Computer Physics Communications 222 (2018) 240–249

Contents lists available at ScienceDirect

Computer Physics Communications

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

Simulating electron wave dynamics in graphene superlattices exploiting parallel processing advantages☆

Manuel J. Rodrigues [a], David E. Fernandes [a], Mário G. Silveirinha [a,b], Gabriel Falcão [a,*]

[a] Instituto de Telecomunicações and Department of Electrical and Computer Engineering, University of Coimbra, 3030-290, Coimbra, Portugal
[b] Instituto Superior Técnico—University of Lisbon, Avenida Rovisco Pais, 1, 1049-001 Lisboa, Portugal

ARTICLE INFO

Article history:
Received 31 January 2017
Received in revised form 8 June 2017
Accepted 24 August 2017
Available online 11 September 2017

Keywords:
Graphene Superlattice (GSL)
Electron wave dynamics
Finite-Difference Time-Domain (FDTD)
Open Computing Language (OpenCL)
Graphics Processing Unit (GPU)
Parallel Processing
Multi-GPU

ABSTRACT

This work introduces a parallel computing framework to characterize the propagation of electron waves in graphene-based nanostructures. The electron wave dynamics is modeled using both "microscopic" and effective medium formalisms and the numerical solution of the two-dimensional massless Dirac equation is determined using a Finite-Difference Time-Domain scheme. The propagation of electron waves in graphene superlattices with localized scattering centers is studied, and the role of the symmetry of the microscopic potential in the electron velocity is discussed. The computational methodologies target the parallel capabilities of heterogeneous multi-core CPU and multi-GPU environments and are built with the OpenCL parallel programming framework which provides a portable, vendor agnostic and high throughput-performance solution. The proposed heterogeneous multi-GPU implementation achieves speedup ratios up to 75x when compared to multi-thread and multi-core CPU execution, reducing simulation times from several hours to a couple of minutes.

Program summary

Program title: GslSim.
Program Files doi: http://dx.doi.org/10.17632/prmfv63nj6.1
Licensing provisions: GPLv3.
Programming language: C, OpenCL and Matlab for results analysis.
Nature of problem: Computing the time evolution of electron waves in graphene superlattices is a time consuming process due to the high number of necessary nodes to discretize the spatial and time domains.
Solution method: We develop a simulator based on the C/OpenCL standards to study the time evolution of electron waves in graphene superlattices by exploiting hardware architectures such as graphics processing units (GPUs) to speedup the computation of the pseudospinor.

1. Introduction

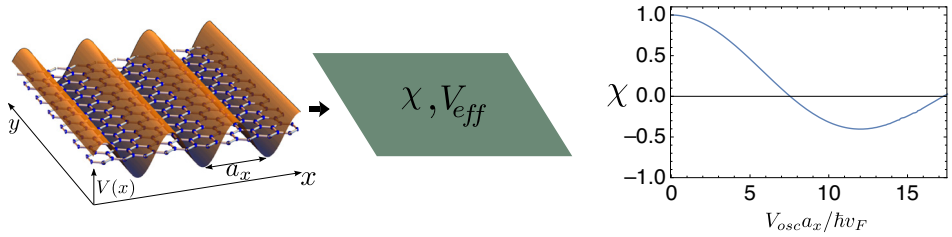
Graphene is a carbon-based two-dimensional material where the carbon atoms are arranged in a hexagonal lattice. Recent studies suggested the possibility of controlling the electronic properties of graphene by applying an external periodic electrostatic potential on its surface with a patterned metallic gate, among other possibilities. These nanostructured materials are known as graphene superlattices (GSLs) [1–10]. The low-energy dynamics of the electrons in GSLs is typically characterized by the massless Dirac equation [11–13] whose solution is usually numerically determined, for instance with the Finite-Difference Time-Domain (FDTD) method [14,15]. The FDTD algorithm is used in a variety of scientific domains and the associated computational complexity is determined by the nature of the problem. Previous works in the context of electromagnetism have shown that it is possible to obtain impressive speedup ratios (on the order of 20–100x) with a single graphics processing unit (GPU) implementation, e.g., [16,17]. Furthermore, a multi-GPU environment enables additional speedup gains, e.g., [18,19].

The application of the FDTD scheme to the electron wave propagation in graphene platforms typically leads to computationally demanding simulations, consuming long periods of processing time. This is mainly due to two factors: first, the computational complexity associated with the density of nodes necessary to accurately discretize the spatial domain and the nature of the FDTD methodology which is based on a leap-frog scheme; second, the hardware and software limitations of the computational resources that are typically used, such as bandwidth constraints or the low

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
E-mail address: gff@co.it.pt (G. Falcão).

http://dx.doi.org/10.1016/j.cpc.2017.08.023
0010-4655/



(a) (b)

Fig. 1. (a) Graphene superlattice characterized by a sinusoidal electrostatic potential in the microscopic model and the corresponding continuum model where the granular details are homogenized. (b) Anisotropy parameter χ of the homogenized superlattice as a function of the peak modulation amplitude V_{osc}.

number of cores available in central processing units (CPUs) or the available sequential programming models.

The purpose of this work is to develop a framework that enables the fast simulation of the electron wave dynamics in GSLs using either a "microscopic" approach (relying on the two-dimensional massless Dirac equation) or an effective medium formalism wherein the microscopic details of the superlattice are described by some effective parameters. Particularly, we focus on parallelization strategies relying on the C/OpenCL standards to exploit higher throughput performance in heterogeneous multi-GPU environments [20–23]. To this end, we propose two distinct models, namely a simulation concurrency model and a device concurrency model that capture different simulation scenarios. Furthermore, we present a detailed study of a time evolution problem in graphene superlattices with localized scattering centers.

The article is organized as follows. In Section 2 we present a brief overview of the electron wave propagation in GSLs and of the FDTD numerical solution. Section 3 describes the adopted computational procedures and parallelization strategies. The proposed methodologies are applied to study the time evolution of electronic states in a superlattice with localized scattering centers in Section 4. Performance metrics are reported and discussed in Section 5. The article ends with a brief conclusion in Section 6.

2. Graphene superlattices and the electron wave propagation

2.1. Formalism

This section identifies and describes the key steps in the formalization of the equations that govern the behavior of electron waves in graphene-based nanostructures. This method was developed in [15] and establishes the basis of our study.

The propagation of charge carriers in graphene superlattices may be characterized in the spatial and time domains by solving the massless Dirac equation [12]:

$$\hat{H}\psi = i\hbar\frac{\partial}{\partial t}\psi, \tag{1}$$

being $\hat{H} = -i\hbar v_F\sigma\cdot\nabla + V(x,y)$ the microscopic Hamiltonian operator near the K point, V the microscopic electric potential, $\psi = \{\Psi_1, \Psi_2\}^T$ the two component pseudospinor, $v_F \approx 10^6 m/s$ is the Fermi velocity, $\sigma = \sigma_x\hat{x} + \sigma_y\hat{y}$ a tensor written in terms of the Pauli matrices and $\nabla = \frac{\partial}{\partial x}\hat{x} + \frac{\partial}{\partial y}\hat{y}$. In GSLs, the potential V is a periodic function of space. A complex spatial dependence of the potential V can increase the computational effort to an undesired level and even limit the understanding of the relevant physical phenomena. A solution to reduce the complexity of the problem is to use effective medium techniques. It was recently shown that electronic states with the pseudo-momentum near the Dirac K point can be accurately modeled using an effective medium framework [24,25]. Within this approach, the microscopic potential is homogenized and the effective Hamiltonian treats the superlattice as a continuum characterized by some effective parameters [24,25]. For the cases of interest in this work, the effective Hamiltonian is of the form:

$$\left(\hat{H}_{eff}\psi\right)(r) = \left[-i\hbar v_F\sigma_{ef}\cdot\nabla + V_{eff}\right]\cdot\psi(r), \tag{2}$$

where $\sigma_{eff} = \chi_{xx}\sigma_x\hat{x} + \chi_{yy}\sigma_y\hat{y}$ and V_{eff} is an effective potential. Moreover, the energy dispersion of the stationary states in the homogenized superlattice can be calculated using [24]:

$$|E - V_{eff}| = \hbar v_F\sqrt{\chi_{xx}^2k_x^2 + \chi_{yy}^2k_y^2}, \tag{3}$$

where $\mathbf{k} = (k_x,\ k_y)$ is the wave vector of the electronic state with respect to the K point and E is the electron energy.

