Computer Physics Communications 211 (2017) 23–30

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Enhanced calculation of eigen-stress field and elastic energy in atomistic interdiffusion of alloys

José M. Cecilia [a], A.M. Hernández-Díaz [a,*,1], Pedro Castrillo [a], J.F. Jiménez-Alonso [b]

[a] Escuela politécnica superior, Universidad Católica de Murcia (UCAM), Spain

[b] Escuela superior de ingeníeros, Universidad de Sevilla, Spain

ARTICLE INFO

Article history:
Received 20 February 2016
Received in revised form
25 May 2016
Accepted 2 July 2016
Available online 11 July 2016

Keywords:
Eigen-stress fields
Material systems
Elastic energy
Atomistic simulation
HPC
GPUs

ABSTRACT

The structural evolution of alloys is affected by the elastic energy associated to eigen-stress fields. However, efficient calculations of the elastic energy in evolving geometries are actually a great challenge in promising atomistic simulation techniques such as Kinetic Monte Carlo (KMC) methods. In this paper, we report two complementary algorithms to calculate the eigen-stress field by linear superposition (a.k.a. LSA, Lineal Superposition Algorithm) and the elastic energy modification in atomistic interdiffusion of alloys (the Atom Exchange Elastic Energy Evaluation (AE^4) Algorithm). LSA is shown to be appropriated for fast incremental stress calculation in highly nanostructured materials, whereas AE^4 provides the required input for KMC and, additionally, it can be used to evaluate the accuracy of the eigen-stress field calculated by LSA. Consequently, they are suitable to be used on-the-fly with KMC. Both algorithms are massively parallel by their definition and thus well-suited for their parallelization on modern Graphics Processing Units (GPUs). Our computational studies confirm that we can obtain significant improvements compared to conventional Finite Element Methods, and the utilization of GPUs opens up new possibilities for the development of these methods in atomistic simulation of materials.

1. Introduction

Eigen-stress fields are generated by the matching conditions of different constituents in material systems [1,2]. In particular, inhomogeneous crystal alloys with composition-dependent lattice parameter are subjected to intrinsic eigen-stress fields, which depend on the spatial composition distribution. This distribution could be rather complex in some cases, such as spinodal decomposed metal alloys [3] or alloy-based nanostructured semiconductors [4].

Predictive simulation of the time-evolution of material nanostructure is critical in certain scenarios such as structural steels in nuclear plants [5] or state-of-the-art integrated circuits during front-end processing [6]. Such evolution is affected by the elastic energy associated to eigen-stress fields [7,8]. However, efficient calculations of the elastic energy in evolving geometries are actually a great challenge for promising simulation techniques such as Kinetic Monte Carlo (KMC) methods [9,10]. In most cases, numerical solutions to this problem have been proposed by using Finite Element Methods (FEM) [11,12]. Nevertheless, the use of both KMC methods and on-the-fly FEM calculations is nowadays computationally prohibitive. Indeed, FEM represents an approach for the solution of a weak form and, hence, any modification of the system configuration implies the whole resolution of the resulting new algebraic system of equations [13]. Consequently, the process would be too time-consuming for estimating the "instantaneous" eigen-stress field in evolving systems.

Eigen-stress fields may be also approximated in the context of the linear elastic theory within the small strain limit [14]. Within this framework, the final configuration coincides with the reference configuration (small displacement hypothesis) and the strain is linear with stress [14]. Therefore, superposition can be applied. In this work, we propose an algorithm based on the superposition of the eigen-stress fields associated to elementary contributions. Within the same context, the elastic energy modifications associated to atom-scale changes can be also estimated from stress field modifications. Our proposal offers several advantages compared to previous approaches [7,11,12,15,16]:

1. It allows incremental calculation of the stress field taking into account only local modifications of the alloy structure. This means higher computational efficiency.

* Corresponding author.
E-mail addresses: jmcecilia@ucam.edu (J.M. Cecilia), amhernandez@ucam.edu (A.M. Hernández-Díaz), pcastrillo@ucam.edu (P. Castrillo), jfjimenez@us.es (J.F. Jiménez-Alonso).

[1] The authors declare that there is no conflict of interests regarding the publication of this article.

http://dx.doi.org/10.1016/j.cpc.2016.07.012
0010-4655/© 2016 Elsevier B.V. All rights reserved.

2. The balance between accuracy and efficiency can be tuned by choosing the truncation radius for superposition.
3. It is massively parallel by its definition and, therefore, an additional performance gain can be obtained by using parallel processors.

These advantages together offer a great opportunity to have a very efficient algorithm to calculate the elastic energy modifications related to atomistic diffusion events, which are an input to the KMC algorithm for every attempt of interdiffusion event (i.e. after every few Monte Carlo steps).

The paper is structured as follows. After this introduction, Section 2 describes the underlying linear superposition model and the method to calculate the energy modifications associated with the eigen-stress. Based on this model, Section 3 shows the corresponding algorithms and their parallelization on modern processors, whereas Section 4 shows the performance and quality evaluation of our algorithm. Finally, we outline the conclusions and some directions for future work.

2. Model

Let us consider an $A_{1-X}B_X$ binary alloy with orthotropic crystal structure, where X is the molar fraction of B atoms, which is a dimensionless measure of alloy composition. Assuming a linear dependence of lattice parameter a with alloy composition (Vegard's law), the lattice parameter can be expressed as:

$$a(X) = a(0) + (a(1) - a(0))X, \tag{1}$$

where a(0) and a(1) are the lattice parameters of the pure constituents of the alloy. The maximum lattice mismatch in the alloy is $\epsilon_{max} = (a(1) - a(0))/a(0.5)$. The spatial composition dependence is accounted dividing the simulation domain into a nanometer-sized uniform cubic mesh. We denote by $\mathbf{n} = (n_x, n_y, n_z)$ the vector index of each mesh element, with n_x, n_y, n_z being integer numbers. Within an atomistic framework, each mesh element has an integer number of alloy atoms, $N_{at} \simeq C_{at}\Omega$, where C_{at} is the atom density of the alloy and Ω is the volume of each mesh element. Likewise, it verifies $N_{at} = N_A + N_B$, with $X = N_B/N_{at}$, where N_A and N_B are, respectively, the integer number of A and B atoms in each mesh element. If moderated lattice mismatch ($|\epsilon_{max}| \ll 1$) and low defect concentration are assumed, N_{at} can be approximated to be constant in all mesh elements. Usual mesh spacing for atomistic simulations are within the nanometer scale and N_{at} is typically ranging from few tens to several hundreds.

