Simulation of Precipitation in an Aluminum Scandium Alloy using Kinetic Monte Carlo and Density-based Clustering with Noise Algorithms

Alfredo de Moura
IPC – Institute of Polymers and Composites
University of Minho
Guimarães, Portugal
alfredo.moura@gmail.com

Antonio Esteves
Computer Science and Technology Center
Informatics Department
University of Minho
Braga, Portugal
esteves@di.uminho.pt

***Abstract*: The present paper reports the precipitation process of Al_3Sc structures in an aluminum scandium alloy, which has been simulated with a kinetic Monte Carlo (kMC) method. The kMC implementation is based on the vacancy diffusion mechanism. To filter the raw data generated by the kMC simulation, the density-based clustering with noise (DBSCAN) method was employed. kMC and DBSCAN algorithms were implemented in the C language. The undertaken simulations were conducted in the SeARCH cluster at the University of Minho. The study covers temperatures, concentrations, and dimensions, ranging from 578K to 873K, 0.25% to 5%, and 50x50x50 to 100x100x100. The Al_3Sc precipitation was successfully simulated at the atomistic scale. DBSCAN revealed to be a valorous aid to identify the precipitates. The achieved results are in good agreement with those reported in the literature, but we went deeper in the evaluation of the influence of all the simulation and analysis parameters. A parallel version of the kMC algorithm using OpenMP was evaluated, which has not proved advantageous compared to the optimized sequential implementation.**

Keywords - Al_3Sc precipitation; kinetic Monte Carlo; cluster analysis; DBSCAN; OpenMP.

I. INTRODUCTION

Precipitate structures play a fundamental role in the material science due to the capacity of representing strong obstacles for dislocations movements within the material structure.

This paper focuses on the elaboration and application of mechanical statistics knowledge, namely the kinetic Monte Carlo method [1], on the study and prediction of the phenomenon of precipitation in an aluminum alloy. The alloy under analysis is the aluminum scandium alloy [2]. The work that will be documented inhere tackles subjects such as computational mechanics, mechanical statistics (the kinetic Monte Carlo method), material science, the precipitation phenomenon, the diffusion phenomenon, what influences this phenomenon and how to control it and also predict it, as well as data mining (namely clustering) the vital information.

OpenMP [3] is an API that allows shared memory parallelization on multi-core machines. It is based on compiler directives, library routines and environmental variables. OpenMP uses multithreading and is based on the fork-join model of parallel execution. It is through directives, added by the programmer to the code, that the compiler adds parallelism to an application. Since the most promising parallelization strategy for the kMC algorithm uses shared memory, OpenMP is a natural choice. Only if the OpenMP implementation of the KMC algorithm accelerates the sequential version in a scalable manner, we will try a distributed memory parallelization strategy, such as Message Passing Interface (MPI) [4].

The outcome of the work undertaken is a set of software applications that allows us (i) to perform Monte Carlo (MC) simulations with and without OpenMP, (ii) to analyze the results using the Density Based Spatial Clustering of Applications with Noise (DBSCAN) technique [5], and (iii) to compare the simulation results with the classical nucleation theory. Practical results obtained with these applications are (i) reports about the simulation, the analysis of clusters and precipitates with DBSCAN algorithm, and the application of the classical nucleation theory; (ii) files for 3D visualization of the simulation (at various stages over time); and (iii) files for 3D visualization of the precipitates.

The rest of the paper is organized as it follows. Section II presents the related work. Section III summarizes the theory behind the simulation of precipitation with kinetic Monte Carlo. Section IV describes the implementation of the simulation and cluster analysis. Section V presents the results of the simulation and analysis. Finally, Section VI points out some conclusions and areas for future research.

II. RELATED WORK

As computation extends its capacities increasingly, so has the scientific field of nucleation and precipitation modeling. The process of modeling nucleation and precipitation has been achieved at different scales, each one having its own advantages and disadvantages. It has increased the number of publications and studies related with the subject of modeling the precipitation kinetics at the atomistic level [6]. At the atomistic level, the simulation model includes (i) the individual atoms, which are organized in a lattice, and (ii) the interactions among atoms, represented by the number of atomic bonds and several energies. The main materials subjected to such studies are alloy materials, such as Fe-Cu, Fe-P-C, Fe-Cu-Ni-Si, Al-Cu.

Aluminum alloys have also their share of studies by which we would like to outline and focus on the Al-Sc alloy.

Binkele and Schmauder studied the precipitation in the Fe-Cu binary system via atomistic Monte Carlo simulations [7][8]. The Fe-Cu system has a BCC structure and the percentages of Fe and Cu used were 90% and 10%, respectively. In our work we have simulated the Al-Sc alloy, which has a FCC structure and a lower supersaturation: the percentage of scandium varied in the range of 0.25% to 5%. Under these conditions, the precipitation is more difficult to observe. We believe that the formula we used to calculate the real-time in simulation is more accurate than the one mentioned in [7]. In [7][8] is not presented a comparison between kMC simulation results and Classical Nucleation Theory (CNT) [9][10], as was done in the present work.

