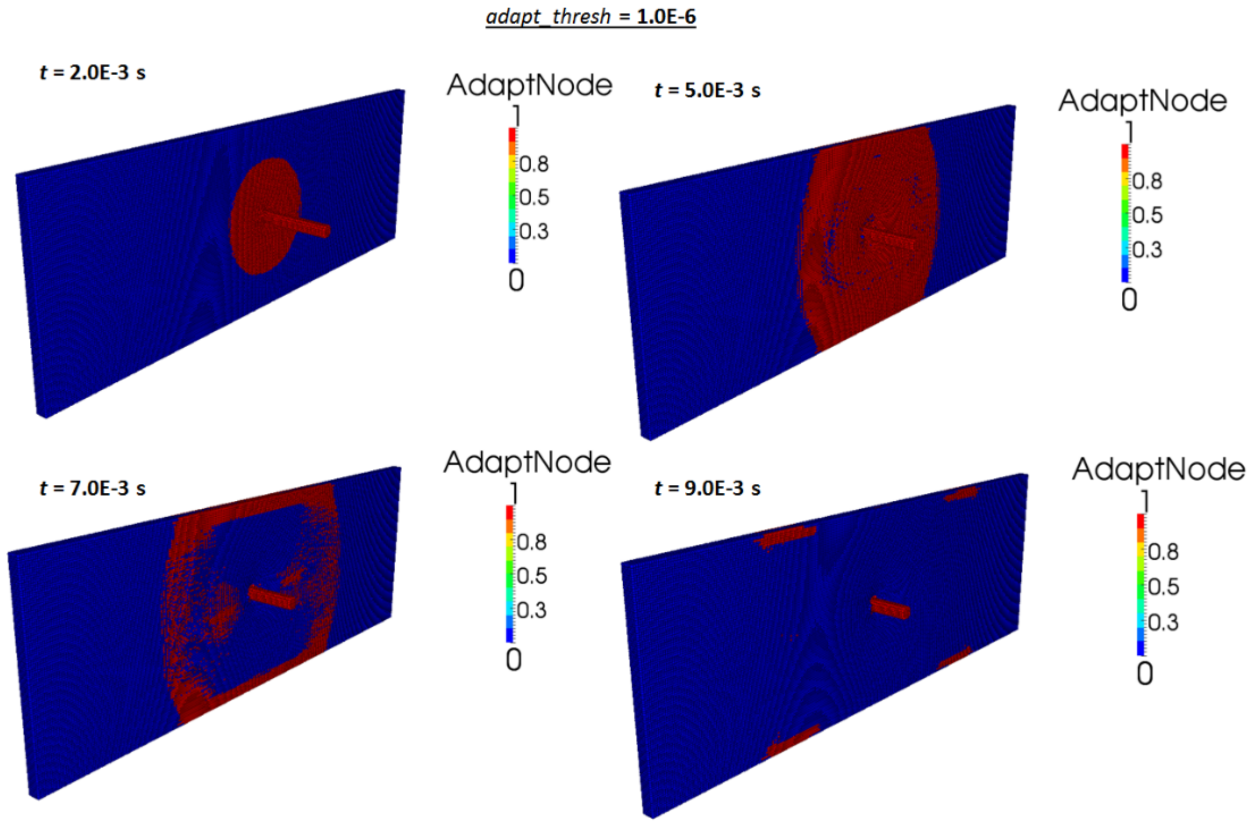


Figure 12 – SPH Elements where the Adaptive Search is applied (*adapt_thresh* = 1.0E-06)

