International Journal of Minerals, Metallurgy and Materials
Volume 19, Number 8, Aug 2012, Page 699
DOI: 10.1007/s12613-012-0616-0

A cellular automata model for simulating grain structures with straight and hyperbolic interfaces

A. Ramírez-López[1, 2, 3], *M. Palomar-Pardavé*[1], *D. Muñoz-Negrón*[2], *C. Duran-Valencia*[3], *S. López-Ramirez*[3], *and G. Soto-Cortés*[1]

1) Materials and Energy Department, Metropolitan Autonomous University, Mexico
2) Department of Industrial Engineering, Technological Autonomous Institute of Mexico, Mexico
3) Department of Molecular Engineering, Mexican Institute of Petroleum, Mexico
(Received: 11 May 2011; revised: 4 July 2011; accepted: 11 July 2011)

Abstract: A description of a mathematical algorithm for simulating grain structures with straight and hyperbolic interfaces is shown. The presence of straight and hyperbolic interfaces in many grain structures of metallic materials is due to different solidification conditions, including different solidification speeds, growth directions, and delaying on the nucleation times of each nucleated node. Grain growth is a complex problem to be simulated; therefore, computational methods based on the chaos theory have been developed for this purpose. Straight and hyperbolic interfaces are between columnar and equiaxed grain structures or in transition zones. The algorithm developed in this work involves random distributions of temperature to assign preferential probabilities to each node of the simulated sample for nucleation according to previously defined boundary conditions. Moreover, more than one single nucleation process can be established in order to generate hyperbolic interfaces between the grains. The appearance of new nucleated nodes is declared in sequences with a particular number of nucleated nodes and a number of steps for execution. This input information influences directly on the final grain structure (grain size and distribution). Preferential growth directions are also established to obtain equiaxed and columnar grains. The simulation is done using routines for nucleation and growth nested inside the main function. Here, random numbers are generated to place the coordinates of each new nucleated node at each nucleation sequence according to a solidification probability. Nucleation and growth routines are executed as a function of nodal availability in order to know if a node will be part of a grain. Finally, this information is saved in a two-dimensional computational array and displayed on the computer screen placing color pixels on the corresponding position forming an image as is done in cellular automaton.

Keywords: grain growth; interfaces; grain size and shape; computational methods; algorithms; cellular automata; computer simulation

1. Introduction

Simulation of grain structures is a very important topic in materials science. Metallic pieces are frequently processed and formed by foundry, laminated, rolling, *etc.* During manufacturing and processing, different grain sizes and morphologies are obtained according to particular solidification conditions and thermal treatments. Fractals, stochastic methods, cellular automata, models based on Monte Carlo, random walk, and the chaos theory have been employed to simulate the anisotropic features of grain structures and the phenomena related such as recrystallization and grain deformation during processing.

Many authors [1-15] have developed mathematical models to simulate grain growth during solidification of metallic materials (metals and alloys). The beginners [2] used basic geometrical models due to the limited computational capacities. Nevertheless, the development of new mathematical methods, more efficient algorithms, and the increment in computational data speed and storage capacities have been possible to solve complex problems that involve no-regular geometries and random processes using Monte Carlo [1, 2, 4, 7] and cellular automaton [5, 8, 11-15] methods in problems related to materials science. Some of these authors [1, 4-8,

Corresponding author: A. Ramírez-López E-mail: adaramil@yahoo.com.mx

10-12] have developed models for solidification processes as a function of solid and liquid fractions for different metals. Others [1-4, 6, 9, 12-15] have been working in models to simulate dendrite growth, which is the basic structure in primary metallic products obtained after foundry.

In a previous work, Ramírez *et al.* [15] explained the development of a basic computational algorithm to create grain structures as a function of an initial stochastic temperature distribution during the solidification of a squared metallic sample. Heterogeneous distributions of temperature provide nodes with particular probabilities to be nucleated. The results are zones with big and small grains. The assignation of preferential growth directions gives samples with different grain sizes. Nevertheless, in the original algorithm, there was only one single nucleation execution.

The grain structures formed in metals are heterogeneous and the grain morphology can also be very different due to the factors involved in the manufacturing process. Grains with equiaxed and columnar morphologies, with different sizes and distributions, can be simulated as shown in Figs. 1(a) and 1(b), respectively, using the original algorithm developed [15]. Nevertheless, the interfaces between the grains (grain boundaries) are always straight (straight lines for two-dimensional (2D) models and flat planes for three-dimensional (3D) models) as shown in better detail in the close-ups, which show the boundaries between these. This work describes an improved algorithm to generate more complex 2D grain structures. The new algorithm includes multiple nucleation sequences to generate different grain sizes and hyperbolic interfaces between them.

