An efficient and parallel level set reinitialization method - Application to micromechanics and microstructural evolutions

Modesar Shakoor, Benjamin Scholtes, Pierre-Olivier Bouchard, Marc Bernacki

An efficient and parallel level set reinitialization method – Application to micromechanics and microstructural evolutions

Modesar Shakoor[a,*], Benjamin Scholtes[a], Pierre-Olivier Bouchard[a], Marc Bernacki[a]

[a] *Mines-ParisTech, PSL-Research University, CEMEF - Centre de mise en forme des matériaux, CNRS UMR 7635, CS 10207 rue Claude Daunesse 06904 Sophia Antipolis Cedex, France.*

Abstract

The paper introduces a new parallel and efficient algorithm for the reinitialization of level set functions on unstructured Finite Element (FE) meshes in two and three dimensions. The originality of this implementation lies in the use of a direct method enhanced by a *k-d tree* space partitioning technique. Different test cases illustrate the potential of the method for typical metallurgical and micromechanical problems with isotropic and anisotropic meshes. Comparison with other classical reinitialization methods, such as Hamilton-Jacobi formulations, proves that the proposed method guarantees optimal accuracy together with importantly reduced computational costs.

Keywords: level set reinitialization, *k-d tree*, finite element method

1. Introduction

Most problems in materials science at the microscale involve multiphase formulation and tracking of dynamic interfaces, e.g. recrystallization, grain growth, fracture mechanics, ductile damage, etc. Thanks to the explosion of large scale parallel computations, these phenomena can now be simulated at a Representative Volume Element (RVE) scale. Among existing approaches, the Level Set (LS) method introduced in 1988 [1] receives a growing attention. In the LS method each phase of the simulation is represented by a LS function evaluated at mesh nodes. This function can simply be defined as a binary function equal to 1 inside the phase and 0 elsewhere, as proposed in [2]. This approach does nevertheless not provide a precision greater than the mesh size. A well-known alternative consists in defining the LS function $\psi(x,t)$ on the domain Ω as a signed distance function to the interface Γ:

$$\forall t \begin{cases} \psi(x,t) = \pm d(x,\Gamma(t)), x \in \Omega, \\ \Gamma(t) = \{x \in \Omega, \psi(x,t) = 0\}, \end{cases} \tag{1}$$

where $d(.,.)$ is the Euclidean distance and the sign depends on whether the node is located inside or outside of the represented object. In an Eulerian context, the evolution of this function is classically computed by solving a transport equation involving the velocity field $\vec{v}$. This formulation has been employed to simulate a wide variety of mechanical and metallurgical phenomena [3, 4, 5, 6].

A major drawback of this formulation lies in the fact that after transport, in general, the function is no longer a distance function. This is particularly problematic when a specific remeshing technique depending on the distance property is used at the interface [3]. In addition, some phenomena, such as curvature-driven interface motions, require a distance function at least in a thin layer around the interface in order to compute properly the corresponding velocity field [7]. Finally, the conditioning of the transport problem also depends

*Corresponding author.
Email address: modesar.shakoor@mines-paristech.fr (Modesar Shakoor)

on the regularity of the LS function [8]

For these reasons, the distance function needs to be reinitialized. Indeed, restoring the metric property is equivalent to solving the following eikonal equation:

$$\forall t \begin{cases} \|\nabla\psi(x,t)\| = 1, x \in \Omega, \\ \psi(x,t) = 0, x \in \Gamma(t). \end{cases} \tag{2}$$

There exist different approaches to solve this equation including the well-known Fast Marching Method introduced by Sethian [9] which propagates a front from the interface and ensures directly a gradient equal to unity. Though this approach has been later extended to unstructured meshes [10], its implementation becomes extremely complicated when it comes to consider anisotropic (*i.e.* obtuse) triangulations [11]. The latter relies on the insertion of numerical supports for obtuse triangle and is mentioned as "cumbersome" in [11]. To our knowledge, this variant is not used in the recent literature. Another major drawback of the Fast Marching Method lies in the parallel implementation. More specifically, the algorithm has to be performed several time on each partition to synchronize the values between the processors, which requires significant implementation effort and poor parallel efficiency.

In [12], a Hamilton-Jacobi (H-J) formulation equivalent to (Eq. 2) was proposed in order to correct iteratively the level set values around the interface by solving a partial differential equation (PDE). This method thus requires the definition of a purely numerical parameter known as the fictive time step for reinitialization $\Delta\tau$. This quantity is generally of order of the mesh size h in the direction normal to the interface. For convenience we assume in the following that $\Delta\tau = h$. By noting ε the reinitialized thickness, $\varepsilon/\Delta\tau$ increments are then needed to reinitialize completely the layer $\psi \in [-\varepsilon, \varepsilon]$.

More recently, coupled convection-reinitialization (CR) methods emerged wherein the LS function is automatically reinitialized during the resolution of the transport equation [8]. Their main advantage lies in the fact that only one solver is needed for the simulation instead of two for the classical H-J technique. The signed distance function can also be replaced by any smooth function which satisfies the metric property, at least in a thin layer around the interface. In the following these two variants will be mentioned: the former using a classical distance function (CR-DF) and the latter working with a hyperbolic tangent distance function $\tilde{\psi} = E \cdot \tanh(\psi/E)$ (CR-HTDF). Since the hyperbolic tangent function has a gradient close to one only in the neighborhood of zero, the truncation thickness E has to be chosen big enough to verify the metric property at least in a thin layer around the interface.

Finally, a natural way to reinitialize LS functions consists in using a brute force algorithm to perform a complete reconstruction of the distance function. This technique works in two steps: discretize the interface (zero-isovalue of the LS function) into a collection of simple elements and, for every node, compute the distance to all elements of the collection and store the smallest one which becomes the updated value of the distance function. Though it guarantees optimal accuracy, this direct reinitialization (DR) technique is generally mentioned as extremely greedy in terms of computational requirements in the literature [12, 13]. Hence it is carefully avoided in most implementations, with the exception of [6]. In [14], a review of various improvements to this method proposed in literature to overcome this difficulty can be found. These works generally address only regular grids or hierarchical meshes [15].

In the following, the DR method is investigated and a new parallel and efficient implementation is proposed for unstructured and possibly anisotropic meshes. It is then compared to other approaches in terms of accuracy and numerical performances. Applications addressed here cover full field grain growth simulations and ductile damage modelling at the microscale.

2. Algorithm

As stated above, the DR method which is a basis for the present work starts from a simple idea, illustrated in Fig. 1. In the frame of P1 (linear by element) interpolation, the LS function is represented by its values at mesh nodes. On each element crossed by the interface, a discrete representation of a portion of the interface is constructed giving in the end a mesh of the whole interface. For sake of clarity, we name mesh the initial finite element (FE) mesh, and collection this 1D or 2D interfacial mesh. Hence, reinitialization of the signed distance function can be performed at any mesh node by searching the closest edge or triangle in the collection. As the computation of the distance to an edge or a triangle will be a critical operation in this part of the algorithm, we chose to use the optimal implementations detailed in [16]. The sign of the reinitialized function can then be taken as the one of the initial LS function. It can be seen that opposed to H-J approaches mentioned above, the DR method is way more accurate: given a P1 representation of a LS function, the DR method performs an analytical resolution. Nevertheless, this method, if used as is, has important costs.

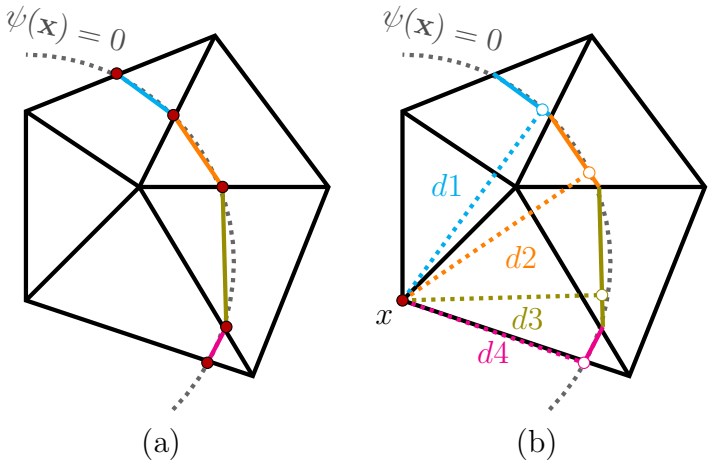

Figure 1: Direct reinitialization method on a P1 mesh: (a) collection construction ; (b) distance computation.