Bombac and Kuglar also simulated a Fe-Cu alloy with MC. The simulation was based on a residence time algorithm, used a BCC rigid lattice structure and applied a temperature of 873K. The outputs of their study are only the number of precipitates and their dimension [11]. They did not compare the simulation results with theory, and several parameters that influence simulation results were not evaluated.

The work by Lae et al. documents a study in which cluster dynamics simulation is applied to Al-Sc and Al-Zr alloys. The achieved results are compared with MC simulation results and they found a good agreement between both simulations results [12]. Comparing with our work, kMC is employed in [12] just as a comparison tool and no details are given about the kMC simulations.

Clouet et al. have published studies of atomistic Monte Carlo simulations not just based on a binary Al-Sc alloy but also on ternary systems [13]. The results of the Al-Sc alloy simulations were compared with the classical nucleation theory. The simulation applied a residence time algorithm using an FCC rigid lattice. Our approach was inspired by Clouet et al. [13] but we evaluated the influence of all the parameters involved in simulation: lattice size, temperature, Sc concentration, and the number of MC steps.

Clouet and Soisson have published a summary of recent applications of the atomistic diffusion model and of the kinetic Monte Carlo method [14]. The summary covers homogeneous and heterogeneous precipitation caused by thermal aging as well as phase transformation caused under irradiation. To conclude this publication the authors mention that atomistic kinetic Monte Carlo simulations provide a convenient way to simulate and model precipitation kinetics in alloys.

Monte Carlo simulations have also been used on the study of other phenomena. Grain growth, abnormal grain growth, thin film deposition and growth, sintering for nuclear fuel aging, bubble formation in nuclear fuels are just some of those phenomena [15].

The three main contributions of the present work to the reviewed literature are (i) the exhaustive evaluation of all the parameters involved in kMC simulation, (ii) the application of a robust and automatic clustering technique, and (iii) the attempt of accelerating the simulation through the parallelization of kMC with OpenMP.

III. THEORETICAL BACKGROUND FOR KMC SIMULATION

This section summarizes the theory, as a set of equations, behind the simulation of Al_3Sc precipitation with kinetic Monte Carlo.

Transition rate for an aluminum atom is calculated by (1) [8].

$$\Gamma_{AlV} = v_{Al} \exp\left(-\frac{\Delta E_{AlV}}{kT}\right) \tag{1}$$

As for (2), it describes the transition rate for a scandium atom [8].

$$\Gamma_{ScV} = v_{Sc} \exp\left(-\frac{\Delta E_{ScV}}{kT}\right) \tag{2}$$

The aluminum activation energy is obtained by (3) [8].

$$\begin{aligned}\Delta E_{AlV} = E_{spAl} - n^{(1)}_{AlAl}\varepsilon^{(1)}_{AlAl} - n^{(1)}_{AlSc}\varepsilon^{(1)}_{AlSc} - n^{(2)}_{AlAl}\varepsilon^{(2)}_{AlAl} \\ -n^{(2)}_{AlSc}\varepsilon^{(2)}_{AlSc} - n^{(1)}_{AlV}\varepsilon^{(1)}_{AlV} - n^{(1)}_{ScV}\varepsilon^{(1)}_{ScV}\end{aligned} \tag{3}$$

Equations (4), (5), and (6) describe relations among the number of bonds and the size of the first and second neighborhoods, for an FCC structure [8].

$$n^{(1)}_{AlAl} + n^{(1)}_{AlSc} = Z_1 - 1 \tag{4}$$

$$n^{(2)}_{AlAl} + n^{(2)}_{AlSc} = Z_2 \tag{5}$$

$$n^{(1)}_{AlV} + n^{(1)}_{ScV} = Z_1 \tag{6}$$

where $n^{(1)}_{AlAl}$ and $n^{(2)}_{AlAl}$ are the number of aluminum-aluminum bonds regarding the first and second neighborhood, respectively. $n^{(1)}_{AlSc}$ and $n^{(2)}_{AlSc}$ are the number of aluminum- scandium bonds regarding the first and second neighborhood. $n^{(1)}_{AlV}$ is the number of aluminum-vacancy bonds and $n^{(1)}_{ScV}$ is the number of scandium-vacancy bonds, regarding the first neighborhood. Z_1 and Z_2 are the size of the first and second neighborhoods, respectively.

The scandium activation energy is obtained by (7) [8].

$$\begin{aligned}\Delta E_{ScV} = E_{spSc} - n^{(1)}_{AlSc}\varepsilon^{(1)}_{AlSc} - n^{(1)}_{ScSc}\varepsilon^{(1)}_{ScSc} - n^{(2)}_{AlSc}\varepsilon^{(2)}_{AlSc} \\ -n^{(2)}_{ScSc}\varepsilon^{(2)}_{ScSc} - n^{(1)}_{AlV}\varepsilon^{(1)}_{AlV} - n^{(1)}_{ScV}\varepsilon^{(1)}_{ScV}\end{aligned} \tag{7}$$

Analogously, (8), (9), and (10) describe the number of scandium-scandium bonds, number of aluminum-scandium bonds, number of aluminum-vacancy bonds, number of scandium-vacancy bonds, regarding the first and second neighborhood [8].