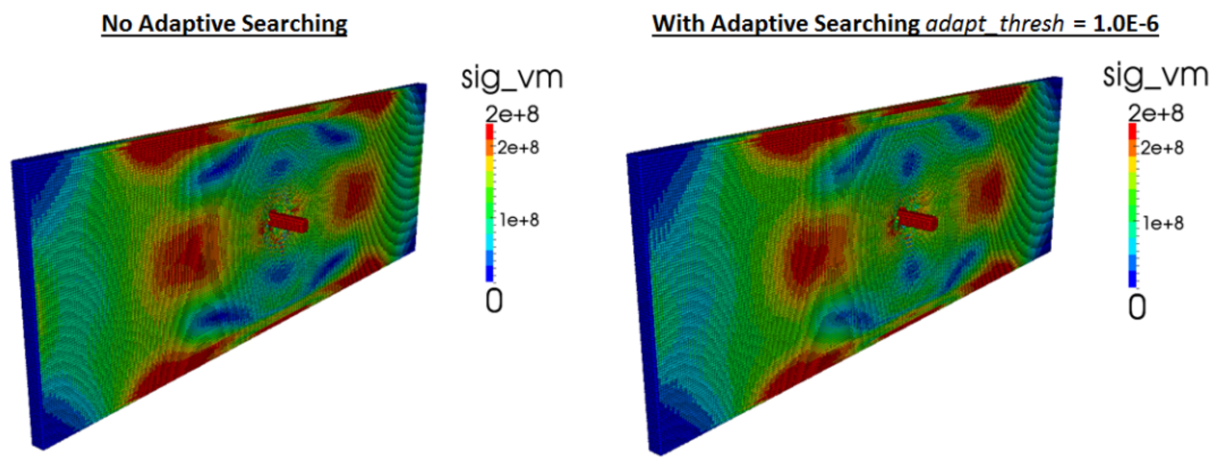


Figure 13 – Comparison of VonMises Stresses

Conclusion

In this work, we have developed an adaptive search method for smoothed particle hydrodynamics for large deformation computational solid mechanics problems. We have shown that the results are qualitatively and quantitatively the same using the adaptive method as they are with the standard search method.

Depending on the precision that is sought from the simulation, an improvement of the total simulation time of 40% to 70% can be attained by changing the threshold criterion for the adaptive search. Using the effective plastic strain increment is a natural threshold criteria for computational solid mechanics problems. $\Delta\varepsilon^p$ is required to be calculated whether the adaptive method is used or not, thus not adding to the computational time. As the *adapt_thresh* is lowered (goes to zero in the limit to obtain the standard non-adaptive search), the results become closer to the standard case.

This work was supported by Predictive Engineering, CQRDA, GRIPS, CURAL and REGAL. Thanks to the technical support of Yves Chiricota for his assistance with the development of the CUDA code. Also thanks to Laszlo Kiss, Lyne St-Georges, Laurie Bedard and Ian Fraser.

References

[1] Gingold RA, Monaghan JJ. Smoothed particle hydrodynamics: theory and application to non-spherical stars. Mon Not R Astron Soc. 1977 181 375–89.
[2] Lucy LB. A numerical approach to the testing of the fission hypothesis. Astron. 1977;J. 82 1013–24.
[3] Hérault A. SPH on GPU with CUDA. Journal of Hydraulic Research. 2009;48:000.
[4] Skillen A, Lind S, Stansby PK, Rogers BD. Incompressible smoothed particle hydrodynamics (SPH) with reduced temporal noise and generalised Fickian smoothing applied to body–water slam and efficient wave–body interaction. Computer Methods in Applied Mechanics and Engineering. 2013;265:163-73.
[5] Domínguez JM, Crespo AJC, Valdez-Balderas D, Rogers BD, Gómez-Gesteira M. New multi-GPU implementation for smoothed particle hydrodynamics on heterogeneous clusters. Computer Physics Communications. 2013;184:1848-60.
[6] Fraser K, St-Georges L, Kiss LI. Prediction of defects in a friction stir welded joint using the Smoothed Particle Hydrodynamics Method. Procedings of the 7th Asia Pacific IIW International Congress on Recent Development in Welding and Joining Methods. 2013.

[7] Pan W, Li D, Tartakovsky AM, Ahzi S, Khraisheh M, Khaleel M. A new smoothed particle hydrodynamics non-Newtonian model for friction stir welding: Process modeling and simulation of microstructure evolution in a magnesium alloy. International Journal of Plasticity. 2013.
[8] Monaghan JJ. An introduction to SPH. Computer Physics Communications. 1988;48:89-96.
[9] Gingold RA, Monaghan JJ. Kernel estimates as a basis for general particle methods in hydrodynamics. Journal of Computational Physics. 1982;46:429-53.
[10] Monaghan JJ. SPH without a Tensile Instability. Journal of Computational Physics. 2000;159:290-311.
[11] J.P. Gray JJM, R.P. Swift. SPH elastic dynamics. Computer Methods in Applied Mechanics and Engineering. 2000.
[12] Pelfrey B, House D. Adaptive neighbor pairing for smoothed particle hydrodynamics. 2010. p. 192-201.