In the following, n represents the number of nodes in the mesh, and e is the number of elements in the collection. Using these notations, collection construction is of complexity $\mathcal{O}(n)$, while distance computation is of complexity $\mathcal{O}(n \cdot e)$. Hence, this last operation is too costly and makes the algorithm unsuitable for computations, especially in 3D. This is basically the reason why H-J methods are usually preferred.

Based on techniques widely used in computer graphics, data mining, and other domains, we propose a new direct reinitialization method, where the cost of distance computation is importantly reduced. This optimization is based on a space partitioning technique. Such technique consists in dividing the space, here the collection, in several parts, thanks to a suitable criterion. More precisely, hierarchical space partitioning techniques consist in dividing the space into p parts, and applying the same procedure to these parts, and so on, until small parts are obtained where searching or pruning can be performed at acceptable costs. Here p is a constant number inherent to the used methodology. For example, one may cite quad-trees in 2D, with $p = 4$, where the whole space is placed in a bounding box, which is divided in 4 boxes of identical dimensions, corresponding to the four quarters of the initial box (these boxes are then divided again, and so on). Using such systematic division, one or several boxes may be empty at some stage, and the division may not be optimal, especially in the present case with unstructured and possibly anisotropic meshes. The same statement can be made regarding oc-trees ($p = 8$), which are the 3D equivalent to quad-trees.

That is the reason why it is chosen to partition the space hierarchically using a k-dimensional (k-d) tree, where the division is always made into $p = 2$ parts, using a criterion which depends on the considered set of elements. This structure was first introduced in [17], with examples of potential applications. Among these applications, we will focus here on Nearest Neighbor Searches (NNS): for each node of the mesh, we want to find the nearest element in the collection. We explain in the following a methodology, named Direct Reinitialization with Trees (DRT), which reduces the complexity of distance computation to $\mathcal{O}(n \log e)$.

Opposed to systematic methods, the division criterion will here be based on an analysis of the space that is to be divided. In the present case, this space is the initial collection, or a sub-set of it, which can both be addressed as a set of elements, and division in 2 parts will be obtained using a division plane. Optimally, this plane should divide the given set into 2 sub-sets containing the same number of elements. To narrow such result without adding too important costs, it is here chosen to use planes normal to the Cartesian axis, and centered respectively to the set of elements. This centering is obtained by computing the barycenter of each element, and choosing a plane that goes through the barycenter of these points. The whole process of tree construction is given below:

(b.1) Set the division plane (line in 2D) as the plane going through the barycenter of all elements and having its direction alternatively defined by the x, y and z directions depending on the depth in the tree.

(b.2) Compute the signed distance from the vertices of all the elements in the collection to this plane.

(b.3) Build a left child to the current tree by going back to (b.1) with all the elements having at least a vertex with negative plane distance.

(b.4) Build a right child to the current tree by going back to (b.1) with all the elements having at least a vertex with positive plane distance.

As illustrated in Fig. 2, this process recursively builds a binary tree. As observed in Fig. 2.c, it ends when at steps (b.3) and (b.4) one of the two subtrees contains the other. In such situation, a leaf is created, and instead of containing a plane and two subtrees, this leaf contains only the remaining elements of the collection. Regarding the division plane, it is easy to find geometric configurations where the used simple definition does not divide the elements in two balanced subtrees. However, such situations are not met in practice, and tests have shown that this definition leads to a globally balanced tree. Regarding costs, the computation of the barycenter is performed in linear time, and because at each stage the number of considered elements should be divided by 2, the global tree construction operation is of optimal complexity $\mathcal{O}(e \log e)$.

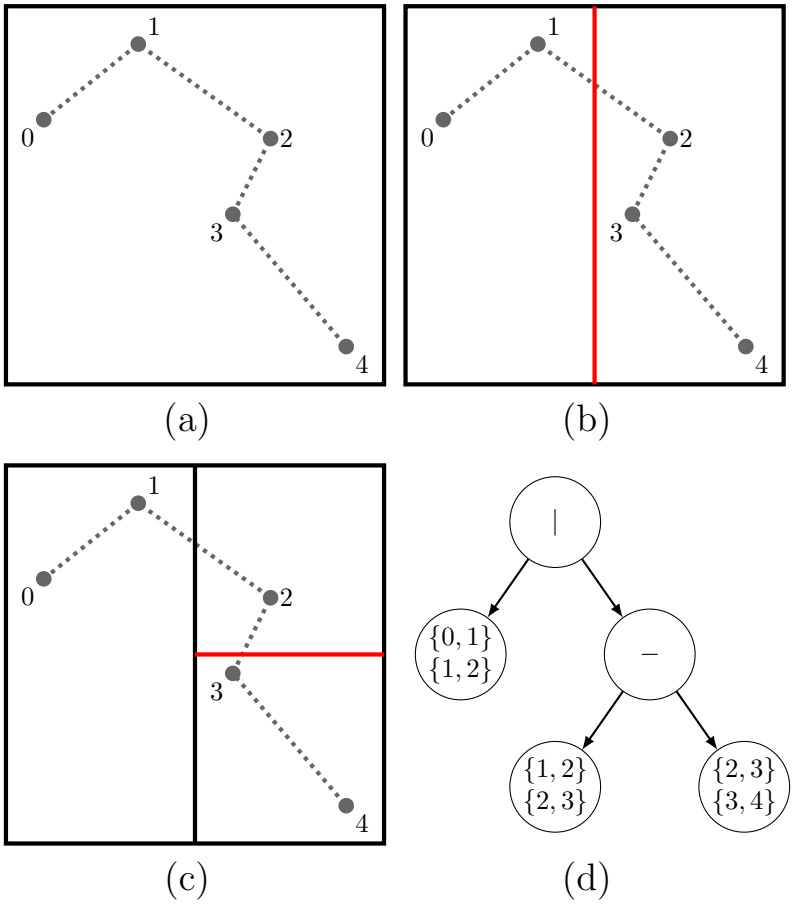

Figure 2: Example of recursive tree construction in 2D.

NNS queries can then be performed for each node of the mesh using the following algorithm at the root of the tree:

(b.5) Compute the signed distance from the node to the division plane.

(b.6) If it is negative, go back to (b.5) with the left subtree.

(b.7) If it is positive, go back to (b.5) with the right subtree.

This recursive process will reach a leaf, where distance computation will be performed by considering one by one all the elements stored in this leaf. Then, it may appear at steps (b.6) or (b.7) that the resulting distance is bigger than the distance to the division plane. In such case, it is required to go back to (b.5) with the other subtree. Though this operation is implemented to ensure consistency, it is not met often if the planes are well defined, as in our applications. Moreover, due to all the divisions, the set of elements stored in any leaf should be small enough to consider that distance computation is of optimal complexity $\mathcal{O}(n \log e)$.