Next we characterize the effective parameters χ_{xx}, χ_{yy} and V_{eff} of the effective Hamiltonian of two distinct superlattices.

2.1.1. Anisotropic superlattices

To begin with, we consider 1D-type graphene superlattices described by a microscopic potential with a spatial variation $V(x) = V_{av} + V_{osc}\sin(2\pi x/a_x)$, as shown in Fig. 1. Here, V_{av} is the average electric potential, V_{osc} is the peak amplitude of the oscillations and a_x is the spatial period. These structures have been extensively studied in the literature and can have strongly anisotropic Dirac cones and particle velocities, allowing for the diffractionless propagation of electron waves [5,6,15,25].

The continuum model for the propagation of electrons in these superlattices was thoroughly discussed in [15,24,25]. In particular, in Ref. [24] it was found that the effective parameters of the stratified superlattice satisfy $\chi_{xx} = 1$ and $\chi_{yy} = \chi$. The anisotropy parameter χ depends on the peak modulation amplitude V_{osc} and can be numerically calculated using the approach described in [24]. The explicit dependence of χ on V_{osc} is represented in Fig. 1, and it varies from $\chi = -0.4$ to $\chi = 1$. The latter value corresponds to pristine graphene. The anisotropy ratio determines the (wave packet) electron velocity, which under the continuum formalism is $\mathbf{v} = \frac{1}{\hbar}\nabla_k E = \mathrm{sgn}(E - V_{av})v_F(k_x^2 + \chi^2k_y^2)^{-\frac{1}{2}}(k_x\hat{x} + \chi^2k_y\hat{y})$ [15,24,25]. Thus, the value of χ determines the degree of anisotropy and a preferred direction of propagation. In particular, in an extreme anisotropy regime, where the anisotropy ratio vanishes, $\chi = 0$, the group velocity is equal to $v = \pm v_F\hat{x}$, so that the electron waves propagate without diffraction along the x-direction [3,15,26–30]. Thus the electron transport differs in a drastic manner from pristine graphene ($\chi = 1$) wherein the electrons propagate parallel to the quasi-momentum $\mathbf{k}$.

2.1.2. Superlattices with localized scattering centers

Next, we characterize the effective Hamiltonian of a superlattice formed by localized scattering centers, which is modeled by an electric potential periodic in the x- and y-coordinates $V(x,y) = V_{av} + V_{osc}\sin(2\pi x/a_x)\sin(2\pi y/a_y)$ as illustrated in Fig. 2. To the best of our knowledge, this superlattice was not previously discussed in detail in the literature. In this work, it is assumed that $a_x = a_y =$

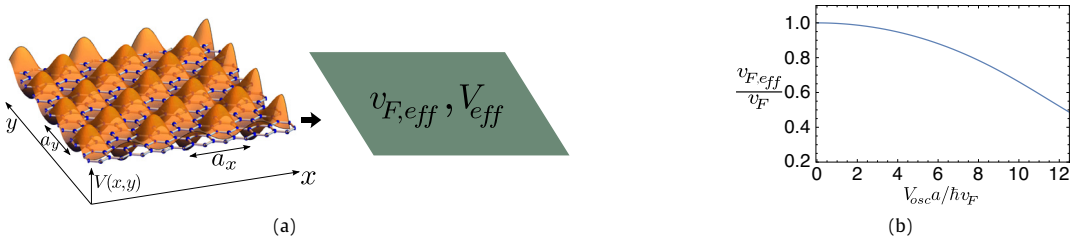


Fig. 2. (a) Graphene superlattice formed by localized scattering centers corresponding to a 2D sinusoidal periodic electrostatic potential in the microscopic model, and the associated continuum platform where all granular details are homogenized. (b) Effective Fermi velocity $v_{F,eff}$ as a function of the peak modulation amplitude V_{osc}.

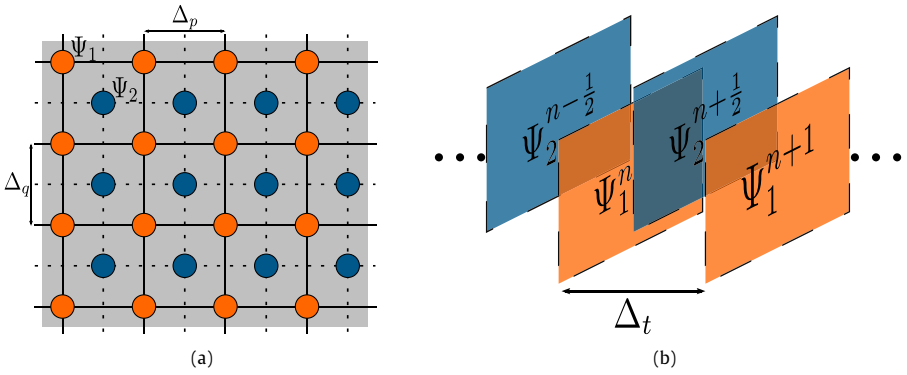


Fig. 3. Geometry of the grid for the discretized two component pseudospinor ψ. (a) Spatial domain. (b) Time domain.

a. This structure is reminiscent of a Moiré graphene superlattice, which is typically obtained when graphene is stacked atop a crystal with a slightly different hexagonal lattice, for instance boron nitride [7–10,31,32]. It should be mentioned that the Hamiltonian of Moiré superlattices also includes σ_z component, which is not featured in our model.

Using the first principles homogenization approach described in Ref. [24] it is possible to determine the parameters χ_{xx} and χ_{yy} as a function of V_{osc}. By symmetry, in this superlattice $\chi_{xx} = \chi_{yy} = \chi$ so that the response is isotropic. Moreover, similar to the anisotropic superlattice, the effective potential is equal to the average potential, i.e., $V_{eff} = V_{av}$. From Eqs. (2)–(3) it is seen that the effect of χ within a continuum approach is equivalent to redefine the Fermi velocity as $v_{F,eff} = v_F\chi$. The explicit dependence of the effective Fermi velocity $v_{F,eff}$ on V_{osc} is shown in Fig. 2(b). As seen, the electrons can be significantly slowed down by the scattering centers. Surprisingly, this happens only for rather large values of the peak normalized microscopic electric potential. Indeed, for moderate values of V_{osc} the electron transport is little affected by the microscopic potential, rather different from the superlattices discussed in the previous subsection.

2.2. The finite-difference time-domain (FDTD) scheme

Here we present a brief review of the FDTD scheme that is used to determine the electron wave propagation in GSLs. This method was proposed and thoroughly discussed in Ref. [15]. The spatial domain is discretized into a given number of nodes that are spread over a rectangular grid. In the time domain, the pseudospinor is sampled with a time step Δ_t. Thus, in the discretized problem the pseudospinor is $\psi(x, y, t) = \psi(p\Delta_p, q\Delta_q, n\Delta_t)$, where Δ_p and Δ_q are the node distances along the x- and y-directions. Since both components of the pseudospinor become coupled in the Dirac equation (1), they are defined in different grids shifted by $\frac{\Delta_p}{2}$, $\frac{\Delta_q}{2}$ and $\frac{\Delta_t}{2}$ (Fig. 3).

