HAL open science

Simplified Marching Cubes: an efficient discretization scheme for simulations of deposition/ablation in complex media

Gerard L. Vignoles, Marc Donias, Christianne Mulat, Christian Germain, Jean-François Delesse

▶ **To cite this version:**

Gerard L. Vignoles, Marc Donias, Christianne Mulat, Christian Germain, Jean-François Delesse. Simplified Marching Cubes: an efficient discretization scheme for simulations of deposition/ablation in complex media. Computational Materials Science, 2011, 50 (3), pp.811-1218. 10.1016/j.commatsci.2010.10.027 . hal-00584763

HAL Id: hal-00584763

https://hal.science/hal-00584763v1

Submitted on 10 Sep 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - NoDerivatives 4.0 International License

Simplified Marching Cubes: an efficient discretization scheme for simulations of deposition/ablation in complex media

Gerard L. Vignoles[a,*], Marc Donias[b], Christianne Mulat[b,a], Christian Germain[b], Jean-François Delesse[c]

[a] *University Bordeaux - Lab. for ThermoStructural Composites (LCTS)*
3, Allée La Boëtie - F33600 Pessac, France
[b] *University Bordeaux - Lab. for Integration from Materials to Systems (IMS)*
351 Cours de la Libération - F33410 Talence Cedex, France
[c] *University Bordeaux - Lab. of Computer Resarch (LaBRI)*
351 Cours de la Libération - F33410 Talence Cedex, France

Abstract

Surface growth or recession models depend on accurate and efficient descriptions of the moving surface. We propose a simplified alternative to the popular marching cubes algorithm for isosurfacing, in which the surface consists of triangles which are composed from vertices of the regular 3D grid on which the data to be processed (*e.g.* the volume of fluid) is defined. Consequently, the new algorithm does not require any interpolation. In contrast with the original method, a switch of the relative status (above or below the threshold value) of vertices does not lead to similar triangle models. The obtained meshes are guaranteed to be manifold *i. e.* to be topologically consistent and with no holes. The implementation of the new method is simple. Comparison of both schemes in a random-walk gas diffusion simulation algorithm shows a substantial time improvement with SMC. An example of Chemical Vapor Infiltration modelling is described.

Keywords: Front tracking, Marching Cube, Volume-Of-Fluid methods (VOF),

*To whom correspondence should be addressed.
Email addresses: `vinhola@lcts.u-bordeaux1.fr` (Gerard L. Vignoles), `marc.donias@ims-bordeaux.fr` (Marc Donias), `christiannemulat@hotmail.com` (Christianne Mulat), `christian.germain@ims-bordeaux.fr` (Christian Germain), `delesse@freesurf.fr` (Jean-François Delesse)

1. Introduction

Triangle models of isosurfaces from volumetric scalar fields are used for scientific visualization (rendering of 3D medical data, geological volumes, synthetic data,...) and for algorithms involving explicit surfaces within volumes like simulations of diffusion processes on or through interfaces. In the case of data defined on a cuberille grid (*i. e.* a regular 3D cubic mesh), Marching Cubes (MC) is the reference algorithm to perform triangle models of isosurfaces defined by a threshold value. Since the early work of Lorensen and Cline[1], many methods [2] have been proposed in order to enhance triangle representations or to obtain more efficient implementations. On the one hand, successive improved versions have completed the various configurations of the original look-up table in order to obtain more accurate models with correct topology[3, 4, 5, 6]. Improvements of the robustness with respect to perturbations of the data and of the threshold value have also been investigated[7]. On the other hand, particularly in the case of large data sets, many works have been carried out to optimize the search process of the cubes intersected by the isosurface[8, 9], in order to reduce the number of the resulting triangles[10] and to decrease dramatically the computation time with hardware-accelerated strategies[11].

As far as visualization and surface extraction are concerned, the MC algorithms are excellent solutions, which have proven their efficiency. However, there is a strong demand from the applied science community to develop software capable of physico-chemical modelling inside complex, heterogeneous media subject to morphological evolution. Among many examples we can cite the clogging of porous media like filters, catalysts or rocks undergoing diagenesis, or the opening of pore space such as in rock dissolution. Fabrication or degradation of materials are other application fields, as will be seen later. Since the considered media exhibit rather complex geometrical features, there is a need to handle large datasets for their internal representation. Simultaneously, trans-

port phenomena involved in the evolution of the materials have to be accounted for. This implies the discretization of conservation equations in the interior of domains which are delimited by the surfaces that an MC algorithm has to produce ; boundary conditions for these equations have also to be stated on those surfaces. For instance, in Finite Volume (FV) scheme, we have to compute in each cell the current volume and the surface (and possibly the normals) of the interfaces with the adjacent cells. These quantities have to be computed as quickly as possible. Moreover, when the surface (*e.g.* some fluid/solid interface) is evolving, remeshing has to occur in some way. Moving boundaries have motivated numerous approaches like, among others, level-set algorithms [12] and Volume-Of-Fluid (VOF) [13] methods. In the case of a surface extracted by an MC-like algorithm which is subject to morphological evolution, there are some conditions that the surface mesh has to verify all the time. First, as far as transfer phenomena are involved, holes are forbidden. Second, the node density has to remain more or less constant on the surface. The recent MC algorithms are consistent with the "no-hole" prerequisite. If it is chosen to modify the data values of the 3D grid from which the surface is extracted, we have a convenient dynamic scheme for moving boundaries, since the MC methods are essentially local. So there is a motivation to keep these interesting properties, while simplifying the MC structure.

Combining the three elements of (*i*) large and complex 3D images, (*ii*) transfer phenomena, and (*iii*) surface evolution, we face the question of optimizing computational time and memory requirements. One of the compromises may be to use a surface discretization scheme which lies halfway between the most accurate ones (*i. e.* MC algorithm and variations) and the simplest one (cuberille), which is known to be inaccurate for the estimation of various geometrical parameters (normals, for instance). This is why, in this article, we propose a triangulation method, called simplified marching cubes (SMC), which can be viewed as a simplified alternative to the MC algorithm. The complete implementation lies on a simpler code than MC, because either no configuration table or no graph permutation lookup is necessary. The accuracy of the obtained meshes

is lower than the standard MC and higher than the cuberille approximation. The principle of the proposed method is similar to the discretized marching cubes (DiscMC) [10]. However it does not require any interpolation, the steps of the algorithm are very different. Triangles are indeed composed of vertices of the cuberille grid on which the processed data is defined. Moreover, meshes are guaranteed to be manifold, *i. e.* to be topologically consistent and without holes. Originally designed for simulations involving triangulated interfaces, our method is also appropriate for fast visualization in order to find the adequate threshold value.

Section II presents the principle and look-up table of the SMC algorithm. The proposed method is then used in section III for visualization of synthetic data and in section IV for simulation of diffusion-reaction processes in evolving porous media, by coupling with a random-walk algorithm.