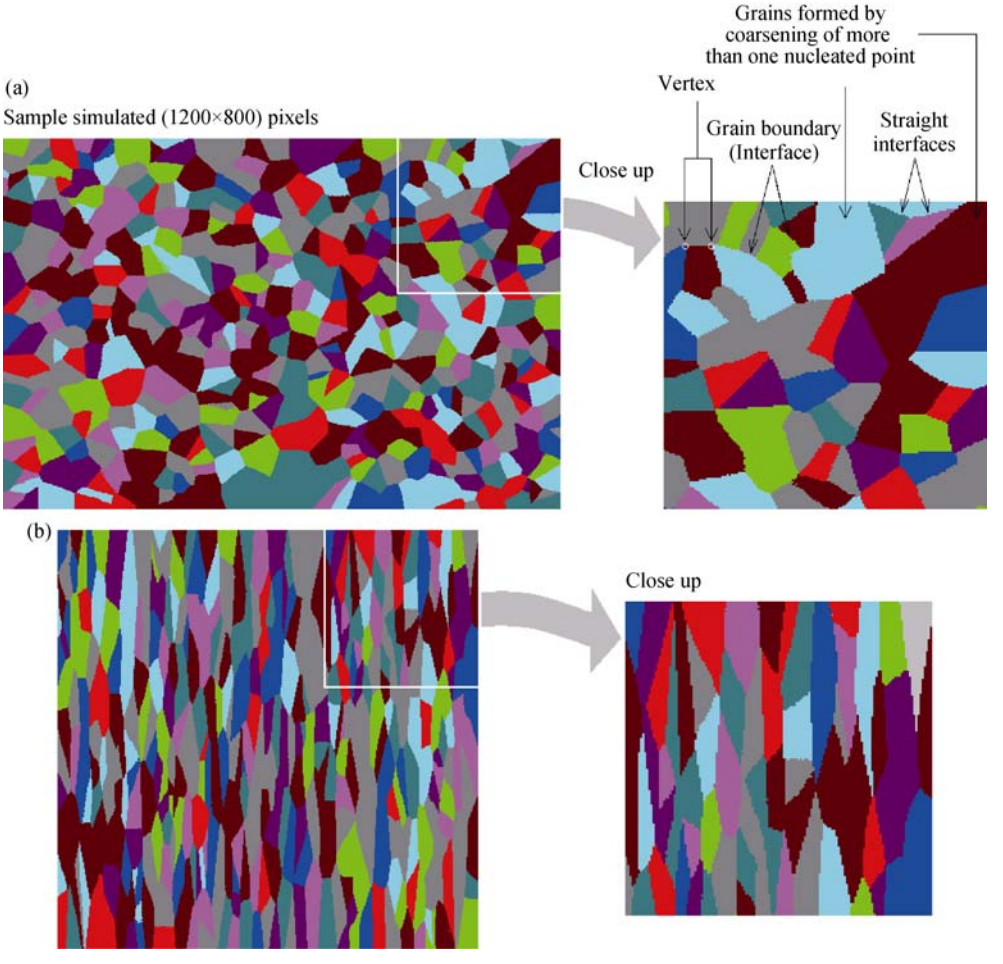

Fig. 1. Different grain structures with straight interfaces: (a) equiaxed grains; (b) columnar grains.

Straight lines are presented in grain boundaries during the evolution of the simulation as shown in Figs. 1(a) and 1(b). These are formed because all the nodes are nucleated in only one single sequence and because all of them have the same growth rate. Nevertheless, this does not happen during real solidification.

Fig. 1(a) shows an equiaxed grain structure. This kind resulted after the metal is foundry and cast. Grain size distribution is frequently a function of the nucleation node distribution in the sample and the growth procedure. Neverthe-

less, Fig. 1(b) shows a columnar structure. This structure is obtained after a secondary process such as lamination, rolling, or extrusion, where the original grain structure is squeezed and deformed.

Although the algorithm developed does not relate solidification with physical evolution, it is evident that the chaos theory and cellular automata models generate samples of grain structures very similar to those in the real metallic specimens. Nucleation and grain growth are influenced by many factors, such as solidification speed and phase transformations in the metallic materials, as described next.

The influence of the previous temperature distribution and solidification speed is related to the number of nucleation nodes to be placed in a region of the sample simulated. The nodes in regions quickly quenched will have a greater probability to be nucleated than those in regions where the solidification speed is slow. This condition will reduce the grain size because this node remains a very short time in mushy. These regions are frequently near the surfaces where heat removal is applied. Here, a great number of nucleation nodes will appear due to the high initial solidification speed. In consequence, the growth process of these nodes will be interrupted because the available space will be quickly occupied; in consequence, the simultaneous growth of many grains will be blocked by each other, forming an interface.

In contrast, the nodes nucleated in regions with a minor population and lower solidification speeds will have larger spaces for growing. Nucleation speed is reduced in zones near the core sample because here the nodes remain a long time in mushy and the evolution of the liquid and solid fractions ($X_{\text{liq}\,I,J}$ and $X_{\text{sol}\,I,J}$) is also slow.

Liquid metallic samples are cast to produce mechanical elements and pieces for machines, tools, *etc.* The regions where heat removal is applied will be quickly quenched; here, equations for forced convection and radiation are frequently used for calculation. However, inside the piece core, latent heat remains; this heat is slowly distributed by conduction from the core towards the surfaces.

2. Initial solidification speed distribution

The model in the present work was developed to create a cellular automaton image using two numerical codes. The first code is used to assign the nucleation probabilities to each node. The second code is used to identify and display the grain structure. In this way, there were two algorithms: the first algorithm was programmed to generate an original stochastic distribution of the temperature and the solidification speed. The second was executed to simulate the nucleation and grain growth processes.

The basic assumptions for the simulation of the solidification speed distribution are the following. (1) A 2D squared sample of a metallic material is simulated. (2) A structured regular squared mesh is used for the discretization. (3) The numbers of nodes used for the discretization are nx and ny for the horizontal and vertical axes, respectively. (4) The sub-indexes I and J are used to identify the nodal position in the computational domain. (5) The values of nx and ny are declared as integer data type; these are fit in order to provide a regular squared mesh. (6) Each node in the temperature distribution will represent the same node in the grain structure and it will be represented with a color pixel in the cellular automata.

Temperature and solidification speed distributions for the simulations are generated using an algorithm based on a stochastic assignation value subroutine. Here, every temperature value is classified as done for populations in a statistical analysis according to their frequency. Then, a probability to be presented in the sample is defined as boundary conditions for the four sides (sample boundaries). Then, the appearing probability in a nodal position $P_{T,I,J}$ is calculated using Eq. (1) as a function of the corresponding nodal position (I,J) in the sample. Here, the sub-index T is used for temperature. Although during solidification of the real sample the nucleation points appear as a function of the heat removal conditions, here it is assumed as a random process.

$$P_{T,I,J}=\frac{\dfrac{I}{\text{nx}}\left(P_{T,I=1}+P_{T,I=\text{nx}}\right)+\dfrac{J}{\text{ny}}\left(P_{T,J=1}+P_{T,J=\text{ny}}\right)}{2} \tag{1}$$

The process to calculate nodal probabilities is executed in a pair of nested loops for all the samples. This process is repeated for all the defined temperatures (nT), and the nodal probability is obtained using Eq. (2). Then, the probability for all the temperatures is given by the sum of the particular probabilities because ($P_{\text{tot},I,J}$=100). Instead, the probabilities are compared with a randomly generated number (Z); if the value of (Z) is in the range between P_{T-1} and P_T ($P_{T-1}<Z<P_T$), the probability of the temperature is assigned to the node. This process is shown in the shaded area of the flowchart in Fig. 2.

$$P_{\text{tot}}=\sum_{T=1}^{nT}P_{T,I,J}=100 \tag{2}$$

The process for generating a random probability for nucleation is also nested in a pair of loops to create regions