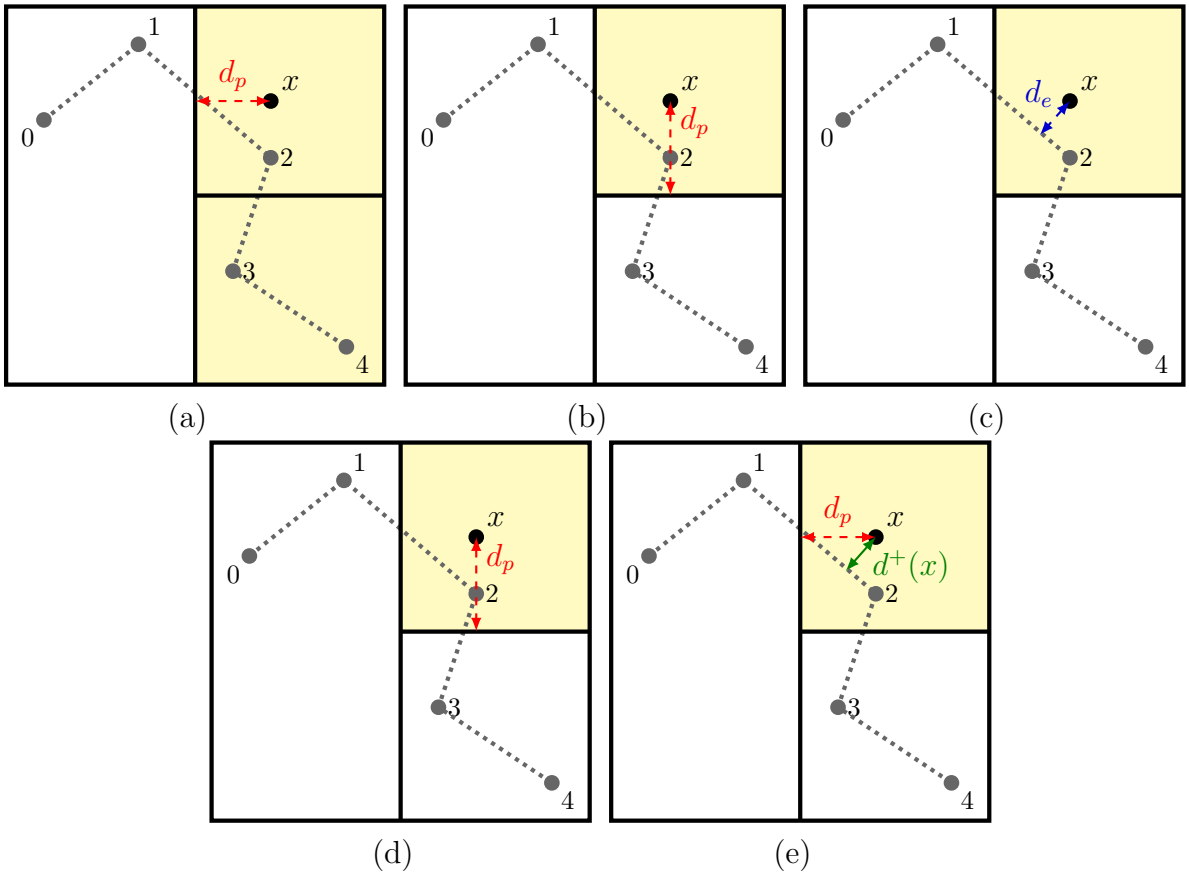

Figure 3: Example of recursive distance computation in 2D with the DRT method, best case scenario.

In the examples shown in Fig. 3 and Fig. 4, the distance from a point x to the interface needs to be computed. At each stage of the NNS, the notation d_p is the distance to a division plane, obtained at an intermediary node of the tree, the distance d_e is the distance to an element, obtained at a leaf of the tree, and d^+ is the final result. In the first case (Fig. 3), the point is optimally located since it is close enough to the interface. Hence, browsing two levels of the tree in (a) and (b) leads directly to the correct leaf in (c), and since when browsing back in (d) and (e), the obtained distance is smaller than the distance to any of the division planes, this distance is the final result. In the second case (Fig. 4), a worst case scenario occurs. The point is a located in a leaf (b) which gives a distance smaller than the distance to the first division plane (a). Hence, the other part of the tree has to be browsed (c). In this part of the tree, an optimal situation is met as the recursive browsing (d) and (e) gives the final result.

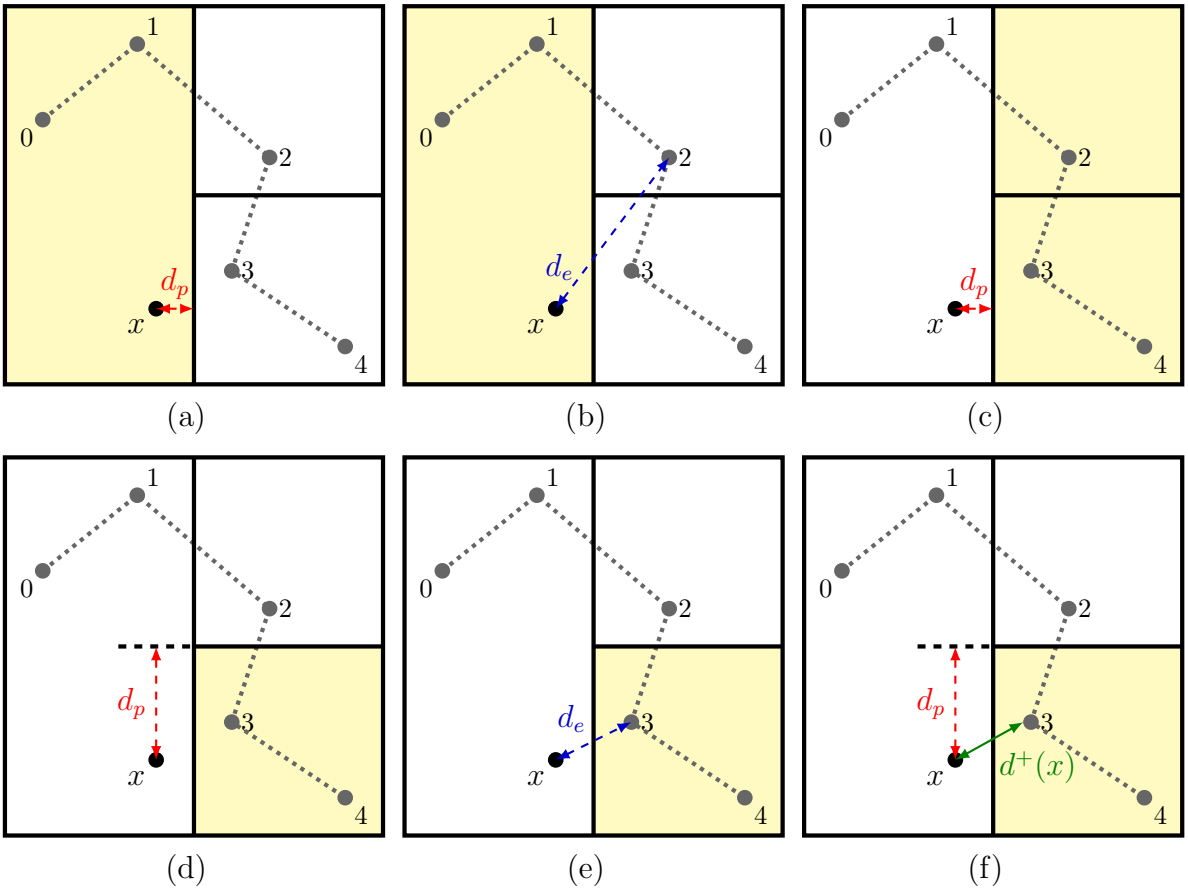

Figure 4: Example of recursive distance computation in 2D with the DRT method, worst case scenario.

In the introduction, it was mentioned that H-J approaches, because of prohibitive costs, are usually applied only in a small thickness around the interface. Implementing the same optimization to the DRT method can drastically reduce distance computation costs. The first step of the implementation consists obviously in executing NNS queries only for the nodes located in the reinitialization zone. Then, at step (6) and (7), if the resulting distance rises a need to look in the other subtree but the distance to the plane is bigger than the thickness used for reinitialization, this operation can be skipped. Because in practice the thickness is small enough to reduce the order of n to the same order as e, the final complexity of the new DRT method is expected to be $\mathcal{O}(e \log e)$. Moreover, using this optimization it appears that the best case scenario illustrated in Fig. 3 occurs more often than the worst case one, because the point x will always be located close to the interface.

Massively multi-domain computations often require an important computational power to obtain a good accuracy. The same remark can be drawn for 3D computations. For this kind of simulations, a classical choice is parallelization using the distributed memory paradigm. Opposed to shared memory, distributed memory enables each parallel unit, called process, to have a separated memory, and to be possibly located on a different machine. This last point is essential for large scale computations, where the whole mesh cannot be stored on a single machine. This paradigm nevertheless raises an issue in our case: since each process will only have the knowledge of a part of the mesh, it will only be able to build a part of the collection; hence it will not be able to compute distances to the collection. To solve this issue, one could simply communicate gather the full collection on each process, and then build the *k-d tree* on each process. Experiments have

showed that though this method has a good performance for a small number of processes (up to 20), it has a poor parallel speed-up.