2. Simplified Marching Cubes Algorithm

2.1. Principle

Considering a data sampled on a cuberille grid, the MC algorithm consists of processing each group of 8 neighboring vertices forming a cube and determining whether the cube edges intersect the isosurface defined by a threshold value. An intersection appears when an edge links a vertex with value below the threshold value to a vertex with value above the threshold value. Intersection points, the coordinates of which are accurately computed using linear interpolations, are collected together to constitute triangles, according to a look-up table containing all different configurations. The original paper [1] presented a first set of 15 configurations, of which one has been later found out to be redundant [7]. Later on, some inconsistencies in the configuration list have been identified, which eventually lead to surfaces with holes. Indeed, there is one face configuration for which the disposition of the triangle edges is ambiguous [3]. This is one of the main reasons that has led various authors to propose improved MC modifications. One of the simplest and coherent solutions is "MC-patch"

[14], which we will retain in the following, though we choose here to consider that configurations 11 and 14 are not distinct since they are symmetrical with respect to a mirror plane [15]. Figure 1 is a description of this configuration list. The 22 configurations correspond to all distinct possible graphs on a cube with black or white vertices, based only on the consideration of the number of first, second and third neighbors with the same color.

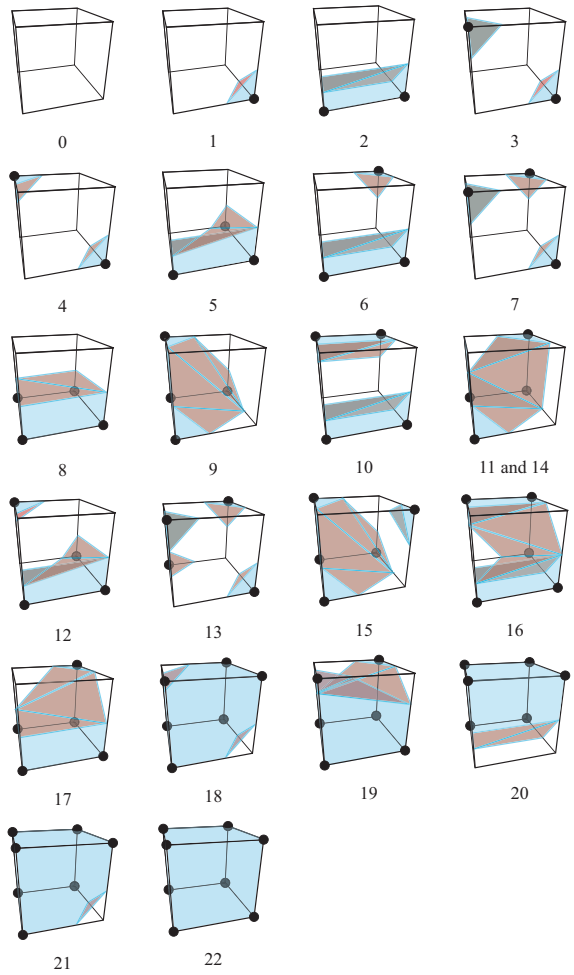


Figure 1: The "MC22" configuration list, with 6-connectivity of nodes respected in all configurations. Note that the 14 first configurations are present in the original report [1]. The numbering is similar to ref. [14] except that our configurations 14-21 match configurations 15-22 of this paper.

The SMC mesh is equivalent to an MC mesh where each intersection point (triangle vertex) has been moved along its edge to the grid vertex above the threshold. Degenerated triangles with collapsed points and null surfaces are naturally removed. The resulting mesh consists in triangles linking only vertices of the cuberille grid.

Let us now give another, more formal formulation of SMC. From a volumetric scalar field $F(x, y, z)$, MC produces a triangle model T of an isosurface $S_\lambda = \{(x, y, z) : F(x, y, z) = \lambda\}$ where λ is a threshold value. Theoretically, the application of the SMC could consist of the following two steps:

- The original data set F is binarized using the threshold value λ and gives a new data set F_B defined by

$$F_B = \begin{cases} 1 & \text{if } F(x, y, z) \geq \lambda \\ 0 & \text{else} \end{cases} .$$

- The standard MC algorithm is applied to F_B using the threshold value $\lambda = 1$.

Contrary to MC, it has to be noted that the difference between the value of vertices and the threshold value is not used by the proposed algorithm. Only the relative status (above or below) is considered.

Actually, implementing SMC by means of MC on thresholded nodes is way too suboptimal. It is just a conceptual model. Details of the actual method can be found in the Appendix. It is not based on the construction of a lookup table, but it is easy to build a lookup table out of it.

Figure 2 describes the SMC configuration list, as obtained from the "MC-patch" lookup table.

2.2. Discussion

The major advantage of the SMC is that all computations can be made with integer arithmetic. The computation of surface area and volume in SMC is quite

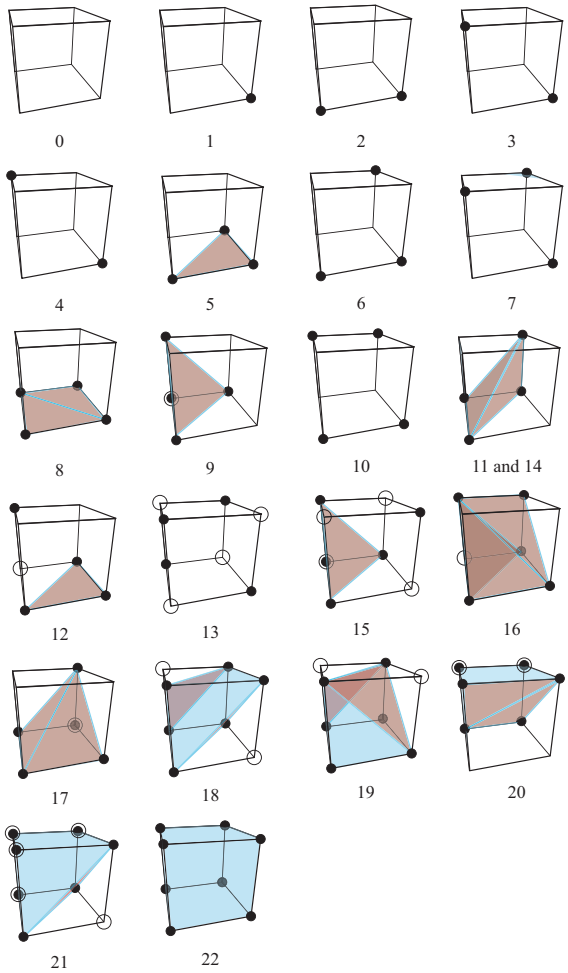


Figure 2: The SMC configuration list, as inferred from "MC-patch". See Fig. A.6 for an explanation of the symbols.

as simple as in the Cuberille case: indeed, for each configuration listed at Fig. 2, the volume and interface area are already known.

3. Application to visualization and computation of surfaces and volumes

3.1. Examples

SMC has been applied to standard datasets [16]. Figure 3 shows the resulting mesh in comparison with the cuberille method and the original MC, which confirms that SMC has intermediate performances between the other two.

