J Comput Electron (2015) 14:922–929
DOI 10.1007/s10825-015-0730-0

Domain decomposition strategies for the two-dimensional Wigner Monte Carlo Method

Josef Weinbub[1] · **Paul Ellinghaus**[1] · **Mihail Nedjalkov**[1]

Published online: 24 July 2015

Abstract A domain decomposition approach for the parallelization of the Wigner Monte Carlo method allows the huge memory requirements to be distributed amongst many computational units, thereby making large multi-dimensional simulations feasible. Two domain decomposition techniques—a uniform slab and uniform block decomposition—are compared and the design and implementation of the block decomposition approach, using the message passing interface, is discussed. The parallel performance of the two approaches is evaluated by simulating a representative physical problem. Our results show that the presumably inferior slab decomposition method is in fact superior to the block decomposition approach, due to the additional overhead incurred by the block decomposition method to set up its communication layer.

Keywords Wigner · Monte Carlo · Domain decomposition · Message passing interface

1 Introduction

The Wigner formalism [1] provides an attractive alternative to the non-equilibrium Green's function formalism, as it provides a reformulation of quantum mechanics—usually defined through operators and wave functions—in the phase space using functions and variables [2]. Thereby, the Wigner formalism gives a more intuitive description, which also facilitates the reuse of many classical concepts and notions. The biggest advantages of using a phase space formulation are that open boundary conditions, transients, and decoherence effects can be considered and are easily implemented from a computational point of view. It is the coherent evolution which presents the computational challenge when solving the Wigner(-Boltzmann) transport equation.

Both stochastic and deterministic methods have been applied to solve the one-dimensional Wigner equation. However, only the Wigner Monte Carlo method, using the signed-particle technique [3,4], has made multi-dimensional Wigner simulations viable thus far [5,6]; a multi-dimensional approach is essential for the simulation of realistic semiconductor devices.

The signed-particle technique is based on the generation of (numerical) particles with + and − signs (weighting), as determined by the Wigner potential, to solve the coherent evolution problem. The number of particles increases exponentially due to particle generation, which occurs at a rate depending on the potential differences present in the spatial domain. This dramatic increase of particles (and the associated computational load) is counteracted by the concept of particle annihilation: particles with opposite signs, within the same cell of the phase space, annihilate each other since they have the same probabilistic future, but their contributions to physical quantities computed from the Wigner function cancel each other. Indeed, it is the annihilation step that has made the signed-particle Wigner Monte Carlo simulations computationally feasible.

The annihilation algorithm [7] requires the entire phase space to be represented by an array whose size is proportional to the dimensionality and resolution of the phase space—this quickly results in exorbitant memory requirements, which

✉ Josef Weinbub
weinbub@iue.tuwien.ac.at

Paul Ellinghaus
ellinghaus@iue.tuwien.ac.at

Mihail Nedjalkov
nedjalkov@iue.tuwien.ac.at

[1] Institute for Microelectronics, TU Wien, Gußhausstraße 27-29/E360, 1040 Vienna, Austria

easily exceed the limited memory of a single workstation [8]. The following (conservative) example clarifies this aspect: a simulation setup is given by a two-dimensional spatial domain of 100 nm × 100 nm with a resolution of $\Delta x = 1$ nm and a three-dimensional k-space with 100 k-values per direction. The associated phase-space grid would therefore consist of $100^2 \times 100^3$ cells, each represented by an integer of (at least) 2 bytes. This would demand a total memory consumption of $\mathcal{O}\left(2^{10}\right)$ bytes, i.e. approximately 20 GB. Each MPI process is assigned to a single CPU core. A supercomputer node typically consists of 16 cores and 32–64 GB of memory; each core has 2–4 GB of memory available, which is insufficient for a domain replication in each process. Although the available node memory is expected to increase in the future, so is the number of computing cores, essentially keeping the memory-per-core-ratio constant. Due to this fact—in addition to the already significant computational demands—an efficient distributed parallel computation approach (limiting the memory demand to a maximum of around 2–4 GB per MPI process) is crucial to facilitate the use of Wigner simulations for realistic, multi-dimensional device structures.

Conventional parallelization approaches for Monte Carlo methods split the particle ensemble amongst computational units, where each subensemble can be treated completely independently and is therefore termed *embarrassingly parallel* [9]. This approach offers excellent parallel efficiency, but necessitates domain replication when working in a distributed-memory context via the message passing interface (MPI)—the de facto standard for large-scale parallel computations—to avoid additional communication. An approach that requires domain replication is not feasible for the Wigner Monte Carlo method due to the huge memory demands associated with the annihilation algorithm. The necessity for domain replication is avoided by a spatial domain-decomposition approach and has been introduced for the Wigner Monte Carlo method in an one-dimensional simulation setting [8] and successfully applied to two-dimensional simulations [10]. The implementations use MPI and are based on the Wigner Ensemble Monte Carlo simulator, which is part of the free open source ViennaWD simulation package [11].