To retrieve parallel efficiency, an advanced technique has been developed. In this method, each process builds its own collection and its own tree, ignoring other processes. Then, a bounding box system is used. Each process computes the minimal box aligned to Cartesian axis that can contain completely its collection. At the step of distance computation, each process will compute first the distance using its tree, and then will interrogate one by one the processes when they can improve this distance. This parallel implementation is summarized in Fig. 5. In this illustration, distance computation to point A is local since it is shorter than the distances to boxes 0, 2 and 3. Regarding point B, the distance to box 2 (in broken line) is shorter than the local distance computation performed inside process 3. Hence, the distance from B to 2's collection is computed and chosen as it is shorter than the previous one.

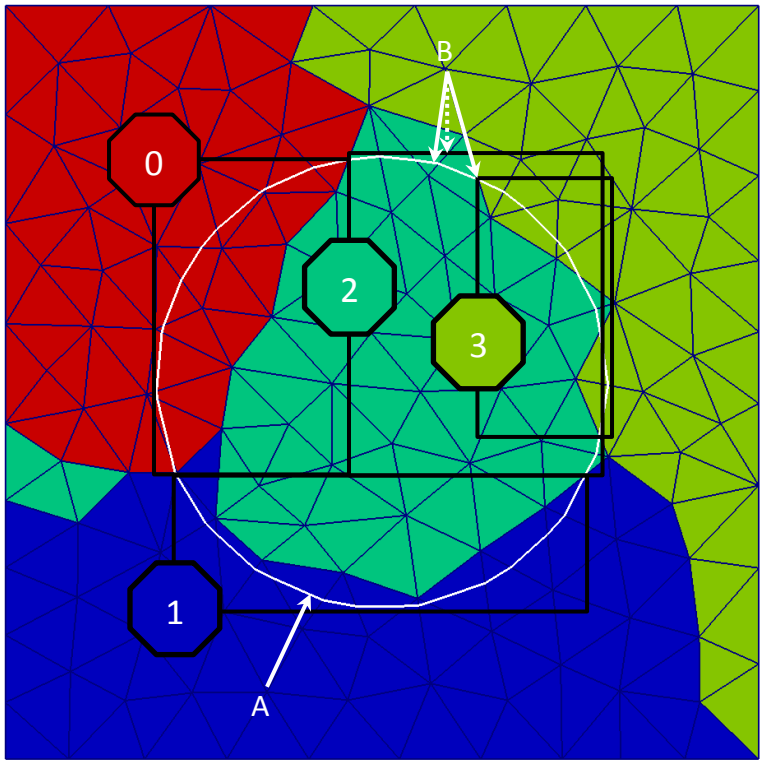

Figure 5: Bounding boxes on a 2D example with 4 processes (identified by the color code). The solid white line is the interface, and the arrows represent distance computation steps.

It can already be seen on this example that bounding boxes are not optimal, as box 2 nearly contains box 3. It is however expected that for a large number of processes and a good repartition of the interface, communications will be minimized and optimal parallel efficiency reached. This bounding box technique at the global level completes the *k-d tree* optimization used at the local level, and the DRT method can now be applied to perform LS reinitialization in parallel.

3. Results

In this part, numerical results obtained with the new DRT method are provided. Both DR and DRT method were implemented within the parallel C++ library Cimlib [8]. Regarding the H-J, CR and CR-

HTDF methods that are used for comparison, parallel implementations were already provided by the library. All the following computations were performed on the same 1.2GHz Intel Xeon Linux cluster.

3.1. Academic cases

The first test case proposed is a square (respectively a cube in 3D) centered in a $1 \times 1mm^2$ (respectively $1 \times 1 \times 1mm^3$ in 3D) domain and subjected to a velocity field equal to the LS function gradient $\vec{v} = \nabla\psi$. The parameter E for the CR-HTDF solver is set to $E = 20h$ and $\varepsilon = E$ is kept for all methods in order to perform a fair comparison. The theoretical values of the LS function $\psi_{theo}(x,t)$ and the theoretical internal area (respectively volume in 3D) f_{theo} are calculated through:

$$\begin{cases} \psi_{theo}(x,t) = \psi(x,0) - t, \\ f_{theo}(t) = (l_0 - 2t)^d, \end{cases} \tag{3}$$

with $t \in [0, l_0/2]$. The terms d and l_0 correspond respectively to the spatial dimension and the initial edge length of the square/cube, which is chosen equal to $0.5mm$ both in two and three dimensions. The simulation time step is calibrated separately so that the global error on the internal area given by (Eq. 4) remains lower than 1%.

$$L^2 = \frac{\|f_{theo}(t) - f(t)\|_2^{t\in[0,0.25]}}{\|f_{theo}(t)\|_2^{t\in[0,0.25]}}. \tag{4}$$

3.1.1. Shrinking square (2D)

In this two-dimensional test case, a fixed unstructured and homogeneous mesh is used. The number of elements is equal to 150000, which corresponds to an averaged mesh size $h \approx 4\mu m$. The main results of this set of simulations are summarized in Table 1.

Method	H-J	CR-DF	CR-HTDF	DRT
Δt (s)	0.01	0.0001	0.0001	0.001
$t_{transport}$ (s)	6.4	85.1	86.7	5.8
t_{reinit} (s)	133			0.4

Table 1: Results of shrinking square simulations run on 4 CPUs. For the CR solvers, the LS function is automatically reinitialized during the transport, which is why there is only one time for these two steps.

The new algorithm appears to be clearly the most efficient among all methods and is up to 300 times faster than the H-J approach. The CR solvers are proving more effective than the H-J method but require a very small time step to guarantee scheme stability. Let $\mathcal{N}$ be the set of nodes belonging to the layer $\pm 5h$ around the interface, the relative discretization error $\mathcal{R}(t)$ between the exact and computed LS functions is calculated through:

$$\mathcal{R}(t) = \frac{\|\psi_{theo}(x,t) - \psi(x,t)\|_2^{\mathcal{N}}}{\|\psi_{theo}(x,t)\|_2^{\mathcal{N}}} \qquad \text{with} \qquad \|u(x,t)\|_2^{\mathcal{N}} = \sqrt{\sum_{n\in\mathcal{N}} u(x_n,t)^2}. \tag{5}$$

Such small layer is chosen because both CR methods are only valid close to the interface [8].

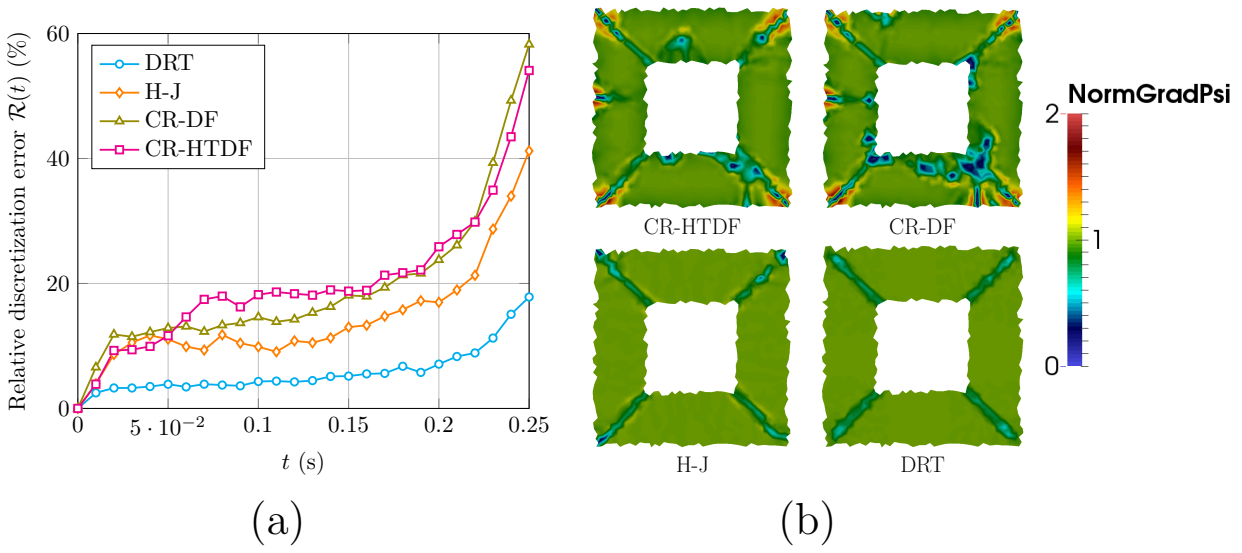

Figure 6: Relative discretization error $\mathcal{R}(t)$ between the exact and computed distance functions around the interface $\pm 5h$; (b) values of $\|\nabla\psi\|$ for each method.