Table 1 gives quantitative data on the number of vertices and triangles. The total memory is also reported in bytes, according to the following assumptions:

- For SMC and Cuberille, the vertex memory requirement is 3 short (2-byte) integers per vertex (*i.e* integer coordinates less than 65535).
- For the MC scheme, the vertex memory requirement is 3 single-precision (4-byte) floats per vertex.
- For all schemes, the triangle memory requirement is 3 long (4-byte) integers per triangle.

Table 1 shows that the SMC memory requirements are half of the other two ; the number of triangles and of vertices is approximately cut by half. Computed surface area and volumes are also reported in this Table.

3.2. Discussion

The accuracy of the obtained SMC meshes is lower than the standard MC and higher than the cuberille approximation. As a consequence, the SMC surface area computation gives values which may lie between the cuberille and MC estimations. The difference between the values of the computed surfaces of a discretized object (*e.g.* a sphere) and its actual theoretical value arises from two sources: one is a possible bias on the interface position, and the other is the "intrinsic roughness" arising from the algorithm.

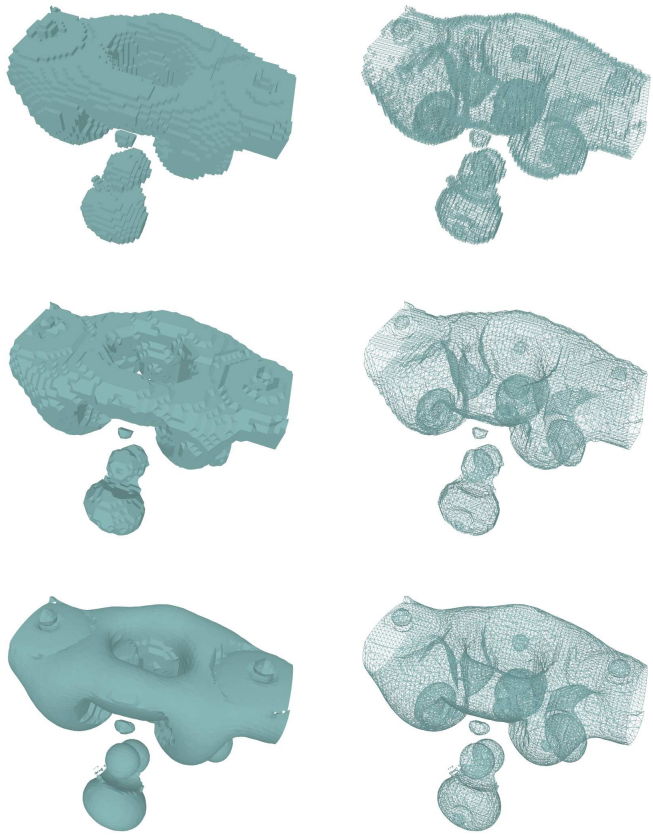


Figure 3: Resulting meshes of the Neghip image [16] with cuberille approximation (first row), SMC (second row) and MC (third row).

	SMC	Cuberille	MC	Rel. diff. w.r.t. Cuberille	Rel. diff. w.r.t. MC
Image "large buckyball", threshold = 80					
points	119 100	119 136	65 880	−44.7%	−44.7%
triangles	238 272	238 344	131 832	−44.7%	−44.7%
memory	3 573 864	4 289 760	1 977 264	−44.7%	−53.9%
surface	119 136	79 823	86 556	−27.3%	+8.4%
volume	711 648	708 248	680 284	−4.4%	−3.9%
time (ms)	43	67	53	+21.7%	−21.2%
Image "neghip", threshold = 40					
points	17 881	17 974	9 641	−46.1%	−46.4%
triangles	35 948	35 734	12 290	−65.8%	−65.6%
memory	538 662	644 496	205 326	−61.9%	−68.1%
surface	17 974	12 250	12 306	−31.5%	+0.5%
volume	33 484	33 526	28 743	−14.2%	−14.3%
time (ms)	5.5	9.0	6.6	+19.4%	−27.0%

Table 1: Comparison of the cuberille, SMC and MC algorithms on classical datasets.

By construction, all discretization schemes produce a bias on the estimation of the volume of an object, which tends towards zero when the sampling rate increases. This leads to the existence of a bias in the surface area estimation. Locally, if the sampling rate is substantially larger than the size of any surface roughness feature, the two estimates are linked by:

$$\Delta S = \Delta V \left(\frac{1}{r_1} + \frac{1}{r_2} \right)$$

where r_1 and r_2 are the algebraic Gaussian curvatures.

Contrary to the other two methods, SMC systematically produces negative volume biases. This ensures that a convex object has a negative surface bias, while a concave object has a positive surface bias. However, the SMC volume bias is always larger than in the other two methods.

Aside from this first effect, approximation of any smooth surface by a collection of triangles gives a systematically positive bias in the surface estimation, whatever the method. Unfortunately, this bias does not vanish when the sampling rate increases.

Let us illustrate these concepts with the case of a discretized sphere. Figure

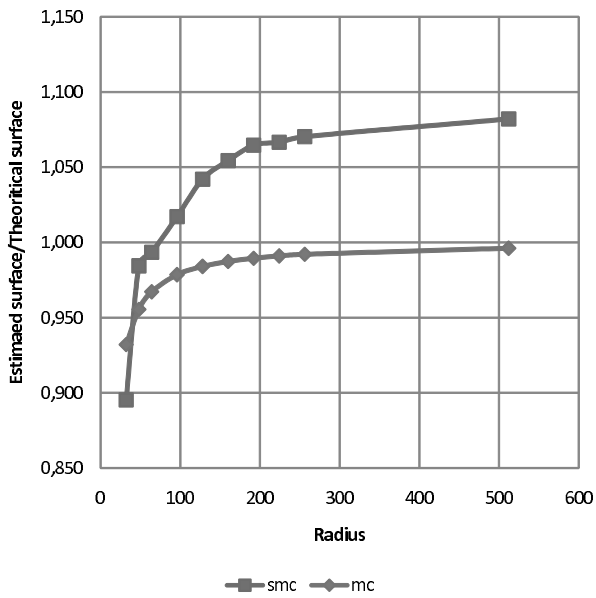


Figure 4: Comparison of surface error with MC and SMC for discretized spheres of growing radius.

4 shows that the SMC scheme goes to an error of 7% while the MC error limit is much lower (around 0.4%). The cuberille method has a much worse bias: the surface area of a sphere computed by this method is exactly the area of the embedding cube, which gives a ratio of $6/\pi$, *i. e.* a error of 90%, whatever the discretization size. Inverting black and white domains gives a different curve for SMC: instead of approaching the non-vanishing bias limit by inferior values, the convergence is by superior values, as can be expected since the sign of the volume estimation bias is reversed.

In the case of the "neghip" image, the surface area estimates of MC and SMC are in close agreement because of error cancellation: the "roughness-induced" surface excess of SMC is compensated by the negative bias originating from the positive curvature zones. Conversely, in the case of the "large buckyball" image, there are less positive curvature regions, which favor a positive bias.