In this context, the minimum composition variation in an element will be given by:

$$\delta X = \frac{1}{N_{at}}, \tag{2}$$

and, thus, the elementary eigen-expansion ratio, related to a composition change of δX will be:

$$\delta\epsilon = \frac{\epsilon_{max}}{N_{at}}. \tag{3}$$

Let us consider now the tensor stress field $\delta\bar{\bar{\sigma}}(\mathbf{n})$ generated by an elementary eigen-expansion $\delta\epsilon$ in the reference mesh-element located at (0, 0, 0) over an infinite (and otherwise unstressed and homogeneous) domain. We will refer to $\delta\bar{\bar{\sigma}}(\mathbf{n})$ as elementary stress field. An analytic solution for $\delta\bar{\bar{\sigma}}(\mathbf{n})$ could be found if the expanded element would have spherical geometry and the material would be isotropic. However, no analytic solution is available in our case, where the expanded mesh element is cubic (instead of spherical) and the material is orthotropic (instead of isotropic). Alternatively, a numerical estimation of $\delta\bar{\bar{\sigma}}(\mathbf{n})$ can be obtained using FEM. Notice that the magnitude of the components of the elementary stress tensor $\delta\bar{\bar{\sigma}}(\mathbf{n})$ are inversely proportional to the mesh element volume Ω. Hence, the product $\Omega\delta\bar{\bar{\sigma}}(\mathbf{n})$ is independent on the mesh spacing and it could be estimated "once and forever" for each material.

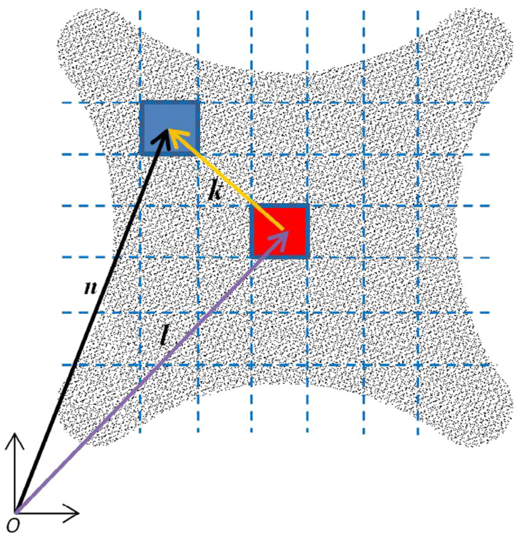


Fig. 1. Schematics of the contribution to the stress $\bar{\bar{\sigma}}$ in a mesh element $\boldsymbol{n}$, induced by an elementary composition variation δX at mesh element $\boldsymbol{l}$. The shadowed region symbolizes the orthotropic stress-field $\delta\bar{\bar{\sigma}}$ related to this δX. The local contribution on element $\boldsymbol{n}$ depends on the relative position vector $\boldsymbol{k} = \boldsymbol{n} - \boldsymbol{l}$.

Consequently, for a non-uniform $X(\mathbf{n})$ alloy composition, the total eigen-stress field $\bar{\bar{\sigma}}(\mathbf{n})$ may be expressed, considering elastic linear superposition, as (see Fig. 1):

$$\bar{\bar{\sigma}}(\boldsymbol{n}) = \sum_{\boldsymbol{l}} \frac{X(\boldsymbol{l}) - \langle X\rangle}{\delta X}\delta\bar{\bar{\sigma}}(\boldsymbol{n} - \boldsymbol{l}). \tag{4}$$

Eq. (4) accounts for the contribution of mesh element $\boldsymbol{l}$ on the stress field at the mesh element $\boldsymbol{n}$. In this sense, $\boldsymbol{n}-\boldsymbol{l}$ represents the vector index of each element relative to the location of the elementary expansion (Fig. 1). Denoting $\langle X\rangle$ the average alloy composition of the system, the magnitude of the eigen-stress field due to element $\boldsymbol{l}$ is proportional to the difference $X(\boldsymbol{l}) - \langle X\rangle$. Under the above mentioned assumptions (and in absence of applied external forces), a mesh element with alloy composition equal to $\langle X\rangle$ does not contribute to the stress field. Since we consider a bulk material, periodic boundary conditions are appropriate and, hence, cyclic values are adopted for the vector $\boldsymbol{n} - \boldsymbol{l}$ in terms of the considered domain.

If the faraway elementary contributions could be neglected, the sum in Eq. (4) may be truncated to small values of $||\boldsymbol{n} - \boldsymbol{l}||$, with $||\ ||$ being a norm of the integer vector. For convenience, the relative vector index $\boldsymbol{k} = \boldsymbol{n} - \boldsymbol{l}$ can be adopted (see Fig. 1) and, then, Eq. (4) can be rewritten as:

$$\bar{\bar{\sigma}}(\boldsymbol{n}) = \sum_{\substack{\boldsymbol{k}\\ ||\boldsymbol{k}|| \le k_{max}}} \frac{X(\boldsymbol{n} - \boldsymbol{k}) - \langle X\rangle}{\delta X}\delta\bar{\bar{\sigma}}(\boldsymbol{k}). \tag{5}$$

For simplicity, the norm we use is $||\boldsymbol{k}|| = max\{|k_x|, |k_y|, |k_z|\}$. Therefore, the condition $||\boldsymbol{k}|| \le k_{max}$ corresponds to a cube with $(2k_{max}+1)^3$ mesh elements. Notice that, in previous equations, the elementary stress tensor $\delta\bar{\bar{\sigma}}(\mathbf{k})$ represents an input of the model and it can be extracted from the previously calculated $\Omega\delta\bar{\bar{\sigma}}(\mathbf{k})$.

Crystal alloy evolution is known to be driven by mobile native defects [17]. Assuming low defect concentration, alloy interdiffusion and precipitation have been modeled as a sequence of defect-driven exchanges of pairs of A–B atoms of neighboring mesh elements [18,19]. A great computational advantage of the model here proposed, with regard to every strategy supported on the Finite Elements technique, resides in the incremental updating of the total stress-field ($\bar{\bar{\sigma}}$). In fact, after an exchange between an A atom in cell $\boldsymbol{p}$ and a B atom in cell $\boldsymbol{q}$ the resulting stress field, $\bar{\bar{\sigma}}_{final}$, may be quickly recalculated from the initial one, $\bar{\bar{\sigma}}_{init}$, as:

$$\bar{\bar{\sigma}}_{final}(\boldsymbol{n}) = \bar{\bar{\sigma}}_{init}(\boldsymbol{n}) + \delta\bar{\bar{\sigma}}(\boldsymbol{n}-\boldsymbol{p}) - \delta\bar{\bar{\sigma}}(\boldsymbol{n}-\boldsymbol{q}) \tag{6}$$

with $\delta\bar{\bar{\sigma}}(\boldsymbol{k})$ is truncated to $||\boldsymbol{k}|| \leq k_{max}$. This strategy allows on-the-fly updating of the total stress field after every atom exchange event in a reasonably short computational time.