Fig. 6.b illustrates the inability of the CR solvers to properly maintain the metric property, especially in the corner vicinity. On the other hand, the DRT and H-J methods exhibit a high level of accuracy. This first 2D test case thus demonstrates the proposed algorithm is both extremely fast (Table 1) and accurate (Fig. 6).

3.1.2. Shrinking cube (3D)

A similar case is now investigated in 3D. In order to limit the number of elements, a remeshing operation is performed at each time step with the parallel topological mesher/remesher MTC implemented in the used library. An a posteriori strategy is employed to construct the metric field needed for the mesh adaptation. The interested reader can find more details in [18]. Elements located near the interface are then stretched to form a refined and anisotropic layer with thickness 2ϵ around the zero iso-level. Outside this zone, the mesh is kept coarse (Fig. 7).

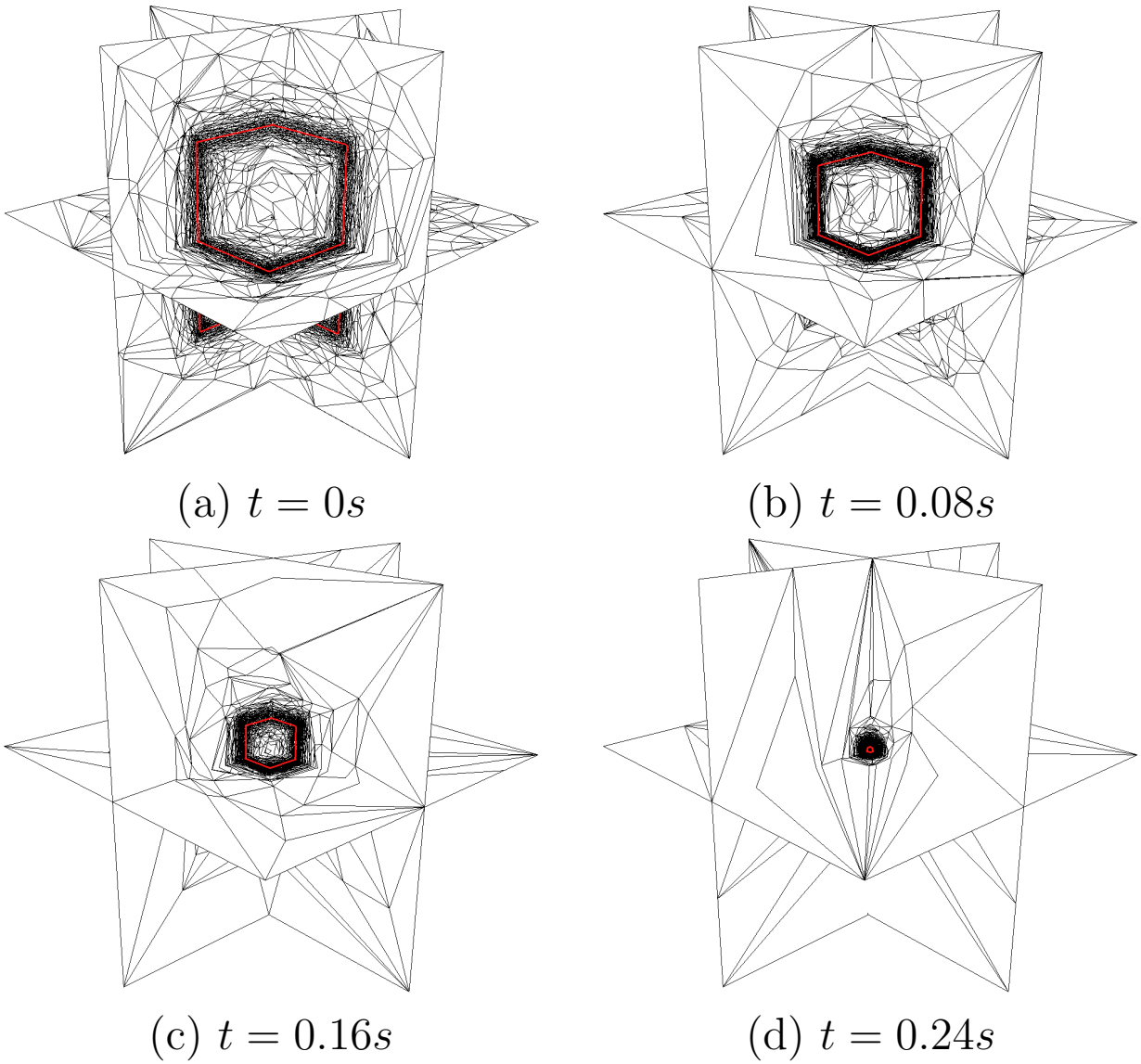

Figure 7: Anisotropic mesh used for the shrinking cube test case. The red thick line indicates the zero iso-level.

For this set of simulations, the total number of elements in the domain is equal to 500000. The local mesh size h in the refined layer thus constantly evolves during the simulation. The parameters ε and $\Delta\tau$ are therefore updated at each time step in order to maintain $\varepsilon = E = 20h$ and $\Delta\tau = h$. The time steps for all methods are chosen identical to the ones calibrated for the two-dimensional case (cf. Table 1).

In the addition to the use of an anisotropic mesh, this case is also critical for non-direct approaches because the gradient is poorly defined along bisecting planes and diagonals. Hence, the linear problems built by the H-J solver also have a poor conditioning (on average 430 iterations are performed by the used GMRES iterative solver in 3D compared to around 20 in 2D). During the simulation, one observes furthermore the appearance of a parasite phase outside the cube (Fig. 8). It proves the function becomes too irregular to be properly reinitialized leading to a modification of sign of the LS function. The interface of this parasite

phase is then automatically detected and captured by the remesher which adapts the mesh around it. As the total number of elements is fixed, the calculation accuracy then falls because fewer elements are used to represent the cube interface.