The application of this scheme results in a pair of update equations that describe the dynamics of the wave function. These equations are given in Appendix A. The stability of the algorithm is guaranteed provided the time step Δ_t satisfies [15]:

$$\Delta_t < \Delta_{t_{max}} = \frac{1}{v_F} \frac{1}{\sqrt{\frac{\chi_{xx}^2}{\Delta_x^2} + \frac{\chi_{yy}^2}{\Delta_y^2}}}. \tag{4}$$

This approach can be applied both to the microscopic and continuum formalisms, simply by adjusting the input parameters. Specifically, the update equations for the microscopic approach are obtained by replacing the effective potential by the corresponding microscopic potential and by setting $\chi_{xx} = \chi_{yy} = 1$ in all space. The numerical method was extensively validated in [15].

3. Computational methods

3.1. Data dependencies

The starting point of this analysis focuses on the update equations (6) and (7) (in Appendix A) and the objective is to capture the behavior of data dependencies. From the update equations we can identify two types of dependencies, namely in the time and spatial domains. The computation of the wave function in an arbitrary node (p, q) at a given time stamp $(n + 1)$ requires the prior knowledge of a set of nodes localized in time at n and $n + 1/2$. Consequently, when computing the pseudospinor ψ it is first necessary to compute the pseudospinor component Ψ_2 and then Ψ_1. Regarding spatial dependencies, to obtain Ψ_1 at node (p, q), the Ψ_2 neighbor values must be accessed. The pseudospinor Ψ_2 has similar data interconnections.

3.2. Parallelization strategy

In this work we explore the C/OpenCL standards to build the computational methodologies that take advantage of GPU devices to implement and expedite the simulation process. The programming model is based on a host that manages all the work performed concurrently by the OpenCL devices. These devices execute functions (kernels) over an index space (NDRange) which, for the purpose of this work, can be seen as the index space of the matrices that hold the wave function's value. This index space is represented in Fig. 4 and the selected data structure is a *float4* vector data type that stores the complex value of both components of the pseudospinor.

At the hardware level, a scheduling of groups of threads/work-items occurs to execute concurrently sub-blocks of data that occupy the GPU resources (called wavefronts by AMD and warps by Nvidia). This granularity can be expressed under the OpenCL standard by appropriately defining the number of work-groups and work-items (Fig. 4).

From the previous analysis we were able to identify the flow and dependencies between tasks. Consequently, the update equations are performed in two independent kernels and the synchronization between them is guaranteed by the host and OpenCL events that track the execution state of each called kernel. This ensures that intra-kernel parallelism is fully implemented and just limited by the hardware capacity and intrinsic data dependencies. Parallelism between tasks cannot be achieved due to the dependencies between them.

Next, we present an overview of the algorithms that describe our approach. We developed two distinct models: (1) a simulation concurrency model; and (2) a device concurrency model. The goal is to explore different simulation scenarios. The main host routine of both models is identical and consists of a main process that manages the entire workload. It is responsible for querying the computer platform, loading simulation data from the pipeline and creating a host thread for each individual simulation setup. The host threads execute concurrently and are implemented by the POSIX Threads application programming interface [33]. This algorithm is described below (Algorithm 1).

Algorithm 1 Host side: main process

1: Load simulation data
2: Computer platform query
3: Load number of simulations to *NSim variable*
4: ***for*** *th_ctr* $= 1$ *to NSim:* ***do***
5: *Create th_ctr thread with respective simulation data*
6: ***end for***
Ensure: *Wait until all threads have finished*

3.2.1. Simulation concurrency model

This model explores concurrency between simulations and targets the distribution of each independent simulation to a unique OpenCl device. Thus, it enables the utilization of all compute resources from the workstation by providing the same number of simulation setups as hardware resources available.

Each host thread handles the assigned simulation and has access to one OpenCL device (Algorithm 2). Before the OpenCL device is called to compute the simulation it is necessary to allocate the pertinent buffers and compile the respective kernels with the input parameters defined in the simulation setup. The process of storing a sample is designed to minimize the time wasted in memory operations. Therefore, a sample is downloaded from the buffer device to a host buffer and passed to a newly created thread that will save the data on the disk. Thus, this job is executed concurrently with the computation of the pseudospinor and the only stall occurs when the data is downloaded from the device to the host process.

Algorithm 2 Host side: simulation thread

1: Lock an available device
2: Compile the kernels that will execute in the OpenCL device
3: Set the respective kernel arguments
4: Create and initialize host side buffers: **psiH**, **vH** and **chiH** {**psiH** - buffer that stores the wave function information; **vH** - buffer that stores the electrostatic potential associated with the structure; **chiH** - buffer that stores the anisotropy parameter associated with the lattice (this only applies to the effective medium model routine).}
5: Create and initialize device side buffers: **psiD**, **vD** and **chiD**
6: **for** $n = 1$ to N_{TI}: **do**{N_{TI} variable gives the number of time iterations}
7: Submit **fdtdPsi2** kernel {Compute over the $N_p * N_q$ index space and updates the pseudospinor component Ψ_2; N_p stores the number of nodes in the p dimension; N_q stores the number of nodes in the q dimension}
Ensure: Wait until **fdtdPsi2** kernel is finished
8: Submit **fdtdPsi1** kernel {Compute over the $N_p * N_q$ index space and updates the pseudospinor component Ψ_1}
Ensure: Wait until **fdtdPsi1** kernel is finished
9: **if** $n ==$ *SampleTI* {*SampleTI* variable holds the iteration values for the pseudospinor that will be stored on disk} **then**
10: Upload the wave function information from device to host
Ensure: Wait until device-host data transfers are complete
11: Create *iSample_ctr* thread with respective wave function data {This thread is launched concurrently to store data on disk}
12: **end if**
13: **end for**
Ensure: Wait until sample store threads are finished
14: Release resources and unlock device

The OpenCL device is called in steps 7 and 8 of Algorithm 2 and executes the kernels over the defined index space $N_p \times N_q$. The kernels' structure is described in Algorithm 3. The **fdtdPsi2** kernel implements Eq. (6) while **fdtdPsi1** kernel is the application of Eq. (7). The synchronization process shown after steps 7 and 8 in Algorithm 2 is accomplished by assigning to each called kernel an OpenCL event object and ensuring that the next kernel is executed only when the previous one has finished.

Algorithm 3 Device side: kernel instance

1: Get index space identifier {Gives the information of the node location in the pseudospinor matrix}
2: Get node potential from the **vD** buffer allocated in the global memory
3: Download information of the node pseudospinor value and neighbors from the **psiD** buffer allocated in the global memory
4: Compute node value {This update is given by the respective update equation in the A}
5: Store node value in **psiD** buffer allocated in the global memory

3.2.2. Device concurrency model

This approach is designed to target multi-GPU platforms such that each simulation has access to all available GPU devices (N_d) and the work load is distributed between them. The idea is to split the pseudospinor matrix into smaller blocks and compute each one in a different device.