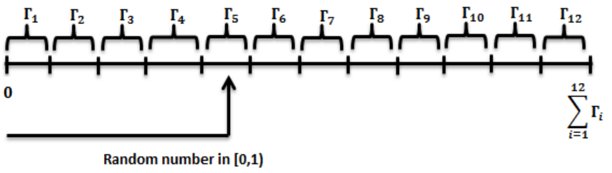
$$n^{(1)}_{ScSc} + n^{(1)}_{AlSc} = Z_1 - 1 \tag{8}$$

$$n_{ScSc}^{(2)} + n_{AlSc}^{(2)} = Z_2 \qquad (9)$$

$$n_{AlV}^{(1)} + n_{ScV}^{(1)} = Z_1 \qquad (10)$$

As a vacancy site is surrounded by twelve nearest neighbors, twelve jump rates are calculated. They are the jump frequency Γ_1, Γ_2, until, Γ_{12}. In the next step of a kMC algorithm, one of these 12 frequencies is selected, based on their values and on a random number: the vacancy will jump to the position of atom *n* that verifies (11) (Figure 1).

Equation (12) describes the computation of the real time of simulation. It is composed by the averaged residence time, multiplied by a factor that takes into account the difference between the simulated vacancy concentration and the real vacancy concentration. Equation (13), which traduces analytically the graphical data vacancy concentration versus temperature obtained in [16], calculates the real vacancy concentration in this kMC algorithm.



Γ_1 Γ_2 Γ_3 Γ_4 Γ_5 Γ_6 Γ_7 Γ_8 Γ_9 Γ_{10} Γ_{11} Γ_{12}
0
$\sum_{i=1}^{12} \Gamma_i$
Random number in [0,1)

Figure 1. Random selection of the jump frequency.

$$\sum_{i=1}^{n} \Gamma_i \leq random\_number \leq \sum_{i=1}^{n+1} \Gamma_i \qquad (11)$$

$$t_{MC}^{real} = \left(\frac{C_V^{sim}}{C_V^{real}} \right) \times \left(\sum_{i=1}^{12} \Gamma_i \right)^{-1} = \left(\frac{C_V^{sim}}{C_V^{real}} \right) \times t_{MC}^{sim} \qquad (12)$$

$$\begin{aligned} C_{Vreal} &= -0.005792301654 + 5.281432466e^{-5} \times T \\ &-1.916781695e^{-7} \times T^2 + 3.466630615e^{-10} \times T^3 \\ &-3.132467044e^{-13} \times T^4 + 1.135950846e^{-16} \times T^5 \end{aligned} \qquad (13)$$

IV. IMPLEMENTATION OF SIMULATION AND ANALYSIS

A. *Simulation with Kinetic Monte Carlo*

The pseudo-code presented in Figure 2 summarizes the implemented kinetic Monte Carlo algorithm in C language. This code enhances the steps that are of upper importance in a kMC simulation: the activation energy calculation, the vacancy exchange frequency calculation, the step time calculation, the swap of positions between the vacancy and the selected first nearest neighbor. Additionally, the code enhances the step of the data input as well as the step of saving the simulated data.

The correspondent C code is portable, in the sense that it can be compiled and run in any system having `gcc` installed: Linux, Windows or other operating system. As so, the submitted simulations were undertaken in the SeARCH cluster. The SeARCH cluster has the advantage that it can be used to accelerate simulations in three ways: (i) running multiple sequential simulations at same time, with different parameters, (ii) running a parallel simulation on the same machine using OpenMP, or (iii) running a parallel simulation on several machines using MPI. The last option was not implemented since the second alternative was implemented and did not succeed on accelerating the sequential version.

```
main:
  Read the configuration file
  Compute the coordinates of all FCC lattice sites
  Compute average step time and rejection step time →
        → avgStepTime, rejectStepTime
  Initialize the simulated time → timeSim=0
  while (mcs < TOTAL_MCS) do
    Calculate the activation energy → Eact
    Calculate the vacancy exchange frequency and the real time of
    this MCS → vEF, ts
    ts = ts*tsCorrection             // corrected simulated time for
                                     // current MCS
    if (ts > rejectStepTime) then    // step time exceeds a threshold
                                     // that is considered a
                                     // computation error
        Increment errorSteps
        ts = avgStepTime             // replace computed step time by
                                     // average step time
    endIf
    timeSim = timeSim + ts           // accumulated simulated time
    Select a 1st nearest neighbor for the new position of vacancy
    Swap the vacancy with the selected neighbor

    if (mcs = snapshots[numSnap]) then  // if it is a snapshot point
        Save simulation data to VTK | PDB | XYZ file(s)
        snapshotTime[numSnap-1] =timeSim // save snapshot time
        Increment numSnap
    endIf
    Increment mcs
  endWhile
  Write a simulation report to file
end main
```

Figure 2. kMC algorithm.

