Procedia Computer Science

Volume 80, 2016, Pages 1897–1905

ICCS 2016. The International Conference on Computational Science

Energy Study of Monte Carlo and Quasi-Monte Carlo Algorithms for Solving Integral Equations

Todor Gurov[1], Aneta Karaivanova[1], and Vassil Alexandrov[2][3]

[1] IICT-BAS, Acad. G. Bonchev St., Bl. 25A, Sofia 1113, Bulgaria
(anet,gurov)@parallel.bas.bg

[2] ICREA-BSC, C/ Jordi Girona 29, 08034, Barcelona, Spain
vassil.alexandrov@bsc.es

[3] Distinguished Visiting Professor, Tecnologico de Monterrey (ITESM), Campus Monterrey, Mexico

Abstract

In the past few years the development of exascale computing technology necessitated to obtain an estimate for the energy consumption when large-scale problems are solved with different high-performance computing (HPC) systems. In this paper we study the energy efficiency of a class of Monte Carlo (MC) and Quasi-Monte Carlo (QMC) algorithms for a given integral equation using hybrid HPC systems. The algorithms are applied to solve quantum kinetic integral equations describing ultra-fast transport in quantum wire. We compare the energy performance of the algorithms using a GPU-based computer platform and CPU-based computer platform both with and without hyper-threading (HT) technology. We use SPRNG library and CURAND generator to produce parallel pseudo-random (PPR) sequences for the MC algorithms on CPU-based and GPU -based platforms, respectively. For our QMC algorithms Sobol and Halton sequences are used to produce parallel quasi-random (PQR) sequences.

We compare the obtained results of the tested algorithms with respect to the given energy metric. The results of our study demonstrate the importance of taking into account not only scalability of the HPC intensive algorithms but also their energy efficiency. They also show the need for further optimisation of the QMC algorithms when GPU-based computing platforms are used.

Keywords: Monte Carlo and Quasi-Monte Carlo algorithms, Integral Equations, Energy Study

1 Introduction

In the past decade the development of energy-efficient algorithms has become more important with increasing power of supercomputer systems and growing size of the applied problems [1, 2]. The newest complex extreme-scale computing systems, based on diverse computing devices (CPU, GPU, accelerators) posed the question of scalability in the light not only of parallel efficiency, but also in terms of energy consumption. The Flops/Watt (F/W) metric

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2016
© The Authors. Published by Elsevier B.V.

doi:10.1016/j.procs.2016.05.492

was introduced and now this metric is used as the standard in measuring the energy efficiency of supercomputing systems (see [3]). Taking into account the probability of an error during a run, some authors [4, 5] propose the *time_to_solution* metric for estimating the algorithm performance. This metric can be written in the following way:

$$F(T, T_0) \times E, \tag{1}$$

where E is the cost of consumed energy; $F(T, T_0)$ is the penalty function; T is the execution time of the given algorithm, and T_0 is the fixed time, i.e. the time in which we want the task (running job) to be completed.

In this work we concentrate on studying energy efficiency of MC and QMC algorithms for integral equations on extreme-scale parallel computing systems with respect to metric (1). The extreme-scale parallel computing systems are HPC systems with low latency interconnection, equipped with GPGPU devices and/or co-processors to speed up the calculations that make use of thousands of processor cores in high-density deployment. Such supercomputers are called hybrid HPC systems if they are equipped additionally with CPU-based platform. The integral equations that we solve with MC and QMC algorithms describe ultrafast carrier transport in semiconductors, namely, a femtosecond relaxation process of optically excited electrons which interact with phonons in one-band semiconductor [6]. In fact we consider the inhomogeneous case - ultrafast transport in quantum wire with applied electric field, which is the more realistic case [7].

The paper is organized as follows: In the next section we present the quantum-kinetic equation in the integral form and the MC and QMC algorithms for solving this equation. The parallel implementation of the algorithms using CPU-based and GPU-based computer platforms is also described. The numerical tests and our findings are presented in section 3. In the conclusion we summarise the main outcomes.