In this work we investigate the feasibility of a block domain decomposition for two-dimensional problems and evaluate its performance relative to the slab domain decomposition approach used in [10] and provide guidance to which domain decomposition approach is best suited for highly memory-intensive, two-dimensional Wigner Monte Carlo quantum simulations. The slab decomposition approach is first summarized in Sect. 2, whereafter the block decomposition approach is introduced in Sect. 3. The parallel efficiency is evaluated based on the execution times of a representative, two-dimensional, physical example in Sect. 4 upon which our conclusions are drawn.

2 Slab decompositioning

The so-called slab or one-dimensional decomposition method partitions the simulation domain in one direction, whereas the second direction (in a two-dimensional setting) is kept untouched (Fig. 1). This method has been successfully applied to one- and two-dimensional Wigner simulations [8, 10].

This slab decomposition approach has two principal limitations: Firstly, the maximum number of subdomains/processes that can be used is limited by the number of mesh points in the (single) direction of decompositioning—each slab has to be at least one mesh cell wide. Secondly, memory intensive applications are more likely to hit a memory-per-process limit with a slab decomposition approach, as only one dimension is partitioned. This memory limitation becomes more relevant for three-dimensional simulation cases, due to the drastic increase in memory demands. Despite these limitations, the slab decomposition approach has proven to be an attractive domain decomposition technique providing excellent parallel scalability for two-dimensional problems [10]. This especially holds true in cases where the decompositioning is aligned with the (dominant) direction of particle propagation, meaning that the majority of particles propagate in the unpartitioned direction. This minimizes the amount of data to be communicated, which is advantageous to parallel scalability.

To enable particles to seamlessly propagate through the decomposed spatial domain, a communication layer is required to enable the transfer of incoming and outgoing particles at every time step between neighboring subdomains

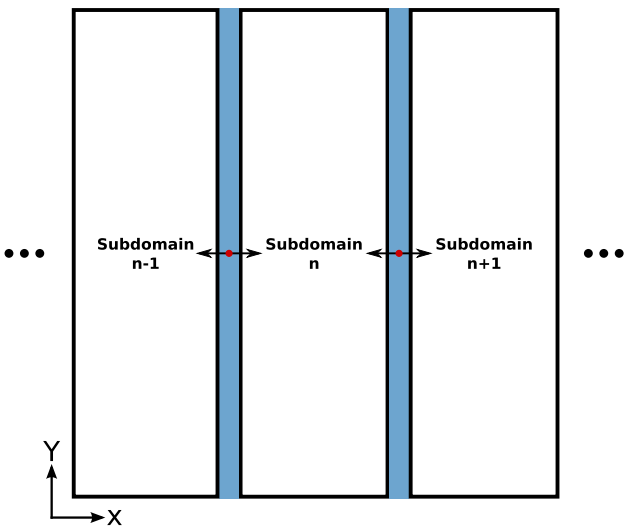

Fig. 1 A uniform slab decomposition approach uses overlap areas (*blue*) to identify and transmit cross-subdomain (*red*) particles. n refers to the MPI rank, which also identifies the subdomain (Color figure online)

(each assigned to one MPI process). A slab decomposition, using N subdomains, requires T_{slab} MPI communication channels (i.e. send and receive operations), to be established during each time step and is given by $T_{slab} = 4 \cdot (N - 1)$; each interior subdomain has to deal with two incoming and two outgoing connections and the two boundary subdomains have to handle only one incoming and one outgoing connection each. Utilizing, for instance, 32 MPI processes ($N = 32$) requires a total number of $T_{slab} = 124$ point-to-point communications for each time step. Although these communication channels are not collective communications (i.e. $N \times N$)—each MPI process only has to communicate with its immediate neighbors and not all other processes—the overhead introduced to first establish these communication channels is unavoidable and potentially limits scalability. The impact of the communication setup overhead is felt even more, if the transmitted data volume is small as the communication is primarily limited by latency instead of bandwidth.

3 Block decompositioning

The logical next step to advance a slab decomposition approach is to also partition the second dimension, extending the method to a block or two-dimensional decomposition approach. A block decomposition approach promises an improved per-process communication volume over a slab decomposition approach [12], making it an interesting candidate for a parallel signed particle Wigner Monte Carlo method.

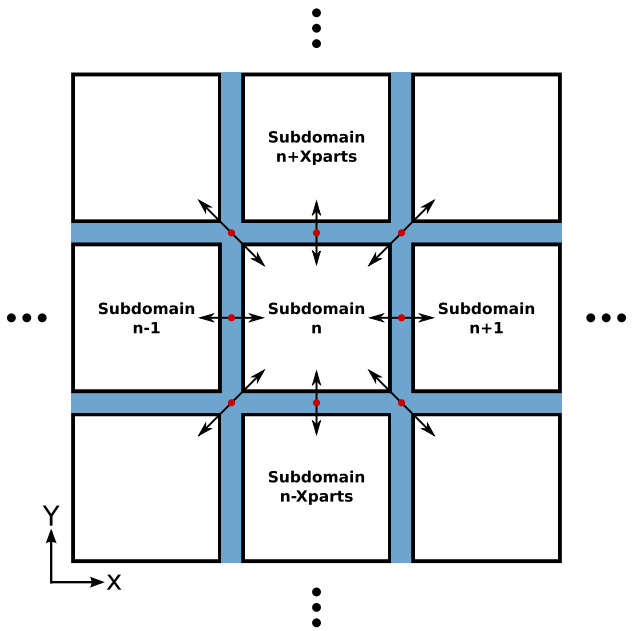

Fig. 2 A uniform block decomposition approach uses overlap areas (*blue*) to identify and transmit cross-subdomain (*red*) particles. n refers to the subdomain identifier, i.e., the MPI process rank, and Xparts denotes the number of subdomains in x-direction (Color figure online)