The performances of both schemes have been also tested with respect to the errors made in approximating normals of discretized ideal objects. Spheres and

	MC		SMC	
	Nb. triangles	Std. error (deg)	Nb. triangles	Std. error (deg)
Sphere				
$r = 25$	2 941	10.1	1 591	10.8
$r = 35$	5 779	10.0	3 135	10.7
$r = 55$	14 125	9.9	7 789	10.6
Torus				
$R = 75$, $r = 25$	26 632	11.3	15 765	12.4
$R = 75$, $r = 35$	37 360	11.3	22 147	12.4
$R = 75$, $r = 55$	58 868	11.4	34 925	12.5

Table 2: Comparison of MC and SMC RMS errors on face normals on binarized images of a sphere and a torus

tori have been generated, then the SMC and MC surfaces have been extracted ; the normals of each face have been computed, as well as the normal that each face center should have on the associated ideal sphere or torus. The root-mean-square of the deviation between the actual and ideal normals is reported in Tables 2 for binarized images and 3 for smooth images. In binarized images, MC and SMC have comparable performances. On the other hand, in smooth images, the error using MC is cut by half, while it is practically unchanged when using SMC. This fact arises from the relative wealth of normal orientation possibilities in MC:*a priori* $511^3 - 1 = 133\,432\,830$ distinct orientations for MC against 26 for SMC, which is fully exploited when extracting a surface from a smooth data set.

4. Application to diffusion-deposition and diffusion-erosion problems

4.1. Context

The SMC algorithm has been developed and used in two practical cases of modeling in materials science, related to thermostructural composites. The first is the matrix fabrication by Chemical Vapor Infiltration (CVI) and the second is the surface degradation by ablation.

A practical problem of interest is the fabrication of ceramic or carbon-matrix composites by the Chemical Vapor Infiltration technique [17, 18, 19]. A preform

	MC		SMC	
	Nb. triangles	Std. error (deg)	Nb. triangles	Std. error (deg)
Sphere				
$r = 25$	2 941	4.8	1 585	10.3
$r = 35$	5 779	4.3	3 135	10.7
$r = 55$	14 125	4.2	7 789	10.6
Torus				
$R = 75$, $r = 25$	26 632	6.6	15 750	12.3
$R = 75$, $r = 35$	37 360	6.4	22 143	12.4
$R = 75$, $r = 55$	58 868	6.5	34 899	12.5

Table 3: Comparison of MC and SMC RMS errors on face normals on smoothed images of a sphere and a torus

made of carbon or ceramic fibers arranged in woven fabrics or mats, possibly punched with needles in order to lock the plies together, is infiltrated by the chemical cracking of a vapor precursor of the matrix material. In order to have a proper insight of the infiltration quality, it is of great interest to perform pore-scale modelling of the gas diffusion and of the deposition process [20].

4.2. Methods

The SMC discretization scheme is used both for the computation of effective diffusivities in porous media, and for the simulation of the matrix deposition, in combination with a Monte-Carlo/Random Walks (MC/RW) technique. The MC/RW technique has been used many times in media described by distributions of ideal objects, mostly overlapping or non-overlapping cylinders [21, 22, 23, 24] ; however, as far as real porous media are concerned, the material surface description by MC algorithms is necessary [25]. One of the interests of SMC in the case of random walks in porous media is that it is an interesting compromise between accuracy and computational speed. Indeed, in rarefied regime, the random walkers go straight from one surface point to another, and the orientation distribution law at surfaces depends on the normal exactly as in Lambert's law in optics. So, very accurate results are not expected for rarefied flow computations in a cuberille surface discretization scheme. On

Reduced configuration	Description	SMC cases
0	Void cube	0;1;2;3;4;6;7;10;13
1	One type I triangle	5;12
2	One rectangle	8;19
3	One type III triangle	9;14;20
4	Two type II triangles	11;16
5	Two type III triangles	17;18
6	One rectangle + one type III triangle	15
7	Solid cube	21

Table 4: Reduced configurations handled by the MC/RW algorithm

the other hand, using a full MC algorithm is a difficult task and leads to rather high memory usage for the storage of the data structure, though efficient techniques have been developed [26]. In the MC/RW - SMC combination, random walkers essentially have to manage their path through contiguous cubes containing up to two distinct surface pieces, either triangles or rectangles. Table 4 summarizes the distinct cases that are handled by SMC: they can be grouped into 8 "reduced" cases. A convenient data structure is then, for each cube, a case indicator in $(0, 7) \cap \mathbb{N}$, and coefficients for the description of at most two planes like $hx + ky + lz + m = 0$ with $(h, k, l) \in \{-1, 0, 1\}^3$ (at most 27 distinct triplets) and $m \in \mathbb{N}$. For example, this can be coded on 7 bytes: one for the case indicator, two for the description of the normal coefficients (h, k, l) of at most two planes, and four bytes for the integer m of at most two planes.

Here the SMC discretization technique, associated to the MC/RW method for fluid diffusion/reaction, is used for the prediction of the structural evolution of a given porous medium sample in CVI conditions. Compared to the diffusion code, the extra feature is the possibility to modify the surface, according to the local flux of reactive walkers received by a surface element. The chemical deposition is simulated by the introduction of a sticking probability associated to the walkers: when they hit a surface, a random number is drawn between 0 and 1; if it is inferior to the sticking probability, then the walker is stuck. In

order to simulate surface growth in accordance with the surface fluxes, there has to be a counter of sticking events associated to each surface element, and the surface should move forward in its normal direction by an amount of distance which is proportional to the received flux. There exists a very simple way to deal with these modeling requisites. Let us consider that we start with a black and white image. Black nodes (value = 255) are solid phase, white negative vertices (value = 0) are fluid vertices. Every time a walker gets stuck, we have to seek for the negative vertex which lies closest to the sticking point, and increase its grey level value by a given quantity. When this quantity exceeds a given threshold, then the vertex is converted into a surface node. The SMC discretization has to be performed again only on the 8 cubes sharing this new node: this provides a flexible, "on-the-fly" surface evolution algorithm.

4.3. Results

Combination of MC/RW and SMC has been used first in media consisting in an idealized description of stacked woven fabrics [27]. More recently, the acquisition [28] and segmentations [29, 30, 31] of high-resolution 3D images of carbon/carbon (C/C) fibrous preforms has been made possible. The SMC algorithm has been used for the computation of the surface area and pore diameter distribution, with an excellent validation with respect to experimental data [32]. quantitative comparison of the above estimates (surface area and pore diameter) using both SMC and MC could be provided.

Then, computations of effective gas diffusivities in the continuum (ordinary) and rarefied regime have been carried out, with an excellent agreement as compared to experimental data [33] and previous results on idealized media [34].

The random-walk algorithm has also been implemented together with the full MC discretization scheme, associated to an efficient memory storage strategy [26, 15], so this gives an opportunity to compare both discretization schemes directly with respect to a physical modelling application.