B. *Clustering Analysis with DBSCAN*

The main goal of clustering analysis is dividing data into groups, or clusters, which share certain characteristics. Clustering is used in the present work to identify Al_3Sc precipitates in a 3D matrix, containing the position of all Sc atoms, generated by the kMC simulation. The implemented clustering algorithm is designated by DBSCAN [5]. CLARANS [17], DBCLASD [18], and OPTICS [19] are other clustering algorithms, adequate for dealing with large spatial datasets.

As a member of the density-based clustering approaches, DBSCAN identifies regions of high density agglomerations in an immense low density surrounding. Its major advantages are (i) it identifies objects with arbitrary shape and (ii) it does not require that the number of clusters to be identified is provided as input, like *k-means* method does. DBSCAN introduces the notion of noise, used to label atoms that are in low dense regions, which revealed to be an adequate feature in our case. In DBSCAN, for each cluster identified, a point of that cluster is a core point if it has in its neighborhood (with a predefined radius *eps*) a predefined minimum number of points (*minPts*). DBSCAN classifies points as being: (i) core point - a point in the interior of the density based cluster, (ii) border point - a point that belongs to the border of the density based cluster, and (iii) noise point - a point that is neither a core point nor a border point.

```
DBSCAN (atoms[], nAtoms, eps, minPts)
  cid = 0          // current cluster ID
  pid = 0          // atom position on the array of atoms

  while (pid < nAtoms) do   // cycle over all atoms
      if (atom 'pid' was not  yet visited) then
          Mark atom 'pid' as visited
          Get the size of neighborhood of atom 'pid' → sizeN
          if (sizeN<minPts) then
              Classify atom 'pid' as NOISE
          else
              resBool = ExpandCluster (atoms, nAtoms, visited, N,
                                       pid, cid, eps, minPts)
              if (resBool = TRUE) then
                  Increment cid
              endIf
          endIf
      endIf
      Increment pid
  endWhile
end DBSCAN
```

Figure 3. Main function of the DBSCAN algorithm.

```
ExpandCluster (atoms[], nAtoms, visited[], N[], pid, cid, eps,
               minPts)
  Get the size of neighborhood of atom 'pid' → sizeN
  Count unclustered  neighbors of atom 'pid' → sizeUnclustered
  if (sizeUnclustered < minPts) then
     Mark atom 'pid' as NOISE
     return FALSE
  else
     Add atom 'pid' to cluster 'cid'
     for (i in [0:sizeN[) do
        nid = neighbor i-th of atom 'pid'
        if (atom 'nid' was not yet visited) then
           Mark atom 'nid' as visited
           Get size of neighborhood of atom 'nid' → sizeNN
           if (sizeNN  >= minPts) then
              for (j in [0:sizeNN[) do
                 nnid = neighbor j-th of atom 'nid'
                 Add atom 'nnid' to neighborhood of atom 'pid'
                 Increment sizeN
              endFor
           endIf
        endIf
        if (atom 'nid' is not yet member of any cluster) then
           Add atom 'nid' to cluster 'cid'
        endIf
     endFor
  endIf
  return TRUE
end ExpandCluster
```

Figure 4. *ExpandCluster* function used by the DBSCAN algorithm.

The pseudo-code included in Figures 3 and 4 presents the main functionalities of DBSCAN, which was implemented in C language. The code follows the main sequence of steps defined by the authors of the algorithm [5].

To save the atoms belonging to each group was used a data structure that varies dynamically, because the clusters are of variable and unknown size. The used data structure was inspired by the Java *ArrayList* class. After applying DBSCAN, the clusters that are split in several parts are merged in a single spatial region per cluster. This is required because we use periodic boundary conditions (PBC) and aims to improve the 3D visualization of clusters [2].

To permit the visualization of the lattice configurations generated by the kinetic Monte Carlo simulations and by the clustering analysis, these configurations are saved to files in a format that can be read and rendered by available visualization tools. The developed code allows us to save data in one of the following formats: **pdb**, **xyz,** and **vtk**. All these data formats can be visualized with the **ParaView** tool, which is an open-source application adequate to the visualization and analysis of multidimensional data.

Beyond the visualization files with precipitates, the analysis carried out by DBSCAN produces other results, such as, the size and radius of the precipitates, the average size and radius among all precipitates, the percentage of Sc atoms in precipitates and in Al solid solution, and the number of small clusters with the same size.

The main inputs necessary to undergo a simulation and posterior cluster analysis, which are supplied in a configuration file, are: the aluminum lattice constant (Angstrom), the number of unit cells in the x/y/z direction, scandium percentage, simulation Monte Carlo steps, simulation temperature (Kelvin), material parameters, the radius used to define the neighborhood of each atom (***eps*** in DBSCAN algorithm), and minimum number of neighbors that makes an atom to be a core atom of a cluster (***minPts*** in DBSCAN).