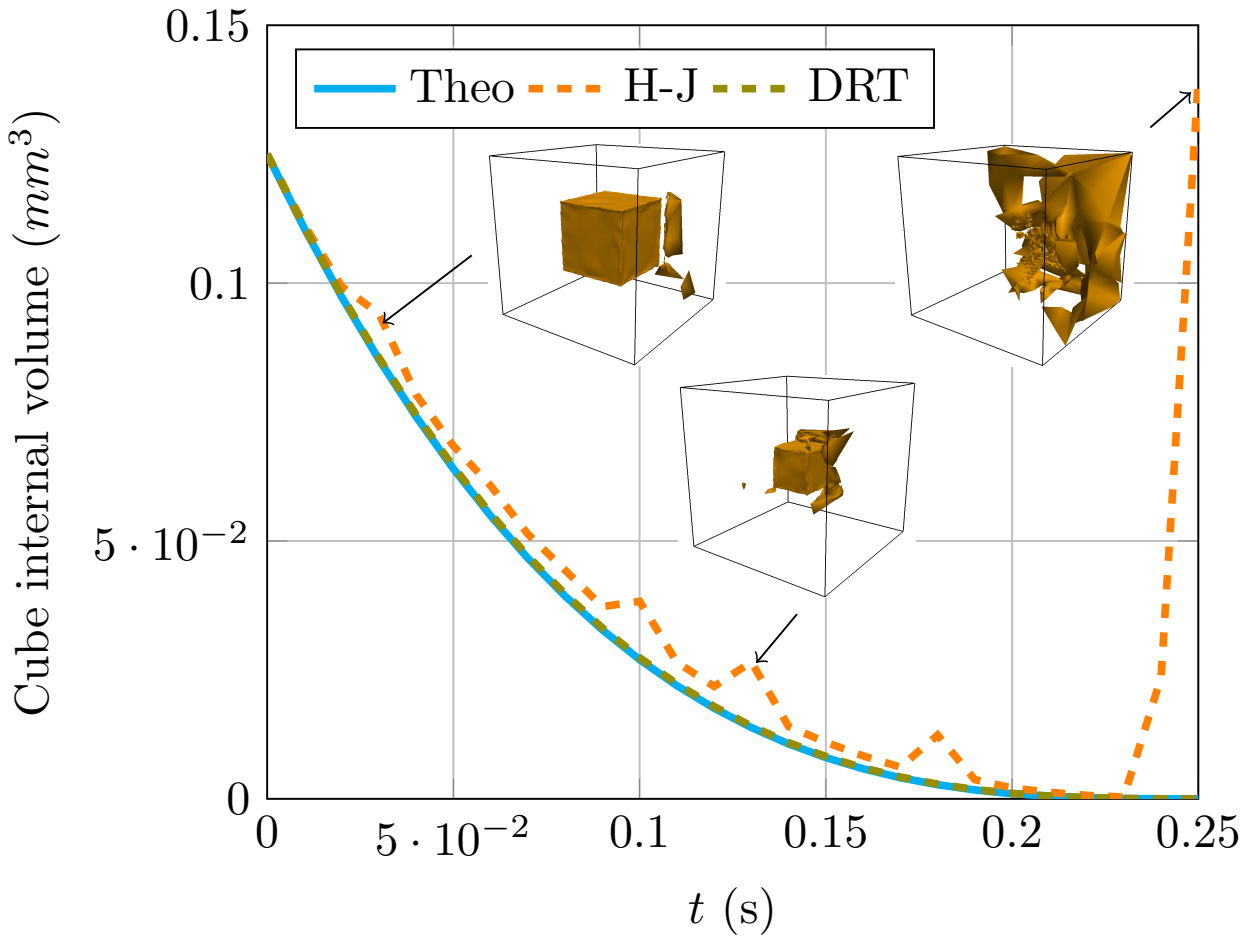

Figure 8: Evolution of the cube internal volume and appearance of a parasite phase with the H-J approach due to an unsatisfying reinitialization of the LS function.

The CR solvers exhibit unstable behaviors although the simulation time step remains an order of magnitude lower than the mesh size in the refined zone (no violation of the stability condition). These approaches seem therefore to be less robust than the H-J and direct ones. In addition they require a good knowledge of the parameters needed for the stabilization. Results obtained with the direct and H-J approaches are summarized in the Table 2.

Method	H-J	DRT
$\mathcal{R}(t)$ (%)	57	0.54
t_{reinit} (s)	263.6	52.8

Table 2: Results of shrinking cube simulations run on 20 CPUs.

In 3D, the DRT algorithm appears to be around five times faster than the H-J approach, while keeping a

very high accuracy. The gain in terms of computational performances is then smaller than in 2D. Nevertheless, one should note this configuration is particularly unfavorable to our algorithm. During the simulation, the cube rapidly shrinks and becomes very small compared to the size of the domain. Considering our partitioning strategy and the number of processors involved (20 CPUs for the cube shrinking case), the whole interface is poorly distributed across the CPUs. The parallel implementation described earlier thus becomes far less efficient. This is no longer true for massively multiphasic systems such as discussed in the next section wherein interfaces are naturally spread in a balanced way across the processes.

These simple academic test cases then confirm the superiority of the proposed algorithm and demonstrate its robustness. It appears to be also well-suited with particular triangulations (anisotropic meshes) and does not require any calibration, contrary to other classical methods which require at least one numerical parameter.

3.2. Ideal grain growth

The second test case is an extremely popular problem in material science: ideal grain growth. As mentioned earlier, this problem is particularly interesting because it involves a large number of LS functions $N_f >> 1$. An efficient reinitialization of the LS functions is then essential when it comes to reduce the computation time. During grain growth, the normal velocity of the grain boundaries (*i.e.* interfaces delimiting the grains) is proportional to their mean local curvature κ which can be expressed as follow:

$$\vec{v} = m\gamma\kappa\vec{n} = -m\gamma\nabla\cdot\left(\frac{\nabla\psi}{\|\nabla\psi\|}\right)\frac{\nabla\psi}{\|\nabla\psi\|}, \tag{6}$$

with $\vec{n}$ the unit outward normal vector and $m\gamma$ a material parameter taken here equal to $8.28\times10^{-7}mm^2/s$ which is representative of a 304L austenitic steel at 1050°C [7]. In a P1 framework such as used in our numerical formulation, computing the mean curvature would rely on a Hessian recovery technique which would induce unacceptable errors [5]. It is thus impossible to use the CR solvers for this application. An alternative approach consists in reformulating the problem as a pure diffusion one, by assuming the LS function satisfies $\|\nabla\psi\| = 1$ around the interface. The reinitialization is therefore fundamental for this application. Further details can be found in [5, 7]. In order to remove kinematics incompatibilities a particular treatment is performed on each LS function $\psi_i(x,t)$:

$$\tilde{\psi_i}(x,t) = \frac{1}{2}\left(\psi_i(x,t) - \max_{j\neq i}\left(\psi_j(x,t)\right)\right). \tag{7}$$

This treatment removes vacuums regions at multiple junctions (with a precision equal to the local mesh size) but also strongly alters the LS functions outside the grains, leading to catastrophic results if the function is not reinitialized (Table 3). Another strategy to deal with these kinematic incompatibilities can be found in [6].This set of simulations is performed on a $8\times8mm^2$ domain with a fixed homogeneous mesh composed of one million elements. The initial microstructure contains around 3600 grains represented by only 27 LS functions thanks to a graph coloring technique (see Fig. 9). The evolution of the mean grain size is tracked and the well-known Burke & Turnbull model is used as reference solution. The reinitialized thickness is fixed to $\varepsilon = 20\mu m$.

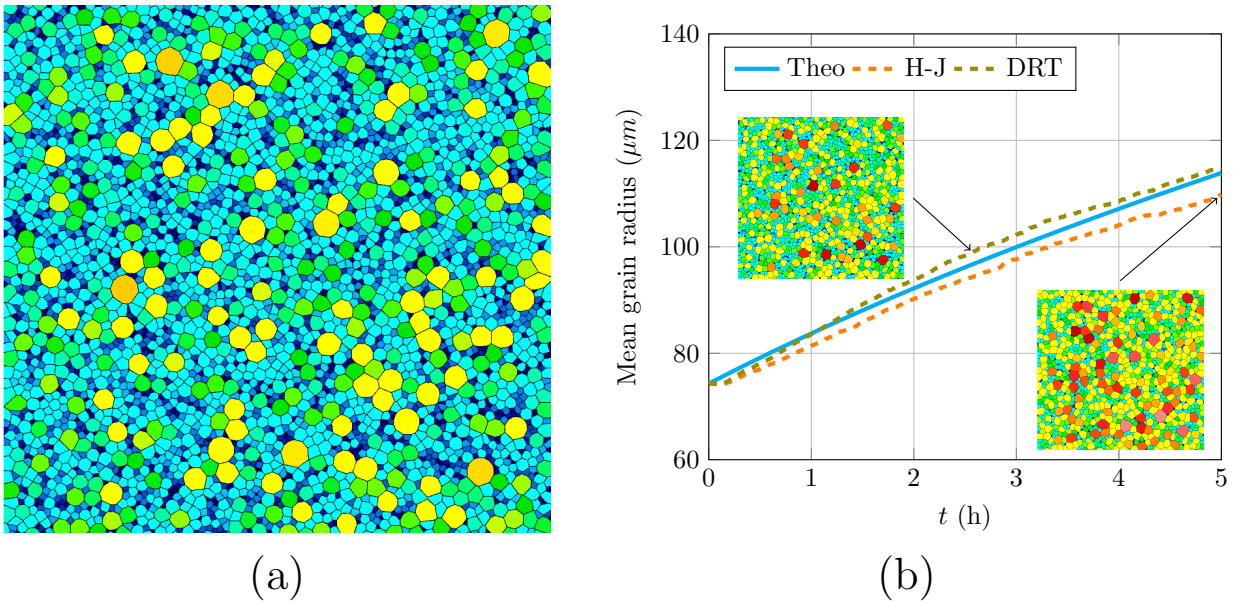

Figure 9: (a) 2D initial microstructure used for grain growth. The color scale corresponds to the grain size ; (b) evolution of the mean grain size and coarsening of the microstructure.