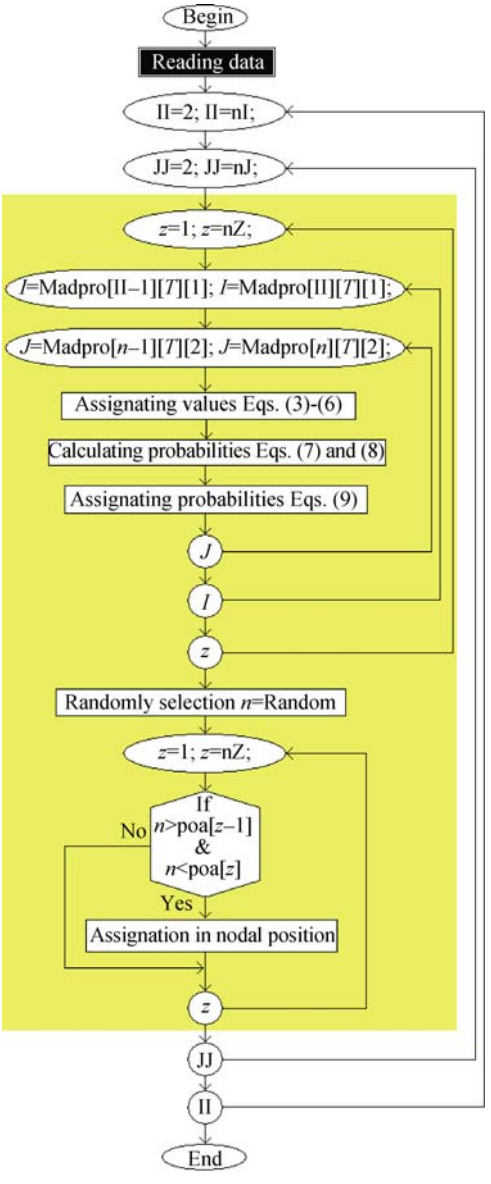

Fig. 2. Flowchart to generate the initial temperature and solidification speed distributions.

with less or greater probabilities for nucleation and growth in the same sample.

These loops rule the process and can be used for becoming an only simple routine in a computational tool to develop a more sophisticated parcel model. In the same way as the previous calculation, the boundary conditions of each parcel are defined independently to calculate the nodal probabilities.

The algorithm for creating the distribution is shown in Fig. 2. Here, the variables nI and nJ are used to know the number of defined regions in the sample. These are used as boundary conditions to command the loops for generating the distributions on each parcel. These go from II=2 to II≤*n*I and from JJ=2 to JJ≤*n*J for horizontal and vertical axes. Here, a 3D array is used to store the values. The computational array was declared as Madpro[*o*][*T*][*A*]. Here, *o* represents a number code used to identify the region on the axes; *o* can take values from II or JJ according to the executing loop. The value of *T* is used for declaring the probability for nucleation of the corresponding class. Finally, the third value *A* corresponds to a numerical code used to identify the axis: "1" is for the horizontal and "2" is for the vertical.

The new process includes the assignation to the boundary conditions for each parcel to the variables I_1, I_2, J_1, and J_2 as shown in Eqs. (3) to (6). Here, the second and third terms correspond to the mathematical and computational notations, respectively.

$$I_1=\mathrm{Md}_{\mathrm{II}-1,T,I}=\mathrm{Madpro}[\mathrm{II}-1][T][1] \tag{3}$$

$$I_2=\mathrm{Md}_{\mathrm{II},T,I}=\mathrm{Madpro}[\mathrm{II}][T][1] \tag{4}$$

$$J_1=\mathrm{Md}_{\mathrm{JJ}-1,T,J}=\mathrm{Madpro}[\mathrm{JJ}-1][T][2] \tag{5}$$

$$J_2=\mathrm{Md}_{\mathrm{JJ},T,J}=\mathrm{Madpro}[\mathrm{JJ}][T][2] \tag{6}$$

Then, the probabilities for horizontal and vertical axes are obtained using Eqs. (7) and (8), respectively. The second and third terms are used for the same notations as Eqs. (3) to (6).

$$\mathrm{prx}=I_1+(I_2-I_1)\left(\frac{I-\mathrm{Md}_{\mathrm{II}-1,T,1}}{\mathrm{Md}_{\mathrm{II},T,1}-\mathrm{Md}_{\mathrm{II}-1,T,1}}\right)= I_1+(I_2-I_1)\left(\frac{I-\mathrm{Madpro}[\mathrm{II}-1][T][1]}{\mathrm{Madpro}[\mathrm{II}][T][1]-\mathrm{Madpro}[\mathrm{II}-1][T][1]}\right) \tag{7}$$

$$\mathrm{pry}=J_1+(J_2-J_1)\left(\frac{J-\mathrm{Md}_{\mathrm{JJ}-1,T,2}}{\mathrm{Md}_{\mathrm{JJ},T,2}-\mathrm{Md}_{\mathrm{JJ}-1,T,2}}\right)= J_1+(J_2-J_1)\left(\frac{J-\mathrm{Madpro}[\mathrm{JJ}-1][T][2]}{\mathrm{Madpro}[\mathrm{JJ}][T][2]-\mathrm{Madpro}[\mathrm{JJ}-1][T][2]}\right) \tag{8}$$

The cumulative nodal probability is obtained by solving Eq. (9) and saved in a one-dimensional array called poa[*T*]. Finally, a random number is generated, and when the sentence "if" in the flowchart is true, the corresponded probability for nucleation is assigned to the node.

$$\mathrm{poa}[T]=\mathrm{poa}\left[T-1\right]+\frac{\mathrm{prx}+\mathrm{pry}}{2} \tag{9}$$

3. Grain growth algorithm

The solidification speeds are used to assign a nodal pro-

bability for nucleation, resulting in a grain structure formed in a cellular automata image. Here, integer data type is used as a code number to identify each grain, and the final values are taken from the final assignation to be stored in a new 2D array. Nodal positions for the nucleated nodes are also stored in this 2D array declared as MaPuNu[*m*][data]. Here, *m* is the nucleated node number and "data" uses a numerical code to save different values in the storage; in this case, the first location (1) is used to store the horizontal position, the second location (2) is for the vertical, and the third (3) is for the color assignation.

The flowchart of the mathematical algorithm developed for nucleation and growth is shown in Fig. 3. The column of the middle is the main function for simulation. The subroutines required are placed beside in shaded zones.