The exchange probability of an A–B atom pair depends on the total energy modification of the system and, thus, the dynamics of the system is driven by the energy changes associated to structural modifications. A contribution to these energy changes is given by the elastic energy, whereas another contribution is coming from the chemical mixing energy [20]. For a cubic material, the local elastic energy density (per unit volume) at every mesh element is given by [14]:

$$U(\bar{\bar{\sigma}}) = \frac{1}{2Y}\sum_i \sigma_i^2 - \frac{\nu}{Y}\sum_i \sum_{\substack{j \\ j>i}} \sigma_i\sigma_j + \frac{1}{G}\sum_i \tau_i^2 \tag{7}$$

where Y is the Young's modulus, ν is the Poisson's ratio and G is the shear modulus. For convenience, we have denoted the cartesian components $\{\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{yz}, \sigma_{xz}, \sigma_{xy}\}$ of the total stress tensor ($\bar{\bar{\sigma}}$) as $\{\sigma_1, \sigma_2, \sigma_3, \tau_1, \tau_2, \tau_3\}$, where σ_i (with $i = 1, 2, 3$) refers to the diagonal stress components whereas τ_i (with $i = 1, 2, 3$) refers to the shear components.

The total variation of the elastic energy of the system due to the exchange of an A-type atom from cell $\boldsymbol{p}$ with a B-type atom from cell $\boldsymbol{q}$ is given by:

$$\Delta E(A_p \rightleftharpoons B_q) = \Omega \sum_{\boldsymbol{n}} \left[U(\bar{\bar{\sigma}}(\boldsymbol{n}) + \delta\bar{\bar{\sigma}}(\boldsymbol{n}-\boldsymbol{p}) - \delta\bar{\bar{\sigma}}(\boldsymbol{n}-\boldsymbol{q})) - U(\bar{\bar{\sigma}}(\boldsymbol{n}))\right]. \tag{8}$$

Taking into consideration the incremental approach established in Eq. (6), it can be shown that Eq. (8) can be approximated by:

$$\Delta E(A_p \rightleftharpoons B_q) = \delta E(\boldsymbol{p}) - \delta E(\boldsymbol{q}) \tag{9}$$

with

$$\delta E(\boldsymbol{p}) = \sum_{\substack{\boldsymbol{k} \\ ||\boldsymbol{k}|| \leq k_{max}}} \left[\frac{1}{Y}\sum_i \sigma_i(\boldsymbol{p}+\boldsymbol{k})\Omega\delta\sigma_i(\boldsymbol{k}) - \frac{\nu}{Y}\sum_i\sum_{\substack{j \\ j\neq i}} \sigma_i(\boldsymbol{p}+\boldsymbol{k})\Omega\delta\sigma_j(\boldsymbol{k}) + \frac{2}{G}\sum_i \tau_i(\boldsymbol{p}+\boldsymbol{k})\Omega\delta\tau_i(\boldsymbol{k}) \right] \tag{10}$$

where second-order terms of $\delta\sigma_i$ and $\delta\tau_i$ have been neglected, and the same values for Y, ν, and G have been assumed in all mesh elements.

As shown in Eq. (10), the elementary energy variation $\delta E(\boldsymbol{p})$ is related to the interaction between the normal (diagonal) and shear components of the preexisting eigen-stress tensor field ($\bar{\bar{\sigma}}$) and the corresponding ones of the elementary stress field ($\delta\bar{\bar{\sigma}}$), associated to an elementary expansion. In this sense, and in an analogous way to the stress variation defined in Eq. (6), $\delta E(\boldsymbol{p})$ accounts for the energy modifications related to all the elements affected by the expansion of element $\boldsymbol{p}$. Consistently to Eq. (5), a truncation distance of k_{max} has been considered. Interestingly, according to Eq. (10), $\delta E(\boldsymbol{p})$ can be seen as a scalar potential for alloy interdiffusion.

Finally, in the context of alloy dynamic evolution, $\delta E(\boldsymbol{p})$ may be used as the physically-relevant magnitude to quantify the agreement between the stress field solution obtained by different numerical methods. In the present case, we will compare the results of the linear superposition strategy, proposed in this work, to those obtained by classical FEM, used as a reference. Moreover, as it has been pointed out above, this energy modification is a scalar (whereas the eigen-stress field is a tensor) and, thus, it allows a straightforward quantification of such agreement.

3. Code design and parallelization techniques

In this section, we present two algorithms based on the models described above, which are aimed to calculate the eigen-stress field (Linear Superposition Algorithm, LSA) and the elastic energy modification in atomistic interdiffusion of alloys (Atom Exchange Elastic Energy Evaluation algorithm, AE^4). Although, in our case, both algorithms work together to model alloy interdiffusion, they can be also used independently to solve other kind of problems. Firstly, we introduce the sequential baselines before we discuss our parallelization strategy based on data-parallelism to leverage horsepower of both Graphics Processing Units (GPUs) and chip multiprocessors (CMPs).

3.1. Sequential baselines

Algorithm 1 The Sequential version of our application that contains both *LSA* and AE^4 algorithms. This application is developed to model alloy interdiffusion.

```
$\delta\bar{\sigma}$ = Load Elementary_Diagonal_Stress
(elementary_stress_tensor.csv);
$\delta\bar{\tau}$ = Load Elementary_Tangential_Stress
(elementary_stress_tensor.csv);
$X$ = Load Concentration_Distribution(concentration.csv);
LSA();
$AE^4$();
printResults();
```

Algorithm 2 Linear Superposition Algorithm (LSA) to calculate the eigen-stress field.

```
$\bar{\sigma} = \mathbf{0}$ $\bar{\tau} = \mathbf{0}$
#pragma omp parallel for collapse (3);
for $n_x = 0$ to $N_x$ do
  for $n_y = 0$ to $N_y$ do
    for $n_z = 0$ to $N_z$ do
      for $k_x = -k_{max}$ to $k_{max}$ do
        for $k_y = -k_{max}$ to $k_{max}$ do
          for $k_z = -k_{max}$ to $k_{max}$ do
            $\boldsymbol{n} = (n_x, n_y, n_z)$; $\boldsymbol{k} = (k_x, k_y, k_z)$
            $\boldsymbol{l} = \boldsymbol{n} - \boldsymbol{k}$
            $\boldsymbol{l} = circularChecking(\boldsymbol{l})$
            $fact = \frac{X(\boldsymbol{l}) - \langle X \rangle}{\delta X}$
            $\boldsymbol{k}_{circular} = circularChecking(\boldsymbol{k})$
            $\bar{\sigma}(\boldsymbol{n}) += fact * \delta\bar{\sigma}(\boldsymbol{k}_{circular})$
            $\bar{\tau}(\boldsymbol{n}) += fact * \delta\bar{\tau}(\boldsymbol{k}_{circular})$
          end for
        end for
      end for
    end for
  end for
end for
```