This benchmark is also used to compare the optimized (DRT) and non-optimized (DR) versions of our algorithm, which have respective complexity of $\mathcal{O}(n \log e)$ and $\mathcal{O}(n \cdot e)$. The parallel implementations are also challenged by running simulations on 4, 8 and 16 CPUs.

Method		Without reinit	H-J	DR	DRT
Comparison with B&T theory (%)		72		< 3	
	4 CPUs		7.74	4.2	0.09
t_{reinit} (h)	8 CPUs		5.02	3	0.07
	16 CPUs		2.48	1.52	0.03
Speedup (4/16 CPUs)			3.12	2.76	3

Table 3: Results of 2D grain growth simulations. The total duration of the heat treatment is equal to 5h.

It appears both the direct and H-J methods are in good agreement with the Burke & Turnbull model. The reinitialization times for the H-J and DR approaches are of the same order while the DRT one is around 80 times faster than the H-J method between 4 and 16 CPUs. The interest of using a *k-d tree* appears clearly here. These results also validate the parallel implementation of our algorithm which exhibits a speedup comparable to the H-J solver.

3.3. Void growth

For this last test case, modelling of ductile damage at the microscale is addressed. In a recent experimental study [19], sheets containing laser drilled holes were submitted to vertical tension. The influence of the angles and distances between voids on growth and coalescence was investigated. Experiments showed that most analytical models fail to predict correctly coalescence in non-academic cases. As these models are the basis of most Gurson-type laws which are used at the macroscale, a better understanding and modelling of ductile failure at the microscale is necessary. In most works, a void is embedded in a small domain called unit cell, where periodic boundary conditions are used to enable the study of multiple voids. In a recent

publication [20], a new Lagrangian framework based on the LS method was developed and applied to the modelling of void growth. In this framework, any configuration of voids was embedded in a larger cubic RVE, and anisotropic mesh adaptation was used to have a finer mesh close to the microstructure and a coarser mesh in the rest of the domain.

In the following, this numerical method is detailed and then enriched with the new DRT method. As stated above, an isotropic mesh refinement is used to progressively refine the mesh from the boundaries of the RVE to its center, where the microstructure is located. At the interfaces between matrix and voids, an anisotropic mesh is built using the methodology proposed in [3]. Close to the interface, a small mesh size is fixed in the normal direction, and the refinement in the other directions depends on local curvatures. To compute the normal vector to the interface and the curvatures, the gradient and the Hessian matrix of the LS function representing the void phase are used. The particularity of this framework is that it is Lagrangian hence the LS function is convected directly by mesh motion. Since the voids grow and may even change shape during simulation, no LS reinitialization may lead to improper mesh adaptation.

In the 2D example shown in Fig. 10, a void coalescence simulation is performed without (a,b,c) and with (d,e,f) LS reinitialization. In pictures (a) and (d), it can be seen that due to the vertical tension applied on the RVE, necking appears at the intervoid ligament. Pictures (b) and (e) show a zoom on this ligament. Without reinitialization, the mesh is tightened, while with a proper distance function, the mesh can be adapted identically on the whole interface. Coalescence is then triggered by inserting small voids inside the ligament. These voids grow rapidly and lead to pictures (c) and (f). Obviously, only LS reinitialization can enable a correct tracking of interfaces.

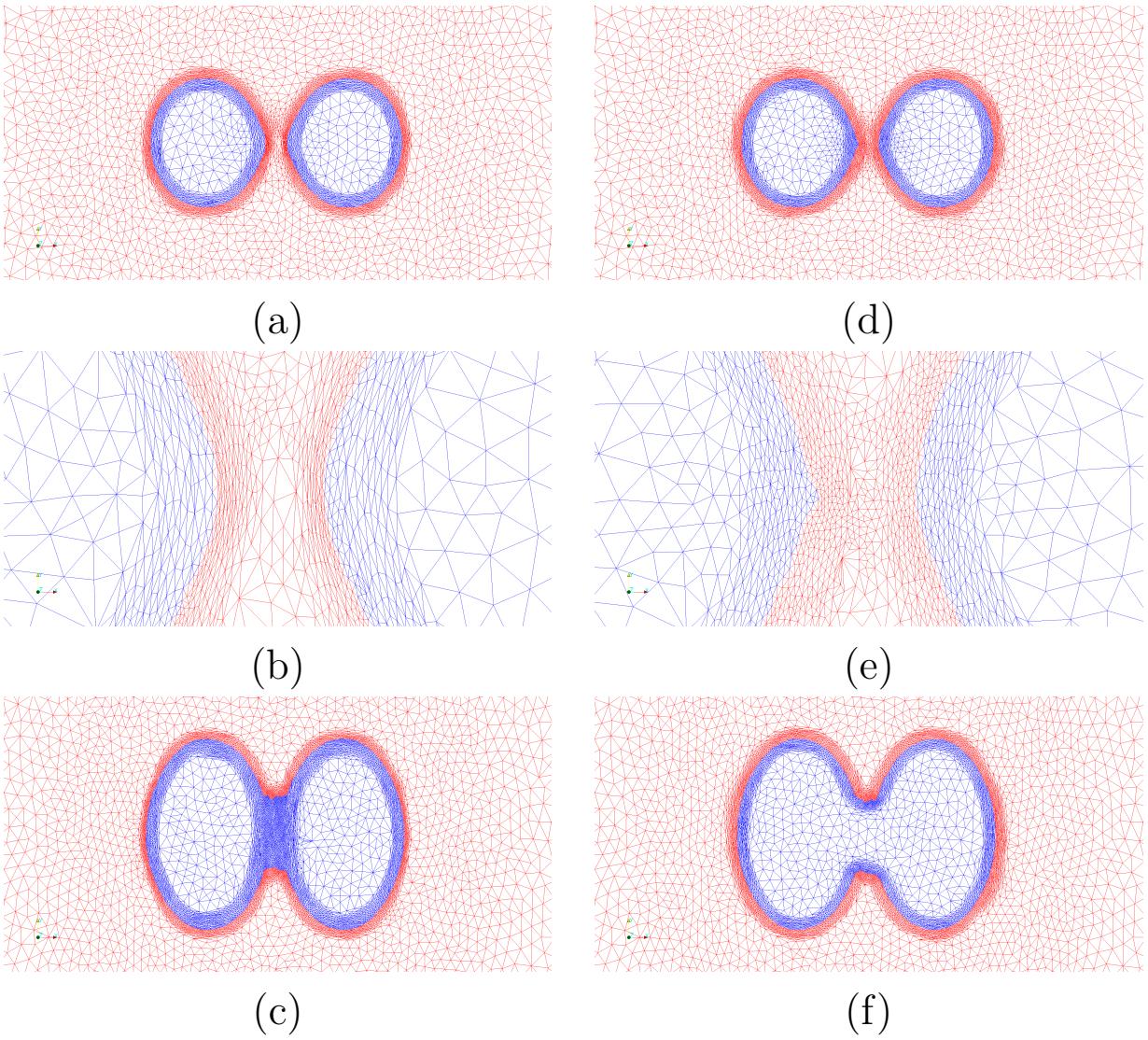

Figure 10: 2D example of void (in blue) coalescence without (a,b,c) and with (d,e,f) reinitialization.

Since the framework is Lagrangian, the CR method cannot be used. Hence, both the new DRT and the classical H-J approaches are applied to the 3D version of the configuration presented in Fig. 10. A $1 \times 1 \times 1 mm^3$ RVE is subjected to vertical tension at a constant velocity with a sticking boundary condition. On this 3D case, no coalescence is triggered, and the resulting void shapes at 20% of elongation are shown in Fig. 11.

To reach 20% of elongation of the RVE, which corresponds to 200 time steps and LS reinitializations, the H-J method takes 127 minutes on 16 processors, while the new DRT method takes 23 minutes in the same configuration. Though the efficiency of the DRT method compared to the H-J method was already proven in previous simulations, this test case confirms the superiority of the DRT approach when the reinitialization thickness becomes large.