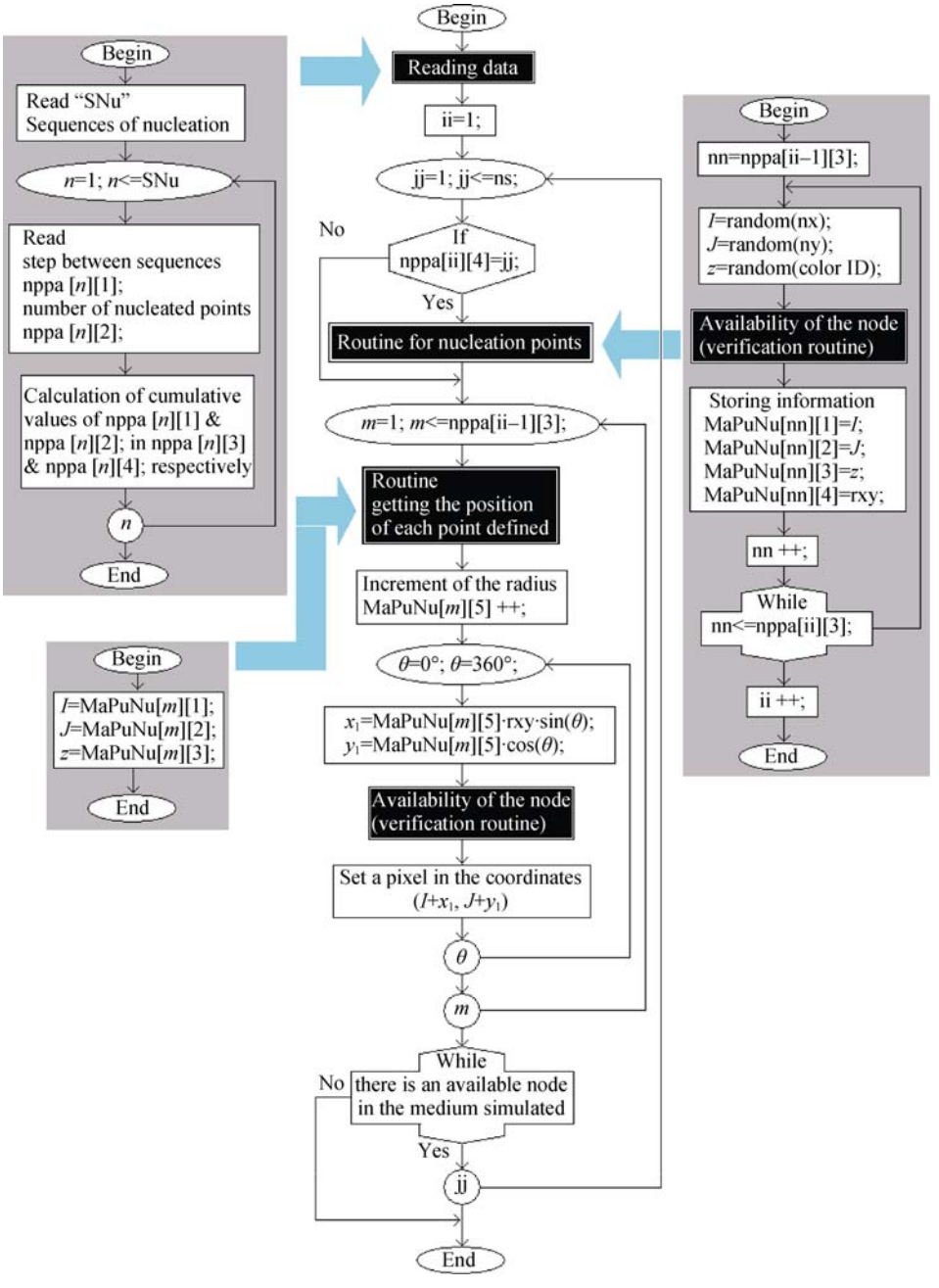

Fig. 3. Flowchart of the computational algorithm to simulate nucleation and grain growth procedures.

The computational source code was written in the programming language C++. Nevertheless, the mathematical and logical procedures can be programmed using any other programming language. This algorithm is applied after the random temperature distribution has been executed.

Instead, the new algorithm was developed to include dif-

ferent nucleation sequences. With a defined number of simulation steps for execution, a different number of nodes for nucleation and with different growth ratios (r_x/r_y) are used in order to obtain grain morphologies with straight and hyperbolic interfaces. This information is stored in a 2D computational array to be used to command the execution of the nucleation and growth routines.

This procedure was also programmed to develop a more efficient algorithm for the simulation. In this algorithm, the nucleated nodes and their characteristics (position on the array and the latest growth radius) are stored in a new 2D computational array to avoid the need for searching them in the sample. Here, the growth radius is stored and updated after each step during simulation.

Initially, a routine cleans all the values previously stored in the memory before the main function is executed in order to avoid returns of false values, which can generate errors or conflict with the actual simulation (all the variables are returned to its initialized values).

The initial assumptions for the simulations of the nucleation and growth procedures are the following. (1) The simulation begins at the step zero. (2) There are no nucleated nodes at the simulation beginning. (3) The nucleated node counter is returned to zero after each nucleation sequence. (4) The step counter is increased after the growth algorithm has been executed. (5) The first nucleation sequence is executed at the first step.

The main function begins executing the reading data subroutine shown at the left upper column of Fig. 3. Here, the user defines the number of sequences for nucleation (SNu), which is defined as an integer data type. (SNu) is the upper limit that rules the reading data loop, and n is the variable used to count the nucleation sequences during simulation.

The steps between nucleation sequences and the number of nucleation nodes are defined by the user for each sequence. These values are stored in a 2D computational array called nppa[n][1 and 2] in the locations indicated, respectively. After the user confirmation, the cumulative values of these variables are calculated and stored in the next available locations of the same array (nppa[n][3 and 4]) in an ordered format. This process is repeated until the sentence "if ($n \leq$SNu)" shown remains being true.

The same loop is used in order for the algorithm to become more efficient and to avoid unnecessary instructions. Here, steps between sequences mean the number of iterations required to execute the next sequence for nucleation. The number of nucleation nodes is the number of new nucleated nodes that appear when the nucleation sequence is executed. Both values are also integers. These values are required for commanding the simulation loops. Logical warnings have been included to avoid error during reading data, e.g., many of these variables cannot be less or equal than zero in order to avoid declaring mistakes.

Immediately, the main function makes ii=1. The variable ii is also an integer data type used for counting the number of sequences (ns) for nucleation during simulation. The assumption ii=1 is taken because ii must be initialized with the first sequence. ii can not be initialized as ii=0 because this sentence does not have a logical sense. ii is used to execute the routine for nucleation nodes illustrated in the right column but only if the sentence "nppa[ii][4]=jj" is true. In the same way, ii is only increased after the execution of the nucleation subroutine. The lower and upper limits are 1 and ns, respectively, for the main calculation loop, and the variable used to count the steps here is jj. This loop is repeated until the condition of available nodes is greater or equal than (1). If there is no one available node for growing or nucleating, a break warning is applied and the simulation is finished. This means that the solidification has been completed and each node forms a part of a grain. The result is an image formed with color pixels on the computer screen in which a numerical correspondence is used as the code to store the corresponding values.

The nucleation and the growth processes are nested inside the main loop. The nucleation routine is executed at the beginning, and growth process is only executed for the nucleated nodes. In this subroutine, nn is the variable used for counting the number of nodes nucleated. nn is initialized as nn=0 for the first sequence. Then, nn will be initialized with the number of the cumulative nucleated points until the last nucleation sequence. The random functions are used to generate random coordinates for placing the new nucleated nodes and a random number is also generated for the color identification (ID-color). Then, these values are stored in the 2D array called MaPuNu[nn][1 to 3] using the number positions in the array indicated in the flowchart for recognition. The only restrictions for the coordinates of the nucleated nodes are given in order to guarantee these will be in the range of the nodes used for discretization.