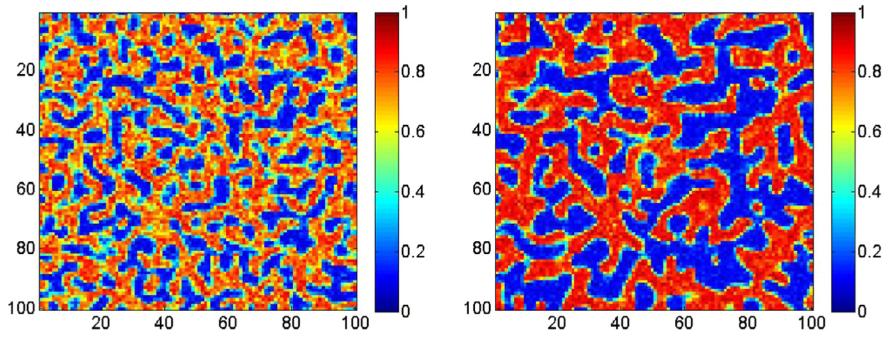


Fig. 2. 100 nm × 100 nm composition maps of a nominally $Fe_{0.5}Cr_{0.5}$ alloy after an anneal at 500 °C for two different evolution stages. Color scale correspond to molar Cr fraction, with blue denoting Fe-rich regions and red indicating Cr-rich regions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Algorithm 1 shows a Single Program Multiple Data (SPMD) pseudocode for the validation of our approach. The elementary stress field $\delta\bar{\bar{\sigma}}$ is calculated offline using Finite Element Methods (FEM) and it is loaded from the *elementary_stress_tensor.csv* file. Diagonal (normal) components are stored in the $\delta\bar{\sigma}$ 3D matrix (with $\delta\bar{\sigma} = (\delta\sigma_{xx}, \delta\sigma_{yy}, \delta\sigma_{zz})$) while the off-diagonal (tangential) components are stored in the $\delta\bar{\tau}$ 3D matrix (with $\delta\bar{\tau} = (\delta\tau_{xy}, \delta\tau_{xz}, \delta\tau_{yz})$). The size of these matrices is $\boldsymbol{N} = (N_x, N_y, N_z)$, which establishes the number of mesh elements in each direction. The alloy concentration distribution is loaded from the *concentration.csv* file and stored in the X, which is also a 3D matrix. In the examples used for illustration (see Fig. 2), the concentration distributions correspond to the iron–chromium alloy (FeCr), and they have been generated by the atomistic simulator MMonCa [19,21].

Algorithm 3 The Atom Exchange Elastic Energy Evaluation (AE^4) algorithm.

```
$n_z = 5$
#pragma omp parallel for collapse (2);
**for** $n_x = 0$ to $N_x$ **do**
  **for** $n_y = 0$ to $N_y$ **do**
    $S_1 = 0; S_2 = 0; S_3 = 0$
    $\boldsymbol{n} = (n_x, n_y, n_z)$
    **for** $k_x = -k_{max}$ to $k_{max}$ **do**
      **for** $k_y = -k_{max}$ to $k_{max}$ **do**
        **for** $k_z = -k_{max}$ to $k_{max}$ **do**
          $\boldsymbol{k} = (k_x, k_y, k_z)$
          $\boldsymbol{m} = circularChecking(\boldsymbol{n} + \boldsymbol{k})$
          $\boldsymbol{k}_{circular} = circularChecking(\boldsymbol{k})$
          $S_1 += \sum_i \sigma_i(\boldsymbol{m})\ \Omega\delta\sigma_i(\boldsymbol{k}_{circular})$
          $S_2 += \sum_i \sum_{j \neq i}^{j} \sigma_i(\boldsymbol{m})\ \Omega\delta\sigma_j(\boldsymbol{k}_{circular})$
          $S_3 += \sum_i \tau_i(\boldsymbol{m})\ \Omega\delta\tau_i(\boldsymbol{k}_{circular})$
        **end for**
      **end for**
    **end for**
    $\delta E[\boldsymbol{n}] += \frac{1}{Y} * S_1 - \frac{\mu}{Y} * S_2 + \frac{2}{G} * S_3$
  **end for**
**end for**
```

Algorithm 2 shows the LSA kernel, which calculates the eigenstress field of the alloy by linear superposition (see Eq. (5)). With this aim, it convolutes the concentration matrix X with the elementary stress matrices ($\delta\bar{\sigma}$ and $\delta\bar{\tau}$). The constants $\langle X \rangle$ and δX are previously described in Section 2. The convolution is performed by iterating over $(2k_{max}+1)^3$ nearest neighbors, which actually correspond to $(2k_{max}+1)^3$ points in a 3D stencil pattern of computation [22,23]. The k_{max} matrix borders in each dimension are also computed as periodic boundary conditions are applied to the simulation domain. Thus, the *circularChecking*() function returns valid matrix indexes by making use of such periodic conditions. The diagonal components of the resulting stress field are stored in the $\bar{\sigma}$ matrix, whereas the tangential components are stored in the $\bar{\tau}$ matrix.

Algorithm 3 shows the computation developed in the AE^4 kernel. This calculates the spatial distribution of the elementary energy variation δE (Eq. (10)), which acts as a potential for alloy interdiffusion (see Eq. (9)). The eigen-stress field of the sample, obtained by LSA (see Algorithm 2) and stored in $\bar{\sigma}$ and $\bar{\tau}$ matrix, is taken as an input. Alternatively, the eigen-stress field obtained by FEM can be taken for comparison and validation tests. Besides, the elementary stress field ($\delta\bar{\sigma}$ and $\delta\bar{\tau}$, already used in Algorithm 2), the mesh element volume (Ω), and the elastic constants of the material (Y, ν and G) are used in the AE^4 algorithm. For validation and illustration purposes, δE is calculated in a 2D plane. Therefore, the computation is based on a two-nested loop (n_x, n_y) that iterates over a $N_x \times N_y$ 2D-matrix.

3.2. Parallelization strategies

Algorithms 2 and 3 are well-suited for data-based parallelism. For instance, in Algorithm 2, the cells of $\bar{\sigma}$ and $\bar{\tau}$ matrices are written independently to each other. Hence, the most-external loops ($n_i = 0$ to $n_i = N_i$, with $i = x, y, z$) can be fully parallelized from a data point of view. This means $O(n^3)$ iterations that can be potentially developed in parallel. δE matrix in Algorithm 3 is also fully parallelized, having $O(n^2)$ iterations that can run in parallel as well. It is also noteworthy that the number of floating-point operations in most-internal loops of Algorithm 3 is higher than in Algorithm 2, which implies heavier threads in an hypothetical parallelization.

With that in mind, we have developed two different parallel implementations. The former is developed using OpenMP for latency-oriented architectures like traditional CPUs. OpenMP development is based on *#pragma* statements that are captured by the compiler, validated and translated to the appropriate function calls to the OpenMP library and runtime system. We refer the reader to [24,25] for insights about OpenMP programming model. The parallelization process consists of dividing the application in different "threads" that run in parallel on the target architecture. Algorithms 2 and 3 show the directive *#pragma omp parallel for* to parallelize the outer-for-loops that iterates over **n** dimension. The *collapse* clause is also included to merge loop iterations, increasing the total work units that will be partitioned across the available threads. The inner loop is left unchanged to be vectorized in case the compiler can do so. We set the -O3 compiler flag that will do for us in case there is no data-dependencies. Moreover, we have set the KMP_AFFINITY to *scatter* to distribute the threads across the processor, maximizing the usage of the cache storage space.