This method requires the division and distribution of N_p/N_d chunks of the wave function matrix by the corresponding devices (Fig. 5). The length of each block (in the p direction) needs to be larger than its normal length to cope with the nature of the accessed nodes (the computation of the pseudospinor component Ψ_1 inquires nodes located at $p - 1$ and Ψ_2 examines nodes at

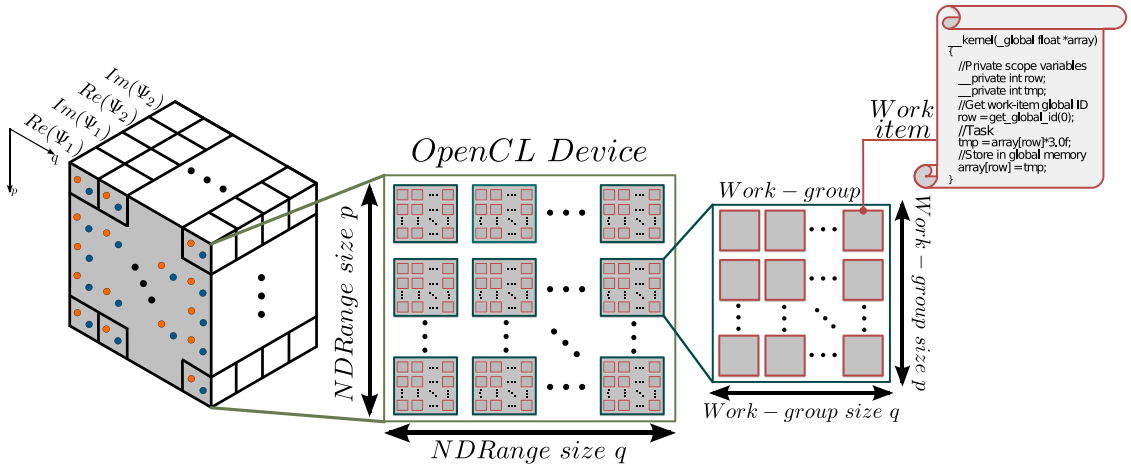


Fig. 4. Representation of the data structure that stores each component of the pseudospinor mapped under the OpenCL index space.

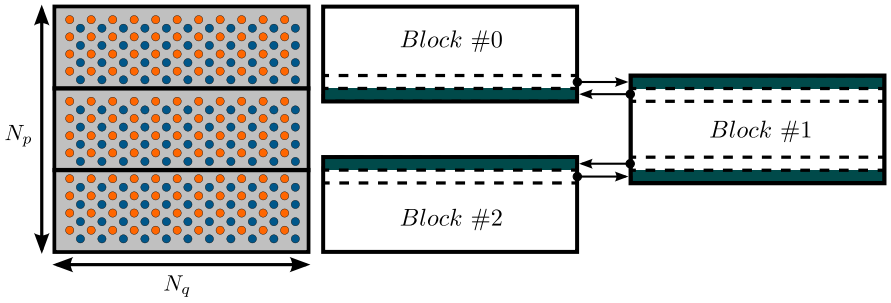


Fig. 5. Division of the Ψ matrix into N_p/N_d chunks (illustrated for $N_d = 3$). The green spaces represent the extra lines of each buffer necessary to cope with the nature of node accesses. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

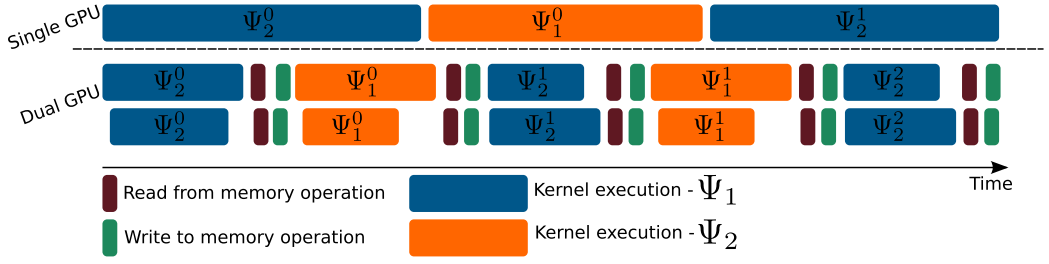


Fig. 6. Time profile of the single and multi-GPU simulation. In the dual-GPU model, tasks are executed concurrently on different devices and synchronized periodically at each time iteration.

$p + 1$). Also, buffer synchronization between devices is required after each kernel is executed in order to keep consistency between buffers. This approach is described in Algorithm 4. The followed approach allows the reutilization of kernels designed to single-GPU implementations. This methodology is enabled by using the host to implement the required control to handle multi-GPU environments. Therefore, the structure of the kernels in steps 8 and 12 of Algorithm 4 is identical to the one in Algorithm 3, thus, showing the flexibility of the proposed solution. To highlight the differences between the single and the multi-GPU implementations, it is shown in Fig. 6 a time profile for the execution of both applications. In a multi-GPU environment (using two different GPUs), the communication between buffers imposes delays in the continuous computation of the pseudospinor. However, the update process is performed much faster when compared to the single-GPU model, which results in an improvement of the overall throughput performance.

4. Simulation results

Next, we apply the proposed methodologies to study the propagation of electronic states in superlattices with localized scattering centers (see Section 2.1.2). In the simulations we consider an unbounded superlattice characterized by an oscillating microscopic potential with normalized amplitude of $V_{osc}a/\hbar v_F = 7.0$. In the continuum model the effective Fermi velocity, $v_{F,eff} = v_F\chi$, equals $0.836v_F$. The initial Gaussian electronic state is of the form:

$$\psi(x, y, t = 0) = \begin{pmatrix} 1 \\ \frac{\hbar v_{F,eff}}{E_0}(k_{x0} + i\chi k_{y0}) \end{pmatrix} e^{i(k_{x0}x + k_{y0}y)} e^{-\frac{x^2+y^2}{R_G^2}}, \quad (5)$$

with $E_0a/\hbar v_{F,eff} = 0.001$, $k_{x0}a = 0.001$ and $k_{y0}a = 0$. The initial state propagates along the x-direction and its characteristic spatial width is determined by the parameter R_G. The results for $R_G = 10a$ and $R_G = 0.5a$ are shown in Fig. 7(a)–7(b) and 7(c)–7(d), respectively.

Algorithm 4 Host side: Simulation thread

1: Lock N_d devices {N_d-Number of available devices}
2: Create and initialize host side buffers: **psiH**, **vH** and **chiH** {**psiH** - buffer that stores the wave function information; **vH** - buffer that stores the electrostatic potential associated with the structure; **chiH** - buffer that stores the anisotropy parameter associated with the lattice (this only applies to the effective medium model routine).}
3: **for** *device_ctr* $= 1$ to N_d: **do**
4: Create and initialize device side buffers: **psiD[***device_ctr***]**, **vD[***device_ctr***]** and **chiD[***device_ctr***]**
5: **end for**
6: **for** $n = 1$ to N_{TI}: **do**{N_{TI} variable gives the number of time iterations}
7: **for** *device_ctr* $= 1$ to N_d: **do**
8: Launch **fdtdPsi2** kernel on device[*device_ctr*]
9: **end for**
10: Buffer synchronization between devices.
11: **for** *device_ctr* $= 1$ to N_d: **do**
12: Launch **fdtdPsi1** kernel on device[*device_ctr*]
13: **end for**
14: Buffer synchronization between devices.
15: **if** $n ==$ *SampleTI* {*SampleTI* variable holds the iteration values for the pseudospinor that will be stored on disk} **then**
16: Upload the wave function information from devices to host
Ensure: Wait until device-host data transfers are complete
17: Create *iSample_ctr* thread with respective wave function data {This thread is launched concurrently to store data on disk}
18: **end if**
19: **end for**
Ensure: Wait until sample store threads are finished
20: Release resources and unlock N_d devices

The microscopic wave function ψ_{mic} has strong fluctuations on the scale of the period of the superlattice. To filter out these spatial oscillations, we represent in Fig. 7(a) the spatially averaged probability density function $|\psi_{mic}|^2_{av}(x, y) = \frac{1}{a}\int_{-a/2}^{a/2} |\psi_{mic}(x + x', y)|^2 dx'$ for $R_G = 10a$. This approach provides a better visualization of the envelope of the wave function calculated with the microscopic theory. Fig. 7(b) shows the same plots but without spatial averaging. Importantly, the results reveal a good agreement between the microscopic and the macroscopic formalisms when the initial state width is larger than the period of the superlattice. In contrast, the results depicted in Fig. 7(c)–(d) for an R_G smaller than the period a, reveal significant differences between the microscopic and continuum approaches (note the different vertical scales in the figures). This is consistent with the fact that the continuum approximation requires that the wave function varies slowly on the scale of the unit cell [24]. To sum up, the numerical results of Fig. 7 reveal that the effective Hamiltonian can only accurately describe the time evolution of an initial wave packet when the characteristic width of the initial state is not more localized than the spatial period of the superlattice. Similar conclusions were drawn in Ref. [25] for the case of anisotropic graphene superlattices.