The values stored in the position MaPuNu[nn][4] correspond to the growth ratio $r_{xy}=r_x/r_y$ initially declared. Finally, nn is increased and the nucleation process is repeated while

the sentence nn≤nppa[ii][3] is valid. The ratio $r_{xy}=r_x/r_y$ that involves the growth radius ratio is included in the same process adding the variable r_{xy}, and all the information about the new nucleated node is updated and stored.

A very important parameter is the radius growth ratio $r_{xy}=r_x/x_y$. If this value is to equal to one ($r_{xy}=1$), all the nucleated nodes grow at the same rate. If the ratio $r_x/r_y=1$, the growth directions form a circumference simulating an isotropical growth. When the growth process is nested in a loop, the new incremented radius (r_x++) is updated at each step of the simulation. Here, a grain structure randomly oriented results as illustrated in Fig. 1(a). The variation of the ratio ($r_x/r_y\neq1$) results in columnar grains with different longitudes and preferential growth directions. However, the interfaces between grains remain straight. If the ratio $r_x/r_y<1$, the growth direction forms an ellipse with its largest radius along vertical axis. A representation of this condition is shown in Fig. 1(b). If the ratio $r_x/r_y>1$, the growth direction forms an ellipse with its largest radius along horizontal axis. The morphology of this condition will be similar to that shown in Fig. 1(b) but with an orientation rotated 90°.

The availability of the node is validated using a basic routine to verify if the node can be a new nucleated node. The following four sentences must be true in order to verify if the node is in the sample: for coordinates on the horizontal axis $I\geq$1 and $I\leq$nx and for coordinates on the vertical axis $J\geq$1 and $J\leq$ny. Then, the numeric value of the pixel on the coordinate is taken to be compared with the numeric code in order to know if the node is a part of a node. If this condition is false, a new nucleated grain is established. However, if this condition is true, a new pair of coordinates is generated randomly and the process is repeated until a new un-solidified node has been found. The availability routine is also employed to verify if a node can be a part of a nucleated grain during grain growth. This routine works verifying the numerical status of the cellular automaton taking the values directly from the screen and comparing with those of the nearest neighbors.

The main function now works using a nested loop that reloads and includes the new nucleated nodes during the growth process. *m* is the variable used to identify all the nucleated nodes. The lower and upper limits are 1 and nppa[i−1][3], respectively, to rule the loop. The number of nucleated nodes is increased after a new execution of the nucleation subroutine due to the updating of ii.

The algorithm can assume an isotropic growth or a predefined growth ratio r_{xy} for the nucleated nodes. The first step to simulate the grain growth process is to increment the radius of each nucleated node (MaPuNu[*m*][4]++). The new coordinates of the points for the circumference generated with the new radius are calculated nesting the θ loop within. The coordinates x_1 and y_1 are obtained trigonometrically using sine and cosine functions as shown in the equations of the flowchart. The inclusion of mathematical libraries is required for the calculation of trigonometric functions and θ is declared as a floating point data type. The coordinates represent the position of the pixel as a function of the corresponding radius and angular position using integer data types. The coordinates x_1 and y_1 are also floating point data types. Nevertheless, the final coordinates of the node (x_1+I, y_1+J) are integers in order to be placed in a nodal position, and the floating part is eliminated. The result of the growth process is a new circumference surrounding the original nucleated node.

Finally, a new pixel according to the node analyzed is placed on the screen if the new grain position is valid. The condition for the validation is that the new pixel position has not been assigned previously as a part of any other grain. This condition is checked by executing the neighboring availability routine.

The use of integer data types for the variables that command the loops and count the nucleated nodes and the steps is because all these values are increased and updated one by one, step by step. These do not need to be declared as floating data type. In the same way, the inclusion of completed subroutines for reading data, nucleation nodes, and loading data reduces the source code becoming the algorithm more efficient and making an easy management.

4. Simulated grain structures

Two temperature distributions shown in Figs. 4(a) and 4(b) were assumed as initial states for simulations in the present work. The first of them is a section of material with a heterogeneous distribution. In the second, there are regions with different probabilities for nucleation as a function of the distribution on the horizontal axis. In consequence, the grain size and morphology will be different. Here, the sample is one single parcel. Only two populations of temperature were assumed for the distribution. The lowest temperatures are in the first population. In consequence, the highest probabilities for nucleation also are presented in this population.

The samples simulated are 1200×800 nodes. The boundary conditions to create the initial temperature distribution

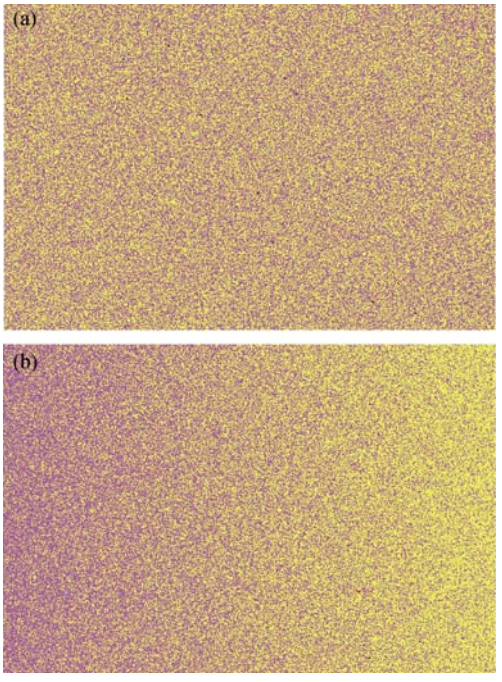

Fig. 4. Original temperature distributions for the simulations: (a) heterogeneous; (b) preferential solidification on the left in comparison with the right due to higher and lower temperature values distributed.

are shown in Table 1; and the number sequences, nucleated nodes, and growth conditions are indicated in Table 2. Also, the results of the initial boundary conditions used for generating the distributions are shown in Figs. 4(a) and 4(b). In Table 2, NNP is the number of nucleated nodes on each nucleation sequence.

Table 1. Boundary conditions to generate the initial temperature distribution

Sample	Population	*I*=1	*I*=nx	*J*=1	*J*=ny
1	1	50	50	50	50
	2	50	50	50	50
2	1	100	0	50	50
	2	0	100	50	50

Figs. 5(a) to 5(c) correspond to the cases (a) to (c) described in Table 2, respectively. Here, a heterogeneous distribution of the grain size and morphology can be appreciated. The nucleated nodes randomly appeared as a function of the original solidification speed distribution. Each grain had an average space to growth according to its appearing moment during the simulation. Nevertheless, the grain size was also influenced by the coarsening of some grains that were defined with the same ID-color during their original nucleation; then, these grains grow until joined and form a bigger complex grain.