We report test results on a typical computation, performed on a 100^3 voxel image shown in fig. 5. The pore volume fraction was 86.9 %, the internal surface

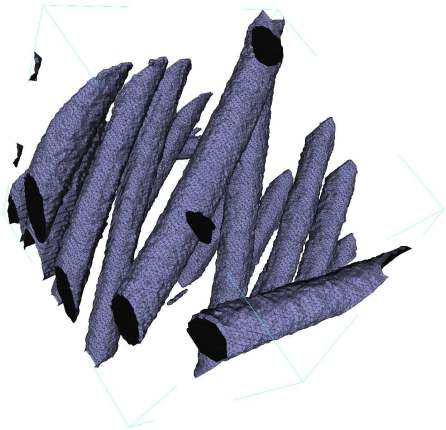


Figure 5: Typical image of a fibrous medium. Size = 100^3 voxels.

area was 0.0528737 pixel^{-1} (total surface = 52873.7 pixel^2) and the hydraulic diameter is around 66 pixels.

The effective diffusion coefficient was determined by averaging on 2,000 random walks, each 10,000 pixels in length ; whenever a random walker exits the image, it is reintroduced by translation ; the mean free path between two random direction changes (*i. e.* scattering events in the gas phase) has been varied from very small to very large values, allowing to cover the whole regime variation between continuum and rarefied regimes. The diffusion coefficient values found with both methods were in close agreement (within numerical noise amplitude) ; the CPU memory and time requirements are reported in table 5. We can see that in the rarefied regime the ratio between execution time for both codes gets worse ; indeed, it is the limit in which the codes spend relatively more time for an inspection of the surface triangles.

Moreover, the code performances have been compared for selected images, all of the same size, with different internal surface area. The results are summarized in Table 6. We can clearly see that the speedup gained by using the SMC scheme is more interesting when the surface area increases.

Table 8 is an example of infiltration movie obtained on a 100×100×100

	MC	SMC
Memory size (Mb)	18	6.7
Mean free path (vox.)	CPU time (s)	
0.05	1213	700
0.1	514	307
0.5	126	83
1.0	86	56
5.0	56	32
10.0	48	29
50.0	58	27
100.0	83	33

Table 5: Comparison of CPU memory and time consumption in a computation of an effective diffusion coefficient by random walks.

	Porosity		Int. surface area (pix^{-1})		Rel. error SMC/MC	
	SMC	MC	SMC	MC	Surface	Porosity
# 1	0.874	0.856	0.051	0.054	5.64%	-2.14%
# 2	0.676	0.628	0.128	0.132	2.88%	-7.67%
# 3	0.475	0.435	0.200	0.198	-0.97%	-9.14%

Table 6: Comparison of SMC and MC RW code performances on selected images with 100^3 voxels in size.

	Mem.size (Mb)		Time (s)		Ratio MC/SMC	
	SMC	MC	SMC	MC	Mem.	Time
# 1	6.7	18	27	77	2.7	2.9
# 2	8.5	19	28	191	2.2	6.8
# 3	10.3	21	31	406	2.0	13.1

Table 7: Comparison of SMC and MC RW code performances on selected images with 100^3 voxels in size.

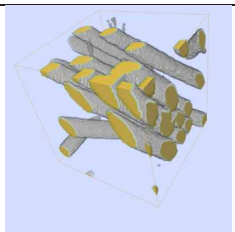
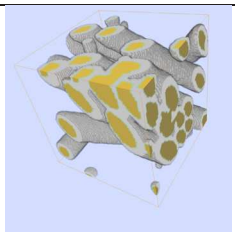
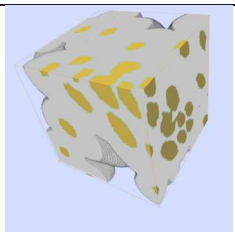
Step	1	2	3
Images			
Solid volume fraction	12.6%	25.3%	86.2%
Internal surface area (voxel edge^{-1})	0.0507	0.0677	0.0549

Table 8: An example of CVI simulation: three stages of the computation have been extracted among the 30 produced images.

cubic voxels image. Moderate values of the rarefaction degree and sticking probabilities have been chosen. The total infiltration has been run in 6 hours 30 on a Pentium4 CPU with 3.2 GHz clock rate. Such a kind of program is able to deliver a precise evolution of the internal surface area and transport properties (diffusivities, etc ...) as a function of infiltration progress, which is a valuable tool for large-scale infiltration modelling [20, 31].

The interest of such a combination of algorithms lies in the fact that it is a compromise between accuracy, CPU time consumption and memory requirements. It has also been used in the frame of C/C composite ablation modeling [35], with excellent results in terms of fidelity with respect to analytical cases [36], and computational speed in large images. Extensions featuring orientation-dependant sticking coefficients have been also tested [37], resulting in an excellent behavior.

5. Conclusion

The construction of a Simplified Marching Cubes algorithm was a demand arising from the community of materials numerical simulation, because of the necessity of managing physically complex simulations inside potentially huge 3D blocks. This has led to analyse thoroughly the nature and properties of

the historical Marching Cubes approach; after this, it was possible to propose a simplified (*i. e.* binarized) version of MC. The Simplified Marching Cubes rely on a mesh consisting in triangles linking only vertices of the cuberille grid. Therefore, computation can be made with integer arithmetic. Moreover, SMC produces a reduced number of points and triangles. These characteristics make the SMC computation faster than the standard MC and with less memory. Besides, the computation of surface area and volume in SMC is quite as simple as in the Cuberille case but much more accurate. Coupled together with a Monte-Carlo/Random Walk algorithm and an efficient surface deposition/erosion procedure, SMC has shown efficiency in performing computations of geometrical characteristics, gas transport by diffusion in rarefied regime, deposition by chemical reaction, and ablation by solid gasification.

One of the perspectives of SMC is to couple it with non-random numerical techniques like Finite Differences, Finite Volumes, Finite Elements or Boundary Integrals, always in the idea of solving moving boundary problems.

Acknowledgements

The authors wish to thank the French Ministry of Education for Ph. D. grants to Jean-François Delesse and Christianne Mulat.

References

[1] W. E. Lorensen, H. E. Cline, Marching cubes: A high resolution 3D surface construction algorithm, in: M. C. Stone (Ed.), SIGGRAPH '87 Proceedings, Vol. 21 of ACM Computer Graphics, ACM Press, New York, 1987, pp. 163–169.

[2] T. S. Newman, H. Yi, A survey of the marching cubes algorithm, Elsevier Computer & Graphics 30 (1996) 854–879.

[3] G. M. Nielson, B. Hamann, The asymptotic decider: Resolving the ambiguity in marching cubes, in: G. M. .Nielson, L. Rosenblum (Eds.), Pro-

ceedings of the conference on Visualization '91, IEEE Computer Society Press, 1991, pp. 83–91.