The latter implementation is tailored to NVIDIA GPUs based on the CUDA programming model [26]. CUDA programming model is very well-suited for data-based parallelism [27] as it is based on the execution of a very large set of lightweight threads that runs in parallel on the different NVIDIA GPU multiprocessors. For insights about CUDA programming model, we refer the reader to [26]. Algorithms 2 and 3 are parallelized on the GPU in two different kernels (i.e. functions that are executed on the GPU). Iterations of outer-loops are defined as threads, and they are equally distributed into blocks. We also use on-chip shared memory to store all the information shared by threads within the same block to increase overall memory bandwidth.

4. Evaluation

This section analyzes the proposed algorithms. Firstly, the hardware and software environment is described to provide fully reproducible results. Performance results are given for single precision numbers and a single iteration run. Nevertheless, these results are obtained as the average of 100 iterations to avoid possible runtime benefits like additional cache hits or operative system overheads.

We focus on the computational features of our algorithms and how it can be efficiently implemented on GPUs. In order to guarantee the correctness of our algorithms, a quality comparison between the results obtained by traditional FEM approaches and our codes is provided. Additionally, we have tested LSA in a simple particular configuration in which an analytic stress solution can be found. FEM calculations have been performed using the commercial-based software ANSYS [28].

4.1. Computational environment

During our computational experimentation, we have used the following computing platforms:

- The C, OpenMP and CUDA codes are executed on a linux-based workstation (Ubuntu 64 bit 14.04.4 LTS) with the following characteristics. On the CPU side, it has an Intel Xeon E3-1220 processor running at 3.10 GHz and endowed with four cores and 16 GBytes of DDR3 memory. On the GPU side, it has two Nvidia GPUs. One of them is a NVIDIA Tesla Kepler K40c with 2880 cores (15 Streaming Multiprocessors (SMs) and 192 Streaming Processors (SPs)) running at boost clock of 0.74 GHz, giving a raw processing power of up to 5068 GFLOPS. The other one is a NVIDIA Fermi GeForce GTX 580 with up to 512 cores (16 SMs and 32 SPs) running at boost clock of 1,5 GHz, giving a raw processing of up to 1581 GFLOPS.
- The ANSYS computations are executed on a desktop-based machine with an Intel Core I7-3630QM running at 2.4 GHz with 12 GB of DDR3 memory and SSD technology. The operative system is Windows 8.1 64 bits.

We use gcc 4.8.4 with the -O3 flag to compile our CPU implementations, and CUDA compilation tools release 6.5 on the GPU side. FEM results are obtained with commercial ANSYS release 16.

4.2. Benchmarking

As pointed out in Section 2, the elementary energy variation $\delta E(\boldsymbol{n})$ defined in Eq. (10) can be used as a merit figure for the validation of the eigen-stress fields computed by LSA, compared to FEM. This gives a scalar map directly describing the stress-related driving force for structural evolution. As a workbench for the comparison, we use a spinodally decomposed metal alloy. Such material systems present a complex vein-like geometry with a high variety of local stress configurations. In particular, we select a simulated sample of a nominally $Fe_{0.5}Cr_{0.5}$ crystal alloy after an anneal at 500 °C, which suffers spontaneous separation into iron-rich and chromium-rich phases (known as α and α' phases, respectively).

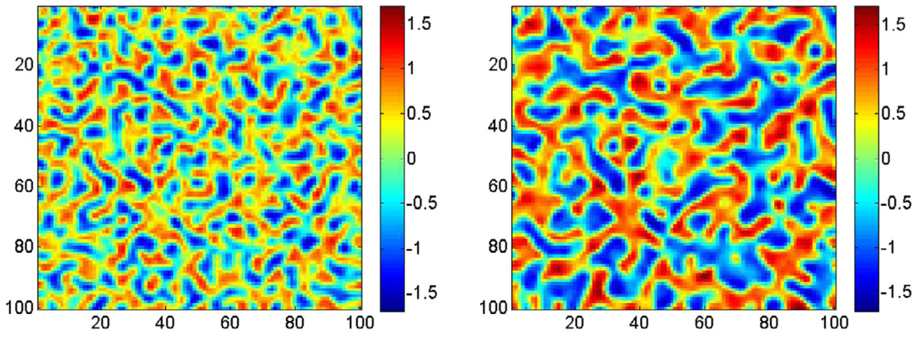
The simulation samples have been prepared using the MMonCa atomistic simulator [21], based on the Object Kinetic Monte Carlo method (OKMC) [9], following the procedure described in [19]. The input parameters of the OKMC simulations are also those listed in Ref. [19]. A simulation box of 100 nm $\times$ 100 nm $\times$ 10 nm with mesh spacing of 1 nm has been set for the simulations. Thus, grid dimensions are $\boldsymbol{N} = (100, 100, 10)$. Assuming an atom concentration of 8.7×10^{22} for the alloy, the number of atoms in each 1 nm^3 cell is $N_{at} = 87$ and, then, the elementary composition variation is $\delta X = 1.15 \times 10^{-2}$.

Typical composition maps are shown in Fig. 2 for different annealing stages, showing average spacial periodicity $\langle\lambda\rangle$ of $\approx$7 nm for the shorter anneal and $\approx$10 nm for the longer one (see Fig. 2). As iron and chromium constituent materials have a certain lattice mismatch ($\epsilon_{max} \approx 0.7\%$), an eigen-stress field will be associated to the spatial composition changes in the structure. This eigen-stress field has been estimated by using both LSA (see Eq. (5)) and FEM, using commercial software ANSYS [28]. Moreover, the elementary $\delta\bar{\bar{\sigma}}(\boldsymbol{k})$, used as the building-block for LSA, was previously calculated off-line by ANSYS. Details about these calculations are summarized below.

Regarding FEM, the infinitesimal strain tensor is considered, since for the above described problem the expected displacement gradient is sufficiently less than the unity [29]. In particular, the governing equations of the problem in hand corresponding to the *linear elastic problem* [29,14], where strains are in geometric compatibility with the displacements, stresses and strains are related via a cubic orthotropic relationship of the considered materials, and the stress tensor verifies the equations of motion, all under the appropriate boundary conditions. The resulting displacement-based boundary value problem may be posed as a weak form whose numerical solution is approximated by the Finite Element Method [13]. To this respect, the simulation domain was discretized into hexahedral finite elements of 1 nm edge. A high order volumetric finite element, with quadratic displacement behavior, has been considered in order to take into account the "bowing effect" of the edges in each cubic mesh element during the eigen-expansion. In this sense, the 3D finite element SOLID186 of the ANSYS library [30], a twenty-node brick element with control nodes located at the corners and at the edges midsize, represents the simplest approach to the modeling of such effect and it has been chosen for our FEM calculations. Moreover, in consonance with the characteristics of the model presented at Section 2, cyclic (periodic) displacement conditions have been applied over the boundaries of the considered domain. The values of elastic parameters for iron ($Y = 2.11 \cdot 10^{11}$ N/m^2, $\nu = 0.29$, and $G = 1.7 \cdot 10^{11}$ N/m^2) have been assumed for the alloy. In this context, the solution of the above mentioned weak form is numerically approximated through the resolution of a linear system of equations [13,31] using the Direct Sparse Solver [32]. The obtained solution provides the nodal degree-of-freedom displacement values. Stresses and strains are derived from the primary solution in displacements. The global stiffness matrix of such system is symmetric and it has been obtained by uniform reduced integration of each element [31,32].