Table 2. Solidification speed distributions and nucleation sequences for the samples simulated

Case	Solidification speed probability								Nucleation Sequences SNu	Number of nucleated nodes (NNP) on each sequence					
										1		2		3	
	I=1		*I*=nx		*J*=1		*J*=ny			Steps	NNP	Steps	NNP	Steps	NNP
	A	B	A	B	A	B	A	B							
a	50	50	50	50	50	50	50	50	2	0	200	5	200	0	0
b	50	50	50	50	50	50	50	50	2	0	300	5	100	0	0
c	50	50	50	50	50	50	50	10	2	0	200	10	200	0	0
d	100	0	0	100	50	50	50	20	2	0	300	20	150	0	0
e	100	0	0	100	50	50	50	10	2	0	200	10	200	0	0
f	100	0	0	100	50	50	50	20	3	0	100	20	100	10	100

Fig. 5(c) shows a close-up of the grain structures where the straight and hyperbolic interfaces between the grains can be appreciated.

Figs. 6(a) and 6(b) show grain structures obtained for the simulation of the cases (d) and (e) described in Table 2. Here, populations of small grain size are shown in the left side of the sample, and populations of bigger grain sizes are shown in the right side. The grain sizes and the curvature of the interfaces are different due to the steps between sequences and the number of nucleated nodes. Finally, Fig. 6(c) shows the simulation of the case (f). Here, three different sequences for nucleation were applied during simulation; in consequence, the presence of different grain sizes is absolutely evident because the growing nodes for the latest nucleated nodes are reduced.

Figs. 7(a) and 7(b) show the formation of straight and hyperbolic interfaces. The same three nucleated grains are shown, but the nucleation event occurs on different solidifi-

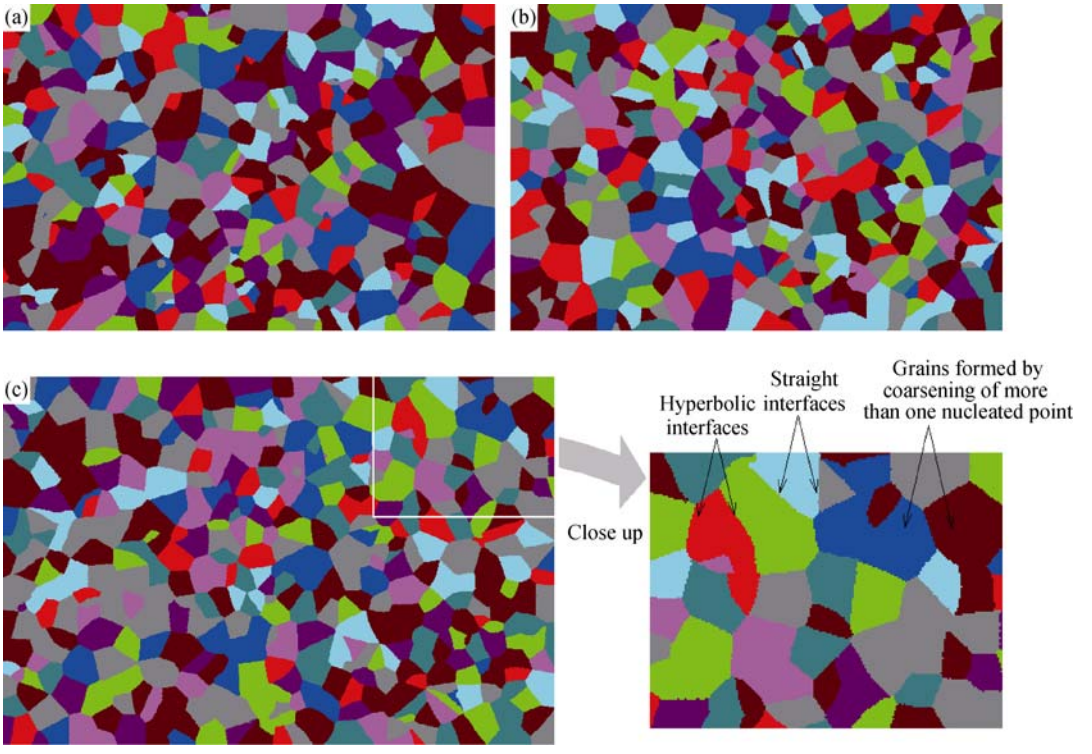

Fig. 5. Simulation of grain structures with hyperbolic interfaces using 2 nucleation sequences according to the nucleation sequences described in cases a to c in Table 2 and considering an heterogeneous original temperature distribution according with sample 1 described in Table 1 (values in Table 1 are used to create an original temperature distribution and information in Table 2 is used to rule the nucleation sequences and the grain growth process).

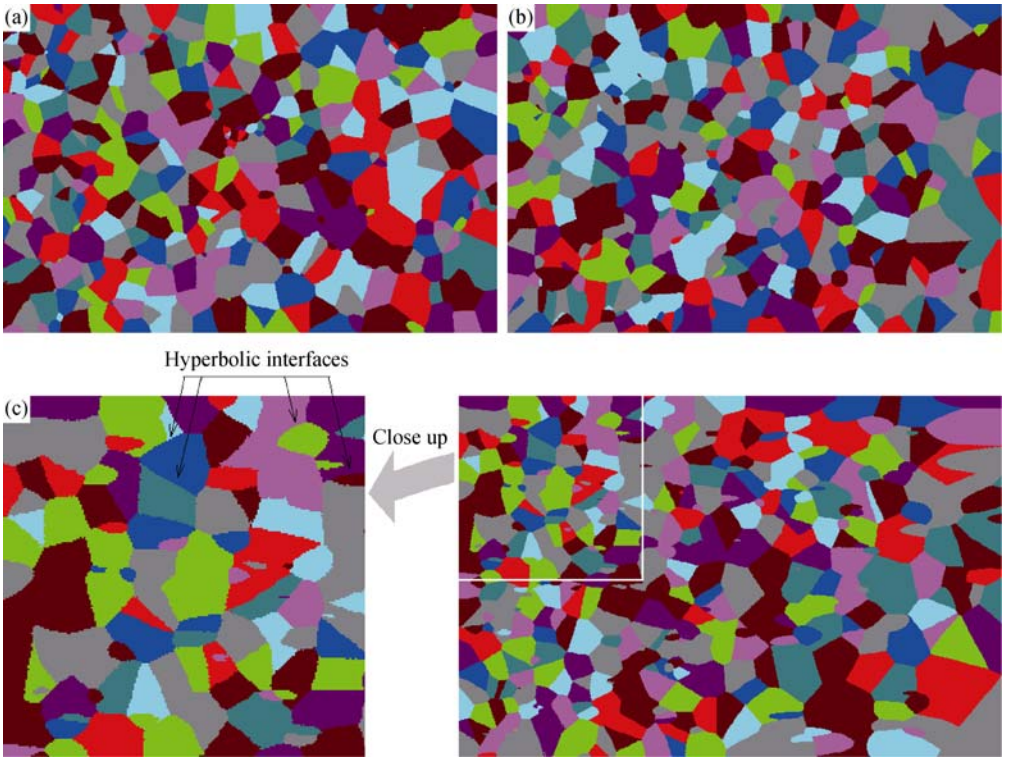

Fig. 6. Simulation of grain structures with different grain sizes at different solidification speeds and nucleation sequences with straight and hyperbolic interfaces: but considering cases d (a), e (b), and f (c) described in Table 2.