The material parameters that supported the previous equations and therefore, the simulations are, first and second nearest-neighbor pair effective energies, saddle point energies and attempt frequencies [13]: $\varepsilon^{(1)}_{AlAl}$ = -0.56 eV; $\varepsilon^{(1)}_{ScSc}$ = -0.65 eV; $\varepsilon^{(1)}_{AlSc}$ =-0.759+21.0x10^{-6}T eV; $\varepsilon^{(1)}_{VV}$ = -0.084 eV; $\varepsilon^{(2)}_{AlSc}$ = 0.113 -33.4x 10^{-6}T eV; $\varepsilon^{(1)}_{AlV}$ =-0.222 eV; $\varepsilon^{(1)}_{ScV}$ = -0.757 eV; e^{sp}_{Al} =-8.219 eV; e^{sp}_{Sc} = -9.434 eV; ν_{Al} = 1.36x10^{14} Hz; ν_{Sc} = 4x10^{15} Hz.

C. Implementation of kMC with OpenMP

Figure 5 presents the algorithm of the main function used to implement the kinetic Monte Carlo simulation with multiple threads of execution, through the OpenMP library. The lines starting with `#pragma omp` specify OpenMP directives, for example to create the parallel threads or to synchronize threads. After the initial steps, which are the same as in the sequential code, it is specified the number of threads to create. The core of the algorithm is a loop that iterates over the number of MC steps. Within this cycle we create parallel threads, each with a private copy of the specified variables. With the aid of the thread ID (*idT*) and the number of threads (*nT*), each thread can execute only a subset of the calculations.

```
main_OMP
[…] // Initial steps are the same as in non OMP code
// Specify the number of threads to be created
 omp_set_num_threads(numThreads)
Initialize the MC step (mcs) to zero

while (mcs<numberMCStoSimulate) do
   if (idT = 0) then // This section is run by thread with id=0 only
      Count the number of vacancy's first neighbors of Al and Sc type
   endIf

// Create multiple threads
#pragma omp parallel private
   (idT, i, j, nPos, nType, nnPos, nnType, n_AlAl_1, n_AlSc_1,
   n_ScSc_1, n_AlV_1a, n_ScV_1a, n_AlAl_2, n_AlSc_2,
   n_ScSc_2, expoent)
{
   idT = omp_get_thread_num() // ID of each thread
   nT = omp_get_num_threads() // Number of threads
   i = idT
   while (i < NUMBER_1ST_NEIGHBORS) do
      Compute Eact[i] associated with i-th vacancy neighbor
      i = i + nT
   endWhile
   Compute absolute vacancy exchange freq. with its 12  1-st neighbors
   #pragma omp barrier
   if (idT = 0) then
      Compute the sum of all 1-st neighbors absolute exchange freq.
   endIf
   #pragma omp barrier
   Compute relative vacancy exchange freq. with its 1-st neighbors
} // (end of) multiple threads

   if (idT = 0) then
      Sum of all relative vacancy exchange freq. with 1-st neighbors
      totalT = totalT + 1/sumAbsoluteVef
      Select randomly a 1-st nearest neighbor for new vacancy
      Swap the vacancy with the selected neighbor
      if (mcs = snapshots[numSnap]) then
         Save simulation data to file at snapshot
         Increment the number of the current snapshot
      endIf
      Increment the MC step (mcs)
   endIf
endWhile // (end of) cycle relative to the number of MCS
[…] // Final steps are the same as in non OMP code
end main_OMP
```

Figure 5. kMC algorithm with OpenMP.

For example, each thread calculates a subset of the activation energies (*Eact[]*) associated with the 12 neighbors of a vacancy. If it is necessary that all threads reach a certain position in the code, at the same time, it is inserted a synchronization barrier. The condition "*idT=0*" is used to force the calculations to be carried out only by thread 0.

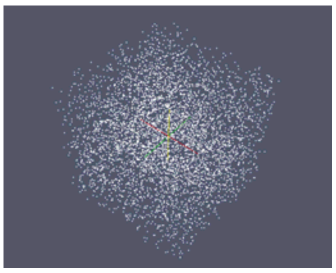
V. RESULTS

Figure 6 illustrates the time evolution of the precipitation phenomenon. The initial random configuration applied to the simulation is shown in Figure 6 (a). The sequence of figures report a simulation that undertook the conditions of 873K, 1%Sc, and over $5x10^{11}$ MCS in a 50x50x50 lattice box ($5x10^5$ atoms). The Sc atoms in raw configurations produced by the simulation are presented in the left part of each figure. The right configuration of each figure demonstrates the application of the DBSCAN algorithm, where the scandium atoms that do not belong to precipitate structures are labeled NOISE and do not appear.