4.3. Qualitative evaluation

The eigen-stress fields corresponding to the two FeCr alloy samples of Fig. 2 have been estimated both using FEM and using LSA with $k_{max} = 4$. In order to compare the results of the two


(a) Shorter anneal—FEM. (b) Longer anneal—FEM.

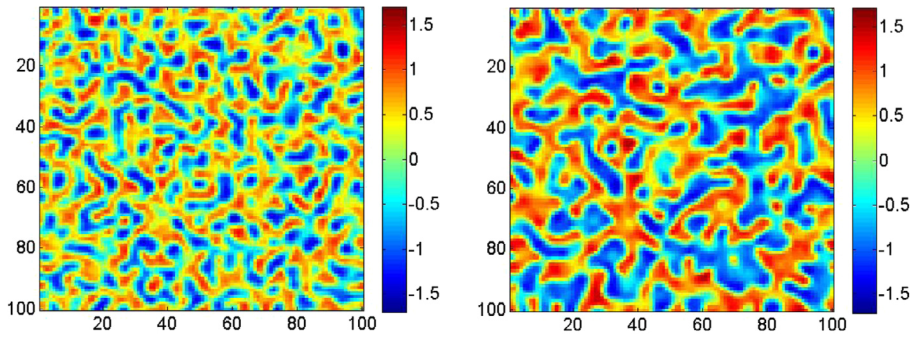

(c) Shorter anneal—LSA. (d) Longer anneal—LSA.

Fig. 3. Comparison of the interdiffusion potential maps $\delta E(\boldsymbol{n})$ calculated from the eigen-stress fields given by FEM (top panels) and those given by LSA with $k_{max} = 4$ (bottom panels) for the two FeCr samples of Fig. 2 (left: shorter anneal, right: longer anneal). Map dimensions are 100 nm × 100 nm and the color scale corresponds to energy in meV units (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

methods, the corresponding interdiffusion potential maps $\delta E(\boldsymbol{n})$ have been calculated using (10) and displayed in Fig. 3. An overall qualitative agreement between LSA and the reference FEM method is observed for both samples.

A quantitative comparison can be also performed as a function of the truncation distance of LSA. To do that, we choose to use two cubic samples corresponding to the same anneals than those of Fig. 2 but with 40 nm × 40 nm × 40 nm dimensions and we vary k_{max} from 0 to 10, going from lower to higher accuracy, but also from higher to lower computation efficiency. The agreement to FEM has been quantified with the relative quadratic error, calculated as:

$$error = \frac{1}{\sqrt{N}} \frac{\sqrt{\sum_{\boldsymbol{n}} (\delta E_{LSA}(\boldsymbol{n}) - \delta E_{FEM}(\boldsymbol{n}))^2}}{max(\delta E_{FEM}(\boldsymbol{n})) - min(\delta E_{FEM}(\boldsymbol{n}))}, \quad (11)$$

where N is the number of addends in the summation, δE_{LSA} are the values calculated by LSA, δE_{FEM} are the values calculated by FEM, and the range of δE_{FEM} has been used for normalization. The results are shown in Fig. 4. It can be observed that, for every k_{max}, the error is lower for the sample with lower average spatial period $\langle\lambda\rangle$. As expected, the error is found to depend on the ratio between the edge of the considered cube, $(2k_{max} + 1)L$, and the $\langle\lambda\rangle$ of the sample, with L being the edge of the mesh elements ($L = 1$ nm in our examples). In both samples, the error falls below 5% for $\frac{(2k_{max}+1)L}{\langle\lambda\rangle} > \frac{1}{2}$, and below 2% for $\frac{(2k_{max}+1)L}{\langle\lambda\rangle} > 1$. In particular, in the cases of Fig. 3 the errors are quantified to be 1.3% for the shorter anneal sample ($\langle\lambda\rangle \approx 7$ nm) and 2.5% for the longer anneal one ($\langle\lambda\rangle \approx 10$ nm).

It is worthy to notice that the lower the spacial period, the lower the required k_{max} to achieve a given accuracy and, thus, the higher the computation efficiency. Hence, the method proposed here is specially suitable for highly nanostructured systems. Conversely,

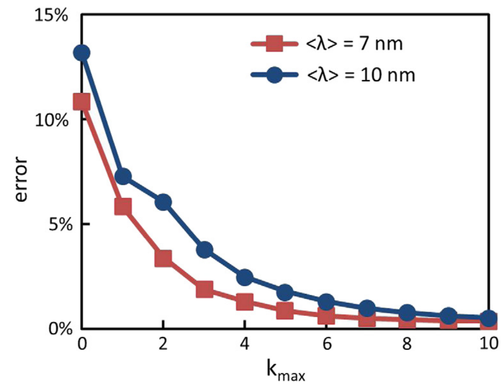


Fig. 4. Mean quadratic error of LSA with respect to FEM as a function of k_{max} for the two samples of Figs. 2 and 3. The samples are denoted by their approximated average periodicity ($\langle\lambda\rangle = 7$ nm for the shorter anneal sample, and 10 nm for the longer anneal sample).

for large $\langle\lambda\rangle$ the efficiency of our approach for reasonable accuracies becomes low. In that case, either a scaling of meshing spacing or the use of other methods should be considered.

The goodness of LSA may be also evaluated for an orthotropic configuration in which the elastic problem has analytic solution [14]. The selected configuration is a biaxially strained alloy superlattice, with periodic one-dimensional composition dependence $X(z)$. The composition profile has been chosen to be $X(z) = \langle X\rangle + \Delta X \cdot sin(2\pi z/\lambda)$. In this case, the stress field is given by $\sigma_{xx}(z) = \sigma_{yy}(z) = \sigma_0 \cdot sin(2\pi z/\lambda)$, with $\sigma_0 = Y\epsilon_{max}\Delta X/(1-\nu)$, and $\sigma_{zz} = \sigma_{xy} = \sigma_{xz} = \sigma_{yz} = 0$. This theoretical solution has been compared to the numerical results obtained by both FEM and LSA. For typical values of $\lambda = 8$ nm, $\langle X\rangle = 0.5$, and $\Delta X = 0.3$, a relative quadratic error of 2.6% with respect to the analytic solution is obtained for FEM, whereas relative quadratic errors of 4.7% and 3.0% are obtained for LSA with $k_{max} = 4$ and $k_{max} = 8$, respectively.