2 Solving Integral Equations by Monte Carlo and Quasi-Monte Carlo algorithms

In our study of energy efficiency we consider MC and QMC algorithms for solving Wigner equation for the nanometer and femtosecond transport regime. For this purpose we use the formulation of the Wigner equation in an inhomogeneous case (in quantum wire - more realistic case) where the electron evolution depends on the energy and space coordinates [7]. The electrons, which can be initially injected or optically generated in the wire, begin to interact with three-dimensional phonons. In this particular case the electron-phonon interaction is described on the quantum-kinetic level by the Levinson equation [9, 10], but the evolution problem becomes inhomogeneous due to the spatial dependence of the initial condition. The direction of the wire is chosen to be z, the corresponding component of the wave vector is k_z. The electrons are in the ground state $\Psi(\mathbf{r}_\perp)$ in the plane normal to the wire, which is an assumption consistent at low temperatures. The initial carrier distribution is assumed Gaussian both in energy and space coordinates, and an electric field can be applied along the wire.

The integral representation of the quantum kinetic equation for the electron Wigner function

f_w in this case has the form [8]:

$$f_w(z,k_z,t) = f_w(z - \frac{\hbar k_z}{m}t + \frac{\hbar \mathbf{F}}{2m}t^2, k_z, 0) + \int_0^t \mathrm{d}t'' \int_{t''}^t \mathrm{d}t' \int d\mathbf{q}'_\perp \int dk'_z \times \tag{2}$$
$$\Big[S(k'_z, k_z, t', t'', \mathbf{q}'_\perp) f_w \left(z - \frac{\hbar k_z}{m}(t-t'') + \frac{\hbar \mathbf{F}}{2m}(t^2 - t''^2) + \frac{\hbar q'_z}{2m}(t'-t''), k'_z, t'' \right)$$
$$-S(k_z, k'_z, t', t'' \mathbf{q}'_\perp) f_w \left(z - \frac{\hbar k_z}{m}(t-t'') + \frac{\hbar \mathbf{F}}{2m}(t^2 - t''^2) - \frac{\hbar q'_z}{2m}(t'-t''), k_z, t'' \right) \Big] .$$

Here, $f_w(z,k_z,t)$ is the Wigner function described in the $2D$ phase space of the carrier wave vector k_z and the position z, and t is the evolution time. The kernel $S(k'_z, k_z, t', t'', \mathbf{q}'_\perp)$ of the integral equation in (2) is well described in [7, 8].

In the inhomogeneous case the wave vector (and respectively the energy) and the density distributions are given by the integrals

$$f(k_z,t) = \int \frac{dz}{2\pi} f_w(z,k_z,t); \qquad n(z,t) = \int \frac{dk_z}{2\pi} f_w(z,k_z,t). \tag{3}$$

MC algorithms [19, 11] and QMC algorithms [16] are developed to estimate the quantities (3) and the Wigner function (2). To sample numerical trajectories in the MC algorithms SPRNG library [14] and CURAND generator is used to produce parallel pseudo-random (PPR) sequences. In the case of QMC algorithms we use low-discrepancy sequences to sample numerical trajectories. The computations were performed with both Sobol and modified Halton sequences. The implementations of these sequences are given in [12, 17] and [13].

2.1 Parallel implementation

The MC and QMC algorithms are perceived as computationally intensive, but naturally parallel. The so-called "master-worker" model is usable for dynamic load-balancing, while static load-balancing is sufficient in the case when the variations in the computational load can be controlled. In this model, a large MC task is split into smaller independent subtasks, which are then executed separately. One process or thread is designated as "master" and is responsible for the communications with the "worker" processes or threads, which perform the actual computations. The partial results are collected and used to assemble an accumulated result with smaller variance than that of a single copy. In the MC algorithms when the subtasks are of the same size, their computational time is also similar, i.e., we can use static load balancing. In the case of QMC computations where one can not afford to lose computations we use the blocking approach in the generation of the sequences, combined with static load balancing. Our parallel implementation uses MPI for the CPU-based parallelisation and CUDA for the GPU-based parallelisation.

2.2 Implementation on CPUs and GPUs

In our numerical tests we compared the CPU performance of the parallel code with and without hyper-threading (HT) for both type algorithms. We note that some codes using HT do not improve the overall speed of calculations because the floating point units of the processor are shared between the threads. In our experience substantial gains may be achieved without any additional coding effort by simply using logical instead of physical cores when sizing the launch of the MPI job. We have observed about 30% improvement of the CPU performance of the MC algorithms [11] when HT is turned on. This result shows that our overall code is reasonably efficient.

In the case of GPGPU-based implementation we used CUDA for parallel computations. Parallel processing is based on splitting the computations on a grid of threads. We use thread size of 256, which is optimal taking into account the relatively large number of registers. Parallel quasi-random (PQR)generators for the scrambled Sobol sequence and modified Halton sequence have been developed and tested in our previous works [12, 13].