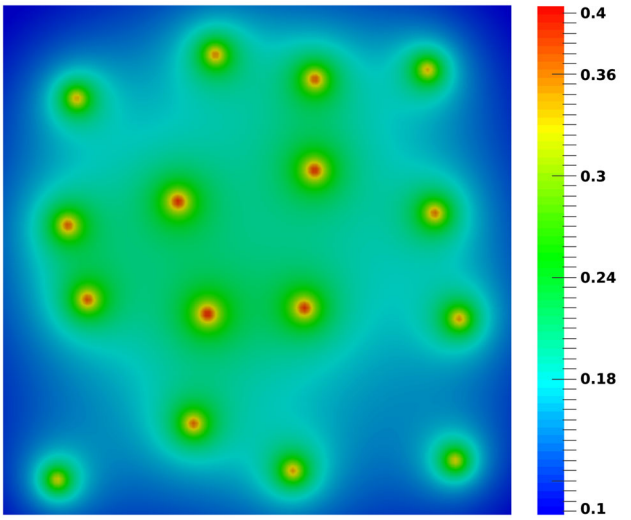

Fig. 3 Sixteen (non-identical) acceptor dopants forming a potential profile with peaks between 0.3 and 0.4 eV

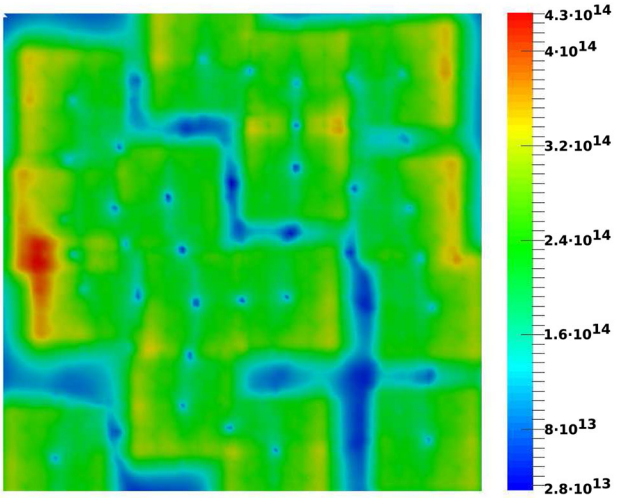

Fig. 4 The particle generation rate (γ [s^{-1}]) of the associated potential profile (Fig. 3). The generation is concentrated around the placed dopants

The partitioning of the second dimension requires additional logic to handle the communication for the exchange of particles moving between the subdomains. Aside from the obvious communication channels in the cardinal directions, the movement of particles in the diagonal directions must also be accounted for (Fig. 2).

The decomposition scheme assigns the rank (process ID) of the MPI processes in an incremental fashion, starting from the bottom left, increasing from left to right and from bottom to top. Due to this specific assignment scheme, each process can identify its direct neighbors by using its own rank and the number of subdomains in the x- and/or y-direction. For

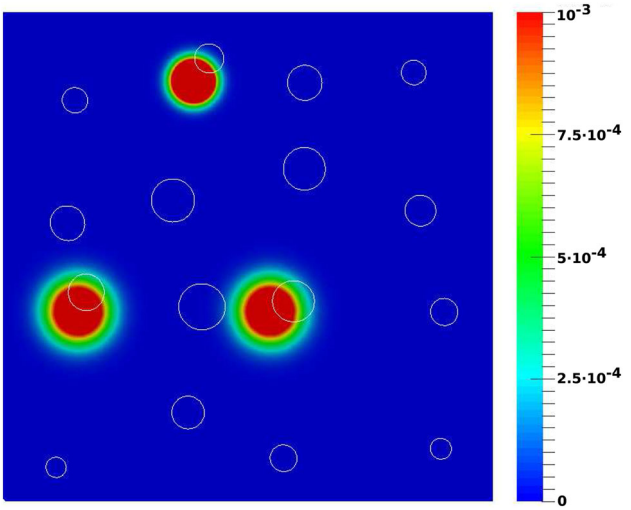

Fig. 5 Normalized density (i.e. probability) at 0 fs, showing three wave packets located in the *upper left* area, serving as the initial condition. *White circles* denote locations of dopants