[4] G. M. Nielson, On marching cubes, IEEE Trans. on Visualization and Computer Graphics 9 (3) (2003) 283–297.

[5] E. V. Chernyaev, Marching cubes 33: Construction of topologically correct isosurfaces, Tech. Rep. CN/95-17, CERN (1995).

[6] T. Lewiner, H. Lopes, A. W. Vieira, G. Tavares, Efficient implementation of marching cubes cases with topological guarantees, Journal of Graphics Tools 8 (2) (2003) 1–15.

[7] A. Lopes, K. Brodlie, Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing, IEEE Transactions on Vizualisation and Computer Graphics 9 (1) (2003) 16–29.

[8] P. Cignoni, P. Marino, C. Montani, E. Puppo, R. Scopigno, Speeding up isosurface extraction using interval trees, IEEE Transactions on Vizualisation and Computer Graphics 3 (2) (1997) 158–170.

[9] Y. Livnat, H.-W. Shen, C. R. Johnson, A near optimal isosurface extraction algorithm using the span space, IEEE Transactions on Vizualisation and Computer Graphics 2 (1996) 73–84.

[10] C. Montani, R. Scateni, R. Scopigno, Discretized marching cubes, in: Proceedings of the IEEE Conference on Visualization '94, IEEE Computer Society Press, 1994, pp. 281–287.

[11] A. del Río, J. Fischer, D. Bartz, W. Straßer, Fast rendering of large encoded isosurfaces from uniform grid datasets, in: G. Greiner (Ed.), Vision, Modelling, Visualization (VMV), AKA, Akademische Verlagsgesellschaft, IOS Press (Berlin, Amsterdam), 2005, pp. 71–78.

[12] S. Osher, R. P. Fedkiw, Level sets and dynamic implicit surfaces, Vol. 153 of Applied Mathematical Sciences, Springer-Verlag, New York, 2002.

[13] C. W. Hirt, B. D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, J. Comput. Phys. 39 (1981) 201–225.

[14] G. M. Nielson, Dual marching cubes, in: Proceedings of the IEEE Conference on Visualization '04, IEEE Computer Society Press, 2004, pp. 489–496.

[15] J.-F. Delesse, Diffusion et réaction des gaz en milieux poreux, PhD thesis n°2598, Université de Bordeaux I (2002).

[16] S. Roettger, The volume library (January 2006).
URL `http://www9.informatik.uni-erlangen.de/External/vollib/`

[17] T. M. Besmann, B. W. Sheldon, R. A. Lowden, D. P. Stinton, Vapor-phase fabrication and properties of continuous-filament ceramic composites, Science 253 (1991) 1104–1109.

[18] R. Naslain, F. Langlais, Fundamental and practical aspects of the chemical vapor infiltration of porous substrates, High Temperature Science 27 (1990) 221–235.

[19] R. Naslain, F. Langlais, CVD-processing of ceramic-ceramic composite materials, in: R. Tressler, G. Messing, C. Pantano, R. Newnham (Eds.), Tailoring multiphase and composite ceramics, Vol. 20 of Mat. Sci. Res., Kluwer Acad. Pub., Dordrecht, The Netherlands, 1986, pp. 145–164.

[20] G. L. Vignoles, Modelling of the CVI processes, Adv. Sci. Technol. 50 (2006) 97–106.

[21] V. N. Burganos, S. V. Sotirchos, Knudsen diffusion in parallel, multidimensional, or randomly oriented pore structures, Chem. Eng. Sci. 44 (1989) 2451.

[22] M. M. Tomadakis, S. V. Sotirchos, Effects of fiber orientation and overlapping on Knudsen, transition, and ordinary regime diffusion in fibrous

structures, in: T. M. Besmann, B. M. Gallois, J. W. Warren (Eds.), Chemical Vapor Deposition of Refractory Metals and Ceramics II, Vol. 250 of Mat. Res. Soc. Symp. Proc., Materials Research Society, Pittsburgh, 1992, pp. 221–226.

[23] R. R. Melkote, K. F. Jensen, Computation of transition and molecular diffusivities in fibrous media, AIChE J. 38 (1992) 56–61.

[24] M. Tassopoulos, D. E. Rosner, Simulation of vapor diffusion in anisotropic particulate deposits, Chem. Eng. Sci. 47 (1992) 421–443.

[25] P. Blasi, B. L. Saëc, G. L. Vignoles, Application of rendering techniques to Monte-Carlo physical simulation of gas diffusion, in: J. Dorsey, P. Slusallek (Eds.), Eurographics Rendering Workshop 1997, Eurographics, Springer Wien, New York City, NY, USA, 1997.

[26] J.-F. Delesse, B. Le Saëc, G. L. Vignoles, A new data structure for the computation of equivalent properties in 3d porous media, in: D. Dutta, H. P. Seidel (Eds.), Solid Modelling 2001 (Proc. 6th ACM symposium on Solid Modelling and Applications), ACM Press, New York, NY, 2001.

[27] G. L. Vignoles, Modelling binary, Knudsen, and transition regime diffusion inside complex porous media, J. de Physique IV C5 (1995) 159–166.

[28] O. Coindreau, G. L. Vignoles, P. Cloetens, Direct 3D microscale imaging of carbon-carbon composites with computed holotomography, Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms 200 (2003) 308–314.

[29] G. L. Vignoles, Image segmentation for hard X-ray phase contrast images of C/C composites, Carbon 39 (2001) 167–173.

[30] J. Martín-Herrero, C. Germain, Microstructure reconstruction of fibrous C/C composites from X-ray microtomography, Carbon 45 (5) (2007) 1242–1253.

[31] G. L. Vignoles, C. Germain, O. Coindreau, C. Mulat, W. Ros, Fibre-scale modelling of C/C processing by chemical vapour infiltration using X-ray CMT images and random walkers, in: M. T. Swihart, D. Barreca, R. A. Adomaitis, K. Wörkhoff (Eds.), Procs. ICVD XVII & EuroCVD 17, Vol. 25 of ECS Transactions, The Electrochemical Society, Pennington, NJ, 2009, pp. 1275–1284.

[32] O. Coindreau, G. L. Vignoles, Assessment of geometrical and transport properties of a fibrous C/C composite preform as digitized by X-ray computed micro-tomography. Part I : Image acquisition and geometrical properties, J. Mater. Res. 20 (2005) 2328–2339.

[33] G. L. Vignoles, O. Coindreau, A. Ahmadi, D. Bernard, Assessment of geometrical and transport properties of a fibrous C/C composite preform as digitized by X-ray computed micro-tomography. Part II : Heat and gas transport, J. Mater. Res. 22 (6) (2007) 1537–1550.

[34] O. Coindreau, G. L. Vignoles, J.-M. Goyhénèche, Multiscale X-ray CMT of C/C composites : a tool for properties assessment, in: N. P. Bansal, J. P. Singh, W. M. Kriven (Eds.), Advances in Ceramic-Matrix Composites XI, Vol. 175 of Ceram. Trans., The American Ceramic Society, Westerville, OH, 2005, pp. 77–84.