3 Numerical Results and Energy Efficiency Study

The MC and QMC algorithms under considaration were tested on the heterogeneous cluster HPCG in the HPC Centre of the Institute of Information and Communication Technologies, Bulgarian Academy of Sciences (IICT-BAS) [15].

This cluster combines CPU-based computing platform with 36 blades BL 280c (dual Intel X5560, 24 GB RAM per a blade,total 576 CPU logic cores) and GPGPU-based platform with high-end GPU computing devices (2 HP ProLiant SL390s G7 servers, each with dual Intel Xeon E5649, 96 GB RAM, with a total of 16 NVIDIA Tesla M2090 graphic cards). Both computing platforms are connected with 2 InfiniBand switches and use 3 storage file systems with a total of 132 TB storage. We note that GPUs have continued to increase in terms of energy usage, while CPU designers have recently focused on improving performance per Watt. High performance GPUs may now be the largest power consumer in a system. The peak performance of any system is essentially limited by the amount of power it can draw. Consequently, performance per watt of a GPU design translates directly into peak performance of a system that uses that design.

The numerical results presented in Tables 1-3 and on Figures 1-3 are obtained for the problem with GaAs material parameters: the electron effective mass is 0.063, the optimal phonon energy is 36 meV, the static and optical dielectric constants are $\varepsilon_s = 12.9$ and $\varepsilon_\infty = 10.92$. The initial condition is a product of two Gaussian distributions of the energy and space. The k_z^2 distribution corresponds to a generating laser pulse with an excess energy of about 180 meV. The z distribution is centered around zero. The side of the wire is chosen to be 10 nanometers. The number of the realized numerical trajectories in MC and QMC algorithms is $N = 10$ million. For the MC algorithms we use SPRNG library [14] to sample numerical trajectories on the CPU-based computing platform while we use CURAND generator on the GPGPU-based platform. Sobol and Halton sequences are used to produce PQR sequences for QMC algorithms.

In our tests the price of energy is assumed to be 10 euro cents per $1KWh$. In case there are no running jobs the energy consumption of n number of CPU blades or GPU devices is denoted by W_0. When we have running jobs using n number of devices than the energy consumption is denoted by W_n. The difference $\Delta W_n = W_n - W_0$ is attributed to the computational workload being run. The function $F(T)$ that penalizes the algorithms is chosen in the following way:

$$F(T) = \exp\left(a|T/T_0 - 1|\right).$$

Using this penalty function allows the metric (1) to find the minimum number of blades and GPU cards necessary for the completion of the task (running job) for a fixed time - T_0, leaving the energy cost close to the minimum. In our test $T_0 = 600s$ and the parameter $a = 1$. The parameter a is chosen so that the results are more easily comparable for all cases.

The test results for MC and QMC algorithms using CPU devices with and without HT are shown in Tables 1 and 2. In each table the optimal values are underlined. Looking at

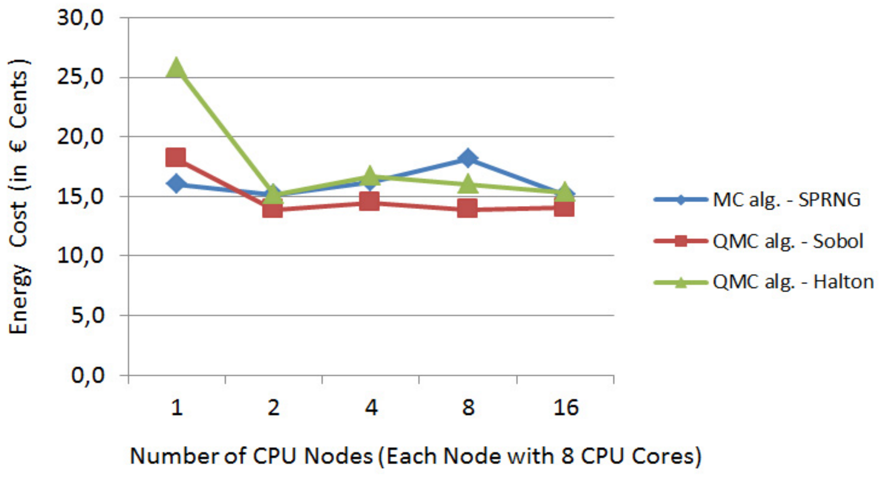

Figure 1: Compering the energy cost using CPU devices without HT for the MC and QMC algorithms.