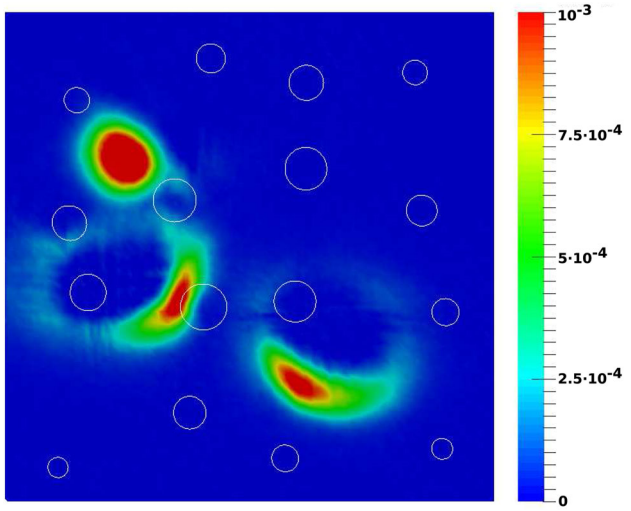

Fig. 7 Normalized density (i.e. probability) at 50 fs, showing a significant change in the shape of the wave packet; interference patterns become evident. *White circles* denote locations of dopants

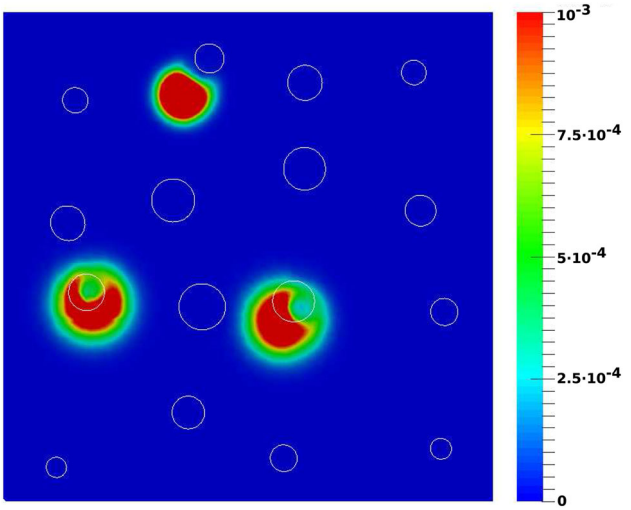

Fig. 6 Normalized density (i.e. probability) at 10 fs, showing the wave packets being warped by the (non-local) influence of the acceptor dopants. *White circles* denote locations of dopants

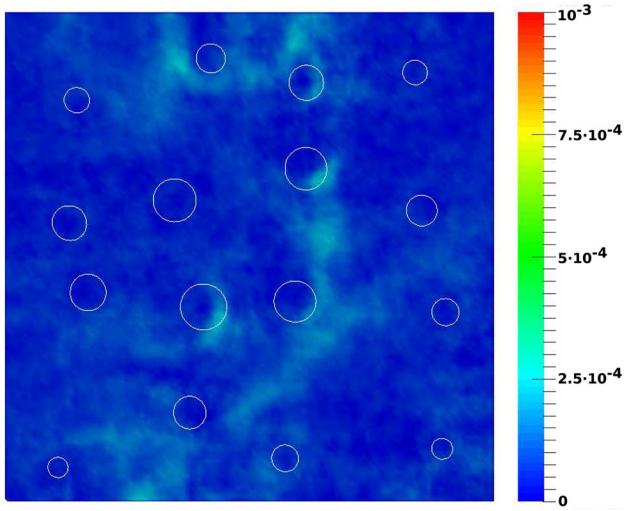

Fig. 8 Normalized density (i.e. probability) at 300 fs, showing a complete diffusion of the wave packets across the entire domain. *White circles* denote locations of dopants

instance, the top neighbor MPI process can be identified by adding the number of subdomains in the x-direction (Xparts) to the rank of the considered process. The other neighbors can be computed analogously.

After each time step, every MPI process evaluates all of its *transfer* boundaries, i.e., boundaries which are in the interior of the simulation domain and require incoming and outgoing communication to facilitate particle movement between processes. Each MPI process identifies particles in its particle subset which will leave its subdomain in the next time step(s): particles must be located within a predetermined *transfer overlap* area (blue region in Fig. 2) and have a momentum directed towards a neighboring subdomain. Every boundary of the subdomain is checked by each MPI process: four transfer boundaries for the interior subdomains, three at the vertical or horizontal domain boundaries, and two at the corner subdomains. Non-blocking point-to-point communications are used to potentially overlap communication with computation for increased efficiency.

A block decomposition, using N subdomains, requires T_{block} MPI communication channels to be established during each time step: $T_{block} = 16 \cdot N_{inner} + 10 \cdot N_{bnd} + 6 \cdot N_{corner}$; each interior subdomain ($inner$) has to deal with a total of 16 connections (4 in cardinal and 4 in diagonal directions for both send and receive operations), each domain boundary subdomain (bnd) has to handle a total of 10 connections