Table 1
Execution time (in seconds) of the Linear Superposition Algorithm (LSA) for a 100 × 100 × 10 mesh. The k_{max} parameter is varied to increase the granularity for each thread.

k_{max}	C (1)	OPENMP (2)	Speed-up (2) vs. (1)	CUDA (3)	Speed-up (3) vs. (1)
$k_{max} = 1$	1.30	0.66	2.0×	2.6×10^{-4}	5016×
$k_{max} = 2$	1.48	0.59	2.5×	1.2×10^{-3}	1230×
$k_{max} = 3$	1.85	0.66	2.8×	3.1×10^{-3}	593×
$k_{max} = 4$	2.53	0.88	3.0×	6.9×10^{-3}	369×
$k_{max} = 5$	3.67	1.23	3.1×	1.2×10^{-2}	297×
$k_{max} = 6$	5.14	1.66	3.1×	2.0×10^{-2}	253×
$k_{max} = 7$	7.24	2.34	3.1×	3.4×10^{-2}	214×
$k_{max} = 8$	9.97	3.22	3.1×	4.5×10^{-2}	220×
$k_{max} = 9$	13.39	4.32	3.1×	6.3×10^{-2}	211×
$k_{max} = 10$	17.53	5.66	3.1×	8.7×10^{-2}	202×

Table 2
Execution time (in seconds) of the Atom Exchange Elastic Energy Evaluation (AE^4) algorithm for a 100 × 100 map. We vary the k_{max} parameter to increase the granularity for each thread.

k_{max}	C (1)	OPENMP (2)	Speed-up (2) vs. (1)	CUDA (3)	Speed-up (3) vs. (1)
$k_{max} = 1$	3.8×10^{-3}	2.1×10^{-3}	1.8×	7.0×10^{-5}	55×
$k_{max} = 2$	1.9×10^{-2}	9.1×10^{-3}	2.1×	2.8×10^{-4}	68×
$k_{max} = 3$	5.6×10^{-2}	2.6×10^{-2}	2.2×	7.4×10^{-4}	76×
$k_{max} = 4$	0.13	6.0×10^{-2}	2.2×	1.7×10^{-3}	78×
$k_{max} = 5$	0.21	0.10	2.2×	3.2×10^{-3}	65×
$k_{max} = 6$	0.42	0.19	2.2×	5.3×10^{-3}	79×
$k_{max} = 7$	0.65	0.29	2.2×	8.2×10^{-3}	79×
$k_{max} = 8$	0.93	0.42	2.2×	1.2×10^{-2}	77×
$k_{max} = 9$	1.31	0.59	2.2×	1.7×10^{-2}	77×
$k_{max} = 10$	1.75	0.80	2.2×	2.3×10^{-2}	76×

Relative quadratic errors of LSA with respect to FEM with values of 7.2% and 1.9% are obtained for $k_{max} = 4$ and $k_{max} = 8$, respectively. The slower convergence of LSA in this case, compared to the examples of Figs. 3 and 4, can be related to the fact that the structure modulation here is only one-dimensional (with infinite period in the plane) whereas in the previous examples the structures are three-dimensionally modulated.

4.4. Performance evaluation

This section shows a performance evaluation for both *LSA* and AE^4 algorithms on different hardware architectures (see Tables 1 and 2). As shown in Eq. (6), $\delta\bar{\bar{\sigma}}(\boldsymbol{k})$ is truncated to $||\boldsymbol{k}|| \leq k_{max}$. Therefore, k_{max} determines the computational cost of our algorithms, and it may limit their scalability. With that in mind, we vary this input parameter to analyze its influence on the system.

Column (1) of Table 1 shows the average execution times of a single core C version of the LSA algorithm running on our server machine for the examples of Fig. 3 with a 100 × 100 × 10 mesh and different values of k_{max}. These execution times are proportional to $(2k_{max} + 1)^3$ plus a constant. $(2k_{max} + 1)^3$ is the number of iterations of the inner-most loops (i.e. nearest neighbors for each cell) in Algorithms 2 and 3.

Regarding FEM, the execution times to solve each example of Fig. 3 using ANSYS is 2020 s in the windows machine specified in Section 4.1. Thus, the LSA approach for $k_{max} = 4$, which provide similar quality results than ANSYS, as previously discussed in Section 4.3, is almost three orders of magnitude faster than FEM resolution with ANSYS.

Furthermore, additional computation speed-up factor of LSA can be obtained by incremental use of the algorithm and by a parallel implementation of the code. As mentioned in Section 2, a great computational advantage of LSA for on-the-fly use in atomistic simulations is the ability for incremental calculation of the stress-field. According to Eq. (6), the resulting stress field after an atom exchange between two cells can be recalculated from the initial one with only 2 steps (instead of 100 × 100 × 10) of the outer loop of Algorithm 2. This poses a 5×10^4 speed-up factor with respect to the values of column (1) of Table 1, bringing the computation times into the scale of tens of microseconds and entering into the requirements to be used within OKMC simulators.

The other acceleration strategy is parallelization. Columns (2) and (3) of Table 1 show the impact of parallelization on the execution time for our LSA algorithm. Execution times of a multicore OpenMP version, column (2), and of a CUDA version that leverage a Kepler-based Nvidia GPU, column (3), are compared to the already mentioned single core C version. OpenMP version uses up to four threads in the same multicore system, providing a 2–3× speed-up factor. As previously commented, the LSA code is massively parallel by its definition. Thus, data parallelism approach developed on the GPU offers performance gains of up to 5016× speed-up factor compared to the C version for $k_{max} = 1$ and in the order of 200× for large values of k_{max}. The execution time of our CUDA version is found to be proportional to $(2k_{max} + 1)^3$ and, in contrast to C and OpenMP versions, no extra term independent on k_{max} is obtained in the fitting. This fact accounts for the variation of the speed-up factor with k_{max}.

The GPU implementation of the LSA kernel deals with 3D matrix (i.e. 100 × 100 × 10), which means a very large number of CUDA thread blocks. Actually, our empirically demonstrated best CUDA configuration for this kernel is 256 threads per block. Thus, this offers up to 391 CUDA threads blocks to be scheduled in SMs that is large enough to fulfill all GPU resources as there is not any other hardware limitation, such as register or shared memory usage. Whenever the k_{max} configuration parameter is increased the thread granularity also increases and those threads become coarser-grain threads, which is not very well-suited for the GPU computation model as it prefers throughput oriented computation.

Table 2 shows the execution time for our AE^4 kernel for a 100 × 100 map. Like in the previous example, three different implementations are under study (C, OpenMP and CUDA) that use a single, multicore and GPU systems, respectively, on our server machine. In this case, FEM calculations are not a reference for kernel performance, although they give an alternative input to the

AE^4 algorithm that may be used in the validation process of LSA algorithm. The evaluation of the elastic energy variation associated to an atom exchange in OKMC simulations requires 2 calculations of $\delta(\boldsymbol{n})$ (instead of the 100×100 of the whole map) and, thus, computation times would be scaled by 2×10^{-4}.

The execution times of the three AE^4 versions are found to be approximately proportional to $(2k_{max} + 1)^3$. Consequently, parallelization-related speed-up factors are hardly dependent on k_{max}. A $2\times$ factor is obtained with OpenMP, whereas a factor about $70\times$ is achieved by the CUDA version. This kernel deals with 2D matrix (100×100) and thus the number of CUDA thread blocks is only 40 in this case. These 40 blocks can be directly mapped to the Stream Multiprocessors (SMs) at once and, therefore, there is no block scheduling latency. This fact makes that thread granularity is not such a critical factor like in LSA kernel.