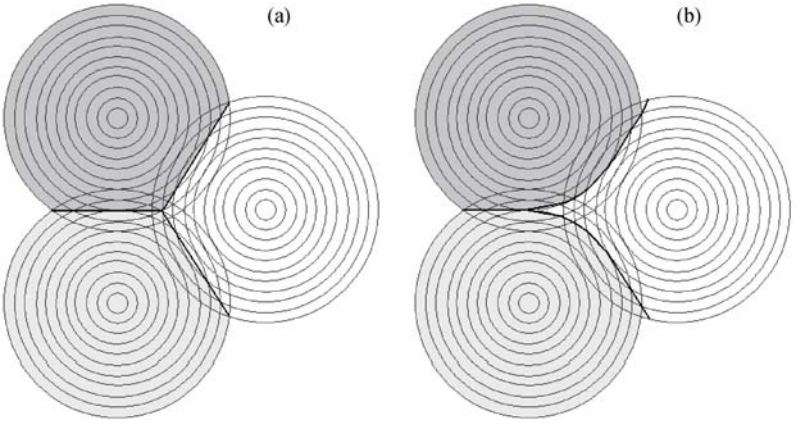

Fig. 7. Formation of grain interfaces between grains: (a) straight interfaces; (b) hyperbolic interfaces.

cation times. If the three grains are nucleated at the same sequence, the three interfaces between these will be straight due to the points where the growing radii are intersected form constant angles with the vertex. However, if the nodes are nucleated in different sequences, this angle will not be constant; in consequence, the interface will be hyperbolic. Finally, all the grain structures simulated in the present work are stored in a 2D array to be characterized computationally. The values of each node are ordered and stored using the numerical code to identify the grain interfaces as shown in Fig. 8.

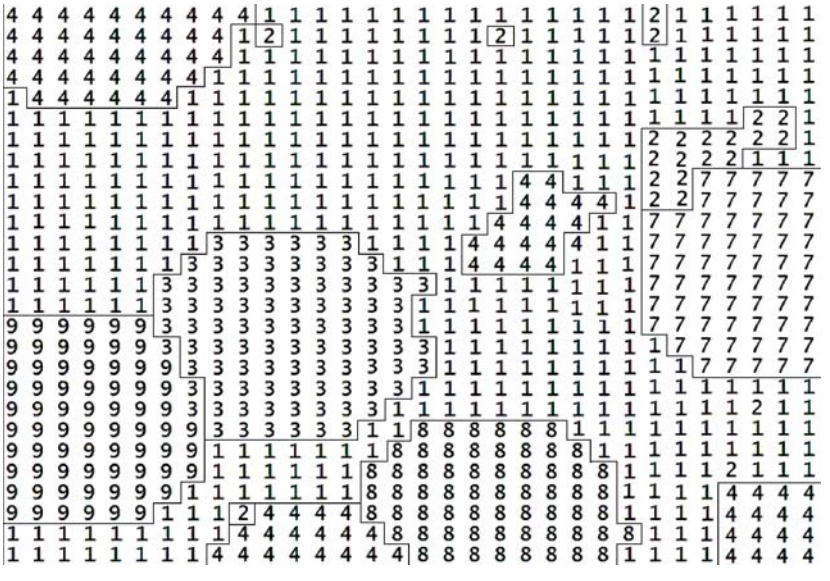

Fig. 8. Grain interfaces identified using the numerical code on the cellular automata.

The model developed in this work is a quasi-dynamic model because nucleation and growth processes are simulated and because the final grain structure is formed during evolution of an original distribution. However, no physical factors are included in the algorithms for simulation. Nevertheless, this kind of models is better than original geometrical models.

Geometrical models were the first approaches developed by the beginners, who relate the chaos theory with applications in materials science. Here, a set of nodes was randomly placed in a sample and these were assumed as vertexes and then were joined with each other according to proximity. In other algorithms, some nodes near the boundaries were taken as initial points and then a random walk algorithm was employed to generate new nodes for joining. Geometrical models are basic algorithms. These are easily programmed and the files generated for saving the information do not require much space on memory or disk.

The information required on geometrical models is the following: the coordinates of each vertex, the ID number for each vertex, the coordination number, and the ID number of the neighbors.

Derivations of these algorithms are used in computer software simulators to generate non-structured meshes in

order to discretize complex solids or surfaces. Although these models can provide a remote approach for a grain structure, these cannot be classified as dynamic models, these cannot be nested or included in sub-routines to show an evolution of an original system, and the interfaces on grain structures simulated are always straight.

Dynamic models give a better approach because it is possible to obtain complex grain interfaces; thus, anisotropy and complex geometry of grain structures will be fidelity represented. In this way, cellular automaton models require larger or huge file sizes because it is needed to save the final status of the system. Here, each nodal value must be saved and the size will be a function of the number of nodes used.

Dynamic models can be appreciated in Figs. 9(a) to 9(c). Fig. 9(a) shows the original temperature distribution at the beginning of the simulation (t=0). Fig. 9(b) shows the execution of the second nucleation sequence. Here, nodes nucleated during the first sequence are with a consolidated advanced front, whereas nodes nucleated in the second sequence are just placed on the sample. In Fig. 9(c), there is a greater population of nucleated nodes in the left side of the sample due to the initial temperature and solidification speed distributions. Moreover, the growth for all the nodes is assumed as isotropic with a constant solidification front. Here, it is also possible to observe the formation of some interfaces due to the interruption on the grain growth (some advanced fronts are blocked).

5. Conclusions and comments

The straight and hyperbolic interfaces between the grains shown in Figs. 7(a) and 7b are formed during grain growth. A grain interface is defined when the growing process between two grains is interrupted because the next nodal position calculated has been occupied with pixel identified with a numerical code as a part of any other grain. The geometrical form of the interface is taken according to the following conditions.

(1) If two grains nucleated at the same sequence are joined, the interface formed between them will be straight.

(2) If two grains nucleated near each other are joined and if these have the same "numeric code", under this unique condition, no interface forms. These grains are joined to form a bigger coarsened single grain. This new grain will frequently be bigger than the average grain size. Coarsening rarely happens during a real solidification. This condition is only during solidification if the fronts of two or more crystals growing independently with the same growth direction are joined.