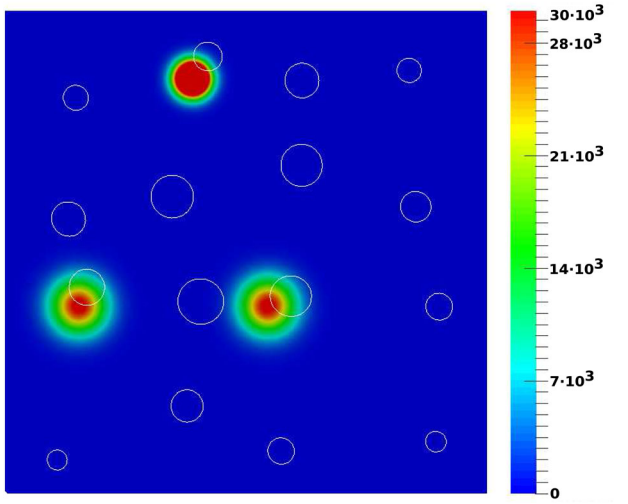

Fig. 9 Total number of particles at 0 fs. *White circles* denote locations of dopants

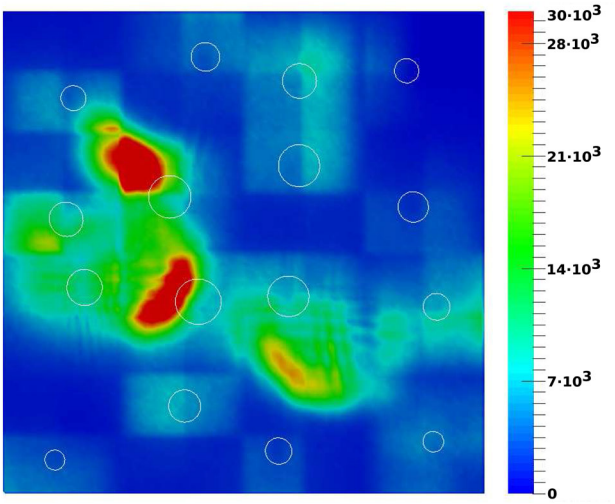

Fig. 11 Total number of particles at 50 fs. *White circles* denote locations of dopants

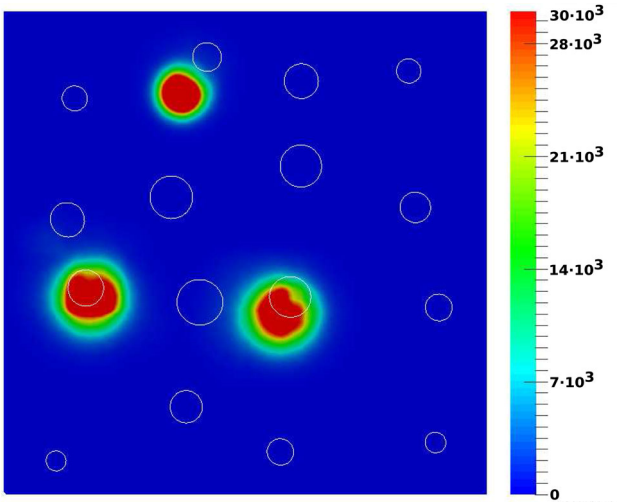

Fig. 10 Total number of particles at 10 fs. *White circles* denote locations of dopants

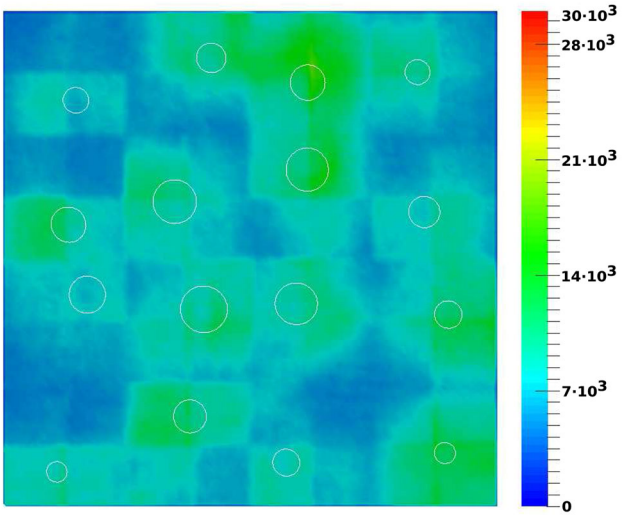

Fig. 12 Total number of particles at 300 fs. *White circles* denote locations of dopants

(3 in cardinal and 2 in diagonal directions for both send and receive operations), and each corner subdomain (*corner*) has to process 6 connections (2 in cardinal and 1 in diagonal directions for both send and receive operations). The number of interior, boundary, and corner subdomain can be computed by $N_{inner} = N - 2(\text{Xparts} + \text{Yparts} - 2)$, $N_{bnd} = 2(\text{Xparts} + \text{Yparts} - 4)$, and $N_{corner} = 4$, where Yparts represents the number of subdomains in the y-direction. As an example: a simulation utilizing 32 MPI processes and a 8×4 (Xparts $= 8$, Yparts $= 4$) decompositioning scheme requires $T_{block} = 376$ point-to-point communications for each time step. T_{block} is approximately three times larger than T_{slab} for this particular scenario, i.e. three times as many communication channels need to be set up at every time step, incurring a considerable additional communication overhead. Furthermore, considering strong scaling (i.e. the number of MPI processes and thus the number of subdomains increases for a given problem), the majority of subdomains are interior subdomains and therefore the number of communication channels increases significantly.