To demonstrate the effect of the localized scattering centers on the electron transport, we show in Fig. 8(bi)–(bii) a snapshot of the probability density function at the fixed time $t = t_2$ calculated with the microscopic and effective medium approaches with $R_G = 10a$. We contrast these results with the propagation of the same initial state in pristine graphene, see Fig. 8(aii). Noticeably, the electron wave is somewhat slowed down by the scattering centers, but not significantly considering the rather strong modulation of the electric potential. More precisely, the electrons propagate in the superlattice with a velocity $0.836v_F$ (i.e., about 84% the velocity in pristine graphene), which is consistent with the theoretical value obtained from the continuum approximation.

Table 1
Apparatus considered in the simulations. In the AMD Platform, GPU 0 is an AMD Radeon R9 280x and GPU 1 is an AMD Radeon R9 390. In the Nvidia Platform, GPU 0 is an Nvidia Geforce GTX Titan, GPU 1 is an Nvidia Tesla k40c and GPU 2 is an Nvidia Geforce Titan X.

	AMD platform		Nvidia platform		
GPU Setup	GPU 0	GPU 1	GPU 0	GPU 1	GPU 2
Cores	2048	2560	2688	2880	3072
Base clock (MHz)	850	1000	837	745	1000
Memory (GB)	3	8	6	12	12
Memory Bandwidth (GB/s)	288	384	288	288	336
Single-Precision (FLOPS)	4096	5120	4500	4291	6144
Double-Precision (FLOPS)	1024	640	1500	1430	192
CPU setup	Intel(R) Core(TM) i7 950 3.07GHz		Intel(R) Core(TM) i7-4790k 4 GHz		
OS	CentOS Linux Release 7.2.1511		CentOS Linux Release 6.7		
OpenCL version	OpenCL 2.0		OpenCL 1.2		

It is interesting to highlight the differences between this superlattice and the anisotropic superlattice discussed in Section 2.1.1. To this end, we studied the propagation of an initial state with $R_G = 10a$ in an anisotropic superlattice characterized by the electric potential $V_{osc}a/\hbar v_F = 7.0$, for which $\chi = 0.093$ in the continuum model. The results are shown in Fig. 8(ci)–(cii) and reveal that not only the (x-component of the) electrons velocity is unaffected by the microscopic electric potential, i.e., the electrons propagate with velocity v_F due to the Klein tunneling effect, but also the matter wave experiences a negligible lateral spreading (the y-component of the velocity is near zero). Finally, it is important to underline the excellent agreement between the continuum and microscopic theories in the anisotropic superlattice case, which further validates the effective medium model.

Performance metrics of these simulations are discussed in Section 5 to evaluate the gains of our method.

5. Throughput performance evaluation

Next, a comprehensive analysis of the performance of the devised solution is disclosed. The apparatus used in the performance tests is summarized in Table 1.

These platforms are intentionally different to demonstrate portability and also to analyze the impact of the devised solution in terms of synchronization and throughput performance for generalized contexts.

The performance tests were based on the time evolution problem reported in the previous section, and the only varying input parameters are the dimensions of the matrices ($N_p \times N_q$) that store the pseudospinor values. The other parameters remain unchanged as they do not influence the performance metrics and can be found in Table 2.

5.1. Multi-core CPU and many-core GPU metrics

This section discusses the throughput performance gains that GPUs present when compared with CPU architectures. The CPU implementation is the basis for this benchmark and is a pseudo-sequential implementation of the OpenCL code that also runs on the GPU. The CPU version is executed on an Intel Core i7 950 that has four cores and uses all eight available threads to perform the computations. The execution times obtained using the three different architectures are indicated in Table 3 and Fig. 9 shows the overall speedup ratios.

The obtained results are a good indicator that GPU architectures are suitable to compute this type of simulations and, if properly

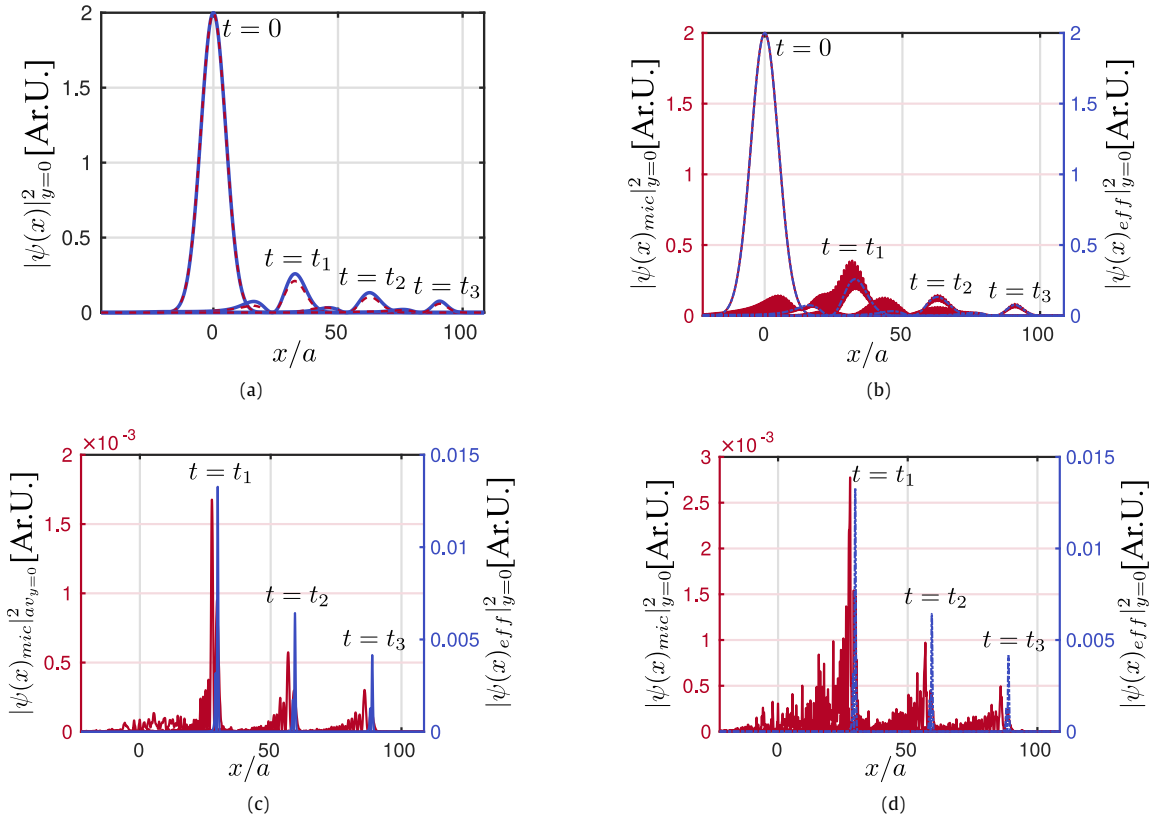


Fig. 7. (a) Longitudinal profiles (along the direction of propagation) of the probability density function sampled at $t = 0$, $t = t_1 = 4000\Delta_t$, $t = t_2 = 8000\Delta_t$ and $t = t_3 = 12\,000\Delta_t$ for the continuum (blue solid line) and microscopic approaches (red dotted line) for an initial electronic state with $R_G = 10a$. (b) Similar to (a) but without the application of a spatial filter. (c) and (d) Similar to (a) and (b) but for an initial electronic state with $R_G = 0.5a$. The time step is $\Delta_t = 0.5\Delta_{t_{max}}$, with $v_{F,eff} = 0.836v_F$ and $\Delta_x = \Delta_y = a/40$.