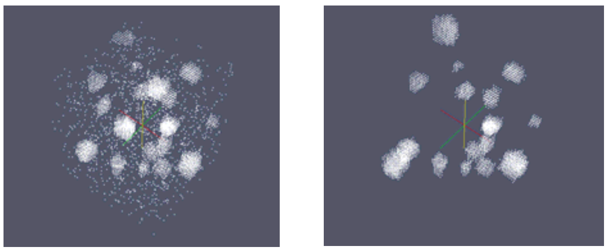
The sequence of graphics from Figure 7 summarizes the analysis undertaken over the simulation outputs. Figure 7 (a) represents the evolution of precipitates dimension in terms of radius measure. Figure 7 (b) acknowledges the evolution of the presence of scandium atoms distributed in the aluminum solid solution. As with Figure 7 (c), it is possible to acknowledge the evolution of the presentage of scandium atoms in precipitate structures. Figure 7 (d) is one of the most important interpretations that is conducted regarding simulation of the nucleation of precipitates as it allows one to undertake comparison analyses with the CNT [9][10]. Two measures are used to efectively compare kMC with CNT: the steady-state nucleation rate (J^{st}), which represents the number of supercritical nuclei formed per unit time in a unit volume and the cluster size distribution (C_{nSc}), which defines the probability to encounter a cluster with a dimension of *n* atoms in a solid solution [2].

The simulations were run on the SeARCH cluster, located at the University of Minho. Table I contains the technical specifications of the SeARCH cluster nodes where we run the kinetic Monte Carlo simulations.

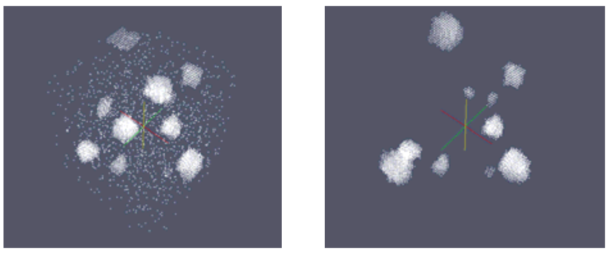
The computation time mainly depends on the number of MC steps. Simulations duration is also influenced by the technical specifications of the machines where the simulations were run. On a `compute-311-X` node of the SeARCH cluster, a simulation with $5x10^{11}$ took around 8 days, and 12 days on a less performing `compute-201-X` node. Computation time does not depend significantly on the scandium percentage, the lattice size or any other parameter of the simulations.



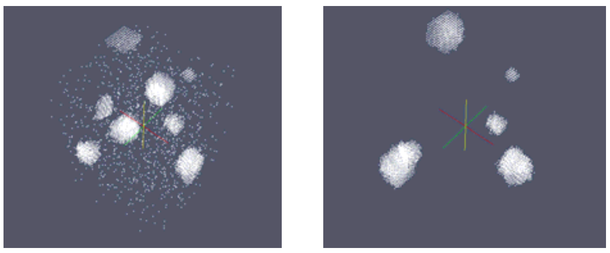
(a)



(b)



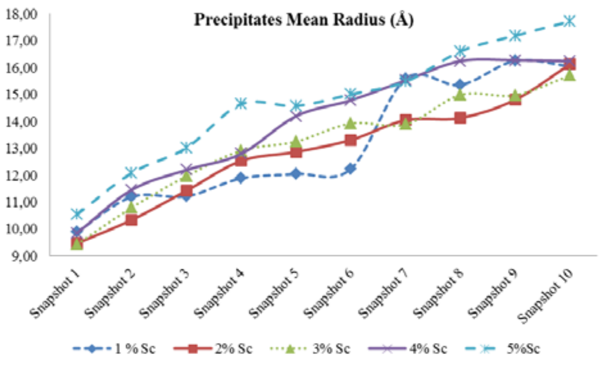
(c)



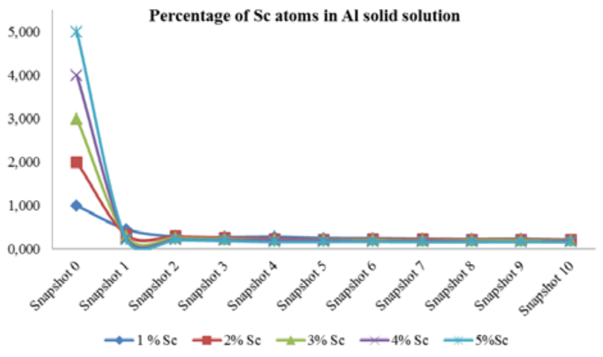
(d)

Figure 6. Evolution of simulation: (a) initial configuration; (b) t=1.55ms; (c) t=3.03ms; (d) t=4.945ms (left/right ⇔ before/after applying DBSCAN).