Aside from the communication overhead, the logic identifying the particles leaving a particular subdomain introduces substantial additional overhead, as the overlap areas have to be specialized to cover diagonal cases and the second cardinal direction as compared to the simpler slab decomposition approach.

All in all, block decomposition reduces the required memory per MPI process and drastically increases the number of MPI processes that can (potentially) be used, albeit at the cost of introducing significant overhead, both for the MPI

communication backend and the logic to drive the communication.

4 Results

This section evaluates the parallel execution performance of the spatial decomposition approaches, presented in the preceding sections, by considering the physical problem of the evolution of three minimum-uncertainty wave packages within a two-dimensional domain. Sixteen acceptor dopants (positive charge) are spread over a 128 nm × 128 nm spatial domain, which yields the potential profile shown in Fig. 3. The particle generation rate γ is proportional to potential differences (through the definition of the Wigner potential) and is depicted in Fig. 4. The particle generation is concentrated around each dopant within the region corresponding to the coherence length. Here, a coherence length of 30 nm is used throughout, resulting in a discrete k-space with a resolution of $\Delta k = \frac{\pi}{30\,\mathrm{nm}}$. Reflective boundary conditions are used, therefore, no particles leave the simulation domain.

The initial condition, shown in Fig. 5, is specified by three minimum uncertainty wave packets placed at (50 nm, 50 nm), (50 nm, 110 nm), and (70 nm, 50 nm), having the wave vectors $(6\Delta k, 3\Delta k)$, $(-4\Delta k, -6\Delta k)$, and $(5\Delta k, -4\Delta k)$, respectively. Each wave packet is initialized using 5×10^6 particles. The evolution of the wave packets over 300 fs, using a 0.1 fs time step, is shown in Figs. 5, 6, 7, and 8. The wave packets get *split up* by the potential peaks; interference patterns also become visible.

The total number of particles gives an indication of the computational load and also its distribution in the domain. The maximum number of particles for the entire domain is limited to 64×10^7 particles; the local maximum for each process depends on the total number of processes used.

Figures 9, 10, 11, and 12 depicts the total number of particles (the sum of positively and negatively signed particles) for different time steps as computed by 64 processes. Over time, the entire simulation domain is filled with particles with local maxima occurring around each dopant, according to the generation rate γ (cf. Fig. 4).

A comparison between the density in Figs. 5, 6, 7, and 8 and the corresponding number of particles in Figs. 9, 10, 11, and 12 reveals that there can be a significant number of particles in regions with a very low density since positive and negative particles can compensate each other when calculating physical quantities, like the density.

All simulations presented were performed on the VSC-3 supercomputer [13], which consists of 2020 nodes. Each node provides 16 cores (two 8-core Intel Xeon Ivy Bridge-EP E5-2650v2, 2.6 GHz, Hyperthreading) and 64 GB of system memory; the nodes are connected via an Intel QDR-80 dual-link high-speed InfiniBand fabric.

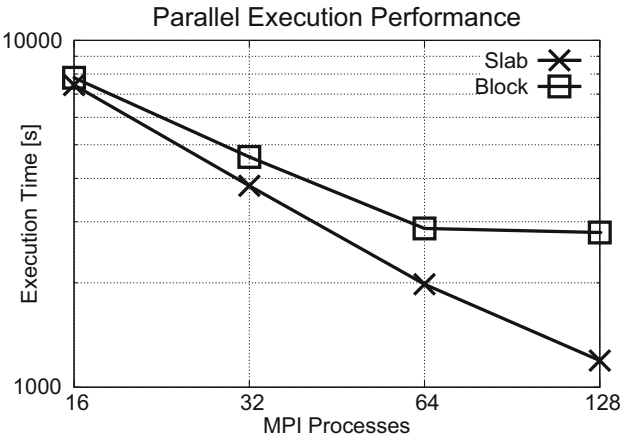

Fig. 13 Comparison of the execution times between the slab and the block decomposition approaches

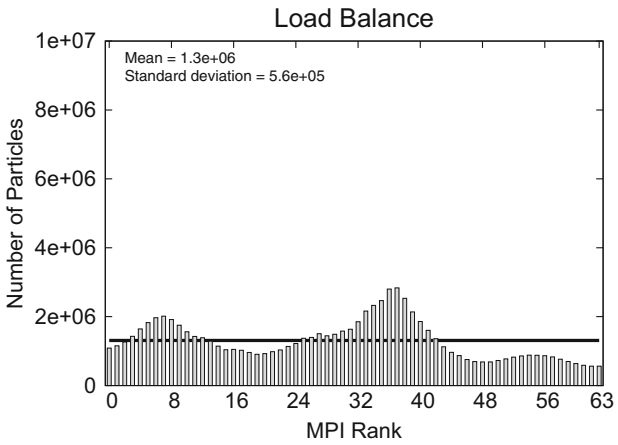

Fig. 14 Load balance of the slab decomposition approach for 64 processes at 100 fs. *Horizontal line* denotes the mean number of particles

The simulation was benchmarked using 16, 32, 64, and 128 MPI processes. Figure 13 depicts the parallel execution performance for the slab and the block decomposition approach. The slab decomposition technique offers a significantly better parallel execution performance. As discussed in Sect. 3, the block decomposition method introduces significant overhead, resulting in inferior performance relative to the slab decomposition approach. Especially for 128 MPI processes, the overhead triggers a stagnation of the scalability as the number of particles (work load) per process is too small relative to the additional communication overhead.