Table 2
Relevant parameters used in all the performance tests.

Time iterations	Δ_x/a	Δ_y/a	$\Delta_t v_F/a$	$E_0 a/\hbar v_{F,eff}$	$k_{x0}a$	$k_{y0}a$	R_G/a
12 000	0.025	0.025	0.0088	0.001	0.001	0.0	10

Table 3
Execution time in seconds obtained with the three different architectures. The CPU execution time was acquired on an Intel Core i7 950 running on the AMD platform. The AMD GPU is a Radeon R9 280X and the Nvidia GPU is a Tesla k40c.

Matrix Dim $N_p \times N_q$	Intel CPU	AMD GPU 0	Nvidia GPU 1
2050×2050	719.36	25.30	28.71
3074×3074	1770.06	56.54	62.39
4098×4098	5171.67	101.48	113.84
6146×6146	16968.17	223.76	259.76

programmed, can provide significant acceleration, demonstrated by the achieved speedups that can reach up to 75x. The execution times and overall speedups also reveal that the nature of these problems is well suited to compute resources that combine a raw amount of processing power rather than highly complex and optimized cores such as the CPU.

5.1.1. Single-Precision vs Double-Precision throughput performance

It is relevant to discuss the difference between using single and double-precision data representation and arithmetic. In this regard, normalizing some of the physical constants (e.g. the reduced Planck's constant $\hbar$, Fermi velocity v_F, among others) allowed us to use single-precision format and arithmetic. Consequently, the simulation results can be correctly represented by the accuracy

Table 4
Execution time in seconds obtained with the same GPU for single-precision and double-precision. Nvidia GPU is a Tesla k40c.

Matrix Dim $N_p \times N_q$	Single-Precision Nvidia GPU 1	Double-Precision Nvidia GPU 1
6146×6146	259.76	544.82

provided by the single-precision format. This approach drastically reduces the amount of memory required to run the simulator and at the same time obtain significant speedup ratios as compared to the double-precision arithmetic. As shown in Table 1, GPUs usually offer higher single-precision throughout performance. In particular, the double-precision performance of the available GPUs in both platforms is less than half of the single-precision performance. To demonstrate these differences we show in Table 4 the execution times for the Nvidia Tesla k40c GPU. The performance gap between the two solutions is evident by the speedup factor of 2×.

5.1.2. GPU results on AMD architecture

In order to extract even more processing power from parallel computing architectures, we studied the possibility of increasing throughput performance gains by using multi-GPU platforms. Table 5 shows the performance metrics for the single and dual-GPU configuration for several matrix dimensions. The results show

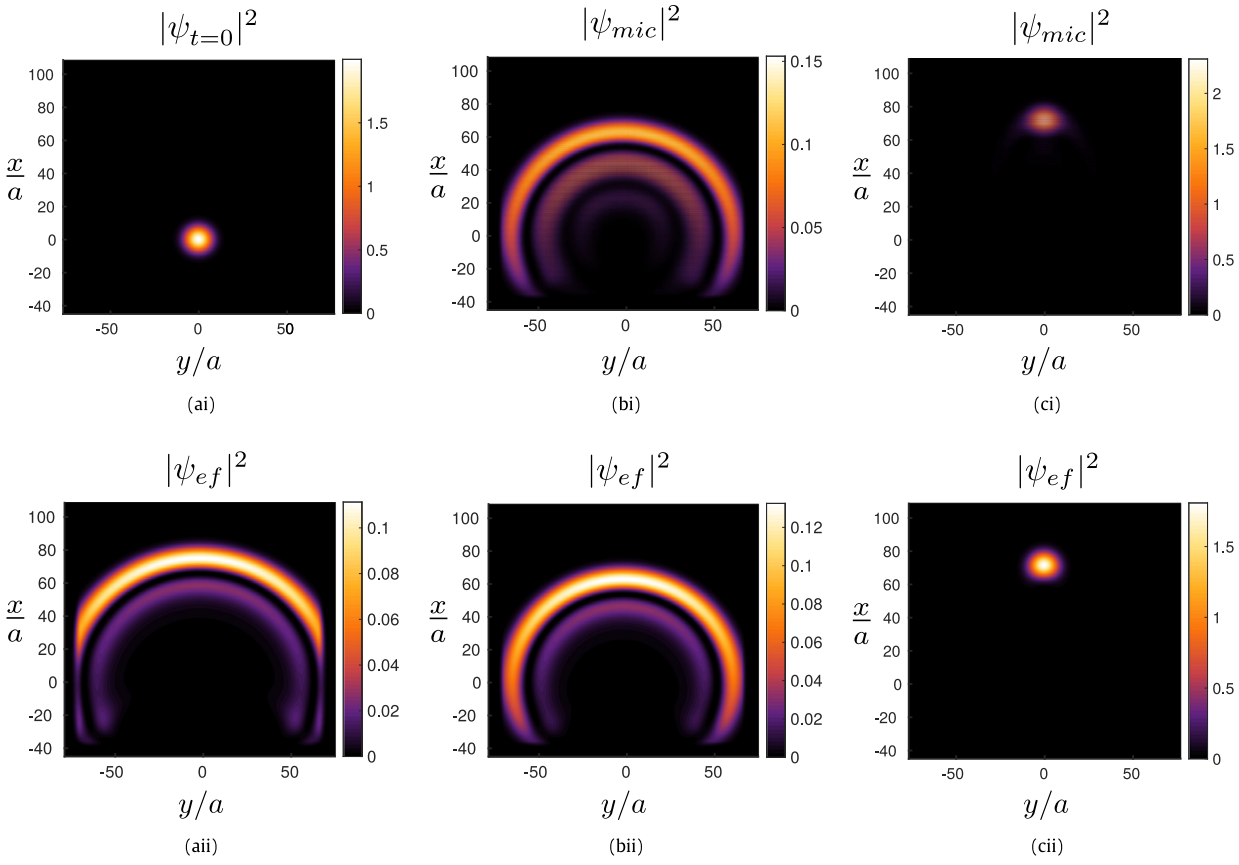


Fig. 8. (ai) Initial electronic state with $R_G = 10a$. (aii) Density plot of $|\psi|^2$ at $t = t_2$ for pristine graphene (here the microscopic and continuum formalisms give coincident results). (bi) and (bii) Density plots of $|\psi|^2$ calculated with the microscopic and continuum formalisms at $t = t_2$, respectively, for a superlattice with localized scattering centers. (ci) and (cii) Similar to (bi) and (bii) but for an anisotropic GSL. The color map of this figure was obtained by using the *othercolor* function from *MATLAB Central File Exchange* [34] which is included in the simulator. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

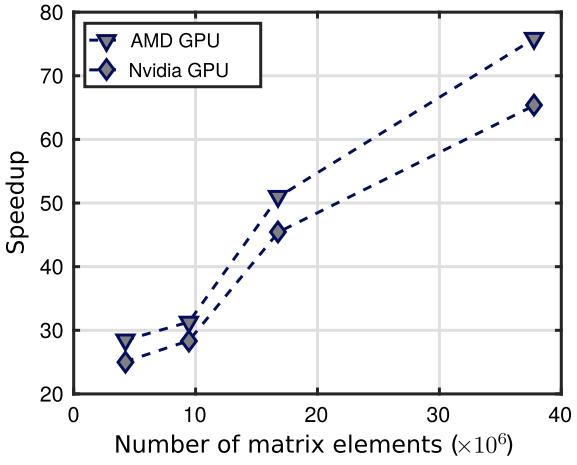


Fig. 9. Speedups obtained by the GPUs compared against the CPU. The line with triangles displays the AMD R9 280x results and the line with diamonds shows the Nvidia Tesla k40c speedups. Each symbol of the figure corresponds to an entry in Table 3.