5. Conclusions and future works

Predictive simulation of alloys structural evolution, involved in critical scenarios such as structural steel members in nuclear plants, can benefit from the great advances in the field of high performance computing in order to overcome computational barriers. Nowadays, the use of KMC methods supported on FEM calculations represents a highly time-consuming process. Alternatively, this work proposes two different algorithms (LSA and AE^4) to calculate the eigen-stress field and the elastic energy modification in atomistic interdiffusion of alloys. In particular, the main contributions of this paper include the following:

1. LSA allows fast calculation eigen-stress field of complex geometry sample. It also offers incremental on-the fly updating of the total stress-field after every atom exchange event in KMC simulations in a reasonably short computational time.
2. AE^4 provides a critical input for KMC. Additionally, it can provide a scalar merit figure in order to evaluate the accuracy of the approximate stress field calculations.
3. Algorithms here proposed are massively parallel by their definition and, therefore, we develop a data-parallel approach to leverage Graphics Processing Units (GPUs).

Predictive simulation of alloy structural evolution on GPUs is still at a relatively early stage and, in this sense, this work provides a relatively simple parallelization. But, with many other types of optimization still to be explored, this field seems to offer a promising and potentially fruitful area of research.

On the physical framework, the present model may be developed through the implementation of another types of boundary conditions for the considered domain. On the hardware side, it is expected to get even higher accelerations on GPUs whenever the problem size keeps growing and larger device memory space is available. Moreover, we may anticipate that the benefits of our approach would also increase when parallelizing k_{max} most-internal loop since it would provide fine-grain parallelism, and also when using future GPU generations endowed with thousands of cores, and eventually grouped into GPU clusters to lift performance into unprecedented gains, where parallelism is called to play a decisive role.

Acknowledgments

This research was supported by the Fundación Séneca (Agencia Regional de Ciencia y Tecnología de la Región de Murcia) under grants 18946/JLI/13 and 19366/PI/14, and by Universidad Católica de Murcia under grant PMAFI/04/14. We also thank NVIDIA for hardware donation under GPU Educational Center 2014–2015 and Research Center 2015–2016.

References

[1] E. Kasper, Physics and Applications of Quantum Wells and Superlattices, Springer, 1987, pp. 101–131.
[2] D.-I.J. Olschewski, et al., Acta mechanica 136 (3–4) (1999) 171–192.
[3] D. Seol, S. Hu, Y. Li, J. Shen, K. Oh, L. Chen, Acta Mater. 51 (17) (2003) 5173–5185.
[4] N. Ledentsov, V. Shchukin, M.e. Grundmann, N. Kirstaedter, J. Böhrer, O. Schmidt, D. Bimberg, V. Ustinov, A.Y. Egorov, A. Zhukov, et al., Phys. Rev. B 54 (12) (1996) 8743.
[5] R.L. Klueh, D.R. Harries, High-Chromium Ferritic and Martensitic Steels for Nuclear Applications, Vol. 3, ASTM International, 2001.
[6] D.C. Gilmer, J.K. Schaeffer, W. Taylor, C. Capasso, K. Junker, J. Hildreth, D. Tekleab, B. Winstead, S. Samavedam, IEEE Trans. Electron Devices 57 (4) (2010) 898–904.
[7] I. Martin-Bragado, I. Avci, K. El Sayed, V. Koltyzhenkov, E. Lyumkis, M. Johnson, J. Comput. Electron. 7 (3) (2008) 103–106.
[8] N. Zographos, C. Zechner, P. Castrillo, I. Martin-Bragado, ION IMPLANTATION TECHNOLOGY 2012: Proceedings of the 19th International Conference on Ion Implantation Technology, vol. 1496, AIP Publishing, 2012, pp. 212–216.
[9] M. Jaraiz, E. Rubio, P. Castrillo, L. Pelaz, L. Bailon, J. Barbolla, G. Gilmer, C. Rafferty, Mater. Sci. Semicond. Process. 3 (1) (2000) 59–63.
[10] E. Martínez, O. Senninger, C.-C. Fu, F. Soisson, Phys. Rev. B 86 (22) (2012) 224109.
[11] M.A. Zaeem, S.D. Mesarovic, J. Comput. Phys. 229 (24) (2010) 9135–9149.
[12] M.R. Tonks, D. Gaston, P.C. Millett, D. Andrs, P. Talbot, Comput. Mater. Sci. 51 (1) (2012) 20–29.
[13] T.J. Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Courier Corporation, 2012.
[14] E.W. Chaves, Notes On Continuum Mechanics, Springer Science & Business Media, 2013.
[15] S. Hu, L. Chen, Acta Mater. 49 (11) (2001) 1879–1890.
[16] J. Wang, S. Bhattacharyya, Q. Li, T. Heo, X. Ma, L.-Q. Chen, Phil. Mag. Lett. 92 (7) (2012) 327–335.
[17] H. Mehrer, Diffusion in Solids: Fundamentals, Methods, Materials, Diffusion-Controlled Processes, Vol. 155, Springer Science & Business Media, 2007.
[18] P. Castrillo, R. Pinacho, M. Jaraiz, J. Rubio, J. Appl. Phys. 109 (10) (2011) 103502.
[19] I. Dopico, P. Castrillo, I. Martin-Bragado, Acta Mater. 95 (2015) 324–334.
[20] G. Bonny, R.C. Pasianot, D. Terentyev, L. Malerba, Phil. Mag. 91 (12) (2011) 1724–1746.
[21] I. Martin-Bragado, A. Rivera, G. Valles, J.L. Gomez-Selles, M.J. Caturla, Comput. Phys. Comm. 184 (12) (2013) 2703–2710.
[22] J.M. Cecilia, J.L. Abellán, J. Fernández, M.E. Acacio, J.M. García, M. Ujaldón, J. Supercomput. 62 (2) (2012) 787–803.
[23] S. Kamil, C. Chan, L. Oliker, J. Shalf, S. Williams, Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on, IEEE, 2010, pp. 1–12.
[24] L. Dagum, R. Enon, IEEE Comput. Sci. Eng. 5 (1) (1998) 46–55.
[25] R. Chandra, Parallel Programming in OpenMP, Morgan kaufmann, 2001.
[26] NVIDIA Corporation, NVIDIA CUDA C Programming Guide 6.5, 2014.
[27] J.M. Cecilia, J.M. García, A. Nisbet, M. Amos, M. Ujaldón, J. Parallel Distrib. Comput. 73 (1) (2013) 42–51.
[28] S. Moaveni, Finite Element Analysis: Theory and Application With ANSYS, Pearson Education India, 2003.
[29] X.O. Olivella, C.A. de Saracíbar Bosch, Mecánica de medios continuos para ingenieros, Univ. Politèc. de Catalunya, 2002.
[30] H.-H. Lee, Finite Element Simulations with ANSYS Workbench 16, SDC publications, 2015.
[31] O.C. Zienkiewicz, R.L. Taylor, The Finite Element Method: Solid Mechanics, Vol. 2, Butterworth-heinemann, 2000.
[32] A. ANSYS, User's manual (ver. 14.0), ANSYS Incorporated, Canonsburg, PA.