Figures 14, 15, 16, 17, 18, and 19 show the load balance of the slab and the block decomposition approach at 100, 200, and 300 fs. The load balance between the processes shows a variation over time, which is due to the annihilation process. Typically, an annihilation step improves the load balance as the processes with the largest number of particles also experience the strongest reduction, effectively *leveling* the number

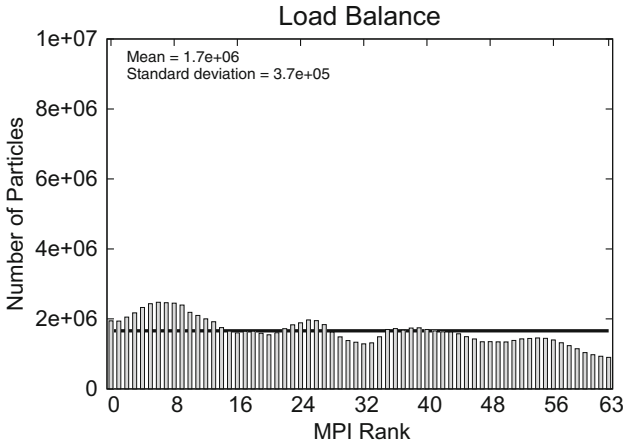

Fig. 15 Load balance of the slab decomposition approach for 64 processes at 200 fs. *Horizontal line* denotes the mean number of particles

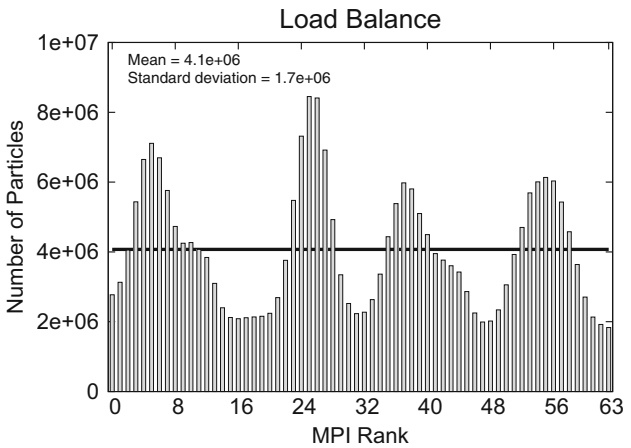

Fig. 16 Load balance of the slab decomposition approach for 64 processes at 300 fs. *Horizontal line* denotes the mean number of particles

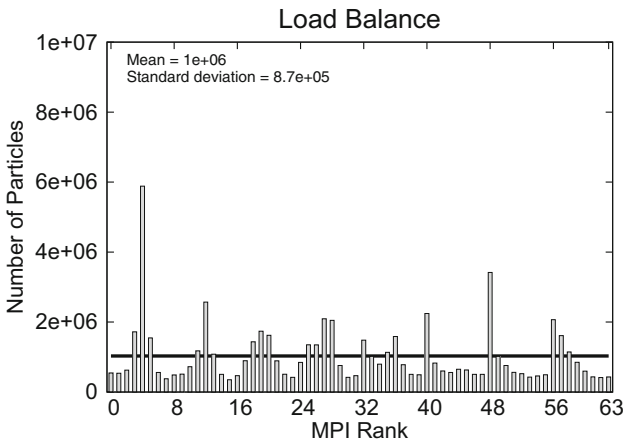

Fig. 17 Load balance of the block decomposition approach for 64 processes at 100 fs. *Horizontal line* denotes the mean number of particles

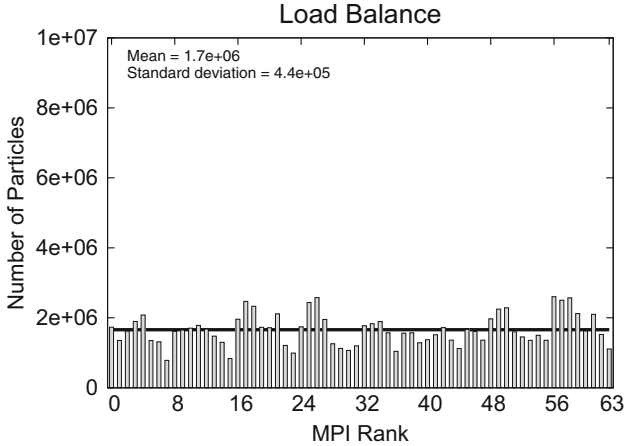

Fig. 18 Load balance of the block decomposition approach for 64 processes at 200 fs. *Horizontal line* denotes the mean number of particles