Table 5
Computation times (seconds) obtained by the AMD devices. Single-GPU metrics were obtained using the AMD Radeon R9 280X.

Matrix Dim $N_p \times N_q$	Single-GPU GPU 0	Dual-GPU GPUs 0 &1
2050×2050	25.30	22.31
3074×3074	56.54	43.65
4098×4098	101.48	65.35
6146×6146	223.76	129.58
8194×8194	393.28	225.43

Table 6
Computation times (seconds) obtained by the Nvidia platform. Single-GPU metrics were obtained by the Nvidia Tesla k40c. Dual-GPU metrics were obtained by the Tesla k40c and GeForce GTX Titan X.

Matrix Dim $N_p \times N_q$	Single-GPU GPU 1	Dual-GPU GPUs 1 & 2	Triple-GPU GPUs 0 & 1 & 2
3074×3074	62.39	37.23	29.17
6146×6146	259.76	137.61	102.41
12290×12290	1056.13	547.71	378.15
18434×18434	2434.48	1246.09	857.32

that performance metrics are directly dependent on the matrix dimensions of the structure.

Based on the results depicted in Fig. 10 it is clear that we can achieve additional speedups by scaling the problem dimensions to a multi-GPU environment. For a small number of matrix elements the performance improvements are not significant, as the available computational resources in all the GPUs are underutilized. By increasing the matrix dimensions, the resources are effectively exploited and a throughput performance increase is obtained, which for this particular platform can reach 1.8x when compared with single-GPU execution.

5.1.3. GPU results on Nvidia architecture

Next, we present the performance metrics for a workstation with three GPUs. The results are shown in Table 6.

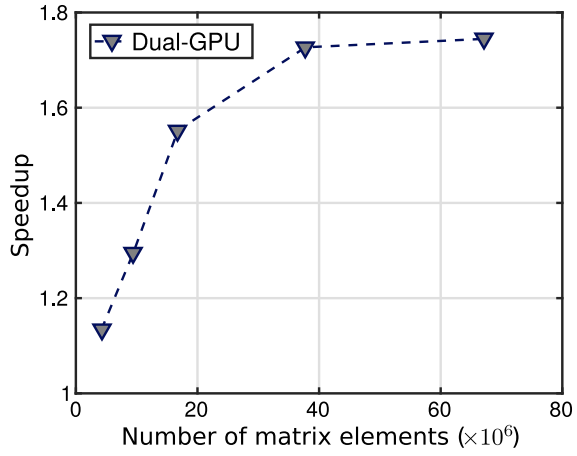


Fig. 10. Speedups obtained by the dual-GPU compared with the single-GPU version. Each symbol of the figure corresponds to an entry in Table 5.

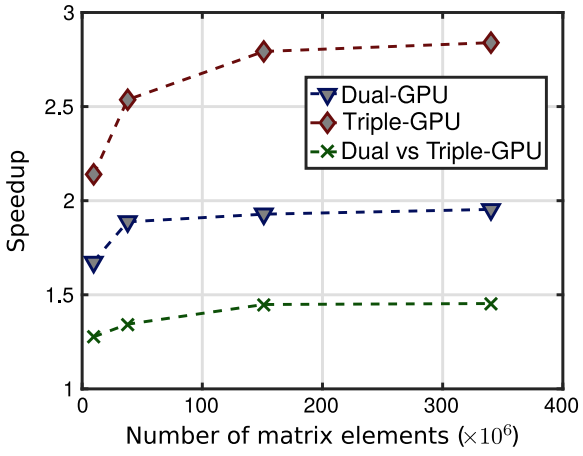


Fig. 11. Speedup obtained by the Nvidia Platform. The red line with diamonds is the speedup of the triple-GPU against single-GPU. The blue line with triangles shows the speedup of the dual-GPU against single-GPU and the green line with the x-marks is the speedup of the triple-GPU against dual-GPU. Each symbol of the figure corresponds to an entry in Table 6. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 11 shows the performance gains obtained using a triple-GPU environment. The results demonstrate the scalability of the devised solution. Comparing with a single GPU implementation it is shown that the triple-GPU version outperforms all the tested scenarios, resulting in speedups that can be as high as 2.7x. Another important feature of a multi-GPU workstation is the possibility of simulating large structures, which with a single-GPU would perhaps be unattainable due to memory size limitations. Furthermore, the multi-GPU approach is designed to automatically distribute the workload by all available devices and prepared to use as much GPU resources as available, thus, proving to be a scalable solution.

6. Conclusions

We proposed novel computational tools to efficiently and rapidly simulate the propagation of electron waves in GSLs based on the two-dimensional massless Dirac equation or alternatively using an effective Hamiltonian. In particular, we investigated the propagation of electron waves in "isotropic" graphene superlattices formed by localized scattering centers. Surprisingly, our results show that for moderate values of the peak microscopic electric potential the effective Fermi velocity remains comparable to the Fermi velocity in pristine graphene. In contrast, for anisotropic superlattices the Fermi velocity along the y-direction can be strongly dependent on the peak amplitude of the microscopic potential and be near zero. These results highlight that the *symmetry* of the microscopic electric potential can have a significant impact on the electron transport properties, and be more influential than the strength of the potential modulation. It was confirmed that the continuum approximation can only be used to characterize the wave propagation in GSLs provided the characteristic width of the initial state is larger than the period of the superlattice. Furthermore, we showed substantial speedups in terms of simulation times, confirming that GPU together with the OpenCL framework is a powerful tool that can offer impressive accelerated execution times. In addition, the developed framework provides portability and scalability to the devised solutions.

Acknowledgments

This work was funded by Fundação para a Ciência e a Tecnologia and Instituto de Telecomunicações under projects PTDC/EEITEL/2764/2012, PTDC/EEI-TEL/4543/2014 and UID/EEA/50008/2013. D.E. Fernandes acknowledges support by Fundação para a Ciência e a Tecnologia, Programa Operacional Capital Humano/POCH, and the cofinancing of Fundo Social Europeu and MCTES under the fellowship SFRH/BPD/116525/2016.

Appendix A. FDTD scheme: Update equations

The time evolution of the pseudospinor within the effective medium framework is given by the following pair of update equations:

$$\begin{aligned}
&\Psi^{n+1}_{1,p,q}\left(1-\frac{V_{p,q}}{2\hbar i}\Delta_t\right)=\Psi^{n}_{1,p,q}\left(1+\frac{V_{p,q}}{2\hbar i}\Delta_t\right)+\\
&\quad-v_{F,eff_{p,q}}\Delta_t\left[\left(\frac{1}{2\Delta_x}-i\frac{\chi_{p,q}+\chi_{p+\frac{1}{2},q+\frac{1}{2}}}{4\Delta_y}\right)\Psi^{n+\frac{1}{2}}_{2,p+\frac{1}{2},q+\frac{1}{2}}+\right.\\
&\qquad-\left(\frac{1}{2\Delta_x}+i\frac{\chi_{p,q}+\chi_{p-\frac{1}{2},q+\frac{1}{2}}}{4\Delta_y}\right)\Psi^{n+\frac{1}{2}}_{2,p-\frac{1}{2},q+\frac{1}{2}}+\\
&\qquad+\left(\frac{1}{2\Delta_x}+i\frac{\chi_{p,q}+\chi_{p+\frac{1}{2},q-\frac{1}{2}}}{4\Delta_y}\right)\Psi^{n+\frac{1}{2}}_{2,p+\frac{1}{2},q-\frac{1}{2}}+\\
&\qquad\left.-\left(\frac{1}{2\Delta_x}-i\frac{\chi_{p,q}+\chi_{p-\frac{1}{2},q-\frac{1}{2}}}{4\Delta_y}\right)\Psi^{n+\frac{1}{2}}_{2,p-\frac{1}{2},q-\frac{1}{2}}\right],
\end{aligned}\tag{6}$$