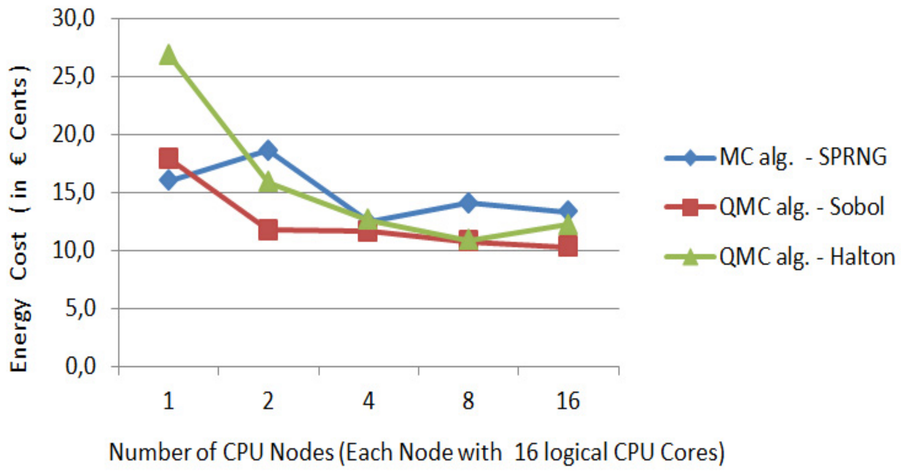

Figure 2: Compering the energy cost using CPU devices with HT for the MC and QMC algorithms.

Table 1: Test results for the MC and QMC algorithmits using CPU devices without Hyper-Threading.

PPR sequence/	Blades/	CPU		Energy	Energy metric
PQR sequence	Cores	Time (s)	ΔW_n	cost: E	$F(T) \times (E)$
	1/8	2728.02	211	16.00	955.95
	2/8	1545.63	353	15.15	92.12
SPRNG	4/8	702.91	831	16.22	24.11
	8/8	350.89	1863	18.16	31.31
	16/8	193.64	2819	15.16	39.78
	1/8	2401.52	272	18.14	365.37
	2/8	1206.56	415	13.91	38.22
Sobol	4/8	605.12	861	14.47	14.60
	8/8	307.55	1624	13.87	22.59
	16/8	160.12	3152	14.02	29.18
	1/8	2499.10	372	25.82	611.84
	2/8	1254.81	435	15.16	45.16
Halton	4/8	631.52	951	16.68	17.58
	8/8	355.82	1622	16.03	24.08
	16/8	179.81	3073	15.35	30.92

Table 2: Test results for the MC and QMC algorithmits using CPU devices with Hyper-Threading.

PPR sequence/	Blades/	CPU		Energy	Energy metric
PQR sequence	Cores/HT	Time (s)	ΔW_n	cost: E	$F(T) \times (E)$
	1/8/2	1818.17	609	15.98	121.71
	2/8/2	904.50	712	18.63	30.94
SPRNG	4/8/2	456.96	1130	12.47	15.82
	8/8/2	231.15	1767	14.10	26.07
	16/8/2	123.20	3624	13.33	29.51
	1/8/2	1555.49	414	17.89	87.94
	2/8/2	780.93	543	11.78	15.92
Sobol	4/8/2	388.36	1081	11.66	16.59
	8/8/2	195.33	1987	10.78	21.16
	16/8/2	100.56	3689	10.30	23.69
	1/8/2	1587.63	609	26.86	139.29
	2/8/2	803.14	712	15.88	22.28
Halton	4/8/2	403.22	1130	12.66	17.57
	8/8/2	222.39	1767	10.91	20.47
	16/8/2	121.20	3624	12.20	27.10

the results we see that there is a difference between the SPRNG, Sobol and Halton sequences. Here we observe that by turning on the hyper-threading in the case of Sobol sequence one can achieve the same metrics with just 2 blades instead of 4. For both the Halton sequences and the SPRNG sequences it is optimal to use 4 blades.

The energy cost for the different algorithms using CPU-based computations with and without HT is compared on Figures 1-2. We see that QMC algorithms consume a little bit less

Table 3: Test results for the MC and QMC algorithms using GPU devices.