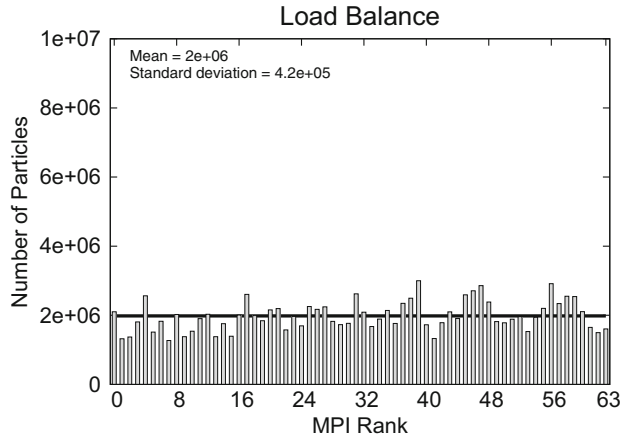

Fig. 19 Load balance of the block decomposition approach for 64 processes at 300 fs. *Horizontal line* denotes the mean number of particles

of particles between the processes. The annihilation intervals for the two decomposition approaches are not the same, which can result in different dynamics of the load balance, but neither one of the two decomposition approaches can be singled out as having a superior load balancing, compared to the other. However, the logic required to drive the block-based communication as well as the communication itself takes about 1.5–4 times longer for the given problem than with the slab decomposition technique.

5 Conclusion

We have investigated the potentially promising uniform block decomposition approach for two-dimensional Wigner Monte Carlo quantum simulations and compared it with our uniform slab decomposition technique. The results show that

the overhead introduced by the block-based communication layer significantly limits parallel performance. For the type of simulation problem considered here, the block decomposition method shows inferior performance when compared to the conventional slab decomposition technique, which, on the other hand, again confirmed its excellent parallel performance. This result is especially interesting, as it shows that the much simpler to implement slab decomposition method is an excellent method for parallelizing highly memory intensive two-dimensional Wigner Monte Carlo quantum simulations based on the signed particle method. Future work will focus on load-balancing approaches, specifically aiming towards three-dimensional problems.

Acknowledgments The computational results presented have been achieved using the Vienna Scientific Cluster (VSC).

References

1. Wigner, E.: On the quantum correction for thermodynamic equilibrium. Phys. Rev. Lett. **40**, 749 (1932). doi:10.1103/PhysRev.40.749
2. Querlioz, D., Dollfus, P.: The Wigner Monte-Carlo Method for Nanoelectronic Devices: Particle Description of Quantum Transport and Decoherence. Wiley, New York (2010). ISBN: 9781848211506
3. Nedjalkov, M., Kosina, H., Selberherr, S., Ringhofer, C., Ferry, D.K.: Unified particle approach to Wigner-Boltzmann transport in small semiconductor devices. Phys. Rev. B **70**, 115319 (2004). doi:10.1103/PhysRevB.70.115319
4. Nedjalkov, M., Schwaha, P., Selberherr, S., Sellier, J.M., Vasileska, D.: Wigner quasi-particle attributes—an asymptotic perspective. Appl. Phys. Lett. **102**(16), 163113 (2013). doi:10.1063/1.4802931
5. Sellier, J., Dimov, I.: The Wigner–Boltzmann Monte Carlo method applied to electron transport in the presence of a single dopant. Comput. Phys. Commun. **185**(10), 2427 (2014). doi:10.1016/j.cpc.2014.05.013
6. Sellier, J., Nedjalkov, M., Dimov, I., Selberherr, S.: Two-dimensional transient Wigner particle model. In: Proceedings of the 18th International Conference on Simulation of Semiconductor Processes and Devices (SISPAD), pp. 404–407 (2013). doi:10.1109/SISPAD.2013.6650660
7. Sellier, J.M., Nedjalkov, M., Dimov, I., Selberherr, S.: The role of annihilation in a Wigner Monte Carlo approach. In: Lirkov, I., Margenov, S., Wasniewski, J. (eds.) Large-Scale Scientific Computing. Springer, Berlin (2014). doi:10.1007/978-3-662-43880-0_20
8. Ellinghaus, P., Weinbub, J., Nedjalkov, M., Selberherr, S., Dimov, I.: Distributed-memory parallelization of the Wigner Monte Carlo method using spatial domain decomposition. J. Comput. Electron. **14**(1), 151 (2015). doi:10.1007/s10825-014-0635-3
9. Dimov, I.: Monte Carlo Methods For Applied Scientists. World Scientific Publishing, Singapore (2005). ISBN: 9789810223298
10. Weinbub, J., Ellinghaus, P., Selberherr, S.: Parallelization of the two-dimensional Wigner Monte Carlo method. In: Lirkov, I., Margenov, S., Wasniewski, J. (eds.) Large-Scale Scientific Computing. Springer, Berlin (2015, in press)
11. Vienna, W.D.: Wigner Ensemble Monte Carlo Simulator. (2015). http://viennawd.sourceforge.net/
12. Hager, G., Wellein, G.: Introduction to High Performance Computing for Scientists and Engineers. CRC Press, Boca Raton, FL (2010). ISBN: 9781439811924
13. Vienna Scientific Cluster. VSC-3. (2015). http://vsc.ac.at/