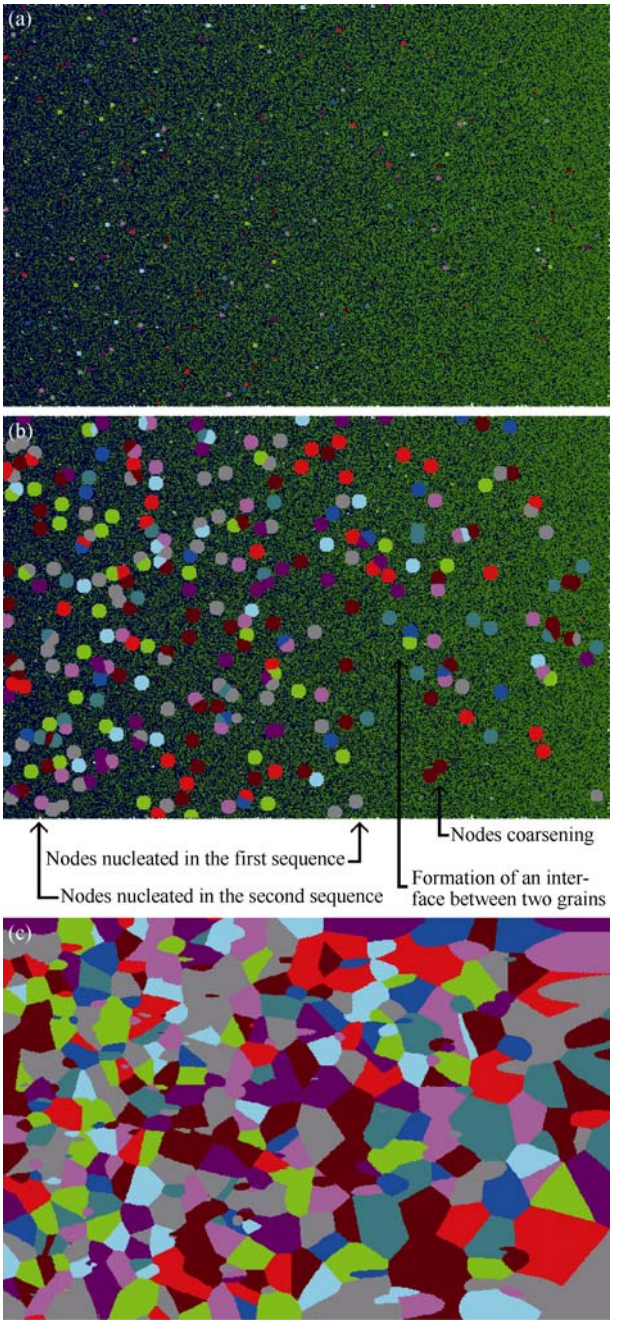

Fig. 9. Grains structure formation: (a) initial temperature distribution; (b) executing the second nucleation sequence; (c) final grain structure.

(3) If two grains are nucleated at different nucleation sequences and their growing fronts are crashed, the interface formed will be hyperbolic.

(4) Bigger coarsened grains can be found in the grain structure if conditions (2) and (3) are present. According to

this, more than two grains nucleated separately can be coarsened to form a bigger one. In the same way, grains nucleated during different sequences can also be coarsened, generating grains with straight and hyperbolic interfaces.

(5) If the advanced fronts of three different grains are joined in a node, a vertex is formed.

The algorithm developed in the present work can give a good approach for grain structure formation in materials with straight and hyperbolic interfaces. It was programmed with an efficient algorithm to allow reducing the computing time and to avoid unnecessary codes.

Finally, the algorithms developed in this work can be used as sub-routines and can also be nested in additional loops if the user likes to build a more sophisticated model for 3D grain structure.

Acknowledgements

The authors wish to thank Consejo Nacional de Ciencia y Tecnología (CONACyT), Universidad Autónoma Metropolitana (UAM-AZC), Instituto Tecnológico Autónomo de México (ITAM), and Instituto Mexicano del Petróleo (IMP).

References

[1] O.M. Ivasishin, S.V. Shevchenko, and N.L. Vasiliev, 3D Monte-Carlo simulation of texture-controlled grain growth, *Acta Mater.*, 51(2003), p.1019.

[2] M.C. Flemings, *Solidification Processing*, McGraw Hill Book Company, New York, 1974.

[3] E.A. Holm, M.A. Miodownik, and A.D. Rollett, On abnormal subgrain growth and the origin of recrystallization nuclei, *Acta Mater.*, 51(2003), p.2701.

[4] S. Mishra and T. DebRoy, Measurements and Monte Carlo simulation of grain growth in the heat-affected zone of Ti-6Al-4V welds, *Acta Mater.*, 52(2004), p.1183.

[5] Y.J. Lan, D.Z. li, and Y.Y. Li, Modeling austenite decomposition into ferrite at different cooling rate in low-carbon steel with cellular automaton method, *Acta Mater.,* 52(2004), p.1721.

[6] H. Yoshioka, Y. Tada, and Y. Hayashi, Crystal growth and its morphology in the mushy zone, *Acta Mater.*, 52(2004), p.1515.

[7] M.M. Tong, D.Z. Li, and Y.Y. Li, Modeling the austenite-ferrite diffusive transformation during continuous cooling on a mesoscale using Monte Carlo method, *Acta Mater.*, 52(2004), p.1155.

[8] L. Zhang, C.B. Zhang, Y.M. Wang, S.Q. Wang, and H.Q. Ye, A cellular automaton investigation of the transformation from austenite to ferrite during continuous cooling, *Acta Mater.,* 51(2003), p.5519.

[9] R. McAfee and I. Nettleship, The simulation and selection of shapes for the unfolding of grain size distributions, *Acta Mater.,* 51(2003), p.4603.

[10] W. Wang, P.D. Lee, and M. McLean, A model of solidification microstructures in nickel-based superalloys: predicting primary dendrite spacing selection, *Acta Mater.*, 51(2003), p.2971.

[11] C.W. Lan, Y.C. Chang, and C.J. Shih, Adaptive phase field simulation of non-isothermal free dendritic growth of a binary alloy, *Acta Mater.*, 51(2003), p.1857.

[12] W. Feng, Q. Xu, and B. Liu, Microstructure simulation of aluminum alloy using parallel computing technique, *ISIJ Int.*, 42(2002), p.702.

[13] K.Y. Lee and C.P. Hong, Stochastic modeling of solidification grain of structures of Al-Cu crystalline ribbons in planar flow casting, *ISIJ Int.*, 37(1997), p.38.

[14] Y.H. Shin and C.P. Hong, Modeling of dendritic growth with convection using a modified cellular automaton model with a diffuse interface, *ISIJ Int.*, 42(2002), p.359.

[15] A. Ramírez, F. Chávez, L. Demedices, A. Cruz, and M. Macias, Randomly grain growth in metallic materials, *Chaos Solitons Fractals*, 42 (2009), p.820.