PPR sequence/ PQR sequence	NVIDIA Tesla M2090	GPGPU Time (s)	ΔW_n	Energy cost: E	Energy metric $F(T) \times (E)$
CURAND	<u>1</u>	564.34	398	6.24	<u>6.62</u>
	2	292.37	555	4.51	7.53
	4	164.41	895	4.09	8.45
	8	105.28	1404	4.11	9.36
	16	87.56	2525	6.14	14.42
Sobol	1	893.24	156	3.87	6.31
	<u>2</u>	456.32	316	4.01	<u>5.09</u>
	4	246.15	638	4.36	7.87
	8	155.88	1295	5.61	11.75
	16	103.97	2419	6.99	15.97
Halton	1	803.85	158	3.53	4.96
	<u>2</u>	411.01	307	3.50	<u>4.80</u>
	4	228.86	614	3.90	7.25
	8	142.07	1094	4.32	9.26
	16	98.24	2494	6.81	15.71

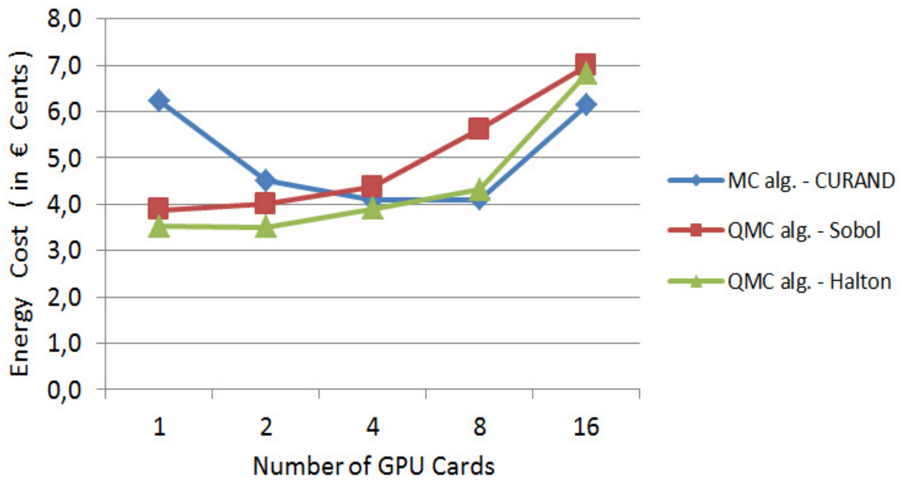

Figure 3: Compering the energy cost using GPU devices for the MC and QMC algorithms.

energy that MC algorithms as in the case of HT the results are improving. Taking in the account that QMC algorithms converge to the exact solution better than MC algorithms [18] we can conclude that QMC algorithms in our case are preferable for CPU-based computational platform.

In the case of GPGPU computations the test results for both algorithms are presented in Table 3 while the energy cost using different number of GPU cards is shown on Figure 3. We observe that in case of small number of GPU devices the energy cost of the QMC algorithms is smaller that for MC algorithm. With increasing the number of GPU devices the picture is changed. This situation can be explained by the fact that algorithms are not optimized for calculations with many cards. In part this is due to the relatively high initial startup overhead for the CUDA and MPI implementations.

Comparing results using both computational platform we can conclude that GPU-based platform is preferable because the energy cost is reduced by 30% to 40%.

4 Conclusion

In this paper we compared the energy performance of a class of MC and QMC algorithms for solving quantum kinetic integral equations describing ultra-fast transport in quantum wire. The algorithms were implemented on a GPU-based computer platform and on CPU-based computer platform with and without hyper-threading. Although the quasi-random sequence generators are not optimized for the GPU-based platform, the QMC algorithms demonstrate better energy efficiency. As a penalty function in our work we used an exponential function. The purpose of this function is to find the minimal number of CPU/GPU devices needed to solve the given problem for a fixed execution time.

The new computing platforms lead to new challenges in programming. That is why it is necessary to optimize the codes (in some cases they need rewriting) of the existing quasi-random generators in order to exploit the advantages of the computer hardware with accelerators. In our future work we plan to compare the energy performance of MC and QMC algorithms for solving linear algebra problems. We also foresee porting these algorithms on new extreme-scale parallel system Avitohol [20], which consists of 150 heterogeneous computing HP Cluster Platform SL250S GEN8 servers, equipped with 2 Intel Xeon E 2650 v2 CPUs and 2 Intel Xeon Phi 7120P coprocessors and study the efficiency there.