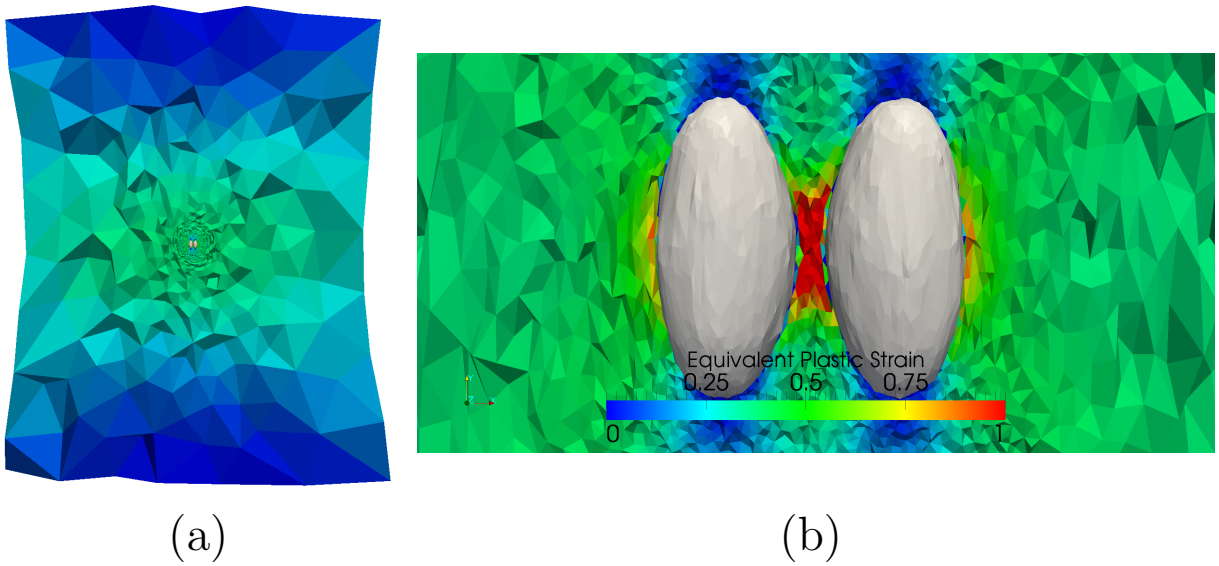

Figure 11: Equivalent plastic strain and void shape after 20% of vertical elongation.

Conclusion

While indirect methods to reinitialize Level Set functions were historically developed to avoid the computational costs required by direct methods, it was proven in this paper that this idea is questionable. For example, it was proven that a simple Direct Reinitialization through a brute force algorithm can compete with a Hamilton-Jacobi approach in terms of performance for massively multi-domains problems such as grain growth in polycrystalline aggregates. Then, based on a *k-d tree* sorting of the interface discretization, a new Direct Reinitialization with Trees method was proposed and applied to three different test cases arising from different contexts. This parallel and optimized DRT approach proved to be as accurate as a classical DR method, while being up to 20 times faster. Computation time reduction was also observed compared to a Hamilton-Jacobi formulation, with speed-ups between 5 in 3D and 300 in 2D. Additionally, direct methods, including the new DRT technique, revealed being the most accurate in theory as in practice. All tests were performed with unstructured 2D or 3D meshes, and anisotropic mesh adaptation for the third one, to illustrate that the proposed method remains as efficient in all these configurations. However, it was mentioned that the efficiency of the bounding boxes method used for the parallel implementation highly depends on mesh partitioning. Taking the interface into account inside the mesh partitioning algorithm should be considered in order to further improve the parallel performance of the algorithm for general applications.

References

[1] S. Osher, J. A. Sethian, Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations, Journal of Computational Physics 79 (1) (1988) 12–49. doi:10.1016/0021-9991(88)90002-2.

[2] B. Merriman, J. K. Bence, S. Osher, Motion of Multiple Junctions: A Level Set Approach, Journal of Computational Physics 112 (2) (1994) 334–363. doi:10.1006/jcph.1994.1105.

[3] D.-L. Quan, T. Toulorge, E. Marchandise, J.-F. Remacle, G. Bricteux, Anisotropic mesh adaptation with optimal convergence for finite elements using embedded geometries, Computer Methods in Applied Mechanics and Engineering 268 (2014) 65–81. doi:10.1016/j.cma.2013.09.007.

[4] A. Yazid, N. Abdelkader, H. Abdelmadjid, A state-of-the-art review of the X-FEM for computational fracture mechanics, Applied Mathematical Modelling 33 (12) (2009) 4269–4282. doi:10.1016/j.apm.2009.02.010.

[5] M. Bernacki, R. Logé, T. Coupez, Level set framework for the finite-element modelling of recrystallization and grain growth in polycrystalline materials, Scripta Materialia 64 (6) (2011) 525–528. doi:10.1016/j.scriptamat.2010.11.032.

[6] H. Hallberg, A modified level set approach to 2D modeling of dynamic recrystallization, Modelling and Simulation in Materials Science and Engineering 21 (8) (2013) 085012. doi:10.1088/0965-0393/21/8/085012.

[7] A.-L. Cruz-Fabiano, R. Logé, M. Bernacki, Assessment of simplified 2D grain growth models from numerical experiments based on a level set framework, Computational Materials Science 92 (2014) 305–312. doi:10.1016/j.commatsci.2014.05.060.

[8] M. Bernacki, Y. Chastel, T. Coupez, R. Logé, Level set framework for the numerical modelling of primary recrystallization in polycrystalline materials, Scripta Materialia 58 (12) (2008) 1129–1132. doi:10.1016/j.scriptamat.2008.02.016.

[9] J. A. Sethian, A fast marching level set method for monotonically advancing fronts., Proceedings of the National Academy of Sciences 93 (4) (1996) 1591–1595. doi:10.1073/pnas.93.4.1591.

[10] R. Kimmel, J. A. Sethian, Computing geodesic paths on manifolds, Proceedings of the National Academy of Sciences 95 (15) (1998) 8431–8435. doi:10.1073/pnas.95.15.8431.

[11] J. A. Sethian, A. Vladimirsky, Fast methods for the Eikonal and related Hamilton- Jacobi equations on unstructured meshes., Proceedings of the National Academy of Sciences of the United States of America 97 (11) (2000) 5699–703. doi:10.1073/pnas.090060097.

[12] M. Sussman, P. Smereka, S. Osher, A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow, Journal of Computational Physics 114 (1) (1994) 146–159. doi:10.1006/jcph.1994.1155.

[13] R. N. Elias, M. A. D. Martins, A. L. G. A. Coutinho, Simple finite element-based computation of distance functions in unstructured grids, International Journal for Numerical Methods in Engineering 72 (9) (2007) 1095–1110. doi:10.1002/nme.2079.

[14] M. W. Jones, J. A. Baerentzen, M. Sramek, 3D distance fields: a survey of techniques and applications., IEEE transactions on visualization and computer graphics 12 (4) (2006) 581–99. doi:10.1109/TVCG.2006.56.

[15] O. Fortmeier, H. Martin Bücker, Parallel re-initialization of level set functions on distributed unstructured tetrahedral grids, Journal of Computational Physics 230 (12) (2011) 4437–4453. doi:10.1016/j.jcp.2011.02.005.

[16] D. H. Eberly, P. J. Schneider, Geometric Tools for Computer Graphics, Kaufmann, Morgan, 2003.

[17] J. L. Bentley, Multidimensional binary search trees used for associative searching, Communications of the ACM 18 (9) (1975) 509–517. doi:10.1145/361002.361007.

[18] E. Hachem, S. Feghali, R. Codina, T. Coupez, Anisotropic adaptive meshing and monolithic Variational Multiscale method for fluid–structure interaction, Computers & Structures 122 (2013) 88–100. doi:10.1016/j.compstruc.2012.12.004.

[19] A. Weck, D. Wilkinson, Experimental investigation of void coalescence in metallic sheets containing laser drilled holes, Acta Materialia 56 (8) (2008) 1774–1784. doi:10.1016/j.actamat.2007.12.035.

[20] E. Roux, M. Shakoor, M. Bernacki, P.-O. Bouchard, A new finite element approach for modelling ductile damage void nucleation and growth—analysis of loading path effect on damage mechanisms, Modelling and Simulation in Materials Science and Engineering 22 (7) (2014) 075001. doi:10.1088/0965-0393/22/7/075001.