[35] G. L. Vignoles, Y. Aspa, J. Lachaud, Roughness evolution in ablation of carbon-based materials: multi-scale modelling and material analysis, in: K. Fletcher (Ed.), 5th European Workshop on Thermal Protection Systems and Hot Structures, Vol. SP-631 of ESA Confs. Procs., ESA-ESTEC, ESA, Noordwijk,The Netherlands, 2006.

[36] J. Lachaud, Y. Aspa, G. L. Vignoles, Analytical modeling of the steady state ablation of a 3D C/C composite, Int. J. of Heat and Mass Transfer 51 (2008) 2614–2627.

[37] J. Lachaud, Y. Aspa, G. L. Vignoles, J.-M. Goyhénèche, 3D modeling of thermochemical ablation in carbon-based materials: effect of anisotropy on

surface roughness onset, in: M. Dinguirard, J. Kleiman (Eds.), Proc. 10th International Symposium on Materials in a Space Environment, Vol. SP-616 of ESA Confs. Procs., ESA Publications, Noordwijk, The Netherlands, 2006, 10 p.

Appendix A. Algorithm

Terminology, concepts and graphical symbols

First of all, let us define terms and introduce concepts or symbols which will be useful for the explanation of the SMC method.

Vertices

- Positive and negative vertices (positive vertices are also called "nodes")

  Vertices with a value above or equal to the threshold value are called positive vertices , while the other ones are called negative vertices [5]. For sake of brevity, we choose to call "nodes" the positive vertices.

- Interface nodes

  An interface node is a positive vertex which has at least one negative vertex as a neighbor (in 6-connectivity, see the section "Neighbors" below). Some nodes lie on the interface whereas the other nodes are outside it.

Coordinates and cumulative coordinates

Without loss of generality, it is convenient to assume that each processed cube is composed of 8 neighboring vertices with normalized coordinates denoted c_x^n ($n = 0...7$) and equal to 0 or 1 along the three directions $x = i$, j or k (Fig. A.7). Cumulative coordinates C_x are defined as the sum of coordinates of the nodes:

$$C_x = \sum_{n \in \{nodes\}} c_x^n.$$

For a given configuration, the first step of MC is to identify the similar case in the lookup-table. In the SMC method each configuration is evaluated without a reference table: the cumulative coordinates are used to identify subcases.

Neighbors

Using 6-connectivity, each vertex of a cube has three immediate neighboring vertices which are linked by edges. Noticing that the number n of a vertex and its coordinates are related by the expressions:

$$n = c_i^n + 2 \times c_j^n + 4 \times c_k^n \longleftrightarrow \begin{cases} c_i^n = n\%2 \\ c_j^n = (n\%4)/2 \\ c_k^n = n/4 \end{cases} \tag{A.1}$$

where / and % denotes respectively the quotient and the remainder of the euclidian division, the number $N_x(n)$, ($x = i, j$, or k) of neighbors of a vertex can be easily computed. For example, the number $N_i(n)$ of neighbors along the direction i is defined as follows:

$$N_i(n) = \overline{c_i^n} + 2 \times c_j^n + 4 \times c_k^n$$

where $\overline{a}$ denotes the complement to 1 of a. Interface nodes are nodes with less than three nodes as immediate neighbors, in 6-connectivity. As a consequence, the negative vertices have a neighboring scheme which is based on 18-connectivity.

Triangle types

By linking vertices of a cube, three types of triangles can be formed: one isosceles-right triangle (type I), one rectangle triangle (type II) and one equilateral triangle (type III). They differ by their total edge length L, surface area S and normal vector orientation. Moreover, their normal vector has coordinates $\frac{1}{N_n}(h, k, l) \in \{-1; 0; 1\}^3$ such that $N_n^2 = h^2 + k^2 + l^2$ equals 1 (type I); 2 (type II) or 3 (type III). In addition to these three criteria, we propose another criterion exclusively based on integer arithmetics, which may help to speed up the

Type	L	S	N_n^2	C_L	Scheme
I	$2+\sqrt{2}$	$\frac{1}{2}$	1	6	
II	$1+\sqrt{2}+\sqrt{3}$	$\frac{\sqrt{2}}{2}$	2	14	
III	$3\sqrt{2}$	$\frac{\sqrt{3}}{2}$	3	12	

Table A.9: Criteria L (edge length), S (surface), N_n^2 and C_L for the three types of triangles.

computations:

$$C_L(n_1, n_2, n_3) = \sum_{l=1}^{3} \sum_{m=l+1}^{3} \left(\sum_{x=i,j,k} (c_x^{n_l} - c_x^{n_m})^2 \right)^2 .$$

For the three types of triangle, the value of the criteria L, S N_n^2 and C_L are summarized in table A.9.

Graphical symbols

Since the status of neighboring vertices is taken into account to determine the triangle model of each case, introduction of some graphical symbols (Fig. A.6) can facilitate the explanation. We choose to draw a node with a dark disc and a vertex (positive or negative) which has three nodes as immediate neighbors with a larger circle.

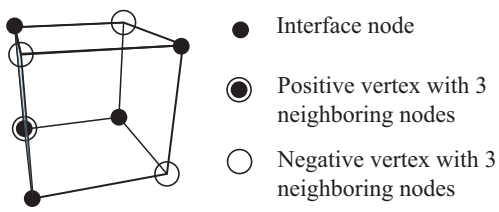


Figure A.6: Symbols.

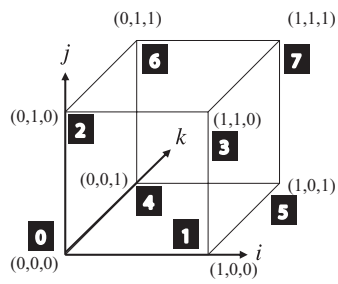


Figure A.7: Coordinates. Numbers on black background refer to n in text and eq. A.1

Algorithm

A major difference between the SMC algorithm and the MC algorithm is the way the configurations are analyzed. MC operates with a look-up table: the first step consists in identifying the equivalent case; the transformation to apply to vertices which leads to it must be inverted in the last step in order to obtain the triangle model. In the SMC, there is no need to identify an equivalent configuration and each case is analyzed independently from the others. However, it is rather illustrative to build an SMC look-up table: it will allow to describe the distinct triangle models and to give a clear comparison with MC algorithms.

Now, the SMC table may be deduced from the MC table, as displayed at Fig. 2 . The SMC algorithm consists in choosing triangles composed from interface nodes according to the number of nodes, the cumulative coordinates, the number of interface nodes and neighborhood of vertices. The first step is to build the list of all nodes: other computations depend on subcases.

Inspection of all configurations

There is no triangle model for configurations with 0, 1, 2 or 8 nodes. For other configurations, all subcases are discussed according to the denomination of the original look-up table of MC taking into account the switch of status of vertices.