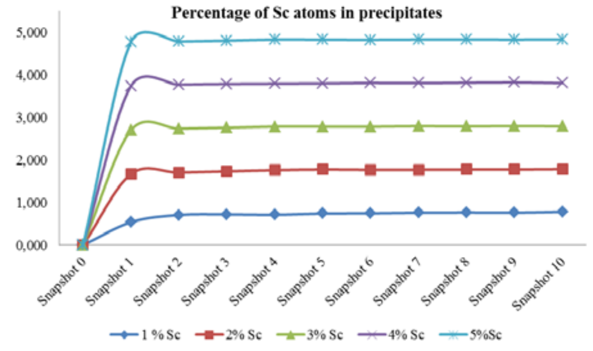
Table II summarizes the computation time needed by a kMC simulation with different number of threads. The number of MC steps simulated was 10^7, the lattice included 10*10*10*4 atoms and we used C code with OpenMP. As we can see from Table II, the utilization of an increasing number of threads is counterproductive. The poor performance achieved by the presented parallel implementation results from 3 facts: (i) the problem we are dealing with is not inherently parallel, since the MC simulation has only one vacancy, (ii) the work assigned to each thread is small and does not compensate the computation overhead introduced by the threads, and (iii) there are several parts of the code that have to be executed by one thread only.



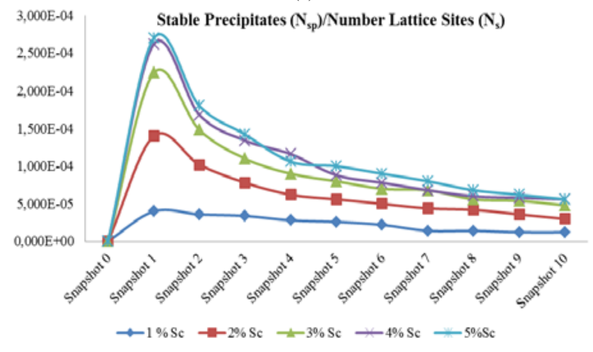
(a)



(b)



(c)



(d)

Figure 7. Simulation metrics: (a) precipitates mean radius; (b) percentage of Sc in Al solid solution; (c) percentage of Sc in precipitates; (d) number of precipitates normalized by the number of lattice sites.

TABLE I. TECHNICAL SPECIFICATIONS OF THE SEARCH NODES USED BY THE KMC SIMULATIONS.

Nodes	*Processors*	*CPUs Number*	L2 Cache	*Operating System*
311–X nodes	Intel Xeon E5420	8	12 MB	Linux x86_64
201–X nodes	Intel Xeon 5130	4	4 MB	Linux x86_64
101–X nodes	Intel Xeon	4	2 MB	Linux x86_64

TABLE II. COMPUTATION TIME, NEEDED BY A MC SIMULATION, AS A FUNCTION OF THE NUMBER OF THREADS.

Number of threads	Average computation time (s)
1	25
2	46
4	52
8	62
12	70

VI. CONCLUSIONS AND FUTURE WORK

kMC simulation of Al_3Sc precipitation on a supersaturated Al solid solution was successfully achieved. This proves that the equations used to model Al_3Sc precipitation are correct. The results from kMC simulations were further improved by the application of DBSCAN, which proved to be a valorous aid to identify the Al_3Sc precipitates, by eliminating the unclustered Sc atoms. The DBSCAN algorithm reveals adequate in the role of identifying, visualizing and measuring (size, radius, and shape) of the precipitates embedded in the Monte Carlo output data. By simulating with various Sc percentages, as well as temperatures, the capacity of clustering Al_3Sc precipitates maintains accurate.

The number of stable precipitates strongly increases in the initial phase. After that, the number of precipitates reduces, as predicted by the theory of nucleation. Consequently the surviving precipitates increase in size, either in number of atoms or in radius. The mean precipitates radius increases almost linearly over time. The number of precipitates normalized by the number of lattice sites increases rapidly in the initial phase of the simulation and then decreases slightly during the rest of the simulation. Temperature has a profound influence on the evolution of the precipitation simulation. As the CNT states, and the simulation graphics do prove, the steady state nucleation rate rises with the temperature increase.

The achieved results are very much in good agreement with those reported by Clouet et al. [13]: the increase of the precipitates average size and the reduction of the Sc concentration in the Al solid solution during the simulation follow the same tendency. The comparison between kMC and CNT are very much similar [13]. Although we have used the same model for Al_3Sc precipitation as [13], it was possible to go deeper in the evaluation of the influence of all the parameters involved in simulation: lattice size, temperature, Sc concentration, number of MC steps, and the technique used in cluster identification and measuring. We also tried strategies to accelerate the simulation, using OpenMP.

Some features of ParaView made it an interesting choice for visualization and even analysis such as its support to the three formats (vtk, pdb, xyz) we used as output of kMC, it is open source and based on a popular framework (VTK) [20], and it supports parallelism as to handle huge files.