Acknowledgment

This work was supported by the National Science Fund of Bulgaria under Grant DFNI-I02/8.

References

[1] Demmel, J., Gearhart, A., Lipshitz, B., Schwartz, O., Perfect Strong Scaling Using No Additional Energy. Proc. of IEEE 27th IPDPS13. IEEE Computer Society, 2013.

[2] Meswani, M., Carrington, L., Unat, D., Peraza, J., Snavely, A., Baden, S., and Poole, S., Modeling and Predicting Application Performance on Hardware Accelerators, International Journal of High Performance Computing (2012).

[3] The Green 500, www.green500.org, EEHPCWG_PowerMeasurementMethodology.pdf, 2015

[4] Bekas, C., Curioni, A., A new energy aware performance metric, Springer, Comput. Sci. Res Dev (2010) 25: 187-195, DOI 10.1007/s00450-010-0119-z.

[5] Bekas, C., Curioni, A., Fedulova, I., Low cost high performance uncertainty quantification, Workshop on HPC finance, SC09, Portland, OR, USA, (November 2009).

[6] Gurov, T.V., Nedjalkov, M., Whitlock, P.A., Kosina, H., Selberherr, S., Femtosecond Relaxation of Hot Electrons by Phonon Emission in Presence of Electric Field, Physica B, vol. 314, pp. 301304, (2002).

[7] Nedjalkov, M., Gurov, T., Kosina, H., Vasileska D., Palankovski, V., Femtosecond Evolution of Spatially Inhomogeneous Carrier Excitations: Part I: Kinetic Approach, Springer LNCS, 3743, pp. 149-156, (2006).

[8] Gurov, T., Atanassov, E., Dimov I., Palankovski, V., Femtosecond Evolution of Spatially Inhomogeneous Carrier Excitations: Part II: Stochastic Approach and GRID Implementation, Springer LNCS, 3743, pp. 157-163, (2006).

[9] Levinson, I., Translational invariance in uniform fields and the equation for the density matrix in the Wigner representation, Sov. Phys. JETP, **30**, pp. 362-367, 1970.

[10] Herbst, M., Glanemann, M., Axt, V., and Kuhn, T., Electron-phonon quantum kinetics for spatially inhomogeneous excitations, Physical Review B, **67**, 195305:1–18, 2003.

[11] Karaivanova, A., Atanassov, E., and Gurov, T., Monte Carlo Simulation of Ultrafast Carrier Transport: Scalability Study, ICCS2013, Published by Elsevier B.V., Procedia Computer Science, v.18, 2298 2306, (2013).

[12] Atanassov, E., et all, Tuning the Generation of Sobol Sequence with Owen Scrambling, LNCS 5910, Springer, 459-466, 2010.

[13] Atanassov, E. I. and Durchova, M. K., Generating and testing the modified Halton sequences, LNCS 2542, Springer, 91-98, 2003.

[14] Scalable Parallel Random Number Generators Library for Parallel Monte Carlo Computations, http://www.sprng.org/ , 2016.

[15] Atanassov, E., Gurov, T., Karaivanova, A., Ivanovska, S., Durchova, M., Georgiev, D., Dimitrov, D., Tuning for Scalability on Hybrid HPC Cluster, Mathematics in Industry, Cambridge Scholar Publishing, pp. 64-77, (2014).

[16] E. Atanassov et all, Ultra-fast Semiconductor Carrier Transport Simulation on the Grid, Sci. Int. J. for Par. and Dist. Comp., SCPE, Vol. 11, no.2, 137-147, 2010.

[17] Sobol, I., Asotsky, D., Kreinin, A., Kucherenko, S., Construction and Comparison of High-Dimensional Sobol Generators. Wilmott Journal Nov: 6479, (2011) .

[18] Karaivanova, A., Atanassov, E., Gurov, T., Ivanovska S., Durchova, M., Parallel Quasi-Random Applications on Heterogeneous Grid, Journal of Scalable Computing: Practice and Experience (SCPE), Vol. 11, no.1, pp.73-80, (2010).

[19] Atanassov, E., Gurov, T., Karaivanova, A., Nedjalkov, M., Vasileska, D., Raleva, K., Electron-phonon interaction in nanowires: A Monte Carlo study of the effect of the field, Mathematics and Computers in Simulation, No: 81, pp. 515-521, (2010).

[20] Avitohol HPC system, http://www.hpc.acad.bg/index.php/system-1/, 2016.