- Configurations with 3 nodes. These configurations are composed of cases 5, 6 and 7. Only the first subcase leads to one triangle (type I). It corresponds to three nodes on a cube face and can be checked by the following test:

$$C_x = 0 \text{ or } 3, \quad x = i, j \text{ or } k.$$

- Configurations with 4 nodes. These configurations are composed of cases 8 to 14.

  - **Case 8.** This first case corresponds to four nodes on a cube face. It can be checked by the following test:

    $$C_x = 0 \text{ or } 4, \quad x = i, j \text{ or } k.$$

    The triangle model is composed of two type I triangles belonging to the adequate face.

  - **Cases 10 and 13.** These cases do no lead to triangles. They correspond to two nodes on all cube faces and can be checked by the following test:

    $$C_x = 2, \quad x = i, j \text{ and } k.$$

  - **Case 9.** Only one node is not an interface node. It can be identified by an exhaustive test of all nodes. This case leads to one type III triangle formed by the interface nodes.

  - **Case 12.** One singular node has only negative vertices as neighbors. This case leads to one type I triangle. After an exhaustive test of all nodes, the simplest way to build the triangle is to switch the status of the isolated node and process again the configuration (3 nodes, case 8).

- **Case 11 and 14.** This last case leads to two type II triangles which can be easily identified by nested loops on nodes.

- Configurations with 5 nodes. These configurations are composed of cases 15 to 17.

  - **Case 17.** Four nodes are on a cube face and the remainder node is on the opposite face. It can be checked by the following test:

    $$C_x = 1 \text{ or } 4, \quad x = i, j \text{ or } k.$$

    The triangle model is composed of two type II triangles which can be easily identified by nested loops on nodes.

  - **Case 15.** One singular node has only negative vertices as neighboors. This case leads to one type III triangle. After an exhaustive test of all nodes, the simplest way to build the triangle is to switch the status of the singular node and process again the configuration (3 nodes, case 9).

  - **Case 16.** This case leads to three triangles (two type II and one type III). The first step consists in identifying the triangle of type III with nested loops on nodes. Then the simplest way to build other triangles is to find the negative vertex which has nodes as neighbors, switch its status and process again the configuration (6 nodes, case 20).

- Configurations with 6 nodes. These configurations are composed of cases 18 to 20.

  - **Cases 18 and 19.** Two negative vertices have only nodes as neighbors. This case leads to two type III triangles. An exhaustive test of all vertices allows to identify them and to build triangles formed by their neighbors.

– **Case 20.** This case leads to two type II triangles which are formed by correctly ordered interface nodes.

- Configuration with 7 nodes. This configuration (21) is the reverse of case 1 (Fig. 2). This case leads to one triangle (type III). The simplest way to build it is to find the only negative vertex and form a triangle with its three neighbors which are interface nodes.

```
int make_SMC() {
         nnodes = compute_number_nodes();
         switch (nnodes) {
    case 0: /* Configuration 0 */
    case 1: /* Configuration 1 */
    case 2: /* Configurations 2,3 and 4 */
    case 8: /* Configuration 22 */
         return 0; /* No triangle */
    case 3 :
        compute_sums(C_i,C_j,C_k);
     if (!(C_i%3) || !(C_k%3) || !(C_k%3)) {
             /* Configuration 5 : One type I triangle */
                    build_triangle(nodes);
                    return 1;
       }
      else {
         /* Configurations 6 and 7 */
return 0;
}
break;
    case 4:
        compute_sums(C_i,C_j,C_k);

     if (!(C_i%4) || !(C_k%4) || !(C_k%4)) {
            /* Configuration 8 : Two type I triangles on a face */
             build_triangles(nodes);
             return 2;
     }
     if ((C_i == 2  && C_j == 2 && C_k == 2)) {
    /* Configurations 10 and 13 : No triangles */
return 0;
}
if (C_i !=2 && C_j !=2 && C_k !=2) {
                 /* Configuration 9 : One type III triangle */
     find_interface_nodes(nodes,interface_nodes);
```

```
     build_triangle(interface_nodes);
     return 1;
     }
         C_t = (C_i%4 + C_j%4 + C_k%4);
         if (!(C_t%2)) {
    /* Configurations 11 and 14 : Two type II triangles */
build_triangles(nodes);/* by nested loops */
return 2;
         }
         else {
            /* Configuration 12 : One type I triangle */
find_isolated_nodes(nodes,isolated_nodes);
             switch_status(isolated_node[0]);
             /* Now we are in Configuration 5 */
             build_triangle(nodes);
             return 1;
     }
     break;
    case 5:
        compute_sums(C_i,C_j,C_k);
     if (!((C_i - 1)%3) || !((C_j - 1)%3) || !((C_k - 1)%3)) {
    /* Configuration 17 : Two type II triangles */
     find_interface_nodes(nodes,interface_nodes);
build_triangles(interface_nodes);/* by nested loops */
return 2;
     }
if (find_isolated_nodes(nodes,isolated_nodes)) {
    /* Configuration 15 : one type III triangle */
            switch_status(isolated_node[0]);
             /* Now we are in Configuration 9 */
             build_triangle(nodes);
             return 1;
}
else {
    /* Configuration 16 : 2 type II and one type III triangles */

With nested loops, build all triangles, compute N_n_squared for each of them

if (N_n_squared == 3) {/* type III identified */
store_triangle();
}
/* Construction of the remaining type II triangles */
/* Find the negative vertex with 3 positive neighbors */
        for (ivertex=0;ivertex<8;ivertex++) {
            if (vertex[ivertex]==NEGATIVE) {
                find_positive_neighbors(ivertex,number_neighbors,neighbors);
```

```
if (number_neighbors) {
    switch_status(ivertex);
    /* Now we are in Configuration 20 */
                    build_triangles(nodes);
                    add_stored_triangle();
                    return 3;
}
    }
}
}
     break;
    case 6 :
        compute_sums(C_i,C_j,C_k);
     C_t = C_i%6 + C_j%6 + C_k%6;
     if ((C_t %2)  && !(C_i%6 == C_j%6 && C_k%6 == C_i%6)) {
   /* Configuration 20 : 2 type II triangles on a plane */
     find_interface_nodes(nodes,interface_nodes);
build_triangles(interface_nodes);
return 2;
     }
     else {
   /* Configurations 18 and 19 : two type III triangles */
        for (ivertex=0;ivertex<8;ivertex++) {
            if (vertex[ivertex]==NEGATIVE) {
                find_positive_neighbors(ivertex,number_neighbors,neighbors);
if (number_neighbors == 3) {
                    build_triangles(neighbors);
}
    }
}
                    return 2;
     }
     break;
    case 7 :
             /* Configuration 21 : one type III triangle */
              for (ivertex=0;ivertex<8;ivertex++) {
                  if (vertex[ivertex]==NEGATIVE) {
                      find_positive_neighbors(ivertex,number_neighbors,neighbors);
      if (number_neighbors == 3) {
                          build_triangle(neighbors);
      }
          }
      }
                         return 1;
     break;
         } /* End switch (nnodes) */
```