A field for future research is the exploration of parallelization techniques for the kMC simulation. Due to the sequential nature of the precipitation problem, a hypothesis is to use multiple vacancies and run multiple simulations in parallel, each one with a vacancy and a sub-lattice. Simulating with multiple vacancies alters the vacancy concentration to a unrealistic value. Thus, the validity of this alternative, used to speed up the simulations, has to be demonstrated. Examples of algorithms that follow this strategy are the optimistic synchronous relaxation (OSR) and the semi-rigorous synchronous sub-lattice (SL) [21]. These approaches have to deal with two critical issues: correct the excessive vacancy concentration and synchronize the parallel instances of the asynchronous kMC simulation. Another future research topic would be extending MC method to simulate ternary alloys, such as Al-Mg-Sc, Al-Sc-Si or Al-Sc-Zr.

ACKNOWLEDGMENTS

This work was funded by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project PEst-OE/EEI/UI0752/2011.

REFERENCES

[1] A. Voter, "Introduction to the Kinetic Monte Carlo Method", Radiation Effects in Solids, Springer, pp. 1-23, 2007.

[2] A. de Moura, “Simulation of the Nucleation of the Precipitate Al3Sc in an Aluminum Scandium Alloy using the Kinetic Monte Carlo Method”, MSc thesis, University of Minho, Dec 2012.

[3] B. Chapman, G. Jost, and R. Pas, "Using OpenMP: Portable Shared Memory Parallel Programming", The MIT Press, 2007.

[4] W. Gropp, E. Lusk, and A. Skjellum, "Using MPI: Portable Parallel Programming with the Message Passing Interface", 2nd edition, The MIT Press, 1999.

[5] M. Ester, H.-P. Kriegel, J. Sanders, and X. Xu, “Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”, Published in Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), 1996.

[6] J. Röyset, “Scandium in aluminum alloys overview: physical metallurgy, properties and applications”, Metallurgical Science and Technology, Hydro Aluminium R&D Sunndal, N-6600 Sunndalsöra, Norway.

[7] P. Binkele and S. Schmauder, “An atomistic Monte Carlo simulation of precipitation in a binary system”, International Journal for Materials Research, 94, pp. 1-6, 2003.

[8] S. Schmauder and P. Binkele, “Atomistic computer simulation of the formulation of Cu-precipitates in steels”, Computational Materials Science 24 (2002), 2002, pp. 42-53.

[9] D.Kashchiev, "Nucleation: Basic Theory with Applications", Butterworth-Heinemann, Oxford, 2000.

[10] E. Clouet and M. Nastar, "Classical nucleation theory in ordering alloys precipitating with L12 structure", Physical Review B 75, 132102, pp. 1-4, 2007.

[11] D. Bombac and Goran Kugler, “Precipitation in Alloys: A kinetic Monte Carlo and Class Model Study”, World Journal of Engineering, 2010.

[12] L. Lae, P. Guyot, and C. Sigli, "Cluster dynamics in AlZr and AlSc alloys", Materials Forum Volume 28, Institute of Materials Engineering Australasia Ltd, pp. 281-286, 2004.

[13] E. Clouet, M. Naster, and C. Sigli, “Nucleation of Al_3Zr and Al_3Sc in aluminum alloys: From kinetic Monte Carlo simulations to classical theory”, Physical Review B 69 (6), 064109, pp. 1-14, 2004.

[14] E. Clouet and F. Soisson, “Atomic simulations of diffusional phase transformations”, C. R. Physique 11 (2010), 2010, pp. 266-235.

[15] S. Plimpton, C. Battoile, M. Chandross, L. Holm, A. Thompson, V. Tikare, G. Wagner, E. Webb, and X. Zhou, “Crossing the Mesoscale No-Man´s Land via Paralelel Kinetic Monte Carlo”, Sandia Report, SAND2009-6226, 2009.

[16] J. E. Hatch, “Properties and Physical Metallurgy”, American Society for Metals, 1984.

[17] R. Ng and J. Han, “Efficient and Effective Clustering Methods for Spatial Data Mining”, Proc. 20th Int. Conf. on Very Large Data Bases, Santiago, Chile, pp. 144-155, 1994.

[18] X. Xu, M. Ester, H. Kriegel, and J. Sander, "A Distribution-Based Clustering Algorithm for Mining in Large Spatial Databases", Proceedings of the 14th International Conference on Data Engineering, pp. 324-331, 1998.

[19] M. Ankerst, M. Breunig, H. Kriegel, and J. Sander, "OPTICS: Ordering Points To Identify the Clustering Structure", Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 49-60, 1999.

[20] W. Schroeder, K. Martin, and B. Lorensen, "The Visualization Toolkit An Object-Oriented Approach To 3D Graphics", 4th Edition, Kitware Inc. publishers, 2006.

[21] G. Nandipati, Y. Shim, J. Amar, A. Karim, A. Kara, T. Rahman, and O. Trushin, “Parallel kinetic Monte Carlo simulations of Ag(111) island coarsening using a large database”, Journal of Physics: Condensed Matter, 21(8):084214, pp.1-12, 2009.