$$\begin{aligned}
&\Psi^{n+\frac{1}{2}}_{2,p+\frac{1}{2},q+\frac{1}{2}}\left(1-\frac{V_{p+\frac{1}{2},q+\frac{1}{2}}}{2\hbar i}\Delta_t\right)\\
&\quad=\Psi^{n-\frac{1}{2}}_{2,p+\frac{1}{2},q+\frac{1}{2}}\left(1+\frac{V_{p+\frac{1}{2},q+\frac{1}{2}}}{2\hbar i}\Delta_t\right)+\\
&\quad-v_{F,eff_{p+\frac{1}{2},q+\frac{1}{2}}}\Delta_t\left[\left(\frac{1}{2\Delta_x}+i\frac{\chi_{p+\frac{1}{2},q+\frac{1}{2}}+\chi_{p+1,q+1}}{4\Delta_y}\right)\Psi^{n}_{1,p+1,q+1}\right.\\
&\qquad+-\left(\frac{1}{2\Delta_x}-i\frac{\chi_{p+\frac{1}{2},q+\frac{1}{2}}+\chi_{p,q+1}}{4\Delta_y}\right)\Psi^{n}_{1,p,q+1}+\\
&\qquad+\left(\frac{1}{2\Delta_x}-i\frac{\chi_{p+\frac{1}{2},q+\frac{1}{2}}+\chi_{p+1,q}}{4\Delta_y}\right)\Psi^{n}_{1,p+1,q}+\\
&\qquad\left.-\left(\frac{1}{2\Delta_x}+i\frac{\chi_{p+\frac{1}{2},q+\frac{1}{2}}+\chi_{p,q}}{4\Delta_y}\right)\Psi^{n}_{1,p,q}\right].
\end{aligned}\tag{7}$$

For an anisotropic superlattice $v_{F,eff}=v_F$, and for a superlattice with localized scattering centers $\chi=1$.

References

[1] M.P. Levendorf, C.-J. Kim, L. Brown, P.Y. Huang, R.W. Havener, D.A. Muller, J. Park, Nature 488 (2012) 627–632.
[2] C. Dean, A. Young, L. Wang, I. Meric, G.-H. Lee, K. Watanabe, T. Taniguchi, K. Shepard, P. Kim, J. Hone, Solid State Commun. 152 (2012) 1275–1282.
[3] L.-G. Wang, S.-Y. Zhu, Phys. Rev. B 81 (2010) 205444.
[4] C.-H. Park, L. Yang, Y.-W. Son, M.L. Cohen, S.G. Louie, Phys. Rev. Lett. 101 (2008) 126804.
[5] C.-H. Park, L. Yang, Y.-W. Son, M.L. Cohen, S.G. Louie, Nat. Phys. 4 (2008) 213–217.
[6] C.-H. Park, Y.-W. Son, L. Yang, M.L. Cohen, S.G. Louie, Nano Lett. 8 (2008) 2920–2924.
[7] C.R. Dean, L. Wang, P. Maher, C. Forsythe, F. Ghahari, Y. Gao, J. Katoch, M. Ishigami, P. Moon, M. Koshino, T. Taniguchi, K. Watanabe, K.L. Shepard, J. Hone, P. Kim, Nature 497 (2013) 598–602.
[8] J. Xue, J. Sanchez-Yamagishi, D. Bulmash, P. Jacquod, A. Deshpande, K. Watanabe, T. Taniguchi, P. Jarillo-Herrero, B.J. LeRoy, Nat. Mater. 10 (2011) 282–285.
[9] J.R. Wallbank, A.A. Patel, M. Mucha-Kruczyński, A.K. Geim, V.I. Fal'ko, Phys. Rev. B 87 (2013) 245408.
[10] G.X. Ni, H. Wang, J.S. Wu, Z. Fei, M.D. Goldflam, F. Keilmann, B. Özyilmaz, A.H. Castro Neto, X.M. Xie, M.M. Fogler, D.N. Basov, Nat. Mater. 14 (2015) 1217–1222.
[11] K. Novoselov, A.K. Geim, S. Morozov, D. Jiang, M. Katsnelson, I. Grigorieva, S. Dubonos, A. Firsov, Nature 438 (2005) 197–200.
[12] A.H. Castro Neto, F. Guinea, N.M.R. Peres, K.S. Novoselov, A.K. Geim, Rev. Modern Phys. 81 (2009) 109–162.
[13] A.K. Geim, K.S. Novoselov, Nat. Mater. 6 (2007) 183–191.
[14] M.S. Jang, H. Kim, H.A. Atwater, W.A. Goddard III, Appl. Phys. Lett. 97 (2010) 043504.
[15] D.E. Fernandes, M. Rodrigues, G. Falcão, M.G. Silveirinha, AIP Adv. 6 (2016) 075109.
[16] M.R. Zunoubi, J. Payne, W.P. Roach, IEEE Antennas Wirel. Propag. Lett. 9 (2010) 756–759.
[17] K.H. Lee, I. Ahmed, R.S.M. Goh, E.H. Khoo, E.P. Li, T.G.G. Hung, Progr. Electromagn. Res. 116 (2011) 441–456.
[18] P. Wahl, D.S.L. Gagnon, C. Debaes, J.V. Erps, N. Vermeulen, D.A.B. Miller, H. Thienpont, Progr. Electromagn. Res. 138 (2013) 467–478.
[19] T. Nagaoka, S. Watanabe, Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Boston, MA, 2011, pp. 401–404.
[20] Khronos OpenCL Working Group, The OpenCL Specification, Version: 1.2, Document Revision: 19, 2012.
[21] A. Munshi, Opencl: Parallel Computing on the GPU and CPU, SIGGRAPH, Tutorial.
[22] J.D. Owens, M. Houston, D. Luebke, S. Green, J.E. Stone, J.C. Phillips, Proc. IEEE 96 (2008) 879–899.
[23] J. Nickolls, W.J. Dally, IEEE Micro 30 (2010) 56–69.
[24] M.G. Silveirinha, N. Engheta, Phys. Rev. B 85 (2012) 195413.
[25] D.E. Fernandes, N. Engheta, M.G. Silveirinha, Phys. Rev. B 90 (2014) 041406.
[26] Y.P. Bliokh, V. Freilikher, S. Savel'ev, F. Nori, Phys. Rev. B 79 (2009) 075123.
[27] L. Brey, H.A. Fertig, Phys. Rev. Lett. 103 (2009) 046809.
[28] P. Burset, A.L. Yeyati, L. Brey, H.A. Fertig, Phys. Rev. B 83 (2011) 195434.
[29] S.P. Milovanović, D. Moldovan, F.M. Peeters, J. Appl. Phys. 118 (2015) 154308.
[30] M. Barbier, F.M. Peeters, P. Vasilopoulos, J.M. Pereira, Phys. Rev. B 77 (2008) 115446.
[31] C. Handschin, P. Makk, P. Rickhaus, M.-H. Liu, K. Watanabe, T. Taniguchi, K. Richter, C. Schönenberger, Nano Lett. 17 (2017) 328–333.
[32] P. Moon, M. Koshino, Phys. Rev. B 90 (2014) 155406.
[33] Blaise Barney, POSIX threads Programming, Lawrence Livermore National Laboratory. https://computing.llnl.gov/tutorials/pthreads/.
[34] Copyright (c) 2011, Joshua. All rights reserved. Othercolor, MATLAB Central File Exchange. Retrieved December 18, 2016.